# StarPU Internal Handbook

This manual documents the internal usage of StarPU version 1.3.7. Its contents was last updated on 13 October 2020.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

# Chapter 2

# StarPU Core

## 2.1 StarPU Core Entities

TODO

### 2.1.1 Overview

Execution entities:

- **worker**: A worker (see Workers, Workers and Scheduling Contexts) entity is a CPU thread created by StarPU to manage one computing unit. The computing unit can be a local CPU core, an accelerator or GPU device, or — on the master side when running in master-slave distributed mode — a remote slave computing node. It is responsible for querying scheduling policies for tasks to execute.

- **sched_context**: A scheduling context (see Scheduling Contexts, Workers and Scheduling Contexts) is a logical set of workers governed by an instance of a scheduling policy. It defines the computing units to which the scheduling policy instance may assign work entities.

- **driver**: A driver is the set of hardware-dependent routines used by a worker to initialize its associated computing unit, execute work entities on it, and finalize the computing unit usage at the end of the session.

Work entities:

- **task**: A task is a high level work request submitted to StarPU by the application, or internally by StarPU itself.

- **job**: A job is a low level view of a work request. It is not exposed to the application. A job structure may be shared among several task structures in the case of a parallel task.

Data entities:

- **data handle**: A data handle is a high-level, application opaque object designating a piece of data currently registered to the StarPU data management layer. Internally, it is a _starpu_data_state structure.

- **data replicate**: A data replicate is a low-level object designating one copy of a piece of data registered to StarPU as a data handle, residing in one memory node managed by StarPU. It is not exposed to the application.

### 2.1.2 Workers

A **worker** is a CPU thread created by StarPU. Its role is to manage one computing unit. This computing unit can be a local CPU core, in which case, the worker thread manages the actual CPU core to which it is assigned; or it can be a computing device such as a GPU or an accelerator (or even a remote computing node when StarPU is running in distributed master-slave mode.) When a worker manages a computing device, the CPU core to which the worker's thread is by default exclusively assigned to the device management work and does not participate to computation.

#### 2.1.2.1 States

**Scheduling operations related state**

While a worker is conducting a scheduling operations, e.g. the worker is in the process of selecting a new task to execute, flag state_sched_op_pending is set to `!0`, otherwise it is set to `0`.

While state_sched_op_pending is !0, the following exhaustive list of operations on that workers are restricted in the stated way:

- adding the worker to a context is not allowed;

- removing the worker from a context is not allowed;

- adding the worker to a parallel task team is not allowed;

- removing the worker from a parallel task team is not allowed;

- querying state information about the worker is only allowed while `state_relax_refcnt > 0`;

  - in particular, querying whether the worker is blocked on a parallel team entry is only allowed while `state_relax_refcnt > 0`.

Entering and leaving the state_sched_op_pending state is done through calls to _starpu_worker_enter_sched_op() and _starpu_worker_leave_sched_op() respectively (see these functions in use in functions _starpu_get_worker←_task() and _starpu_get_multi_worker_task()). These calls ensure that any pending conflicting operation deferred while the worker was in the state_sched_op_pending state is performed in an orderly manner.

**Scheduling contexts related states**

Flag `state_changing_ctx_notice` is set to `!0` when a thread is about to add the worker to a scheduling context or remove it from a scheduling context, and is currently waiting for a safe window to do so, until the targeted worker is not in a scheduling operation or parallel task operation anymore. This flag set to `!0` will also prevent the targeted worker to attempt a fresh scheduling operation or parallel task operation to avoid starving conditions. However, a scheduling operation that was already in progress before the notice is allowed to complete.

Flag `state_changing_ctx_waiting` is set to `!0` when a scheduling context worker addition or removal involving the targeted worker is about to occur and the worker is currently performing a scheduling operation to tell the targeted worker that the initiator thread is waiting for the scheduling operation to complete and should be woken up upon completion.

**Relaxed synchronization related states**

Any StarPU worker may participate to scheduling operations, and in this process, may be forced to observe state information from other workers. A StarPU worker thread may therefore be observed by any thread, even other StarPU workers. Since workers may observe each other in any order, it is not possible to rely exclusively on the `sched_mutex` of each worker to protect the observation of worker state flags by other workers, because worker A observing worker B would involve locking workers in (A B) sequence, while worker B observing worker A would involve locking workers in (B A) sequence, leading to lock inversion deadlocks.

In consequence, no thread must hold more than one worker's sched_mutex at any time. Instead, workers implement a relaxed locking scheme based on the `state_relax_refcnt` counter, itself protected by the worker's sched←_mutex. When `state_relax_refcnt > 0`, the targeted worker state flags may be observed, otherwise the thread attempting the observation must repeatedly wait on the targeted worker's `sched_cond` condition until `state_relax_refcnt > 0`.

The relaxed mode, while on, can actually be seen as a transactional consistency model, where concurrent accesses are authorized and potential conflicts are resolved after the fact. When the relaxed mode is off, the consistency model becomes a mutual exclusion model, where the sched_mutex of the worker must be held in order to access or change the worker state.

**Parallel tasks related states**

When a worker is scheduled to participate to the execution of a parallel task, it must wait for the whole team of workers participating to the execution of this task to be ready. While the worker waits for its teammates, it is not available to run other tasks or perform other operations. Such a waiting operation can therefore not start while conflicting operations such as scheduling operations and scheduling context resizing involving the worker are ongoing. Conversely these operations and other may query weather the worker is blocked on a parallel task entry with starpu_worker_is_blocked_in_parallel().

The starpu_worker_is_blocked_in_parallel() function is allowed to proceed while and only while `state_relax`↵
`_refcnt > 0`. Due to the relaxed worker locking scheme, the `state_blocked_in_parallel` flag of the
targeted worker may change after it has been observed by an observer thread. In consequence, flag `state_`↵
`blocked_in_parallel_observed` of the targeted worker is set to `1` by the observer immediately after the
observation to "taint" the targeted worker. The targeted worker will clear the `state_blocked_in_parallel`↵
`_observed` flag tainting and defer the processing of parallel task related requests until a full scheduling operation
shot completes without the `state_blocked_in_parallel_observed` flag being tainted again. The pur-
pose of this tainting flag is to prevent parallel task operations to be started immediately after the observation of a
transient scheduling state.

Worker's management of parallel tasks is governed by the following set of state flags and counters:

- `state_blocked_in_parallel`: set to `!0` while the worker is currently blocked on a parallel task;

- `state_blocked_in_parallel_observed`: set to `!0` to taint the worker when a thread has ob-
  served the state_blocked_in_parallel flag of this worker while its `state_relax_refcnt` state counter
  was $>0$. Any pending request to add or remove the worker from a parallel task team will be deferred until a
  whole scheduling operation shot completes without being tainted again.

- `state_block_in_parallel_req`: set to `!0` when a thread is waiting on a request for the worker to
  be added to a parallel task team. Must be protected by the worker's `sched_mutex`.

- `state_block_in_parallel_ack`: set to `!0` by the worker when acknowledging a request for being
  added to a parallel task team. Must be protected by the worker's `sched_mutex`.

- `state_unblock_in_parallel_req`: set to `!0` when a thread is waiting on a request for the worker
  to be removed from a parallel task team. Must be protected by the worker's `sched_mutex`.

- `state_unblock_in_parallel_ack`: set to `!0` by the worker when acknowledging a request for
  being removed from a parallel task team. Must be protected by the worker's `sched_mutex`.

- `block_in_parallel_ref_count`: counts the number of consecutive pending requests to enter par-
  allel task teams. Only the first of a train of requests for entering parallel task teams triggers the transition of
  the `state_block_in_parallel_req` flag from `0` to `1`. Only the last of a train of requests to leave a
  parallel task team triggers the transition of flag `state_unblock_in_parallel_req` from `0` to `1`. Must
  be protected by the worker's `sched_mutex`.

### 2.1.2.2 Operations

**Entry point**

All the operations of a worker are handled in an iterative fashion, either by the application code on a thread launched
by the application, or automatically by StarPU on a device-dependent CPU thread launched by StarPU. Whether a
worker's operation cycle is managed automatically or not is controlled per session by the field `not_launched`↵
`_drivers` of the `starpu_conf` struct, and is decided in _starpu_launch_drivers() function.

When managed automatically, cycles of operations for a worker are handled by the corresponding driver specific
`_starpu_<DRV>_worker()` function, where `DRV` is a driver name such as cpu (`_starpu_cpu_worker`)
or cuda (`_starpu_cuda_worker`), for instance. Otherwise, the application must supply a thread which will
repeatedly call starpu_driver_run_once() for the corresponding worker.

In both cases, control is then transferred to _starpu_cpu_driver_run_once() (or the corresponding driver specific
func). The cycle of operations typically includes, at least, the following operations:

- **task scheduling**

- **parallel task team build-up**

- **task input processing**

- **data transfer processing**

- **task execution**

When the worker cycles are handled by StarPU automatically, the iterative operation processing ends when the
`running` field of `_starpu_config` becomes false. This field should not be read directly, instead it should be
read through the [_starpu_machine_is_running()](#) function.

**Task scheduling**

If the worker does not yet have a queued task, it calls _starpu_get_worker_task() to try and obtain a task. This may involve scheduling operations such as stealing a queued but not yet executed task from another worker. The operation may not necessarily succeed if no tasks are ready and/or suitable to run on the worker's computing unit.

**Parallel task team build-up**

If the worker has a task ready to run and the corresponding job has a size $>1$, then the task is a parallel job and the worker must synchronize with the other workers participating to the parallel execution of the job to assign a unique rank for each worker. The synchronization is done through the job's `sync_mutex` mutex.

**Task input processing**

Before the task can be executed, its input data must be made available on a memory node reachable by the worker's computing unit. To do so, the worker calls _starpu_fetch_task_input()

**Data transfer processing**

The worker makes pending data transfers (involving memory node(s) that it is driving) progress, with a call to __↩ starpu_datawizard_progress(),

**Task execution**

Once the worker has a pending task assigned and the input data for that task are available in the memory node reachable by the worker's computing unit, the worker calls _starpu_cpu_driver_execute_task() (or the corresponding driver specific function) to proceed to the execution of the task.

### 2.1.3 Scheduling Contexts

A scheduling context is a logical set of workers governed by an instance of a scheduling policy. Tasks submitted to a given scheduling context are confined to the computing units governed by the workers belonging to this scheduling context at the time they get scheduled.

A scheduling context is identified by an unsigned integer identifier between $0$ and `STARPU_NMAX_SCHED_CT↩ XS - 1`. The `STARPU_NMAX_SCHED_CTXS` identifier value is reserved to indicated an unallocated, invalid or deleted scheduling context.

Accesses to the scheduling context structure are governed by a multiple-readers/single-writer lock (`rwlock` field). Changes to the structure contents, additions or removals of workers, statistics updates, all must be done with proper exclusive write access.

### 2.1.4 Workers and Scheduling Contexts

A worker can be assigned to one or more **scheduling contexts**. It exclusively receives tasks submitted to the scheduling context(s) it is currently assigned at the time such tasks are scheduled. A worker may add itself to or remove itself from a scheduling context.

**Locking and synchronization rules between workers and scheduling contexts**

A thread currently holding a worker sched_mutex must not attempt to acquire a scheduling context rwlock, neither for writing nor for reading. Such an attempt constitutes a lock inversion and may result in a deadlock.

A worker currently in a scheduling operation must enter the relaxed state before attempting to acquire a scheduling context rwlock, either for reading or for writing.

When the set of workers assigned to a scheduling context is about to be modified, all the workers in the union between the workers belonging to the scheduling context before the change and the workers expected to belong to the scheduling context after the change must be notified using the notify_workers_about_changing_ctx_pending() function prior to the update. After the update, all the workers in that same union must be notified for the update completion with a call to notify_workers_about_changing_ctx_done().

The function notify_workers_about_changing_ctx_pending() places every worker passed in argument in a state compatible with changing the scheduling context assignment of that worker, possibly blocking until that worker leaves incompatible states such as a pending scheduling operation. If the caller of `notify_workers_about↩ _changing_ctx_pending()` is itself a worker included in the set of workers passed in argument, it does not notify itself, with the assumption that the worker is already calling `notify_workers_about_changing↩ _ctx_pending()` from a state compatible with a scheduling context assignment update. Once a worker has

been notified about a scheduling context change pending, it cannot proceed with incompatible operations such as a scheduling operation until it receives a notification that the context update operation is complete.

### 2.1.5 Drivers

Each driver defines a set of routines depending on some specific hardware. These routines include hardware discovery/initialization, task execution, device memory management and data transfers.

While most hardware dependent routines are in source files located in the `/src/drivers` subdirectory of the StarPU tree, some can be found elsewhere in the tree such as `src/datawizard/malloc.c` for memory allocation routines or the subdirectories of `src/datawizard/interfaces/` for data transfer routines.

The driver ABI defined in the [_starpu_driver_ops](#) structure includes the following operations:

- .init: initialize a driver instance for the calling worker managing a hardware computing unit compatible with this driver.

- .run_once: perform a single driver progress cycle for the calling worker (see [Operations](#)).

- .deinit: deinitialize the driver instance for the calling worker

- .run: executes the following sequence automatically: call .init, repeatedly call .run_once until the function [_starpu_machine_is_running()](#) returns false, call .deinit.

The source code common to all drivers is shared in `src/drivers/driver_common/driver_←common.[ch]`. This file includes services such as grabbing a new task to execute on a worker, managing statistics accounting on job startup and completion and updating the worker status

#### 2.1.5.1 Master/Slave Drivers

A subset of the drivers corresponds to drivers managing computing units in master/slave mode, that is, drivers involving a local master instance managing one or more remote slave instances on the targeted device(s). This includes devices such as discrete manycore accelerators (e.g. Intel's Knight Corners board, for instance), or pseudo devices such as a cluster of cpu nodes driver through StarPU's MPI master/slave mode. A driver instance on the master side is named the **source**, while a driver instances on the slave side is named the **sink**.

A significant part of the work realized on the source and sink sides of master/slave drivers is identical among all master/slave drivers, due to the similarities in the software pattern. Therefore, many routines are shared among all these drivers in the `src/drivers/mp_common` subdirectory. In particular, a set of default commands to be used between sources and sinks is defined, assuming the availability of some communication channel between them (see enum _starpu_mp_command)

TODO

### 2.1.6 Tasks and Jobs

TODO

### 2.1.7 Data

TODO

# Chapter 3

# Module Index

## 3.1 Modules

Here is a list of all modules:

# Chapter 4

# Module Documentation

## 4.1   Workers

**Data Structures**

- struct _starpu_worker
- struct _starpu_combined_worker
- struct _starpu_worker_set
- struct _starpu_machine_topology
- struct _starpu_machine_config
- struct _starpu_machine_config.bindid_workers

**Macros**

- #define **STARPU_MAX_PIPELINE**
- #define **starpu_worker_get_count**
- #define **starpu_worker_get_id**
- #define **_starpu_worker_get_id_check**(f, l)
- #define **starpu_worker_relax_on**
- #define **starpu_worker_relax_off**
- #define **starpu_worker_get_relax_state**

**Enumerations**

- enum **initialization** { **UNINITIALIZED**, **CHANGING**, **INITIALIZED** }

**Functions**

- void _starpu_set_argc_argv (int ∗argc, char ∗∗∗argv)
- int ∗ **_starpu_get_argc** ()
- char ∗∗∗ **_starpu_get_argv** ()
- void _starpu_conf_check_environment (struct starpu_conf ∗conf)
- void _starpu_may_pause (void)
- static unsigned _starpu_machine_is_running (void)
- void _starpu_worker_init (struct _starpu_worker ∗workerarg, struct _starpu_machine_config ∗pconfig)
- uint32_t _starpu_worker_exists (struct starpu_task ∗)
- uint32_t _starpu_can_submit_cuda_task (void)
- uint32_t _starpu_can_submit_cpu_task (void)
- uint32_t _starpu_can_submit_opencl_task (void)
- unsigned _starpu_worker_can_block (unsigned memnode, struct _starpu_worker ∗worker)
- void _starpu_block_worker (int workerid, starpu_pthread_cond_t ∗cond, starpu_pthread_mutex_t ∗mutex)
- void _starpu_driver_start (struct _starpu_worker ∗worker, unsigned fut_key, unsigned sync)
- void _starpu_worker_start (struct _starpu_worker ∗worker, unsigned fut_key, unsigned sync)

- static unsigned **_starpu_worker_get_count** (void)
- static void _starpu_set_local_worker_key (struct _starpu_worker ∗worker)
- static struct _starpu_worker ∗ _starpu_get_local_worker_key (void)
- static void _starpu_set_local_worker_set_key (struct _starpu_worker_set ∗worker)
- static struct _starpu_worker_set ∗ _starpu_get_local_worker_set_key (void)
- static struct _starpu_worker ∗ _starpu_get_worker_struct (unsigned id)
- static struct _starpu_sched_ctx ∗ _starpu_get_sched_ctx_struct (unsigned id)
- struct _starpu_combined_worker ∗ **_starpu_get_combined_worker_struct** (unsigned id)
- static struct _starpu_machine_config ∗ _starpu_get_machine_config (void)
- static int _starpu_get_disable_kernels (void)
- static enum _starpu_worker_status _starpu_worker_get_status (int workerid)
- static void _starpu_worker_set_status (int workerid, enum _starpu_worker_status status)
- static struct _starpu_sched_ctx ∗ _starpu_get_initial_sched_ctx (void)
- int **starpu_worker_get_nids_by_type** (enum starpu_worker_archtype type, int ∗workerids, int maxsize)
- int starpu_worker_get_nids_ctx_free_by_type (enum starpu_worker_archtype type, int ∗workerids, int maxsize)
- static unsigned **_starpu_worker_mutex_is_sched_mutex** (int workerid, starpu_pthread_mutex_t ∗mutex)
- static int **_starpu_worker_get_nsched_ctxs** (int workerid)
- static unsigned _starpu_get_nsched_ctxs (void)
- static int _starpu_worker_get_id (void)
- static unsigned __starpu_worker_get_id_check (const char ∗f, int l)
- enum starpu_node_kind **_starpu_worker_get_node_kind** (enum starpu_worker_archtype type)
- void **_starpu_worker_set_stream_ctx** (unsigned workerid, struct _starpu_sched_ctx ∗sched_ctx)
- struct _starpu_sched_ctx ∗ **_starpu_worker_get_ctx_stream** (unsigned stream_workerid)
- static void _starpu_worker_request_blocking_in_parallel (struct _starpu_worker ∗const worker)
- static void _starpu_worker_request_unblocking_in_parallel (struct _starpu_worker ∗const worker)
- static void _starpu_worker_process_block_in_parallel_requests (struct _starpu_worker ∗const worker)
- static void _starpu_worker_enter_sched_op (struct _starpu_worker ∗const worker)
- void _starpu_worker_apply_deferred_ctx_changes (void)
- static void **_starpu_worker_leave_sched_op** (struct _starpu_worker ∗const worker)
- static int **_starpu_worker_sched_op_pending** (void)
- static void _starpu_worker_enter_changing_ctx_op (struct _starpu_worker ∗const worker)
- static void _starpu_worker_leave_changing_ctx_op (struct _starpu_worker ∗const worker)
- static void _starpu_worker_relax_on (void)
- static void _starpu_worker_relax_on_locked (struct _starpu_worker ∗worker)
- static void **_starpu_worker_relax_off** (void)
- static void **_starpu_worker_relax_off_locked** (void)
- static int **_starpu_worker_get_relax_state** (void)
- static void _starpu_worker_lock (int workerid)
- static int **_starpu_worker_trylock** (int workerid)
- static void **_starpu_worker_unlock** (int workerid)
- static void **_starpu_worker_lock_self** (void)
- static void **_starpu_worker_unlock_self** (void)
- static int **_starpu_wake_worker_relax** (int workerid)
- int **starpu_wake_worker_relax_light** (int workerid)
- void _starpu_worker_refuse_task (struct _starpu_worker ∗worker, struct starpu_task ∗task)

## Variables

- int **_starpu_worker_parallel_blocks**
- struct _starpu_machine_config _starpu_config **STARPU_ATTRIBUTE_INTERNAL**

### 4.1.1 Detailed Description

### 4.1.2 Data Structure Documentation

#### 4.1.2.1 struct _starpu_worker

This is initialized by [_starpu_worker_init()](#)

Data Fields

| struct _starpu_machine_config ∗ | config | |
|---:|:---|:---|
| starpu_pthread_mutex_t | mutex | |
| enum starpu_worker_archtype | arch | what is the type of worker ? |
| uint32_t | worker_mask | what is the type of worker ? |
| struct starpu_perfmodel_arch | perf_arch | in case there are different models of the same arch |
| starpu_pthread_t | worker_thread | the thread which runs the worker |
| unsigned | devid | which cpu/gpu/etc is controlled by the worker ? |
| unsigned | subworkerid | which sub-worker this one is for the cpu/gpu |
| int | bindid | which cpu is the driver bound to ? (logical index) |
| int | workerid | uniquely identify the worker among all processing units types |
| int | combined_workerid | combined worker currently using this worker |
| int | current_rank | current rank in case the worker is used in a parallel fashion |
| int | worker_size | size of the worker in case we use a combined worker |
| starpu_pthread_cond_t | started_cond | indicate when the worker is ready |
| starpu_pthread_cond_t | ready_cond | indicate when the worker is ready |
| unsigned | memory_node | which memory node is the worker associated with ? |
| unsigned | numa_memory_node | which numa memory node is the worker associated with? (logical index) |
| starpu_pthread_cond_t | sched_cond | condition variable used for passive waiting operations on worker STARPU_PTHREAD_COND_B↩ ROADCAST must be used instead of STARPU_PTHREAD_COND↩ _SIGNAL, since the condition is shared for multiple purpose |
| starpu_pthread_mutex_t | sched_mutex | mutex protecting sched_cond |
| unsigned | state_relax_refcnt | mark scheduling sections where other workers can safely access the worker state |
| unsigned | state_sched_op_pending | a task pop is ongoing even though sched_mutex may temporarily be unlocked |
| unsigned | state_changing_ctx_waiting | a thread is waiting for operations such as pop to complete before acquiring sched_mutex and modifying the worker ctx |
| unsigned | state_changing_ctx_notice | the worker ctx is about to change or being changed, wait for flag to be cleared before starting new scheduling operations |
| unsigned | state_blocked_in_parallel | worker is currently blocked on a parallel section |

**Data Fields**

| | | |
|---:|---|---|
| unsigned | state_blocked_in_parallel_observed | the blocked state of the worker has been observed by another worker during a relaxed section |
| unsigned | state_block_in_parallel_req | a request for state transition from unblocked to blocked is pending |
| unsigned | state_block_in_parallel_ack | a block request has been honored |
| unsigned | state_unblock_in_parallel_req | a request for state transition from blocked to unblocked is pending |
| unsigned | state_unblock_in_parallel_ack | an unblock request has been honored |
| unsigned | block_in_parallel_ref_count | cumulative blocking depth<br><br>• =0 worker unblocked<br><br>• >0 worker blocked<br><br>• transition from 0 to 1 triggers a block_req<br><br>• transition from 1 to 0 triggers a unblock_req |
| starpu_pthread_t | thread_changing_ctx | thread currently changing a sched_ctx containing the worker |
| struct _starpu_ctx_change_list | ctx_change_list | list of deferred context changes when the current thread is a worker, _and_ this worker is in a scheduling operation, new ctx changes are queued to this list for subsequent processing once worker completes the ongoing scheduling operation |
| struct starpu_task_list | local_tasks | this queue contains tasks that have been explicitly submitted to that queue |
| struct starpu_task ** | local_ordered_tasks | this queue contains tasks that have been explicitly submitted to that queue with an explicit order |
| unsigned | local_ordered_tasks_size | this records the size of local_ordered_tasks |
| unsigned | current_ordered_task | this records the index (within local_ordered_tasks) of the next ordered task to be executed |
| unsigned | current_ordered_task_order | this records the order of the next ordered task to be executed |
| struct starpu_task * | current_task | task currently executed by this worker (non-pipelined version) |
| struct starpu_task * | current_tasks[STARPU_MAX_PIPELINE] | tasks currently executed by this worker (pipelined version) |
| starpu_pthread_wait_t | wait | |
| struct timespec | cl_start | Codelet start time of the task currently running |
| struct timespec | cl_end | Codelet end time of the last task running |
| unsigned char | first_task | Index of first task in the pipeline |

**Data Fields**

| | | |
|---:|---|---|
| unsigned char | ntasks | number of tasks in the pipeline |
| unsigned char | pipeline_length | number of tasks to be put in the pipeline |
| unsigned char | pipeline_stuck | whether a task prevents us from pipelining |
| struct _starpu_worker_set ∗ | set | in case this worker belongs to a set |
| unsigned | worker_is_running | |
| unsigned | worker_is_initialized | |
| enum _starpu_worker_status | status | what is the worker doing now ? (eg. CALLBACK) |
| unsigned | state_keep_awake | !0 if a task has been pushed to the worker and the task has not yet been seen by the worker, the worker should no go to sleep before processing this task |
| char | name[128] | |
| char | short_name[32] | |
| unsigned | run_by_starpu | Is this run by StarPU or directly by the application ? |
| struct _starpu_driver_ops ∗ | driver_ops | |
| struct _starpu_sched_ctx_list ∗ | sched_ctx_list | |
| int | tmp_sched_ctx | |
| unsigned | nsched_ctxs | the no of contexts a worker belongs to |
| struct _starpu_barrier_counter | tasks_barrier | wait for the tasks submitted |
| unsigned | has_prev_init | had already been inited in another ctx |
| unsigned | removed_from_ctx[STARPU_NMAX_SCHED_CTXS+1] | |
| unsigned | spinning_backoff | number of cycles to pause when spinning |
| unsigned | nb_buffers_transferred | number of piece of data already send to worker |
| unsigned | nb_buffers_totransfer | number of piece of data already send to worker |
| struct starpu_task ∗ | task_transferring | The buffers of this task are being sent |
| unsigned | shares_tasks_lists[STARPU_NMAX_SCHED_CTXS+1] | indicate whether the workers shares tasks lists with other workers in this case when removing him from a context it disapears instantly |
| unsigned | poped_in_ctx[STARPU_NMAX_SCHED_CTXS+1] | boolean to chose the next ctx a worker will pop into |
| unsigned | reverse_phase[2] | boolean indicating at which moment we checked all ctxs and change phase for the booleab poped_in_ctx one for each of the 2 priorities |
| unsigned | pop_ctx_priority | indicate which priority of ctx is currently active: the values are 0 or 1 |

**Data Fields**

| | | |
|---:|---|---|
| unsigned | is_slave_somewhere | bool to indicate if the worker is slave in a ctx |
| struct _starpu_sched_ctx ∗ | stream_ctx | |
| hwloc_bitmap_t | hwloc_cpu_set | |
| hwloc_obj_t | hwloc_obj | |
| char | padding[STARPU_CACHELINE_SIZE] | Keep this last, to make sure to separate worker data in separate cache lines. |

**4.1.2.2 struct _starpu_combined_worker**

**Data Fields**

| | | |
|---:|---|---|
| struct starpu_perfmodel_arch | perf_arch | in case there are different models of the same arch |
| uint32_t | worker_mask | what is the type of workers ? |
| int | worker_size | |
| unsigned | memory_node | which memory node is associated that worker to ? |
| int | combined_workerid[STARPU_NMAXWORKERS] | |
| hwloc_bitmap_t | hwloc_cpu_set | |
| char | padding[STARPU_CACHELINE_SIZE] | Keep this last, to make sure to separate worker data in separate cache lines. |

**4.1.2.3 struct _starpu_worker_set**

in case a single CPU worker may control multiple accelerators

**Data Fields**

| | | |
|---:|---|---|
| starpu_pthread_mutex_t | mutex | |
| starpu_pthread_t | worker_thread | the thread which runs the worker |
| unsigned | nworkers | |
| unsigned | started | Only one thread for the whole set |
| void ∗ | retval | |
| struct _starpu_worker ∗ | workers | |
| starpu_pthread_cond_t | ready_cond | indicate when the set is ready |
| unsigned | set_is_initialized | |

**4.1.2.4 struct _starpu_machine_topology**

**Data Fields**

| | | |
|---:|---|---|
| unsigned | nworkers | Total number of workers. |
| unsigned | ncombinedworkers | Total number of combined workers. |
| unsigned | nsched_ctxs | |
| hwloc_topology_t | hwtopology | Topology as detected by hwloc. |
| struct starpu_tree ∗ | tree | custom hwloc tree |

**Data Fields**

| | | | |
|---:|---|---|---|
| unsigned | nhwcpus | | Total number of CPU cores, as detected by the topology code. May be different from the actual number of CPU workers. |
| unsigned | nhwpus | | Total number of PUs (i.e. threads), as detected by the topology code. May be different from the actual number of PU workers. |
| unsigned | nhwcudagpus | | Total number of CUDA devices, as detected. May be different from the actual number of CUDA workers. |
| unsigned | nhwopenclgpus | | Total number of OpenCL devices, as detected. May be different from the actual number of OpenCL workers. |
| unsigned | nhwmpi | | Total number of MPI nodes, as detected. May be different from the actual number of node workers. |
| unsigned | ncpus | | Actual number of CPU workers used by StarPU. |
| unsigned | ncudagpus | | Actual number of CUDA GPUs used by StarPU. |
| unsigned | nworkerpercuda | | |
| int | cuda_th_per_stream | | |
| int | cuda_th_per_dev | | |
| unsigned | nopenclgpus | | Actual number of OpenCL workers used by StarPU. |
| unsigned | nmpidevices | | Actual number of MPI workers used by StarPU. |
| unsigned | nhwmpidevices | | |
| unsigned | nhwmpicores[STARPU_MAXMPIDEVS] | | Each MPI node has its set of cores. |
| unsigned | nmpicores[STARPU_MAXMPIDEVS] | | |
| unsigned | nhwmicdevices | | Topology of MP nodes (MIC) as well as necessary objects to communicate with them. |
| unsigned | nmicdevices | | |
| unsigned | nhwmiccores[STARPU_MAXMICDEVS] | | Each MIC node has its set of cores. |
| unsigned | nmiccores[STARPU_MAXMICDEVS] | | |
| unsigned | workers_bindid[STARPU_NMAXWORKERS] | | Indicates the successive logical PU identifier that should be used to bind the workers. It is either filled according to the user's explicit parameters (from starpu_conf) or according to the STARPU_WORKERS_CPUID env. variable. Otherwise, a round-robin policy is used to distributed the workers over the cores. |
| unsigned | workers_cuda_gpuid[STARPU_NMAXWORKERS] | | Indicates the successive CUDA identifier that should be used by the CUDA driver. It is either filled according to the user's explicit parameters (from starpu_conf) or according to the STARPU_WORKERS_CUDAID env. variable. Otherwise, they are taken in ID order. |

**Data Fields**

| | | |
|---:|---|---|
| unsigned | workers_opencl_gpuid[STARPU_NMAXWORKERS] | Indicates the successive OpenCL identifier that should be used by the OpenCL driver. It is either filled according to the user's explicit parameters (from starpu_conf) or according to the STARPU_WORKERS_OPENCLID env. variable. Otherwise, they are taken in ID order. |
| unsigned | workers_mpi_ms_deviceid[STARPU_NMAXWORKERS] | unsigned workers_mic_← deviceid[STARPU_NMAXWORKERS]; |

### 4.1.2.5 struct _starpu_machine_config

**Data Fields**

| | | |
|---:|---|---|
| struct _starpu_machine_topology | topology | |
| int | cpu_depth | |
| int | pu_depth | |
| int | current_bindid | Where to bind next worker ? |
| char | currently_bound[STARPU_NMAXWORKERS] | |
| char | currently_shared[STARPU_NMAXWORKERS] | |
| int | current_cuda_gpuid | Which GPU(s) do we use for CUDA ? |
| int | current_opencl_gpuid | Which GPU(s) do we use for OpenCL ? |
| int | current_mic_deviceid | Which MIC do we use? |
| int | current_mpi_deviceid | Which MPI do we use? |
| int | cpus_nodeid | Memory node for cpus, if only one |
| int | cuda_nodeid | Memory node for CUDA, if only one |
| int | opencl_nodeid | Memory node for OpenCL, if only one |
| int | mic_nodeid | Memory node for MIC, if only one |
| int | mpi_nodeid | Memory node for MPI, if only one |
| char | padding1[STARPU_CACHELINE_SIZE] | Separate out previous variables from per-worker data. |
| struct _starpu_worker | workers[STARPU_NMAXWORKERS] | Basic workers : each of this worker is running its own driver and can be combined with other basic workers. |
| struct _starpu_combined_worker | combined_workers[STARPU_NMAX_COMBINEDWORKERS] | Combined workers : these worker are a combination of basic workers that can run parallel tasks together. |
| starpu_pthread_mutex_t | submitted_mutex | |
| char | padding2[STARPU_CACHELINE_SIZE] | Separate out previous mutex from the rest of the data. |
| struct _starpu_machine_config | bindid_workers | Translation table from bindid to worker IDs |
| unsigned | nbindid | size of bindid_workers |

**Data Fields**

| uint32_t | worker_mask | This bitmask indicates which kinds of worker are available. For instance it is possible to test if there is a CUDA worker with the result of (worker_mask & STARPU_CUDA). |
| --- | --- | --- |
| struct starpu_conf | conf | either the user given configuration passed to starpu_init or a default configuration |
| unsigned | running | this flag is set until the runtime is stopped |
| int | disable_kernels | |
| int | pause_depth | Number of calls to starpu_pause() - calls to starpu_resume(). When >0, StarPU should pause. |
| struct _starpu_sched_ctx | sched_ctxs[STARPU_NMAX_SCHED_CTXS+1] | The sched ctx of the current instance of starpu |
| unsigned | submitting | this flag is set until the application is finished submitting tasks |
| int | watchdog_ok | |

#### 4.1.2.6  struct _starpu_machine_config.bindid_workers

Translation table from bindid to worker IDs

**Data Fields**

| int ∗ | workerids | |
| --- | --- | --- |
| unsigned | nworkers | size of workerids |

### 4.1.3  Function Documentation

#### 4.1.3.1  _starpu_set_argc_argv()

```
void _starpu_set_argc_argv (
            int * argc,
            char *** argv )
```
Three functions to manage argv, argc

#### 4.1.3.2  _starpu_conf_check_environment()

```
void _starpu_conf_check_environment (
            struct starpu_conf * conf )
```
Fill conf with environment variables

#### 4.1.3.3  _starpu_may_pause()

```
void _starpu_may_pause (
            void  )
```
Called by the driver when it is ready to pause

### 4.1.3.4 _starpu_machine_is_running()

```
static unsigned _starpu_machine_is_running (
            void  )  [inline], [static]
```
Has starpu_shutdown already been called ?

### 4.1.3.5 _starpu_worker_init()

```
void _starpu_worker_init (
            struct _starpu_worker * workerarg,
            struct _starpu_machine_config * pconfig )
```
initialise a worker

### 4.1.3.6 _starpu_worker_exists()

```
uint32_t _starpu_worker_exists (
            struct starpu_task *  )
```
Check if there is a worker that may execute the task.

### 4.1.3.7 _starpu_can_submit_cuda_task()

```
uint32_t _starpu_can_submit_cuda_task (
            void  )
```
Is there a worker that can execute CUDA code ?

### 4.1.3.8 _starpu_can_submit_cpu_task()

```
uint32_t _starpu_can_submit_cpu_task (
            void  )
```
Is there a worker that can execute CPU code ?

### 4.1.3.9 _starpu_can_submit_opencl_task()

```
uint32_t _starpu_can_submit_opencl_task (
            void  )
```
Is there a worker that can execute OpenCL code ?

### 4.1.3.10 _starpu_worker_can_block()

```
unsigned _starpu_worker_can_block (
            unsigned memnode,
            struct _starpu_worker * worker )
```
Check whether there is anything that the worker should do instead of sleeping (waiting on something to happen).

### 4.1.3.11 _starpu_block_worker()

```
void _starpu_block_worker (
            int workerid,
            starpu_pthread_cond_t * cond,
            starpu_pthread_mutex_t * mutex )
```
This function must be called to block a worker. It puts the worker in a sleeping state until there is some event that forces the worker to wake up.

### 4.1.3.12 _starpu_driver_start()

```
void _starpu_driver_start (
            struct _starpu_worker * worker,
            unsigned fut_key,
            unsigned sync )
```
This function initializes the current driver for the given worker

**4.1.3.13 _starpu_worker_start()**

```
void _starpu_worker_start (
            struct _starpu_worker * worker,
            unsigned fut_key,
            unsigned sync )
```
This function initializes the current thread for the given worker

**4.1.3.14 _starpu_set_local_worker_key()**

```
static void _starpu_set_local_worker_key (
            struct _starpu_worker * worker ) [inline], [static]
```
The _starpu_worker structure describes all the state of a StarPU worker. This function sets the pthread key which stores a pointer to this structure.

**4.1.3.15 _starpu_get_local_worker_key()**

```
static struct _starpu_worker* _starpu_get_local_worker_key (
            void ) [static]
```
Returns the _starpu_worker structure that describes the state of the current worker.

**4.1.3.16 _starpu_set_local_worker_set_key()**

```
static void _starpu_set_local_worker_set_key (
            struct _starpu_worker_set * worker ) [inline], [static]
```
The _starpu_worker_set structure describes all the state of a StarPU worker_set. This function sets the pthread key which stores a pointer to this structure.

**4.1.3.17 _starpu_get_local_worker_set_key()**

```
static struct _starpu_worker_set* _starpu_get_local_worker_set_key (
            void ) [static]
```
Returns the _starpu_worker_set structure that describes the state of the current worker_set.

**4.1.3.18 _starpu_get_worker_struct()**

```
static struct _starpu_worker* _starpu_get_worker_struct (
            unsigned id ) [static]
```
Returns the _starpu_worker structure that describes the state of the specified worker.

**4.1.3.19 _starpu_get_sched_ctx_struct()**

```
static struct _starpu_sched_ctx* _starpu_get_sched_ctx_struct (
            unsigned id ) [static]
```
Returns the starpu_sched_ctx structure that describes the state of the specified ctx

**4.1.3.20 _starpu_get_machine_config()**

```
static struct _starpu_machine_config* _starpu_get_machine_config (
            void ) [static]
```
Returns the structure that describes the overall machine configuration (eg. all workers and topology).

**4.1.3.21 _starpu_get_disable_kernels()**

```
static int _starpu_get_disable_kernels (
            void ) [inline], [static]
```
Return whether kernels should be run ($<=0$) or not ($>0$)

**4.1.3.22 _starpu_worker_get_status()**

```
static enum _starpu_worker_status _starpu_worker_get_status (
            int workerid ) [inline], [static]
```
Retrieve the status which indicates what the worker is currently doing.

**4.1.3.23 _starpu_worker_set_status()**

```
static void _starpu_worker_set_status (
            int workerid,
            enum _starpu_worker_status status ) [inline], [static]
```
Change the status of the worker which indicates what the worker is currently doing (eg. executing a callback).

**4.1.3.24 _starpu_get_initial_sched_ctx()**

```
static struct _starpu_sched_ctx* _starpu_get_initial_sched_ctx (
            void ) [static]
```
We keep an initial sched ctx which might be used in case no other ctx is available

**4.1.3.25 starpu_worker_get_nids_ctx_free_by_type()**

```
int starpu_worker_get_nids_ctx_free_by_type (
            enum starpu_worker_archtype type,
            int * workerids,
            int maxsize )
```
returns workers not belonging to any context, be careful no mutex is used, the list might not be updated

**4.1.3.26 _starpu_get_nsched_ctxs()**

```
static unsigned _starpu_get_nsched_ctxs (
            void ) [inline], [static]
```
Get the total number of sched_ctxs created till now

**4.1.3.27 _starpu_worker_get_id()**

```
static int _starpu_worker_get_id (
            void ) [inline], [static]
```
Inlined version when building the core.

**4.1.3.28 __starpu_worker_get_id_check()**

```
static unsigned __starpu_worker_get_id_check (
            const char * f,
            int l ) [inline], [static]
```
Similar behaviour to starpu_worker_get_id() but fails when called from outside a worker This returns an unsigned object on purpose, so that the caller is sure to get a positive value

**4.1.3.29 _starpu_worker_request_blocking_in_parallel()**

```
static void _starpu_worker_request_blocking_in_parallel (
            struct _starpu_worker *const worker ) [inline], [static]
```
Send a request to the worker to block, before a parallel task is about to begin.
Must be called with worker's sched_mutex held.

**4.1.3.30 _starpu_worker_request_unblocking_in_parallel()**

```
static void _starpu_worker_request_unblocking_in_parallel (
            struct _starpu_worker *const worker ) [inline], [static]
```
Send a request to the worker to unblock, after a parallel task is complete.
Must be called with worker's sched_mutex held.

### 4.1.3.31 _starpu_worker_process_block_in_parallel_requests()

```
static void _starpu_worker_process_block_in_parallel_requests (
            struct _starpu_worker *const worker )  [inline], [static]
```
Called by the the worker to process incoming requests to block or unblock on parallel task boundaries.
Must be called with worker's sched_mutex held.

### 4.1.3.32 _starpu_worker_enter_sched_op()

```
static void _starpu_worker_enter_sched_op (
            struct _starpu_worker *const worker )  [inline], [static]
```
Mark the beginning of a scheduling operation by the worker. No worker blocking operations on parallel tasks and no scheduling context change operations must be performed on contexts containing the worker, on contexts about to add the worker and on contexts about to remove the worker, while the scheduling operation is in process. The sched mutex of the worker may only be acquired permanently by another thread when no scheduling operation is in process, or when a scheduling operation is in process _and_ worker->state_relax_refcnt!=0. If a scheduling operation is in process _and_ worker->state_relax_refcnt==0, a thread other than the worker must wait on condition worker->sched_cond for worker->state_relax_refcnt!=0 to become true, before acquiring the worker sched mutex permanently.
Must be called with worker's sched_mutex held.

### 4.1.3.33 _starpu_worker_apply_deferred_ctx_changes()

```
void _starpu_worker_apply_deferred_ctx_changes (
            void  )
```
Mark the end of a scheduling operation by the worker.
Must be called with worker's sched_mutex held.

### 4.1.3.34 _starpu_worker_enter_changing_ctx_op()

```
static void _starpu_worker_enter_changing_ctx_op (
            struct _starpu_worker *const worker )  [inline], [static]
```
Must be called before altering a context related to the worker whether about adding the worker to a context, removing it from a context or modifying the set of workers of a context of which the worker is a member, to mark the beginning of a context change operation. The sched mutex of the worker must be held before calling this function.
Must be called with worker's sched_mutex held.

### 4.1.3.35 _starpu_worker_leave_changing_ctx_op()

```
static void _starpu_worker_leave_changing_ctx_op (
            struct _starpu_worker *const worker )  [inline], [static]
```
Mark the end of a context change operation.
Must be called with worker's sched_mutex held.

### 4.1.3.36 _starpu_worker_relax_on()

```
static void _starpu_worker_relax_on (
            void  )  [inline], [static]
```
Temporarily allow other worker to access current worker state, when still scheduling, but the scheduling has not yet been made or is already done

### 4.1.3.37 _starpu_worker_relax_on_locked()

```
static void _starpu_worker_relax_on_locked (
            struct _starpu_worker * worker )  [inline], [static]
```
Same, but with current worker mutex already held

### 4.1.3.38 _starpu_worker_lock()

```
static void _starpu_worker_lock (
            int workerid )  [inline], [static]
```
lock a worker for observing contents
notes:

- if the observed worker is not in state_relax_refcnt, the function block until the state is reached

### 4.1.3.39 _starpu_worker_refuse_task()

```
void _starpu_worker_refuse_task (
            struct _starpu_worker * worker,
            struct starpu_task * task )
```
Allow a worker pulling a task it cannot execute to properly refuse it and send it back to the scheduler.

# Chapter 5

# File Index

## 5.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 6

# StarPU File Documentation

## 6.1 barrier.h File Reference

```
#include <starpu_thread.h>
```

**Data Structures**

- struct _starpu_barrier

**Functions**

- int **_starpu_barrier_init** (struct _starpu_barrier ∗barrier, int count)
- int **_starpu_barrier_destroy** (struct _starpu_barrier ∗barrier)
- int **_starpu_barrier_wait** (struct _starpu_barrier ∗barrier)

### 6.1.1 Data Structure Documentation

#### 6.1.1.1 struct _starpu_barrier

**Data Fields**

| | | |
|---:|---|---|
| unsigned | count | |
| unsigned | reached_start | |
| unsigned | reached_exit | |
| double | reached_flops | |
| starpu_pthread_mutex_t | mutex | |
| starpu_pthread_mutex_t | mutex_exit | |
| starpu_pthread_cond_t | cond | |

## 6.2 barrier_counter.h File Reference

```
#include <common/utils.h>
#include <common/barrier.h>
```

**Data Structures**

- struct _starpu_barrier_counter

**Functions**

- int **_starpu_barrier_counter_init** (struct _starpu_barrier_counter ∗barrier_c, unsigned count)
- int **_starpu_barrier_counter_destroy** (struct _starpu_barrier_counter ∗barrier_c)
- int **_starpu_barrier_counter_wait_for_empty_counter** (struct _starpu_barrier_counter ∗barrier_c)
- int **_starpu_barrier_counter_wait_until_counter_reaches_down_to_n** (struct _starpu_barrier_counter ∗barrier_c, unsigned n)
- int **_starpu_barrier_counter_wait_until_counter_reaches_up_to_n** (struct _starpu_barrier_counter ∗barrier_c, unsigned n)
- int **_starpu_barrier_counter_wait_for_full_counter** (struct _starpu_barrier_counter ∗barrier_c)
- int **_starpu_barrier_counter_decrement_until_empty_counter** (struct _starpu_barrier_counter ∗barrier↩_c, double flops)
- int **_starpu_barrier_counter_increment_until_full_counter** (struct _starpu_barrier_counter ∗barrier_↩c, double flops)
- int **_starpu_barrier_counter_increment** (struct _starpu_barrier_counter ∗barrier_c, double flops)
- int **_starpu_barrier_counter_check** (struct _starpu_barrier_counter ∗barrier_c)
- int **_starpu_barrier_counter_get_reached_start** (struct _starpu_barrier_counter ∗barrier_c)
- double **_starpu_barrier_counter_get_reached_flops** (struct _starpu_barrier_counter ∗barrier_c)

### 6.2.1 Data Structure Documentation

#### 6.2.1.1 struct _starpu_barrier_counter

**Data Fields**

| struct _starpu_barrier | barrier | |
| ---: | --- | --- |
| unsigned | min_threshold | |
| unsigned | max_threshold | |
| starpu_pthread_cond_t | cond2 | |

## 6.3 bound.h File Reference

```
#include <starpu.h>
#include <starpu_bound.h>
#include <core/jobs.h>
```

**Functions**

- void _starpu_bound_record (struct _starpu_job ∗j)
- void _starpu_bound_tag_dep (starpu_tag_t id, starpu_tag_t dep_id)
- void _starpu_bound_task_dep (struct _starpu_job ∗j, struct _starpu_job ∗dep_j)
- void _starpu_bound_job_id_dep (starpu_data_handle_t handle, struct _starpu_job ∗dep_j, unsigned long job_id)
- void starpu_bound_clear (void)

**Variables**

- int _starpu_bound_recording

### 6.3.1 Function Documentation

**6.3.1.1 _starpu_bound_record()**

```
void _starpu_bound_record (
            struct _starpu_job * j )
```
Record task for bound computation

**6.3.1.2 _starpu_bound_tag_dep()**

```
void _starpu_bound_tag_dep (
            starpu_tag_t id,
            starpu_tag_t dep_id )
```
Record tag dependency: id depends on dep_id

**6.3.1.3 _starpu_bound_task_dep()**

```
void _starpu_bound_task_dep (
            struct _starpu_job * j,
            struct _starpu_job * dep_j )
```
Record task dependency: j depends on dep_j

**6.3.1.4 _starpu_bound_job_id_dep()**

```
void _starpu_bound_job_id_dep (
            starpu_data_handle_t handle,
            struct _starpu_job * dep_j,
            unsigned long job_id )
```
Record job id dependency: j depends on job_id

**6.3.1.5 starpu_bound_clear()**

```
void starpu_bound_clear (
            void  )
```
Clear recording

## 6.3.2 Variable Documentation

**6.3.2.1 _starpu_bound_recording**

```
int _starpu_bound_recording
```
Are we recording?

## 6.4 cg.h File Reference

```
#include <starpu.h>
#include <common/config.h>
```

**Data Structures**

- struct _starpu_cg_list
- struct _starpu_cg
- union _starpu_cg.succ
- struct _starpu_cg.succ.succ_apps

**Macros**

- #define STARPU_DYNAMIC_DEPS_SIZE

**Typedefs**

- typedef struct _starpu_notify_job_start_data **_starpu_notify_job_start_data**

**Enumerations**

- enum **_starpu_cg_type** { **STARPU_CG_APPS**, **STARPU_CG_TAG**, **STARPU_CG_TASK** }

**Functions**

- void **_starpu_notify_dependencies** (struct _starpu_job *j)
- void **_starpu_job_notify_start** (struct _starpu_job *j, struct starpu_perfmodel_arch *perf_arch)
- void **_starpu_job_notify_ready_soon** (struct _starpu_job *j, _starpu_notify_job_start_data *data)
- void **_starpu_cg_list_init** (struct _starpu_cg_list *list)
- void **_starpu_cg_list_deinit** (struct _starpu_cg_list *list)
- int **_starpu_add_successor_to_cg_list** (struct _starpu_cg_list *successors, struct _starpu_cg *cg)
- int **_starpu_list_task_successors_in_cg_list** (struct _starpu_cg_list *successors, unsigned ndeps, struct starpu_task *task_array[ ])
- int **_starpu_list_task_scheduled_successors_in_cg_list** (struct _starpu_cg_list *successors, unsigned ndeps, struct starpu_task *task_array[ ])
- int **_starpu_list_tag_successors_in_cg_list** (struct _starpu_cg_list *successors, unsigned ndeps, starpu_tag_t tag_array[ ])
- void **_starpu_notify_cg** (void *pred, struct _starpu_cg *cg)
- void **_starpu_notify_cg_list** (void *pred, struct _starpu_cg_list *successors)
- void **_starpu_notify_job_start_cg_list** (void *pred, struct _starpu_cg_list *successors, _starpu_notify_↩ job_start_data *data)
- void **_starpu_notify_task_dependencies** (struct _starpu_job *j)
- void **_starpu_notify_job_start_tasks** (struct _starpu_job *j, _starpu_notify_job_start_data *data)

### 6.4.1 Data Structure Documentation

#### 6.4.1.1 struct _starpu_cg_list

Completion Group list, records both the number of expected notifications before the completion can start, and the list of successors when the completion is finished.

**Data Fields**

| struct _starpu_spinlock | lock | Protects atomicity of the list and the terminated flag |
|---|---|---|
| unsigned | ndeps | Number of notifications to be waited for |
| unsigned | ndeps_completed | |
| unsigned | terminated | Whether the completion is finished. For restartable/restarted tasks, only the first iteration is taken into account here. |
| unsigned | nsuccs | List of successors |
| unsigned | succ_list_size | How many allocated items in succ |
| struct _starpu_cg ** | succ | |

#### 6.4.1.2 struct _starpu_cg

Completion Group

**Data Fields**

| unsigned | ntags | number of tags depended on |
|---|---|---|
| unsigned | remaining | number of remaining tags |
| enum _starpu_cg_type | cg_type | |

**Data Fields**

| | | |
|---|---|---|
| union _starpu_cg | succ | |

**6.4.1.3 union _starpu_cg.succ**

**Data Fields**

| | | |
|---|---|---|
| struct _starpu_tag ∗ | tag | STARPU_CG_TAG |
| struct _starpu_job ∗ | job | STARPU_CG_TASK |
| succ | succ_apps | STARPU_CG_APPS in case this completion group is related to an application, we have to explicitly wake the waiting thread instead of reschedule the corresponding task |

**6.4.1.4 struct _starpu_cg.succ.succ_apps**

STARPU_CG_APPS in case this completion group is related to an application, we have to explicitly wake the waiting thread instead of reschedule the corresponding task

**Data Fields**

| | | |
|---|---|---|
| unsigned | completed | |
| starpu_pthread_mutex_t | cg_mutex | |
| starpu_pthread_cond_t | cg_cond | |

## 6.4.2 Macro Definition Documentation

**6.4.2.1 STARPU_DYNAMIC_DEPS_SIZE**

```
#define STARPU_DYNAMIC_DEPS_SIZE
```
we do not necessarily want to allocate room for 256 dependencies, but we want to handle the few situation where there are a lot of dependencies as well

## 6.5 coherency.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/starpu_spinlock.h>
#include <common/rwlock.h>
#include <common/timing.h>
#include <common/fxt.h>
#include <common/list.h>
#include <datawizard/interfaces/data_interface.h>
#include <datawizard/datastats.h>
#include <datawizard/memstats.h>
#include <datawizard/data_request.h>
```

**Data Structures**

- struct _starpu_data_replicate

- struct _starpu_jobid_list
- struct _starpu_task_wrapper_list
- struct _starpu_task_wrapper_dlist
- struct _starpu_data_state

## Typedefs

- typedef void(∗ **_starpu_data_handle_unregister_hook**) (starpu_data_handle_t)

## Enumerations

- enum **_starpu_cache_state** { **STARPU_OWNER**, **STARPU_SHARED**, **STARPU_INVALID** }

## Functions

- int _starpu_fetch_data_on_node (starpu_data_handle_t handle, int node, struct _starpu_data_replicate ∗replicate, enum starpu_data_access_mode mode, unsigned detached, enum _starpu_is_prefetch is_↩ prefetch, unsigned async, void(∗callback_func)(void ∗), void ∗callback_arg, int prio, const char ∗origin)
- void _starpu_release_data_on_node (struct _starpu_data_state ∗state, uint32_t default_wt_mask, struct _↩ starpu_data_replicate ∗replicate)
- void **_starpu_update_data_state** (starpu_data_handle_t handle, struct _starpu_data_replicate ∗requesting↩ _replicate, enum starpu_data_access_mode mode)
- uint32_t **_starpu_get_data_refcnt** (struct _starpu_data_state ∗state, unsigned node)
- size_t **_starpu_data_get_size** (starpu_data_handle_t handle)
- size_t **_starpu_data_get_alloc_size** (starpu_data_handle_t handle)
- uint32_t **_starpu_data_get_footprint** (starpu_data_handle_t handle)
- void **__starpu_push_task_output** (struct _starpu_job ∗j)
- void _starpu_push_task_output (struct _starpu_job ∗j)
- void **_starpu_release_nowhere_task_output** (struct _starpu_job ∗j)
- STARPU_ATTRIBUTE_WARN_UNUSED_RESULT int **_starpu_fetch_task_input** (struct starpu_task ∗task, struct _starpu_job ∗j, int async)
- void **_starpu_fetch_task_input_tail** (struct starpu_task ∗task, struct _starpu_job ∗j, struct _starpu_worker ∗worker)
- void **_starpu_fetch_nowhere_task_input** (struct _starpu_job ∗j)
- int **_starpu_select_src_node** (struct _starpu_data_state ∗state, unsigned destination)
- int **_starpu_determine_request_path** (starpu_data_handle_t handle, int src_node, int dst_node, enum starpu_data_access_mode mode, int max_len, unsigned ∗src_nodes, unsigned ∗dst_nodes, unsigned ∗handling_nodes, unsigned write_invalidation)
- struct _starpu_data_request ∗ _starpu_create_request_to_fetch_data (starpu_data_handle_t handle, struct _starpu_data_replicate ∗dst_replicate, enum starpu_data_access_mode mode, enum _starpu_is_prefetch is_prefetch, unsigned async, void(∗callback_func)(void ∗), void ∗callback_arg, int prio, const char ∗origin)
- void **_starpu_redux_init_data_replicate** (starpu_data_handle_t handle, struct _starpu_data_replicate ∗replicate, int workerid)
- void **_starpu_data_start_reduction_mode** (starpu_data_handle_t handle)
- void **_starpu_data_end_reduction_mode** (starpu_data_handle_t handle)
- void **_starpu_data_end_reduction_mode_terminate** (starpu_data_handle_t handle)
- void **_starpu_data_set_unregister_hook** (starpu_data_handle_t handle, _starpu_data_handle_↩ unregister_hook func)
- struct _starpu_data_replicate ∗ **get_replicate** (starpu_data_handle_t handle, enum starpu_data_access_↩ mode mode, int workerid, unsigned node)

## Variables

- int **_starpu_has_not_important_data**

## 6.5.1 Data Structure Documentation

### 6.5.1.1 struct _starpu_data_replicate

this should contain the information relative to a given data replicate

**Data Fields**

| | | |
|---:|---|---|
| starpu_data_handle_t | handle | |
| void ∗ | data_interface | describe the actual data layout, as manipulated by data interfaces in ∗_interface.c |
| int | refcnt | How many requests or tasks are currently working with this replicate |
| char | memory_node | |
| enum _starpu_cache_state | state: 2 | describes the state of the local data in term of coherency |
| unsigned | relaxed_coherency:2 | A buffer that is used for SCRATCH or reduction cannnot be used with filters. |
| unsigned | initialized:1 | We may need to initialize the replicate with some value before using it. |
| unsigned | allocated:1 | is the data locally allocated ? |
| unsigned | automatically_allocated:1 | was it automatically allocated ? (else it's the application-provided buffer, don't ever try to free it!) perhaps the allocation was perform higher in the hiearchy for now this is just translated into !automatically_allocated |
| uint32_t | requested | To help the scheduling policies to make some decision, we may keep a track of the tasks that are likely to request this data on the current node. It is the responsability of the scheduling _policy_ to set that flag when it assigns a task to a queue, policies which do not use this hint can simply ignore it. |
| struct _starpu_data_request ∗ | request[STARPU_MAXNODES] | |
| struct _starpu_mem_chunk ∗ | mc | Pointer to memchunk for LRU strategy |

### 6.5.1.2 struct _starpu_jobid_list

**Data Fields**

| | | |
|---:|---|---|
| unsigned long | id | |
| struct _starpu_jobid_list ∗ | next | |

### 6.5.1.3 struct _starpu_task_wrapper_list

This structure describes a simply-linked list of task

**Data Fields**

| | | |
|---:|---|---|
| struct starpu_task ∗ | task | |
| struct _starpu_task_wrapper_list ∗ | next | |

**6.5.1.4   struct _starpu_task_wrapper_dlist**

This structure describes a doubly-linked list of task

**Data Fields**

| struct starpu_task ∗ | task | |
| --- | --- | --- |
| struct _starpu_task_wrapper_dlist ∗ | next | |
| struct _starpu_task_wrapper_dlist ∗ | prev | |

**6.5.1.5   struct _starpu_data_state**

This is initialized in both _starpu_register_new_data and _starpu_data_partition

**Data Fields**

| int | magic | |
| --- | --- | --- |
| struct _starpu_data_requester_prio_list | req_list | |
| unsigned | refcnt | the number of requests currently in the scheduling engine (not in the req_list anymore), i.e. the number of holders of the current_mode rwlock |
| unsigned | unlocking_reqs | whether we are already unlocking data requests |
| enum starpu_data_access_mode | current_mode | Current access mode. Is always either STARPU_R, STARPU_W, STARPU_SCRATCH or STARPU_REDUX, but never a combination such as STARPU_RW. |
| struct _starpu_spinlock | header_lock | protect meta data |
| unsigned | busy_count | Condition to make application wait for all transfers before freeing handle busy_count is the number of handle->refcnt, handle->per_node[∗]->refcnt, number of starpu_data_requesters, and number of tasks that have released it but are still registered on the implicit data dependency lists. Core code which releases busy_count has to call _starpu_data_check_not_busy to let starpu_data_unregister proceed |
| unsigned | busy_waiting | Is starpu_data_unregister waiting for busy_count? |
| starpu_pthread_mutex_t | busy_mutex | |
| starpu_pthread_cond_t | busy_cond | |
| struct _starpu_data_state ∗ | root_handle | In case we user filters, the handle may describe a sub-data |
| struct _starpu_data_state ∗ | father_handle | root of the tree |

**Data Fields**

| | | |
|---:|---|---|
| starpu_data_handle_t * | active_children | father of the node, NULL if the current node is the root |
| starpu_data_handle_t ** | active_readonly_children | The currently active set of read-write children |
| unsigned | nactive_readonly_children | The currently active set of read-only children |
| unsigned | nsiblings | Size of active_readonly_children array Our siblings in the father partitioning |
| starpu_data_handle_t * | siblings | How many siblings |
| unsigned | sibling_index | |
| unsigned | depth | indicate which child this node is from the father's perpsective (if any) |
| starpu_data_handle_t | children | what's the depth of the tree ? Synchronous partitioning |
| unsigned | nchildren | |
| unsigned | nplans | How many partition plans this handle has |
| struct starpu_codelet * | switch_cl | Switch codelet for asynchronous partitioning |
| unsigned | switch_cl_nparts | size of dyn_nodes recorded in switch_cl |
| unsigned | partitioned | Whether a partition plan is currently submitted and the corresponding unpartition has not been yet<br>Or the number of partition plans currently submitted in readonly mode. |
| unsigned | readonly:1 | Whether a partition plan is currently submitted in readonly mode |
| unsigned | active:1 | Whether our father is currently partitioned into ourself |
| unsigned | active_ro:1 | |
| struct _starpu_data_replicate | per_node[STARPU_MAXNODES] | describe the state of the data in term of coherency |
| struct _starpu_data_replicate * | per_worker | |
| struct starpu_data_interface_ops * | ops | |
| uint32_t | footprint | Footprint which identifies data layout |
| int | home_node | where is the data home, i.e. which node it was registered from ? -1 if none yet |
| uint32_t | wt_mask | what is the default write-through mask for that data ? |
| unsigned | is_not_important | in some case, the application may explicitly tell StarPU that a piece of data is not likely to be used soon again |

**Data Fields**

| | | |
|---:|---|---|
| unsigned | sequential_consistency | Does StarPU have to enforce some implicit data-dependencies ? |
| unsigned | initialized | Is the data initialized, or a task is already submitted to initialize it |
| unsigned | ooc | Can the data be pushed to the disk? |
| starpu_pthread_mutex_t | sequential_consistency_mutex | This lock should protect any operation to enforce sequential_consistency |
| enum starpu_data_access_mode | last_submitted_mode | The last submitted task (or application data request) that declared it would modify the piece of data ? Any task accessing the data in a read-only mode should depend on that task implicitely if the sequential_consistency flag is enabled. |
| struct starpu_task ∗ | last_sync_task | |
| struct _starpu_task_wrapper_dlist | last_submitted_accessors | |
| unsigned | last_submitted_ghost_sync_id_is_valid | If FxT is enabled, we keep track of "ghost dependencies": that is to say the dependencies that are not needed anymore, but that should appear in the post-mortem DAG. For instance if we have the sequence f(Aw) g(Aw), and that g is submitted after the termination of f, we want to have f->g appear in the DAG even if StarPU does not need to enforce this dependency anymore. |
| unsigned long | last_submitted_ghost_sync_id | |
| struct _starpu_jobid_list ∗ | last_submitted_ghost_accessors_id | |
| struct _starpu_task_wrapper_list ∗ | post_sync_tasks | protected by sequential_consistency_mutex |
| unsigned | post_sync_tasks_cnt | |
| struct starpu_codelet ∗ | redux_cl | During reduction we need some specific methods: redux_func performs the reduction of an interface into another one (eg. "+="), and init_func initializes the data interface to a default value that is stable by reduction (eg. 0 for +=). |
| struct starpu_codelet ∗ | init_cl | |
| unsigned | reduction_refcnt | Are we currently performing a reduction on that handle ? If so the reduction_refcnt should be non null until there are pending tasks that are performing the reduction. |
| struct | | |

**Data Fields**

| | | |
|---|---|---|
| _starpu_data_requester_prio_list | reduction_req_list | List of requesters that are specific to the pending reduction. This list is used when the requests in the req_list list are frozen until the end of the reduction. |
| starpu_data_handle_t ∗ | reduction_tmp_handles | |
| struct _starpu_data_request ∗ | write_invalidation_req | Final request for write invalidation |
| unsigned | lazy_unregister | |
| unsigned | removed_from_context_hash | |
| void ∗ | mpi_data | Used for MPI |
| _starpu_memory_stats_t | memory_stats | |
| unsigned int | mf_node | |
| _starpu_data_handle_unregister_hook | unregister_hook | hook to be called when unregistering the data |
| struct starpu_arbiter ∗ | arbiter | |
| struct _starpu_data_requester_prio_list | arbitered_req_list | This is protected by the arbiter mutex |
| int | last_locality | Data maintained by schedulers themselves Last worker that took this data in locality mode, or -1 if nobody took it yet |
| int | partition_automatic_disabled | |
| unsigned | dimensions | Application-provided coordinates. The maximum dimension (5) is relatively arbitrary. |
| int | coordinates[5] | |
| void ∗ | user_data | A generic pointer to data in the user land (could be anything and this is not manage by StarPU) |

## 6.5.2 Function Documentation

### 6.5.2.1 _starpu_fetch_data_on_node()

```
int _starpu_fetch_data_on_node (
          starpu_data_handle_t handle,
          int node,
          struct _starpu_data_replicate * replicate,
          enum starpu_data_access_mode mode,
          unsigned detached,
          enum _starpu_is_prefetch is_prefetch,
          unsigned async,
          void(*)(void *) callback_func,
          void * callback_arg,
          int prio,
          const char * origin )
```

This does not take a reference on the handle, the caller has to do it, e.g. through _starpu_attempt_to_submit↩
_data_request_from_apps() detached means that the core is allowed to drop the request. The caller should thus
∗not∗ take a reference since it can not know whether the request will complete async means that _starpu_fetch_↩
data_on_node will wait for completion of the request

**6.5.2.2 _starpu_release_data_on_node()**

```
void _starpu_release_data_on_node (
            struct _starpu_data_state * state,
            uint32_t default_wt_mask,
            struct _starpu_data_replicate * replicate )
```
This releases a reference on the handle

**6.5.2.3 _starpu_push_task_output()**

```
void _starpu_push_task_output (
            struct _starpu_job * j )
```
Version with driver trace

**6.5.2.4 _starpu_create_request_to_fetch_data()**

```
struct _starpu_data_request* _starpu_create_request_to_fetch_data (
            starpu_data_handle_t handle,
            struct _starpu_data_replicate * dst_replicate,
            enum starpu_data_access_mode mode,
            enum _starpu_is_prefetch is_prefetch,
            unsigned async,
            void(*)(void *) callback_func,
            void * callback_arg,
            int prio,
            const char * origin )
```
is_prefetch is whether the DSM may drop the request (when there is not enough memory for instance async is whether the caller wants a reference on the last request, to be able to wait for it (which will release that reference).

## 6.6 combined_workers.h File Reference

```
#include <starpu.h>
#include <common/config.h>
```

## 6.7 config.h File Reference

**Macros**

- #define **CONFIG_FUT**
- #define **HAVE_AIO_H**
- #define **HAVE_AYUDAME_H**
- #define **HAVE_CLENQUEUEMARKERWITHWAITLIST**
- #define **HAVE_CLGETEXTENSIONFUNCTIONADDRESSFORPLATFORM**
- #define **HAVE_CLOCK_GETTIME**
- #define **HAVE_CL_CL_EXT_H**
- #define **HAVE_COPY_FILE_RANGE**
- #define **HAVE_CUDA_GL_INTEROP_H**
- #define **HAVE_CXX11**
- #define **HAVE_DECL_CUSPARSESETSTREAM**
- #define **HAVE_DECL_ENABLE_FUT_FLUSH**
- #define **HAVE_DECL_FUT_SETUP_FLUSH_CALLBACK**
- #define **HAVE_DECL_FUT_SET_FILENAME**
- #define **HAVE_DECL_HWLOC_CUDA_GET_DEVICE_OSDEV_BY_INDEX**
- #define **HAVE_DECL_NVMLDEVICEGETTOTALENERGYCONSUMPTION**
- #define **HAVE_DECL_SMPI_PROCESS_SET_USER_DATA**

- #define **HAVE_DLB_H**
- #define **HAVE_DLFCN_H**
- #define **HAVE_ENABLE_FUT_FLUSH**
- #define **HAVE_FUT_SETUP_FLUSH_CALLBACK**
- #define **HAVE_FUT_SET_FILENAME**
- #define **HAVE_FXT_CLOSE**
- #define **HAVE_GETRLIMIT**
- #define **HAVE_GLPK_H**
- #define **HAVE_HDF5_H**
- #define **HAVE_HWLOC_GLIBC_SCHED_H**
- #define **HAVE_HWLOC_TOPOLOGY_DUP**
- #define **HAVE_HWLOC_TOPOLOGY_SET_COMPONENTS**
- #define **HAVE_INTTYPES_H**
- #define **HAVE_LEVELDB_DB_H**
- #define **HAVE_LIBATLAS**
- #define **HAVE_LIBBLAS_OPENBLAS**
- #define **HAVE_LIBCBLAS**
- #define **HAVE_LIBCUSPARSE**
- #define **HAVE_LIBDLB**
- #define **HAVE_LIBGFORTRAN**
- #define **HAVE_LIBGL**
- #define **HAVE_LIBGLPK**
- #define **HAVE_LIBGLU**
- #define **HAVE_LIBGLUT**
- #define **HAVE_LIBGOTO**
- #define **HAVE_LIBGOTO2**
- #define **HAVE_LIBHDF5**
- #define **HAVE_LIBIFCORE**
- #define **HAVE_LIBLEVELDB**
- #define **HAVE_LIBNVIDIA_ML**
- #define **HAVE_LIBOPENBLAS**
- #define **HAVE_LIBRT**
- #define **HAVE_LIBSIMGRID**
- #define **HAVE_LIBWS2_32**
- #define **HAVE_MALLOC_H**
- #define **HAVE_MEMALIGN**
- #define **HAVE_MEMORY_H**
- #define **HAVE_MKDTEMP**
- #define **HAVE_MKOSTEMP**
- #define **HAVE_MPI_COMM_F2C**
- #define **HAVE_MSG_ENVIRONMENT_GET_ROUTING_ROOT**
- #define **HAVE_MSG_GET_AS_BY_NAME**
- #define **HAVE_MSG_HOST_GET_SPEED**
- #define **HAVE_MSG_MSG_H**
- #define **HAVE_MSG_PROCESS_ATTACH**
- #define **HAVE_MSG_PROCESS_SELF_NAME**
- #define **HAVE_MSG_PROCESS_USERDATA_INIT**
- #define **HAVE_MSG_ZONE_GET_BY_NAME**
- #define **HAVE_MSG_ZONE_GET_HOSTS**
- #define **HAVE_NVMLDEVICEGETTOTALENERGYCONSUMPTION**
- #define **HAVE_PIOM_LTASK_SET_BOUND_THREAD_OS_INDEXES**
- #define **HAVE_POSIX_MEMALIGN**
- #define **HAVE_POTI_INIT_CUSTOM**
- #define **HAVE_POTI_USER_NEWEVENT**
- #define **HAVE_PREAD**

- #define **HAVE_PTHREAD_SETAFFINITY_NP**
- #define **HAVE_PTHREAD_SPIN_LOCK**
- #define **HAVE_PWRITE**
- #define **HAVE_SCANDIR**
- #define **HAVE_SG_ACTOR_ATTACH**
- #define **HAVE_SG_ACTOR_DATA**
- #define **HAVE_SG_ACTOR_EXECUTE**
- #define **HAVE_SG_ACTOR_INIT**
- #define **HAVE_SG_ACTOR_ON_EXIT**
- #define **HAVE_SG_ACTOR_REF**
- #define **HAVE_SG_ACTOR_SELF**
- #define **HAVE_SG_ACTOR_SELF_EXECUTE**
- #define **HAVE_SG_ACTOR_SLEEP_FOR**
- #define **HAVE_SG_CFG_SET_INT**
- #define **HAVE_SG_CONFIG_CONTINUE_AFTER_HELP**
- #define **HAVE_SG_HOST_GET_PROPERTIES**
- #define **HAVE_SG_HOST_LIST**
- #define **HAVE_SG_HOST_ROUTE**
- #define **HAVE_SG_HOST_SELF**
- #define **HAVE_SG_HOST_SENDTO**
- #define **HAVE_SG_HOST_SEND_TO**
- #define **HAVE_SG_HOST_SPEED**
- #define **HAVE_SG_LINK_BANDWIDTH_SET**
- #define **HAVE_SG_LINK_NAME**
- #define **HAVE_SG_ZONE_GET_BY_NAME**
- #define **HAVE_SG_ZONE_GET_HOSTS**
- #define **HAVE_SIMCALL_PROCESS_CREATE**
- #define **HAVE_SIMGRID_ACTOR_H**
- #define **HAVE_SIMGRID_BARRIER_H**
- #define **HAVE_SIMGRID_COND_H**
- #define **HAVE_SIMGRID_ENGINE_H**
- #define **HAVE_SIMGRID_GET_CLOCK**
- #define **HAVE_SIMGRID_HOST_H**
- #define **HAVE_SIMGRID_INIT**
- #define **HAVE_SIMGRID_MSG_H**
- #define **HAVE_SIMGRID_MUTEX_H**
- #define **HAVE_SIMGRID_SEMAPHORE_H**
- #define **HAVE_SIMGRID_SIMDAG_H**
- #define **HAVE_SIMGRID_VERSION_H**
- #define **HAVE_SIMGRID_ZONE_H**
- #define **HAVE_SMPI_PROCESS_SET_USER_DATA**
- #define **HAVE_SMPI_THREAD_CREATE**
- #define **HAVE_SMX_ACTOR_T**
- #define **HAVE_STDINT_H**
- #define **HAVE_STDLIB_H**
- #define **HAVE_STRINGS_H**
- #define **HAVE_STRING_H**
- #define **HAVE_SYSCONF**
- #define **HAVE_SYS_STAT_H**
- #define **HAVE_SYS_TYPES_H**
- #define **HAVE_UNISTD_H**
- #define **HAVE_VALGRIND_HELGRIND_H**
- #define **HAVE_VALGRIND_MEMCHECK_H**
- #define **HAVE_VALGRIND_VALGRIND_H**
- #define **HAVE_XBT_BARRIER_INIT**

- #define **HAVE_XBT_BASE_H**
- #define **HAVE_XBT_CONFIG_H**
- #define **HAVE_XBT_MUTEX_TRY_ACQUIRE**
- #define **HAVE_XBT_SYNCHRO_H**
- #define **LT_OBJDIR**
- #define **PACKAGE**
- #define **PACKAGE_BUGREPORT**
- #define **PACKAGE_NAME**
- #define **PACKAGE_STRING**
- #define **PACKAGE_TARNAME**
- #define **PACKAGE_URL**
- #define **PACKAGE_VERSION**
- #define **SIZEOF_VOID_P**
- #define **STARPURM_DLB_VERBOSE**
- #define **STARPURM_HAVE_DLB**
- #define **STARPURM_HAVE_DLB_CALLBACK_ARG**
- #define **STARPURM_STARPU_HAVE_WORKER_CALLBACKS**
- #define **STARPURM_VERBOSE**
- #define **STARPU_ARMPL**
- #define **STARPU_ATLAS**
- #define **STARPU_BUILD_DIR**
- #define **STARPU_BUILT_IN_MIN_DGELS**
- #define **STARPU_CLUSTER**
- #define **STARPU_DEBUG**
- #define **STARPU_DEVEL**
- #define **STARPU_DISABLE_ASYNCHRONOUS_COPY**
- #define **STARPU_DISABLE_ASYNCHRONOUS_CUDA_COPY**
- #define **STARPU_DISABLE_ASYNCHRONOUS_MIC_COPY**
- #define **STARPU_DISABLE_ASYNCHRONOUS_MPI_MS_COPY**
- #define **STARPU_DISABLE_ASYNCHRONOUS_OPENCL_COPY**
- #define **STARPU_EXTRA_VERBOSE**
- #define **STARPU_FXT_LOCK_TRACES**
- #define **STARPU_GDB_PATH**
- #define **STARPU_GOTO**
- #define **STARPU_HAVE_ATOMIC_COMPARE_EXCHANGE_N**
- #define **STARPU_HAVE_ATOMIC_EXCHANGE_N**
- #define **STARPU_HAVE_ATOMIC_FETCH_ADD**
- #define **STARPU_HAVE_ATOMIC_FETCH_OR**
- #define **STARPU_HAVE_ATOMIC_TEST_AND_SET**
- #define **STARPU_HAVE_BUSID**
- #define **STARPU_HAVE_CBLAS_H**
- #define **STARPU_HAVE_CUDA_MEMCPY_PEER**
- #define **STARPU_HAVE_CUFFTDOUBLECOMPLEX**
- #define **STARPU_HAVE_CURAND**
- #define **STARPU_HAVE_CXX11**
- #define **STARPU_HAVE_DARWIN**
- #define **STARPU_HAVE_DOMAINID**
- #define **STARPU_HAVE_F77_H**
- #define **STARPU_HAVE_FC**
- #define **STARPU_HAVE_FFTW**
- #define **STARPU_HAVE_FFTWF**
- #define **STARPU_HAVE_FFTWL**
- #define **STARPU_HAVE_GLPK_H**
- #define **STARPU_HAVE_HDF5**
- #define **STARPU_HAVE_HELGRIND_H**

- #define **STARPU_HAVE_HWLOC**
- #define **STARPU_HAVE_ICC**
- #define **STARPU_HAVE_LEVELDB**
- #define **STARPU_HAVE_LIBNUMA**
- #define **STARPU_HAVE_MAGMA**
- #define **STARPU_HAVE_MALLOC_H**
- #define **STARPU_HAVE_MEMALIGN**
- #define **STARPU_HAVE_MEMCHECK_H**
- #define **STARPU_HAVE_MSG_MSG_H**
- #define **STARPU_HAVE_NEARBYINTF**
- #define **STARPU_HAVE_POSIX_MEMALIGN**
- #define **STARPU_HAVE_POTI**
- #define **STARPU_HAVE_PTHREAD_BARRIER**
- #define **STARPU_HAVE_PTHREAD_SETNAME_NP**
- #define **STARPU_HAVE_PTHREAD_SPIN_LOCK**
- #define **STARPU_HAVE_RINTF**
- #define **STARPU_HAVE_SCHED_YIELD**
- #define **STARPU_HAVE_SETENV**
- #define **STARPU_HAVE_SIMGRID_ACTOR_H**
- #define **STARPU_HAVE_SIMGRID_BARRIER_H**
- #define **STARPU_HAVE_SIMGRID_COND_H**
- #define **STARPU_HAVE_SIMGRID_ENGINE_H**
- #define **STARPU_HAVE_SIMGRID_HOST_H**
- #define **STARPU_HAVE_SIMGRID_MSG_H**
- #define **STARPU_HAVE_SIMGRID_MUTEX_H**
- #define **STARPU_HAVE_SIMGRID_SEMAPHORE_H**
- #define **STARPU_HAVE_SIMGRID_SIMDAG_H**
- #define **STARPU_HAVE_SIMGRID_VERSION_H**
- #define **STARPU_HAVE_SIMGRID_ZONE_H**
- #define **STARPU_HAVE_SMX_ACTOR_T**
- #define **STARPU_HAVE_STATEMENT_EXPRESSIONS**
- #define **STARPU_HAVE_STRERROR_R**
- #define **STARPU_HAVE_STRUCT_TIMESPEC**
- #define **STARPU_HAVE_SYNC_BOOL_COMPARE_AND_SWAP**
- #define **STARPU_HAVE_SYNC_FETCH_AND_ADD**
- #define **STARPU_HAVE_SYNC_FETCH_AND_OR**
- #define **STARPU_HAVE_SYNC_LOCK_TEST_AND_SET**
- #define **STARPU_HAVE_SYNC_SYNCHRONIZE**
- #define **STARPU_HAVE_SYNC_VAL_COMPARE_AND_SWAP**
- #define **STARPU_HAVE_UNISTD_H**
- #define **STARPU_HAVE_UNSETENV**
- #define **STARPU_HAVE_VALGRIND_H**
- #define **STARPU_HAVE_WINDOWS**
- #define **STARPU_HAVE_X11**
- #define **STARPU_HAVE_XBT_BASE_H**
- #define **STARPU_HAVE_XBT_CONFIG_H**
- #define **STARPU_HAVE_XBT_SYNCHRO_H**
- #define **STARPU_HISTORYMAXERROR**
- #define **STARPU_LINUX_SYS**
- #define **STARPU_LONG_CHECK**
- #define **STARPU_MAJOR_VERSION**
- #define **STARPU_MAXCPUS**
- #define **STARPU_MAXCUDADEVS**
- #define **STARPU_MAXIMPLEMENTATIONS**
- #define **STARPU_MAXMICCORES**

- #define **STARPU_MAXMICDEVS**
- #define **STARPU_MAXMPIDEVS**
- #define **STARPU_MAXMPKERNELS**
- #define **STARPU_MAXNODES**
- #define **STARPU_MAXNUMANODES**
- #define **STARPU_MAXOPENCLDEVS**
- #define **STARPU_MEMORY_STATS**
- #define **STARPU_MIC_USE_RMA**
- #define **STARPU_MINOR_VERSION**
- #define **STARPU_MKL**
- #define **STARPU_MLR_MODEL**
- #define **STARPU_MODEL_DEBUG**
- #define **STARPU_MPI_EXTRA_VERBOSE**
- #define **STARPU_MPI_MASTER_SLAVE_MULTIPLE_THREAD**
- #define **STARPU_MPI_PEDANTIC_ISEND**
- #define **STARPU_MPI_VERBOSE**
- #define **STARPU_NATIVE_WINTHREADS**
- #define **STARPU_NEW_CHECK**
- #define **STARPU_NMAXBUFS**
- #define **STARPU_NMAXWORKERS**
- #define **STARPU_NMAX_COMBINEDWORKERS**
- #define **STARPU_NMAX_SCHED_CTXS**
- #define **STARPU_NON_BLOCKING_DRIVERS**
- #define **STARPU_NO_ASSERT**
- #define **STARPU_OPENBLAS**
- #define **STARPU_OPENBSD_SYS**
- #define **STARPU_OPENCL_SIMULATOR**
- #define **STARPU_OPENGL_RENDER**
- #define **STARPU_OPENMP**
- #define **STARPU_PERF_DEBUG**
- #define **STARPU_PERF_MODEL_DIR**
- #define **STARPU_QUICK_CHECK**
- #define **STARPU_RELEASE_VERSION**
- #define **STARPU_SC_HYPERVISOR_DEBUG**
- #define **STARPU_SIMGRID**
- #define **STARPU_SIMGRID_HAVE_SIMGRID_INIT**
- #define **STARPU_SIMGRID_HAVE_XBT_BARRIER_INIT**
- #define **STARPU_SIMGRID_MC**
- #define **STARPU_SPINLOCK_CHECK**
- #define **STARPU_SRC_DIR**
- #define **STARPU_STATIC_ONLY**
- #define **STARPU_SYSTEM_BLAS**
- #define **STARPU_USE_ALLOCATION_CACHE**
- #define **STARPU_USE_AYUDAME1**
- #define **STARPU_USE_AYUDAME2**
- #define **STARPU_USE_CPU**
- #define **STARPU_USE_CUDA**
- #define **STARPU_USE_DRAND48**
- #define **STARPU_USE_ERAND48_R**
- #define **STARPU_USE_FXT**
- #define **STARPU_USE_MIC**
- #define **STARPU_USE_MP**
- #define **STARPU_USE_MPI**
- #define **STARPU_USE_MPI_MASTER_SLAVE**
- #define **STARPU_USE_MPI_MPI**

- #define **STARPU_USE_MPI_NMAD**
- #define **STARPU_USE_OPENCL**
- #define **STARPU_USE_SC_HYPERVISOR**
- #define **STARPU_VALGRIND_FULL**
- #define **STARPU_VERBOSE**
- #define **STARPU_WORKER_CALLBACKS**
- #define **STDC_HEADERS**
- #define **VERSION**
- #define **X_DISPLAY_MISSING**
- #define **restrict**

## 6.8 copy_driver.h File Reference

```
#include <common/config.h>
#include <common/list.h>
#include <cuda.h>
#include <cuda_runtime.h>
#include <starpu_opencl.h>
```

**Data Structures**

- struct _starpu_mic_async_event
- struct _starpu_disk_backend_event
- struct _starpu_disk_async_event
- union _starpu_async_channel_event
- struct _starpu_async_channel
- struct _starpu_async_channel_event.__unnamed__

**Enumerations**

- enum _starpu_is_prefetch { STARPU_FETCH, STARPU_PREFETCH, STARPU_IDLEFETCH, **STARPU_↩ NFETCH** }

**Functions**

- void **_starpu_wake_all_blocked_workers_on_node** (unsigned nodeid)
- int **_starpu_driver_copy_data_1_to_1** (starpu_data_handle_t handle, struct _starpu_data_replicate ∗src↩ _replicate, struct _starpu_data_replicate ∗dst_replicate, unsigned donotread, struct _starpu_data_request ∗req, unsigned may_alloc, enum _starpu_is_prefetch prefetch)
- unsigned **_starpu_driver_test_request_completion** (struct _starpu_async_channel ∗async_channel)
- void **_starpu_driver_wait_request_completion** (struct _starpu_async_channel ∗async_channel)

### 6.8.1 Data Structure Documentation

#### 6.8.1.1 struct _starpu_mic_async_event

MIC needs memory_node to know which MIC is concerned. mark is used to wait asynchronous request. signal is used to test asynchronous request.

**Data Fields**

| unsigned | memory_node | |
|---:|---|---|
| int | mark | |
| uint64_t ∗ | signal | |

**6.8.1.2 struct _starpu_disk_backend_event**

**Data Fields**

| void ∗ | backend_event | |
|---|---|---|

**6.8.1.3 struct _starpu_disk_async_event**

**Data Fields**

| unsigned | memory_node | |
|---|---|---|
| struct _starpu_disk_backend_event_list ∗ | requests | |
| void ∗ | ptr | |
| unsigned | node | |
| size_t | size | |
| starpu_data_handle_t | handle | |

**6.8.1.4 union _starpu_async_channel_event**

this is a structure that can be queried to see whether an asynchronous transfer has terminated or not

**Data Fields**

| struct _starpu_async_channel_event | __unnamed__ | |
|---|---|---|
| cudaEvent_t | cuda_event | |
| cl_event | opencl_event | |
| struct _starpu_mic_async_event | mic_event | |
| struct _starpu_disk_async_event | disk_event | |

**6.8.1.5 struct _starpu_async_channel**

**Data Fields**

| union _starpu_async_channel_event | event | |
|---|---|---|
| struct _starpu_node_ops ∗ | node_ops | |
| struct _starpu_mp_node ∗ | polling_node_sender | Which node to polling when needing ACK msg |
| struct _starpu_mp_node ∗ | polling_node_receiver | |
| volatile int | starpu_mp_common_finished_sender | Used to know if the acknowlegdment msg is arrived from sinks |
| volatile int | starpu_mp_common_finished_receiver | |

**6.8.1.6 struct _starpu_async_channel_event.__unnamed__**

**Data Fields**

| unsigned | finished | |
|---|---|---|
| starpu_pthread_queue_t ∗ | queue | |

### 6.8.2 Enumeration Type Documentation

#### 6.8.2.1 _starpu_is_prefetch

enum _starpu_is_prefetch

**Enumerator**

| STARPU_FETCH | A task really needs it now! |
|---|---|
| STARPU_PREFETCH | It is a good idea to have it asap |
| STARPU_IDLEFETCH | Get this here when you have time to |

## 6.9 data_concurrency.h File Reference

#include <core/jobs.h>

**Functions**

- void **_starpu_job_set_ordered_buffers** (struct _starpu_job ∗j)
- unsigned **_starpu_submit_job_enforce_data_deps** (struct _starpu_job ∗j)
- void **_starpu_submit_job_enforce_arbitered_deps** (struct _starpu_job ∗j, unsigned buf, unsigned nbuffers)
- void **_starpu_enforce_data_deps_notify_job_ready_soon** (struct _starpu_job ∗j, _starpu_notify_job_↩ start_data ∗data)
- int **_starpu_notify_data_dependencies** (starpu_data_handle_t handle)
- void **_starpu_notify_arbitered_dependencies** (starpu_data_handle_t handle)
- unsigned **_starpu_attempt_to_submit_data_request_from_apps** (starpu_data_handle_t handle, enum starpu_data_access_mode mode, void(∗callback)(void ∗), void ∗argcb)
- unsigned **_starpu_attempt_to_submit_arbitered_data_request** (unsigned request_from_codelet, starpu_data_handle_t handle, enum starpu_data_access_mode mode, void(∗callback)(void ∗), void ∗argcb, struct _starpu_job ∗j, unsigned buffer_index)

## 6.10 data_interface.h File Reference

#include <starpu.h>
#include <common/config.h>
#include <common/uthash.h>
#include <util/openmp_runtime_support.h>

**Data Structures**

- union _starpu_interface

**Macros**

- #define **_starpu_data_check_not_busy**(handle)
- #define **_starpu_data_is_multiformat_handle**(handle)

## Functions

- void **_starpu_data_free_interfaces** (starpu_data_handle_t handle) STARPU_ATTRIBUTE_INTERNAL
- int **_starpu_data_handle_init** (starpu_data_handle_t handle, [struct](#) starpu_data_interface_ops ∗interface↩
  _ops, unsigned int mf_node)
- void **_starpu_data_initialize_per_worker** (starpu_data_handle_t handle)
- void **_starpu_data_interface_init** (void) STARPU_ATTRIBUTE_INTERNAL
- int **__starpu_data_check_not_busy** (starpu_data_handle_t handle) STARPU_ATTRIBUTE_INTERNAL
  STARPU_ATTRIBUTE_WARN_UNUSED_RESULT
- void **_starpu_data_interface_shutdown** (void) STARPU_ATTRIBUTE_INTERNAL
- void **_starpu_omp_unregister_region_handles** ([struct](#) starpu_omp_region ∗region)
- void **_starpu_omp_unregister_task_handles** ([struct](#) starpu_omp_task ∗task)
- [struct](#) starpu_data_interface_ops ∗ **_starpu_data_interface_get_ops** (unsigned interface_id)
- void **_starpu_data_register_ram_pointer** (starpu_data_handle_t handle, void ∗ptr) STARPU_ATTRIBU↩
  TE_INTERNAL
- void **_starpu_data_unregister_ram_pointer** (starpu_data_handle_t handle, unsigned node) STARPU_A↩
  TTRIBUTE_INTERNAL

## Variables

- [struct](#) starpu_data_interface_ops [starpu_interface_matrix_ops](#)
- [struct](#) starpu_data_interface_ops **starpu_interface_block_ops**
- [struct](#) starpu_data_interface_ops **starpu_interface_vector_ops**
- [struct](#) starpu_data_interface_ops **starpu_interface_csr_ops**
- [struct](#) starpu_data_interface_ops **starpu_interface_bcsr_ops**
- [struct](#) starpu_data_interface_ops **starpu_interface_variable_ops**
- [struct](#) starpu_data_interface_ops **starpu_interface_void_ops**
- [struct](#) starpu_data_interface_ops **starpu_interface_multiformat_ops**
- [struct](#) starpu_arbiter ∗ **_starpu_global_arbiter**

### 6.10.1 Data Structure Documentation

#### 6.10.1.1 union _starpu_interface

Generic type representing an interface, for now it's only used before execution on message-passing devices but it
can be useful in other cases.

**Data Fields**

| | | |
|---|---|---|
| [struct](#) starpu_matrix_interface | matrix | |
| [struct](#) starpu_block_interface | block | |
| [struct](#) starpu_vector_interface | vector | |
| [struct](#) starpu_csr_interface | csr | |
| [struct](#) starpu_coo_interface | coo | |
| [struct](#) starpu_bcsr_interface | bcsr | |
| [struct](#) starpu_variable_interface | variable | |
| [struct](#) starpu_multiformat_interface | multiformat | |

### 6.10.2 Variable Documentation

#### 6.10.2.1 starpu_interface_matrix_ops

[struct](#) starpu_data_interface_ops starpu_interface_matrix_ops

Some data interfaces or filters use this interface internally

## 6.11 data_request.h File Reference

```
#include <datawizard/coherency.h>
#include <semaphore.h>
#include <datawizard/copy_driver.h>
#include <common/list.h>
#include <common/prio_list.h>
#include <common/starpu_spinlock.h>
```

**Data Structures**

- struct _starpu_callback_list

**Macros**

- #define **MAX_PENDING_REQUESTS_PER_NODE**
- #define **MAX_PENDING_PREFETCH_REQUESTS_PER_NODE**
- #define **MAX_PENDING_IDLE_REQUESTS_PER_NODE**
- #define MAX_PUSH_TIME

### 6.11.1 Macro Definition Documentation

#### 6.11.1.1 MAX_PUSH_TIME

```
#define MAX_PUSH_TIME
```
Maximum time in us that we can afford pushing requests before going back to the driver loop, e.g. for checking GPU task termination

## 6.12 datastats.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <stdint.h>
#include <stdlib.h>
```

**Macros**

- #define **_starpu_msi_cache_hit**(node)
- #define **_starpu_msi_cache_miss**(node)
- #define **_starpu_allocation_cache_hit**(node)
- #define **_starpu_data_allocation_inc_stats**(node)

**Functions**

- void **_starpu_datastats_init** ()
- static int **starpu_enable_stats** (void)
- void **__starpu_msi_cache_hit** (unsigned node)
- void **__starpu_msi_cache_miss** (unsigned node)
- void **_starpu_display_msi_stats** (FILE ∗stream)
- void **__starpu_allocation_cache_hit** (unsigned node STARPU_ATTRIBUTE_UNUSED)

- void **__starpu_data_allocation_inc_stats** (unsigned node STARPU_ATTRIBUTE_UNUSED)
- void **_starpu_display_alloc_cache_stats** (FILE ∗stream)

**Variables**

- int **_starpu_enable_stats**

## 6.13 datawizard.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/utils.h>
#include <datawizard/coherency.h>
#include <datawizard/filters.h>
#include <datawizard/copy_driver.h>
#include <datawizard/footprint.h>
#include <datawizard/data_request.h>
#include <datawizard/interfaces/data_interface.h>
#include <core/dependencies/implicit_data_deps.h>
```

**Functions**

- int **___starpu_datawizard_progress** (unsigned memory_node, unsigned may_alloc, unsigned push_↩
  requests)
- int **__starpu_datawizard_progress** (unsigned may_alloc, unsigned push_requests)
- void **_starpu_datawizard_progress** (unsigned may_alloc)

## 6.14 debug.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <common/config.h>
#include <core/workers.h>
```

**Macros**

- #define **STARPU_AYU_EVENT**
- #define **STARPU_AYU_PREINIT**()
- #define **STARPU_AYU_INIT**()
- #define **STARPU_AYU_FINISH**()
- #define **STARPU_AYU_ADDDEPENDENCY**(previous, handle, next_job)
- #define **STARPU_AYU_REMOVETASK**(job_id)
- #define **STARPU_AYU_ADDTASK**(job_id, task)
- #define **STARPU_AYU_PRERUNTASK**(job_id, workerid)
- #define **STARPU_AYU_RUNTASK**(job_id)
- #define **STARPU_AYU_POSTRUNTASK**(job_id)
- #define **STARPU_AYU_ADDTOTASKQUEUE**(job_id, worker_id)
- #define **STARPU_AYU_BARRIER**()

**Functions**

- void [_starpu_open_debug_logfile](void)
- void [_starpu_close_debug_logfile](void)
- void [_starpu_print_to_logfile](const char ∗format,...) STARPU_ATTRIBUTE_FORMAT(printf
- void **_starpu_watchdog_init** (void)
- void **_starpu_watchdog_shutdown** (void)

**Variables**

- void int [_starpu_use_fxt](void)

### 6.14.1 Function Documentation

#### 6.14.1.1 _starpu_open_debug_logfile()

```
void _starpu_open_debug_logfile (
            void  )
```
Create a file that will contain StarPU's log

#### 6.14.1.2 _starpu_close_debug_logfile()

```
void _starpu_close_debug_logfile (
            void  )
```
Close StarPU's log file

#### 6.14.1.3 _starpu_print_to_logfile()

```
void _starpu_print_to_logfile (
            const char ∗ format,
            ... )
```
Write into StarPU's log file

### 6.14.2 Variable Documentation

#### 6.14.2.1 _starpu_use_fxt

```
void int _starpu_use_fxt
```
Tell gdb whether FXT is compiled in or not

## 6.15 detect_combined_workers.h File Reference

```
#include <starpu.h>
```

**Functions**

- void [_starpu_sched_find_worker_combinations](int ∗workerids, int nworkers)

**Variables**

- int **_starpu_initialized_combined_workers**

### 6.15.1 Function Documentation

#### 6.15.1.1 _starpu_sched_find_worker_combinations()

```
void _starpu_sched_find_worker_combinations (
            int * workerids,
            int nworkers )
```
Initialize combined workers

## 6.16 disk.h File Reference

```
#include <datawizard/copy_driver.h>
#include <datawizard/malloc.h>
```

**Macros**

- #define **STARPU_DISK_ALL**
- #define **STARPU_DISK_NO_RECLAIM**

**Functions**

- void ∗ _starpu_disk_alloc (unsigned node, size_t size) STARPU_ATTRIBUTE_MALLOC
- void **_starpu_disk_free** (unsigned node, void ∗obj, size_t size)
- int _starpu_disk_read (unsigned src_node, unsigned dst_node, void ∗obj, void ∗buf, off_t offset, size_t size, struct _starpu_async_channel ∗async_channel)
- int _starpu_disk_write (unsigned src_node, unsigned dst_node, void ∗obj, void ∗buf, off_t offset, size_t size, struct _starpu_async_channel ∗async_channel)
- int **_starpu_disk_full_read** (unsigned src_node, unsigned dst_node, void ∗obj, void ∗∗ptr, size_t ∗size, struct _starpu_async_channel ∗async_channel)
- int **_starpu_disk_full_write** (unsigned src_node, unsigned dst_node, void ∗obj, void ∗ptr, size_t size, struct _starpu_async_channel ∗async_channel)
- int **_starpu_disk_copy** (unsigned node_src, void ∗obj_src, off_t offset_src, unsigned node_dst, void ∗obj↩_dst, off_t offset_dst, size_t size, struct _starpu_async_channel ∗async_channel)
- void starpu_disk_wait_request (struct _starpu_async_channel ∗async_channel)
- int starpu_disk_test_request (struct _starpu_async_channel ∗async_channel)
- void **starpu_disk_free_request** (struct _starpu_async_channel ∗async_channel)
- int _starpu_disk_can_copy (unsigned node1, unsigned node2)
- void _starpu_set_disk_flag (unsigned node, int flag)
- int **_starpu_get_disk_flag** (unsigned node)
- void _starpu_disk_unregister (void)
- void **_starpu_swap_init** (void)

### 6.16.1 Function Documentation

#### 6.16.1.1 _starpu_disk_alloc()

```
void* _starpu_disk_alloc (
            unsigned node,
            size_t size )
```
interface to manipulate memory disk

**6.16.1.2 _starpu_disk_read()**

```
int _starpu_disk_read (
            unsigned src_node,
            unsigned dst_node,
            void * obj,
            void * buf,
            off_t offset,
            size_t size,
            struct _starpu_async_channel * async_channel )
```
src_node is a disk node, dst_node is for the moment the STARPU_MAIN_RAM

**6.16.1.3 _starpu_disk_write()**

```
int _starpu_disk_write (
            unsigned src_node,
            unsigned dst_node,
            void * obj,
            void * buf,
            off_t offset,
            size_t size,
            struct _starpu_async_channel * async_channel )
```
src_node is for the moment the STARU_MAIN_RAM, dst_node is a disk node

**6.16.1.4 starpu_disk_wait_request()**

```
void starpu_disk_wait_request (
            struct _starpu_async_channel * async_channel )
```
force the request to compute

**6.16.1.5 starpu_disk_test_request()**

```
int starpu_disk_test_request (
            struct _starpu_async_channel * async_channel )
```
return 1 if the request is finished, 0 if not finished

**6.16.1.6 _starpu_disk_can_copy()**

```
int _starpu_disk_can_copy (
            unsigned node1,
            unsigned node2 )
```
interface to compare memory disk

**6.16.1.7 _starpu_set_disk_flag()**

```
void _starpu_set_disk_flag (
            unsigned node,
            int flag )
```
change disk flag

**6.16.1.8 _starpu_disk_unregister()**

```
void _starpu_disk_unregister (
            void  )
```
unregister disk

## 6.17 disk_unistd_global.h File Reference

```
#include <fcntl.h>
```

**Data Structures**

- struct [starpu_unistd_global_obj](#)

**Macros**

- #define **O_BINARY**
- #define **STARPU_UNISTD_USE_COPY**

**Functions**

- void ∗ **starpu_unistd_global_alloc** ([struct starpu_unistd_global_obj](#) ∗obj, void ∗base, size_t size)
- void **starpu_unistd_global_free** (void ∗base, void ∗obj, size_t size)
- void ∗ **starpu_unistd_global_open** ([struct starpu_unistd_global_obj](#) ∗obj, void ∗base, void ∗pos, size_t size)
- void **starpu_unistd_global_close** (void ∗base, void ∗obj, size_t size)
- int **starpu_unistd_global_read** (void ∗base, void ∗obj, void ∗buf, off_t offset, size_t size)
- int **starpu_unistd_global_write** (void ∗base, void ∗obj, const void ∗buf, off_t offset, size_t size)
- void ∗ **starpu_unistd_global_plug** (void ∗parameter, starpu_ssize_t size)
- void **starpu_unistd_global_unplug** (void ∗base)
- int **get_unistd_global_bandwidth_between_disk_and_main_ram** (unsigned node, void ∗base)
- void ∗ **starpu_unistd_global_async_read** (void ∗base, void ∗obj, void ∗buf, off_t offset, size_t size)
- void ∗ **starpu_unistd_global_async_write** (void ∗base, void ∗obj, void ∗buf, off_t offset, size_t size)
- void ∗ **starpu_unistd_global_async_full_write** (void ∗base, void ∗obj, void ∗ptr, size_t size)
- void ∗ **starpu_unistd_global_async_full_read** (void ∗base, void ∗obj, void ∗∗ptr, size_t ∗size, unsigned dst_node)
- void **starpu_unistd_global_wait_request** (void ∗async_channel)
- int **starpu_unistd_global_test_request** (void ∗async_channel)
- void **starpu_unistd_global_free_request** (void ∗async_channel)
- int **starpu_unistd_global_full_read** (void ∗base, void ∗obj, void ∗∗ptr, size_t ∗size, unsigned dst_node)
- int **starpu_unistd_global_full_write** (void ∗base, void ∗obj, void ∗ptr, size_t size)

### 6.17.1 Data Structure Documentation

#### 6.17.1.1 struct starpu_unistd_global_obj

**Data Fields**

| | | |
|---:|---|---|
| int | descriptor | |
| char ∗ | path | |
| size_t | size | |
| int | flags | |
| starpu_pthread_mutex_t | mutex | |

## 6.18 driver_common.h File Reference

```
#include <starpu.h>
#include <starpu_util.h>
#include <core/jobs.h>
#include <common/utils.h>
```

**Functions**

- void **_starpu_driver_start_job** (struct _starpu_worker *args, struct _starpu_job *j, struct starpu_↩
perfmodel_arch *perf_arch, int rank, int profiling)
- void **_starpu_driver_end_job** (struct _starpu_worker *args, struct _starpu_job *j, struct starpu_perfmodel↩
_arch *perf_arch, int rank, int profiling)
- void **_starpu_driver_update_job_feedback** (struct _starpu_job *j, struct _starpu_worker *worker_args,
struct starpu_perfmodel_arch *perf_arch, int profiling)
- struct starpu_task * **_starpu_get_worker_task** (struct _starpu_worker *args, int workerid, unsigned
memnode)
- int **_starpu_get_multi_worker_task** (struct _starpu_worker *workers, struct starpu_task **tasks, int
nworker, unsigned memnode)

## 6.19   driver_cpu.h File Reference

```
#include <common/config.h>
#include <datawizard/node_ops.h>
```

**Functions**

- void * **_starpu_cpu_worker** (void *)
- int **_starpu_cpu_copy_interface** (starpu_data_handle_t handle, void *src_interface, unsigned src_node,
void *dst_interface, unsigned dst_node, struct _starpu_data_request *req)
- int **_starpu_cpu_copy_data** (uintptr_t src_ptr, size_t src_offset, unsigned src_node, uintptr_t dst_ptr, size↩
_t dst_offset, unsigned dst_node, size_t ssize, struct _starpu_async_channel *async_channel)
- int **_starpu_cpu_is_direct_access_supported** (unsigned node, unsigned handling_node)
- uintptr_t **_starpu_cpu_malloc_on_node** (unsigned dst_node, size_t size, int flags)
- void **_starpu_cpu_free_on_node** (unsigned dst_node, uintptr_t addr, size_t size, int flags)

**Variables**

- struct _starpu_driver_ops **_starpu_driver_cpu_ops**
- struct _starpu_node_ops **_starpu_driver_cpu_node_ops**

## 6.20   driver_cuda.h File Reference

```
#include <common/config.h>
#include <cuda.h>
#include <cuda_runtime_api.h>
#include <cublas.h>
#include <starpu.h>
#include <core/workers.h>
#include <datawizard/node_ops.h>
```

**Functions**

- void **_starpu_cuda_init** (void)
- unsigned **_starpu_get_cuda_device_count** (void)
- void **_starpu_cuda_discover_devices** (struct _starpu_machine_config *)
- void **_starpu_init_cuda** (void)
- void * **_starpu_cuda_worker** (void *)
- cudaStream_t **starpu_cuda_get_local_in_transfer_stream** (void)

- cudaStream_t **starpu_cuda_get_in_transfer_stream** (unsigned dst_node)
- cudaStream_t **starpu_cuda_get_local_out_transfer_stream** (void)
- cudaStream_t **starpu_cuda_get_out_transfer_stream** (unsigned src_node)
- cudaStream_t **starpu_cuda_get_peer_transfer_stream** (unsigned src_node, unsigned dst_node)
- unsigned **_starpu_cuda_test_request_completion** (struct _starpu_async_channel ∗async_channel)
- void **_starpu_cuda_wait_request_completion** (struct _starpu_async_channel ∗async_channel)
- int **_starpu_cuda_copy_interface_from_cpu_to_cuda** (starpu_data_handle_t handle, void ∗src_interface, unsigned src_node, void ∗dst_interface, unsigned dst_node, struct _starpu_data_request ∗req)
- int **_starpu_cuda_copy_interface_from_cuda_to_cuda** (starpu_data_handle_t handle, void ∗src_↩ interface, unsigned src_node, void ∗dst_interface, unsigned dst_node, struct _starpu_data_request ∗req)
- int **_starpu_cuda_copy_interface_from_cuda_to_cpu** (starpu_data_handle_t handle, void ∗src_interface, unsigned src_node, void ∗dst_interface, unsigned dst_node, struct _starpu_data_request ∗req)
- int **_starpu_cuda_copy_data_from_cuda_to_cuda** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, struct _starpu_async_channel ∗async_↩ channel)
- int **_starpu_cuda_copy_data_from_cuda_to_cpu** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, struct _starpu_async_channel ∗async_↩ channel)
- int **_starpu_cuda_copy_data_from_cpu_to_cuda** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, struct _starpu_async_channel ∗async_↩ channel)
- int **_starpu_cuda_copy2d_data_from_cuda_to_cuda** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks, size_t ld_src, size_t ld_dst, struct _starpu_async_channel ∗async_channel)
- int **_starpu_cuda_copy2d_data_from_cuda_to_cpu** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks, size_t ld_src, size_t ld_dst, struct _starpu_async_channel ∗async_channel)
- int **_starpu_cuda_copy2d_data_from_cpu_to_cuda** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks, size_t ld_src, size_t ld_dst, struct _starpu_async_channel ∗async_channel)
- int **_starpu_cuda_copy3d_data_from_cuda_to_cuda** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks_1, size_t ld1_src, size_t ld1_dst, size_t numblocks_2, size_t ld2_src, size_t ld2_dst, struct _starpu_async_channel ∗async_↩ channel)
- int **_starpu_cuda_copy3d_data_from_cuda_to_cpu** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks_1, size_t ld1_src, size_t ld1_dst, size_t numblocks_2, size_t ld2_src, size_t ld2_dst, struct _starpu_async_channel ∗async_↩ channel)
- int **_starpu_cuda_copy3d_data_from_cpu_to_cuda** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks_1, size_t ld1_src, size_t ld1_dst, size_t numblocks_2, size_t ld2_src, size_t ld2_dst, struct _starpu_async_channel ∗async_↩ channel)
- int **_starpu_cuda_is_direct_access_supported** (unsigned node, unsigned handling_node)
- uintptr_t **_starpu_cuda_malloc_on_node** (unsigned dst_node, size_t size, int flags)
- void **_starpu_cuda_free_on_node** (unsigned dst_node, uintptr_t addr, size_t size, int flags)

**Variables**

- struct _starpu_driver_ops **_starpu_driver_cuda_ops**
- struct _starpu_node_ops **_starpu_driver_cuda_node_ops**
- int **_starpu_cuda_bus_ids** [STARPU_MAXCUDADEVS+STARPU_MAXNUMANODES][STARPU_MAX↩ CUDADEVS+STARPU_MAXNUMANODES]

## 6.21 driver_disk.h File Reference

```
#include <datawizard/node_ops.h>
```

**Functions**

- int **_starpu_disk_copy_src_to_disk** (void ∗src, unsigned src_node, void ∗dst, size_t dst_offset, unsigned dst_node, size_t size, void ∗async_channel)
- int **_starpu_disk_copy_disk_to_src** (void ∗src, size_t src_offset, unsigned src_node, void ∗dst, unsigned dst_node, size_t size, void ∗async_channel)
- int **_starpu_disk_copy_disk_to_disk** (void ∗src, size_t src_offset, unsigned src_node, void ∗dst, size_t dst_offset, unsigned dst_node, size_t size, void ∗async_channel)
- unsigned **_starpu_disk_test_request_completion** ([struct _starpu_async_channel](#) ∗async_channel)
- void **_starpu_disk_wait_request_completion** ([struct _starpu_async_channel](#) ∗async_channel)
- int **_starpu_disk_copy_interface_from_disk_to_cpu** (starpu_data_handle_t handle, void ∗src_interface, unsigned src_node, void ∗dst_interface, unsigned dst_node, [struct](#) _starpu_data_request ∗req)
- int **_starpu_disk_copy_interface_from_disk_to_disk** (starpu_data_handle_t handle, void ∗src_interface, unsigned src_node, void ∗dst_interface, unsigned dst_node, [struct](#) _starpu_data_request ∗req)
- int **_starpu_disk_copy_interface_from_cpu_to_disk** (starpu_data_handle_t handle, void ∗src_interface, unsigned src_node, void ∗dst_interface, unsigned dst_node, [struct](#) _starpu_data_request ∗req)
- int **_starpu_disk_copy_data_from_disk_to_cpu** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, [struct _starpu_async_channel](#) ∗async_↩ channel)
- int **_starpu_disk_copy_data_from_disk_to_disk** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, [struct _starpu_async_channel](#) ∗async_↩ channel)
- int **_starpu_disk_copy_data_from_cpu_to_disk** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, [struct _starpu_async_channel](#) ∗async_↩ channel)
- int **_starpu_disk_is_direct_access_supported** (unsigned node, unsigned handling_node)
- uintptr_t **_starpu_disk_malloc_on_node** (unsigned dst_node, size_t size, int flags)
- void **_starpu_disk_free_on_node** (unsigned dst_node, uintptr_t addr, size_t size, int flags)

**Variables**

- [struct _starpu_node_ops](#) **_starpu_driver_disk_node_ops**

## 6.22 driver_mic_common.h File Reference

```
#include <common/config.h>
#include <source/COIProcess_source.h>
```

**Data Structures**

- struct [_starpu_mic_free_command](#)

**Macros**

- #define **STARPU_TO_MIC_ID**(id)
- #define **STARPU_MIC_PORTS_BEGIN**
- #define **STARPU_MIC_SOURCE_PORT_NUMBER**
- #define **STARPU_MIC_SINK_PORT_NUMBER**(id)
- #define **STARPU_MIC_SOURCE_DT_PORT_NUMBER**
- #define **STARPU_MIC_SINK_DT_PORT_NUMBER**(id)
- #define **STARPU_MIC_SINK_SINK_DT_PORT_NUMBER**(me, peer_id)
- #define **STARPU_MIC_PAGE_SIZE**
- #define **STARPU_MIC_GET_PAGE_SIZE_MULTIPLE**(size)
- #define **STARPU_MIC_COMMON_REPORT_SCIF_ERROR**(status)

**Functions**

- void **_starpu_mic_common_report_scif_error** (const char ∗func, const char ∗file, int line, const int status)
- int **_starpu_mic_common_recv_is_ready** (const struct _starpu_mp_node ∗mp_node)
- void **_starpu_mic_common_send** (const struct _starpu_mp_node ∗node, void ∗msg, int len)
- void **_starpu_mic_common_recv** (const struct _starpu_mp_node ∗node, void ∗msg, int len)
- void **_starpu_mic_common_dt_send** (const struct _starpu_mp_node ∗node, void ∗msg, int len, void ∗event)
- void **_starpu_mic_common_dt_recv** (const struct _starpu_mp_node ∗node, void ∗msg, int len, void ∗event)
- void **_starpu_mic_common_connect** (scif_epd_t ∗endpoint, uint16_t remote_node, COIPROCESS process, uint16_t local_port_number, uint16_t remote_port_number)
- void **_starpu_mic_common_accept** (scif_epd_t ∗endpoint, uint16_t port_number)

### 6.22.1 Data Structure Documentation

#### 6.22.1.1 struct _starpu_mic_free_command

**Data Fields**

| void ∗ | addr | |
|---|---|---|
| size_t | size | |

## 6.23 driver_mic_sink.h File Reference

```
#include <common/config.h>
#include <scif.h>
#include <drivers/mp_common/mp_common.h>
#include <drivers/mp_common/sink_common.h>
```

**Macros**

- #define **STARPU_MIC_SINK_REPORT_ERROR**(status)

**Functions**

- void **_starpu_mic_sink_report_error** (const char ∗func, const char ∗file, const int line, const int status)
- void **_starpu_mic_sink_init** (struct _starpu_mp_node ∗node)
- void **_starpu_mic_sink_launch_workers** (struct _starpu_mp_node ∗node)
- void **_starpu_mic_sink_deinit** (struct _starpu_mp_node ∗node)
- void **_starpu_mic_sink_allocate** (const struct _starpu_mp_node ∗mp_node, void ∗arg, int arg_size)
- void **_starpu_mic_sink_free** (const struct _starpu_mp_node ∗mp_node STARPU_ATTRIBUTE_UNUSED, void ∗arg, int arg_size)
- void **_starpu_mic_sink_bind_thread** (const struct _starpu_mp_node ∗mp_node STARPU_ATTRIBUTE_UNUSED, int coreid, int ∗core_table, int nb_core)

**Variables**

- void(∗)(void) **_starpu_mic_sink_lookup** (const struct _starpu_mp_node ∗node STARPU_ATTRIBUTE_UNUSED, char ∗func_name)

## 6.24 driver_mic_source.h File Reference

```
#include <starpu_mic.h>
#include <common/config.h>
#include <source/COIProcess_source.h>
```

```
#include <source/COIEngine_source.h>
#include <core/workers.h>
#include <drivers/mp_common/mp_common.h>
#include <datawizard/node_ops.h>
```

## Macros

- #define **STARPU_MIC_REQUEST_COMPLETE**
- #define **STARPU_MIC_SRC_REPORT_COI_ERROR**(status)
- #define **STARPU_MIC_SRC_REPORT_SCIF_ERROR**(status)

## Functions

- struct _starpu_mp_node ∗ **_starpu_mic_src_get_actual_thread_mp_node** ()
- struct _starpu_mp_node ∗ **_starpu_mic_src_get_mp_node_from_memory_node** (int memory_node)
- int **_starpu_mic_src_register_kernel** (starpu_mic_func_symbol_t ∗symbol, const char ∗func_name)
- starpu_mic_kernel_t **_starpu_mic_src_get_kernel** (starpu_mic_func_symbol_t symbol)
- void **_starpu_mic_src_report_coi_error** (const char ∗func, const char ∗file, int line, const COIRESULT status)
- void **_starpu_mic_src_report_scif_error** (const char ∗func, const char ∗file, int line, const int status)
- unsigned **_starpu_mic_src_get_device_count** (void)
- starpu_mic_kernel_t **_starpu_mic_src_get_kernel_from_codelet** (struct starpu_codelet ∗cl, unsigned nimpl)
- void **_starpu_mic_src_init** (struct _starpu_mp_node ∗node)
- void **_starpu_mic_clear_kernels** (void)
- void **_starpu_mic_src_deinit** (struct _starpu_mp_node ∗node)
- size_t **_starpu_mic_get_global_mem_size** (int devid)
- size_t **_starpu_mic_get_free_mem_size** (int devid)
- int **_starpu_mic_allocate_memory** (void ∗∗addr, size_t size, unsigned memory_node)
- void **_starpu_mic_free_memory** (void ∗addr, size_t size, unsigned memory_node)
- int **_starpu_mic_copy_ram_to_mic** (void ∗src, unsigned src_node STARPU_ATTRIBUTE_UNUSED, void ∗dst, unsigned dst_node, size_t size)
- int **_starpu_mic_copy_mic_to_ram** (void ∗src, unsigned src_node, void ∗dst, unsigned dst_node STAR←PU_ATTRIBUTE_UNUSED, size_t size)
- int **_starpu_mic_copy_ram_to_mic_async** (void ∗src, unsigned src_node STARPU_ATTRIBUTE_UNU←SED, void ∗dst, unsigned dst_node, size_t size)
- int **_starpu_mic_copy_mic_to_ram_async** (void ∗src, unsigned src_node, void ∗dst, unsigned dst_node STARPU_ATTRIBUTE_UNUSED, size_t size)
- int **_starpu_mic_init_event** (struct _starpu_mic_async_event ∗event, unsigned memory_node)
- void ∗ **_starpu_mic_src_worker** (void ∗arg)
- unsigned **_starpu_mic_test_request_completion** (struct _starpu_async_channel ∗async_channel)
- void **_starpu_mic_wait_request_completion** (struct _starpu_async_channel ∗async_channel)
- int **_starpu_mic_copy_data_from_mic_to_cpu** (starpu_data_handle_t handle, void ∗src_interface, un-signed src_node, void ∗dst_interface, unsigned dst_node, struct _starpu_data_request ∗req)
- int **_starpu_mic_copy_data_from_cpu_to_mic** (starpu_data_handle_t handle, void ∗src_interface, un-signed src_node, void ∗dst_interface, unsigned dst_node, struct _starpu_data_request ∗req)
- int **_starpu_mic_copy_interface_from_mic_to_cpu** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, struct _starpu_async_channel ∗async_←channel)
- int **_starpu_mic_copy_interface_from_cpu_to_mic** (uintptr_t src, size_t src_offset, unsigned src_node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, struct _starpu_async_channel ∗async_←channel)
- int **_starpu_mic_is_direct_access_supported** (unsigned node, unsigned handling_node)
- uintptr_t **_starpu_mic_malloc_on_node** (unsigned dst_node, size_t size, int flags)
- void **_starpu_mic_free_on_node** (unsigned dst_node, uintptr_t addr, size_t size, int flags)

**Variables**

- struct _starpu_node_ops **_starpu_driver_mic_node_ops**
- struct _starpu_mp_node ∗ _starpu_mic_nodes [STARPU_MAXMICDEVS]
- struct _starpu_mic_async_event ∗ **event**
- void(∗)(void) **_starpu_mic_src_get_kernel_from_job** (const struct _starpu_mp_node ∗node STARPU_↩
  ATTRIBUTE_UNUSED, struct _starpu_job ∗j)

**6.24.1 Variable Documentation**

**6.24.1.1 _starpu_mic_nodes**

```
struct _starpu_mp_node* _starpu_mic_nodes[STARPU_MAXMICDEVS]
```
Array of structures containing all the informations useful to send and receive informations with devices

## 6.25 driver_mpi_common.h File Reference

```
#include <drivers/mp_common/mp_common.h>
#include <drivers/mpi/driver_mpi_source.h>
```

## 6.26 driver_mpi_sink.h File Reference

```
#include <drivers/mp_common/sink_common.h>
```

## 6.27 driver_mpi_source.h File Reference

```
#include <drivers/mp_common/mp_common.h>
#include <starpu_mpi_ms.h>
#include <datawizard/node_ops.h>
```

## 6.28 driver_opencl.h File Reference

```
#include <CL/cl.h>
#include <core/workers.h>
#include <datawizard/node_ops.h>
```

**Macros**

- #define **_GNU_SOURCE**
- #define **CL_TARGET_OPENCL_VERSION**

**Functions**

- void **_starpu_opencl_discover_devices** (struct _starpu_machine_config ∗config)
- unsigned **_starpu_opencl_get_device_count** (void)
- void **_starpu_opencl_init** (void)
- void ∗ **_starpu_opencl_worker** (void ∗)
- int **_starpu_run_opencl** (struct _starpu_worker ∗)
- int **_starpu_opencl_driver_init** (struct _starpu_worker ∗)

- int **_starpu_opencl_driver_run_once** ([struct _starpu_worker](#) ∗)
- int **_starpu_opencl_driver_deinit** ([struct _starpu_worker](#) ∗)
- int **_starpu_opencl_init_context** (int devid)
- int **_starpu_opencl_deinit_context** (int devid)
- cl_device_type **_starpu_opencl_get_device_type** (int devid)
- unsigned **_starpu_opencl_test_request_completion** ([struct _starpu_async_channel](#) ∗async_channel)
- void **_starpu_opencl_wait_request_completion** ([struct _starpu_async_channel](#) ∗async_channel)
- int **_starpu_opencl_copy_interface_from_opencl_to_opencl** (starpu_data_handle_t handle, void ∗src←
  _interface, unsigned src_node, void ∗dst_interface, unsigned dst_node, [struct](#) _starpu_data_request ∗req)
- int **_starpu_opencl_copy_interface_from_opencl_to_cpu** (starpu_data_handle_t handle, void ∗src←
  interface, unsigned src_node, void ∗dst_interface, unsigned dst_node, [struct](#) _starpu_data_request ∗req)
- int **_starpu_opencl_copy_interface_from_cpu_to_opencl** (starpu_data_handle_t handle, void ∗src←
  interface, unsigned src_node, void ∗dst_interface, unsigned dst_node, [struct](#) _starpu_data_request ∗req)
- int **_starpu_opencl_copy_data_from_opencl_to_cpu** (uintptr_t src, size_t src_offset, unsigned src_node,
  uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, [struct _starpu_async_channel](#) ∗async←
  channel)
- int **_starpu_opencl_copy_data_from_opencl_to_opencl** (uintptr_t src, size_t src_offset, unsigned src←
  node, uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, [struct _starpu_async_channel](#) ∗async←
  _channel)
- int **_starpu_opencl_copy_data_from_cpu_to_opencl** (uintptr_t src, size_t src_offset, unsigned src_node,
  uintptr_t dst, size_t dst_offset, unsigned dst_node, size_t size, [struct _starpu_async_channel](#) ∗async←
  channel)
- int **_starpu_opencl_is_direct_access_supported** (unsigned node, unsigned handling_node)
- uintptr_t **_starpu_opencl_malloc_on_node** (unsigned dst_node, size_t size, int flags)
- void **_starpu_opencl_free_on_node** (unsigned dst_node, uintptr_t addr, size_t size, int flags)

**Variables**

- [struct _starpu_node_ops](#) **_starpu_driver_opencl_node_ops**
- [struct _starpu_driver_ops](#) **_starpu_driver_opencl_ops**
- char ∗ **_starpu_opencl_program_dir**

## 6.29   driver_opencl_utils.h File Reference

**Macros**

- #define **_STARPU_OPENCL_PLATFORM_MAX**

**Functions**

- char ∗ **_starpu_opencl_get_device_type_as_string** (int id)

## 6.30   drivers.h File Reference

**Data Structures**

- struct [_starpu_driver_ops](#)

## 6.31   errorcheck.h File Reference

```
#include <starpu.h>
```

**Enumerations**

- enum _starpu_worker_status {
  STATUS_INVALID, STATUS_UNKNOWN, STATUS_INITIALIZING, STATUS_EXECUTING,
  STATUS_CALLBACK, STATUS_SCHEDULING, STATUS_WAITING, STATUS_SLEEPING_SCHEDULI↩
  NG,
  STATUS_SLEEPING }

**Functions**

- void _starpu_set_worker_status (struct _starpu_worker ∗worker, enum _starpu_worker_status st)
- void **_starpu_set_local_worker_status** (enum _starpu_worker_status st)
- enum _starpu_worker_status _starpu_get_local_worker_status (void)
- unsigned _starpu_worker_may_perform_blocking_calls (void)

## 6.31.1 Enumeration Type Documentation

### 6.31.1.1 _starpu_worker_status

enum _starpu_worker_status
This type describes in which state a worker may be.

**Enumerator**

| STATUS_INVALID | invalid status (for instance if we request the status of some thread that is not controlled by StarPU |
| --- | --- |
| STATUS_UNKNOWN | everything that does not fit the other status |
| STATUS_INITIALIZING | during the initialization |
| STATUS_EXECUTING | during the execution of a codelet |
| STATUS_CALLBACK | during the execution of the callback |
| STATUS_SCHEDULING | while executing the scheduler code |
| STATUS_WAITING | while waiting for a data transfer |
| STATUS_SLEEPING_SCHEDULING | while sleeping because there is nothing to do, but looking for tasks to do |
| STATUS_SLEEPING | while sleeping because there is nothing to do, and not even scheduling |

## 6.31.2 Function Documentation

### 6.31.2.1 _starpu_set_worker_status()

void _starpu_set_worker_status (
            struct _starpu_worker ∗ *worker,*
            enum _starpu_worker_status *st* )
Specify what the local worker is currently doing (eg. executing a callback). This permits to detect if this is legal to do a blocking call for instance.

### 6.31.2.2 _starpu_get_local_worker_status()

enum _starpu_worker_status _starpu_get_local_worker_status (
            void  )
Indicate what type of operation the worker is currently doing.

**6.31.2.3 _starpu_worker_may_perform_blocking_calls()**

```
unsigned _starpu_worker_may_perform_blocking_calls (
            void )
```

It is forbidden to do blocking calls during some operations such as callback or during the execution of a task. This function indicates whether it is legal to call a blocking operation in the current context.

## 6.32 fifo_queues.h File Reference

```
#include <starpu.h>
#include <core/task.h>
```

**Data Structures**

- struct _starpu_fifo_taskq

**Functions**

- struct _starpu_fifo_taskq ∗ **_starpu_create_fifo** (void) STARPU_ATTRIBUTE_MALLOC
- void **_starpu_destroy_fifo** (struct _starpu_fifo_taskq ∗fifo)
- int **_starpu_fifo_empty** (struct _starpu_fifo_taskq ∗fifo)
- double **_starpu_fifo_get_exp_len_prev_task_list** (struct _starpu_fifo_taskq ∗fifo_queue, struct starpu_↩
  task ∗task, int workerid, int nimpl, int ∗fifo_ntasks)
- int **_starpu_fifo_push_sorted_task** (struct _starpu_fifo_taskq ∗fifo_queue, struct starpu_task ∗task)
- int **_starpu_fifo_push_task** (struct _starpu_fifo_taskq ∗fifo, struct starpu_task ∗task)
- int **_starpu_fifo_push_back_task** (struct _starpu_fifo_taskq ∗fifo_queue, struct starpu_task ∗task)
- int **_starpu_fifo_pop_this_task** (struct _starpu_fifo_taskq ∗fifo_queue, int workerid, struct starpu_task
  ∗task)
- struct starpu_task ∗ **_starpu_fifo_pop_task** (struct _starpu_fifo_taskq ∗fifo, int workerid)
- struct starpu_task ∗ **_starpu_fifo_pop_local_task** (struct _starpu_fifo_taskq ∗fifo)
- struct starpu_task ∗ **_starpu_fifo_pop_every_task** (struct _starpu_fifo_taskq ∗fifo, int workerid)
- int **_starpu_normalize_prio** (int priority, int num_priorities, unsigned sched_ctx_id)
- int **_starpu_count_non_ready_buffers** (struct starpu_task ∗task, unsigned worker)
- struct starpu_task ∗ **_starpu_fifo_pop_first_ready_task** (struct _starpu_fifo_taskq ∗fifo_queue, unsigned
  workerid, int num_priorities)

### 6.32.1 Data Structure Documentation

**6.32.1.1 struct _starpu_fifo_taskq**

**Data Fields**

| | | |
|---|---|---|
| struct starpu_task_list | taskq | the actual list |
| unsigned | ntasks | the number of tasks currently in the queue |
| unsigned ∗ | ntasks_per_priority | the number of tasks currently in the queue corresponding to each priority |
| unsigned | nprocessed | the number of tasks that were processed |
| double | exp_start | only meaningful if the queue is only used by a single worker |
| double | exp_end | Expected start date of next item to do in the queue (i.e. not started yet). This is thus updated when we start it. |
| double | exp_len | Expected end date of last task in the queue |
| double ∗ | exp_len_per_priority | Expected duration of the set of tasks in the queue |
| double | pipeline_len | Expected duration of the set of tasks in the queue corresponding to each priority |

## 6.33 filters.h File Reference

```
#include <stdarg.h>
#include <datawizard/coherency.h>
#include <datawizard/memalloc.h>
#include <starpu.h>
#include <common/config.h>
```

**Functions**

- void _starpu_data_partition_access_submit (starpu_data_handle_t target, int write)

### 6.33.1 Function Documentation

#### 6.33.1.1 _starpu_data_partition_access_submit()

```
void _starpu_data_partition_access_submit (
            starpu_data_handle_t target,
            int write )
```
submit asynchronous unpartitioning / partitioning to make target active read-only or read-write

## 6.34 footprint.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <core/jobs.h>
```

**Functions**

- uint32_t _starpu_compute_buffers_footprint (struct starpu_perfmodel ∗model, struct starpu_perfmodel_arch ∗arch, unsigned nimpl, struct _starpu_job ∗j)
- uint32_t _starpu_compute_data_footprint (starpu_data_handle_t handle)
- uint32_t _starpu_compute_data_alloc_footprint (starpu_data_handle_t handle)

### 6.34.1 Function Documentation

#### 6.34.1.1 _starpu_compute_buffers_footprint()

```
uint32_t _starpu_compute_buffers_footprint (
            struct starpu_perfmodel * model,
            struct starpu_perfmodel_arch * arch,
            unsigned nimpl,
            struct _starpu_job * j )
```
Compute the footprint that characterizes the job and cache it into the job structure.

#### 6.34.1.2 _starpu_compute_data_footprint()

```
uint32_t _starpu_compute_data_footprint (
            starpu_data_handle_t handle )
```
Compute the footprint that characterizes the layout of the data handle.

**6.34.1.3    _starpu_compute_data_alloc_footprint()**

```
uint32_t _starpu_compute_data_alloc_footprint (
            starpu_data_handle_t handle )
```
Compute the footprint that characterizes the allocation of the data handle.

## 6.35    fxt.h File Reference

```
#include <string.h>
#include <sys/types.h>
#include <stdlib.h>
#include <common/config.h>
#include <common/utils.h>
#include <starpu.h>
```

**Macros**

- #define **_GNU_SOURCE**
- #define **_STARPU_FUT_APPS_KEY**
- #define **_STARPU_FUT_CPU_KEY**
- #define **_STARPU_FUT_CUDA_KEY**
- #define **_STARPU_FUT_OPENCL_KEY**
- #define **_STARPU_FUT_MIC_KEY**
- #define **_STARPU_FUT_MPI_KEY**
- #define **_STARPU_FUT_WORKER_INIT_START**
- #define **_STARPU_FUT_WORKER_INIT_END**
- #define **_STARPU_FUT_START_CODELET_BODY**
- #define **_STARPU_FUT_END_CODELET_BODY**
- #define **_STARPU_FUT_JOB_PUSH**
- #define **_STARPU_FUT_JOB_POP**
- #define **_STARPU_FUT_UPDATE_TASK_CNT**
- #define **_STARPU_FUT_START_FETCH_INPUT_ON_TID**
- #define **_STARPU_FUT_END_FETCH_INPUT_ON_TID**
- #define **_STARPU_FUT_START_PUSH_OUTPUT_ON_TID**
- #define **_STARPU_FUT_END_PUSH_OUTPUT_ON_TID**
- #define **_STARPU_FUT_TAG**
- #define **_STARPU_FUT_TAG_DEPS**
- #define **_STARPU_FUT_TASK_DEPS**
- #define **_STARPU_FUT_DATA_COPY**
- #define **_STARPU_FUT_WORK_STEALING**
- #define **_STARPU_FUT_WORKER_DEINIT_START**
- #define **_STARPU_FUT_WORKER_DEINIT_END**
- #define **_STARPU_FUT_WORKER_SLEEP_START**
- #define **_STARPU_FUT_WORKER_SLEEP_END**
- #define **_STARPU_FUT_TASK_SUBMIT**
- #define **_STARPU_FUT_CODELET_DATA_HANDLE**
- #define **_STARPU_FUT_MODEL_NAME**
- #define **_STARPU_FUT_DATA_NAME**
- #define **_STARPU_FUT_DATA_COORDINATES**
- #define **_STARPU_FUT_HANDLE_DATA_UNREGISTER**
- #define **_STARPU_FUT_USER_DEFINED_START**
- #define **_STARPU_FUT_USER_DEFINED_END**
- #define **_STARPU_FUT_NEW_MEM_NODE**
- #define **_STARPU_FUT_START_CALLBACK**

- #define **_STARPU_FUT_END_CALLBACK**
- #define **_STARPU_FUT_TASK_DONE**
- #define **_STARPU_FUT_TAG_DONE**
- #define **_STARPU_FUT_START_ALLOC**
- #define **_STARPU_FUT_END_ALLOC**
- #define **_STARPU_FUT_START_ALLOC_REUSE**
- #define **_STARPU_FUT_END_ALLOC_REUSE**
- #define **_STARPU_FUT_USED_MEM**
- #define **_STARPU_FUT_TASK_NAME**
- #define **_STARPU_FUT_DATA_WONT_USE**
- #define **_STARPU_FUT_TASK_COLOR**
- #define **_STARPU_FUT_DATA_DOING_WONT_USE**
- #define **_STARPU_FUT_START_MEMRECLAIM**
- #define **_STARPU_FUT_END_MEMRECLAIM**
- #define **_STARPU_FUT_START_DRIVER_COPY**
- #define **_STARPU_FUT_END_DRIVER_COPY**
- #define **_STARPU_FUT_START_DRIVER_COPY_ASYNC**
- #define **_STARPU_FUT_END_DRIVER_COPY_ASYNC**
- #define **_STARPU_FUT_START_PROGRESS_ON_TID**
- #define **_STARPU_FUT_END_PROGRESS_ON_TID**
- #define **_STARPU_FUT_USER_EVENT**
- #define **_STARPU_FUT_SET_PROFILING**
- #define **_STARPU_FUT_TASK_WAIT_FOR_ALL**
- #define **_STARPU_FUT_EVENT**
- #define **_STARPU_FUT_THREAD_EVENT**
- #define **_STARPU_FUT_CODELET_DETAILS**
- #define **_STARPU_FUT_CODELET_DATA**
- #define **_STARPU_FUT_LOCKING_MUTEX**
- #define **_STARPU_FUT_MUTEX_LOCKED**
- #define **_STARPU_FUT_UNLOCKING_MUTEX**
- #define **_STARPU_FUT_MUTEX_UNLOCKED**
- #define **_STARPU_FUT_TRYLOCK_MUTEX**
- #define **_STARPU_FUT_RDLOCKING_RWLOCK**
- #define **_STARPU_FUT_RWLOCK_RDLOCKED**
- #define **_STARPU_FUT_WRLOCKING_RWLOCK**
- #define **_STARPU_FUT_RWLOCK_WRLOCKED**
- #define **_STARPU_FUT_UNLOCKING_RWLOCK**
- #define **_STARPU_FUT_RWLOCK_UNLOCKED**
- #define **_STARPU_FUT_LOCKING_SPINLOCK**
- #define **_STARPU_FUT_SPINLOCK_LOCKED**
- #define **_STARPU_FUT_UNLOCKING_SPINLOCK**
- #define **_STARPU_FUT_SPINLOCK_UNLOCKED**
- #define **_STARPU_FUT_TRYLOCK_SPINLOCK**
- #define **_STARPU_FUT_COND_WAIT_BEGIN**
- #define **_STARPU_FUT_COND_WAIT_END**
- #define **_STARPU_FUT_MEMORY_FULL**
- #define **_STARPU_FUT_DATA_LOAD**
- #define **_STARPU_FUT_START_UNPARTITION_ON_TID**
- #define **_STARPU_FUT_END_UNPARTITION_ON_TID**
- #define **_STARPU_FUT_START_FREE**
- #define **_STARPU_FUT_END_FREE**
- #define **_STARPU_FUT_START_WRITEBACK**
- #define **_STARPU_FUT_END_WRITEBACK**
- #define **_STARPU_FUT_SCHED_COMPONENT_PUSH_PRIO**
- #define **_STARPU_FUT_SCHED_COMPONENT_POP_PRIO**

- #define **_STARPU_FUT_START_WRITEBACK_ASYNC**
- #define **_STARPU_FUT_END_WRITEBACK_ASYNC**
- #define **_STARPU_FUT_HYPERVISOR_BEGIN**
- #define **_STARPU_FUT_HYPERVISOR_END**
- #define **_STARPU_FUT_BARRIER_WAIT_BEGIN**
- #define **_STARPU_FUT_BARRIER_WAIT_END**
- #define **_STARPU_FUT_WORKER_SCHEDULING_START**
- #define **_STARPU_FUT_WORKER_SCHEDULING_END**
- #define **_STARPU_FUT_WORKER_SCHEDULING_PUSH**
- #define **_STARPU_FUT_WORKER_SCHEDULING_POP**
- #define **_STARPU_FUT_START_EXECUTING**
- #define **_STARPU_FUT_END_EXECUTING**
- #define **_STARPU_FUT_SCHED_COMPONENT_NEW**
- #define **_STARPU_FUT_SCHED_COMPONENT_CONNECT**
- #define **_STARPU_FUT_SCHED_COMPONENT_PUSH**
- #define **_STARPU_FUT_SCHED_COMPONENT_PULL**
- #define **_STARPU_FUT_TASK_SUBMIT_START**
- #define **_STARPU_FUT_TASK_SUBMIT_END**
- #define **_STARPU_FUT_TASK_BUILD_START**
- #define **_STARPU_FUT_TASK_BUILD_END**
- #define **_STARPU_FUT_TASK_MPI_DECODE_START**
- #define **_STARPU_FUT_TASK_MPI_DECODE_END**
- #define **_STARPU_FUT_TASK_MPI_PRE_START**
- #define **_STARPU_FUT_TASK_MPI_PRE_END**
- #define **_STARPU_FUT_TASK_MPI_POST_START**
- #define **_STARPU_FUT_TASK_MPI_POST_END**
- #define **_STARPU_FUT_TASK_WAIT_START**
- #define **_STARPU_FUT_TASK_WAIT_END**
- #define **_STARPU_FUT_TASK_WAIT_FOR_ALL_START**
- #define **_STARPU_FUT_TASK_WAIT_FOR_ALL_END**
- #define **_STARPU_FUT_HANDLE_DATA_REGISTER**
- #define **_STARPU_FUT_START_FETCH_INPUT**
- #define **_STARPU_FUT_END_FETCH_INPUT**
- #define **_STARPU_FUT_TASK_THROTTLE_START**
- #define **_STARPU_FUT_TASK_THROTTLE_END**
- #define **_STARPU_FUT_DATA_STATE_INVALID**
- #define **_STARPU_FUT_DATA_STATE_OWNER**
- #define **_STARPU_FUT_DATA_STATE_SHARED**
- #define **_STARPU_FUT_DATA_REQUEST_CREATED**
- #define **_STARPU_FUT_TASK_EXCLUDE_FROM_DAG**
- #define **_STARPU_TRACE_NEW_MEM_NODE**(nodeid)
- #define **_STARPU_TRACE_WORKER_INIT_START**(a, b, c, d, e, f)
- #define **_STARPU_TRACE_WORKER_INIT_END**(workerid)
- #define **_STARPU_TRACE_START_CODELET_BODY**(job, nimpl, perf_arch, workerid)
- #define **_STARPU_TRACE_END_CODELET_BODY**(job, nimpl, perf_arch, workerid)
- #define **_STARPU_TRACE_START_EXECUTING**()
- #define **_STARPU_TRACE_END_EXECUTING**()
- #define **_STARPU_TRACE_START_CALLBACK**(job)
- #define **_STARPU_TRACE_END_CALLBACK**(job)
- #define **_STARPU_TRACE_JOB_PUSH**(task, prio)
- #define **_STARPU_TRACE_JOB_POP**(task, prio)
- #define **_STARPU_TRACE_UPDATE_TASK_CNT**(counter)
- #define **_STARPU_TRACE_START_FETCH_INPUT**(job)
- #define **_STARPU_TRACE_END_FETCH_INPUT**(job)
- #define **_STARPU_TRACE_START_PUSH_OUTPUT**(job)

- #define **_STARPU_TRACE_END_PUSH_OUTPUT**(job)
- #define **_STARPU_TRACE_TAG**(tag, job)
- #define **_STARPU_TRACE_TAG_DEPS**(a, b)
- #define **_STARPU_TRACE_TASK_DEPS**(a, b)
- #define **_STARPU_TRACE_GHOST_TASK_DEPS**(a, b)
- #define **_STARPU_TRACE_TASK_EXCLUDE_FROM_DAG**(a)
- #define **_STARPU_TRACE_TASK_NAME**(a)
- #define **_STARPU_TRACE_TASK_COLOR**(a)
- #define **_STARPU_TRACE_TASK_DONE**(a)
- #define **_STARPU_TRACE_TAG_DONE**(a)
- #define **_STARPU_TRACE_DATA_NAME**(a, b)
- #define **_STARPU_TRACE_DATA_COORDINATES**(a, b, c)
- #define **_STARPU_TRACE_DATA_COPY**(a, b, c)
- #define **_STARPU_TRACE_DATA_WONT_USE**(a)
- #define **_STARPU_TRACE_DATA_DOING_WONT_USE**(a)
- #define **_STARPU_TRACE_START_DRIVER_COPY**(a, b, c, d, e, f)
- #define **_STARPU_TRACE_END_DRIVER_COPY**(a, b, c, d, e)
- #define **_STARPU_TRACE_START_DRIVER_COPY_ASYNC**(a, b)
- #define **_STARPU_TRACE_END_DRIVER_COPY_ASYNC**(a, b)
- #define **_STARPU_TRACE_WORK_STEALING**(a, b)
- #define **_STARPU_TRACE_WORKER_DEINIT_START**
- #define **_STARPU_TRACE_WORKER_DEINIT_END**(a)
- #define **_STARPU_TRACE_WORKER_SCHEDULING_START**
- #define **_STARPU_TRACE_WORKER_SCHEDULING_END**
- #define **_STARPU_TRACE_WORKER_SCHEDULING_PUSH**
- #define **_STARPU_TRACE_WORKER_SCHEDULING_POP**
- #define **_STARPU_TRACE_WORKER_SLEEP_START**
- #define **_STARPU_TRACE_WORKER_SLEEP_END**
- #define **_STARPU_TRACE_TASK_SUBMIT**(job, a, b)
- #define **_STARPU_TRACE_TASK_SUBMIT_START**()
- #define **_STARPU_TRACE_TASK_SUBMIT_END**()
- #define **_STARPU_TRACE_TASK_THROTTLE_START**()
- #define **_STARPU_TRACE_TASK_THROTTLE_END**()
- #define **_STARPU_TRACE_TASK_BUILD_START**()
- #define **_STARPU_TRACE_TASK_BUILD_END**()
- #define **_STARPU_TRACE_TASK_MPI_DECODE_START**()
- #define **_STARPU_TRACE_TASK_MPI_DECODE_END**()
- #define **_STARPU_TRACE_TASK_MPI_PRE_START**()
- #define **_STARPU_TRACE_TASK_MPI_PRE_END**()
- #define **_STARPU_TRACE_TASK_MPI_POST_START**()
- #define **_STARPU_TRACE_TASK_MPI_POST_END**()
- #define **_STARPU_TRACE_TASK_WAIT_START**(job)
- #define **_STARPU_TRACE_TASK_WAIT_END**()
- #define **_STARPU_TRACE_TASK_WAIT_FOR_ALL_START**()
- #define **_STARPU_TRACE_TASK_WAIT_FOR_ALL_END**()
- #define **_STARPU_TRACE_USER_DEFINED_START**()
- #define **_STARPU_TRACE_USER_DEFINED_END**()
- #define **_STARPU_TRACE_START_ALLOC**(memnode, size, handle, is_prefetch)
- #define **_STARPU_TRACE_END_ALLOC**(memnode, handle, r)
- #define **_STARPU_TRACE_START_ALLOC_REUSE**(a, size, handle, is_prefetch)
- #define **_STARPU_TRACE_END_ALLOC_REUSE**(a, handle, r)
- #define **_STARPU_TRACE_START_FREE**(memnode, size, handle)
- #define **_STARPU_TRACE_END_FREE**(memnode, handle)
- #define **_STARPU_TRACE_START_WRITEBACK**(memnode, handle)
- #define **_STARPU_TRACE_END_WRITEBACK**(memnode, handle)

- #define **_STARPU_TRACE_USED_MEM**(memnode, used)
- #define **_STARPU_TRACE_START_MEMRECLAIM**(memnode, is_prefetch)
- #define **_STARPU_TRACE_END_MEMRECLAIM**(memnode, is_prefetch)
- #define **_STARPU_TRACE_START_WRITEBACK_ASYNC**(memnode)
- #define **_STARPU_TRACE_END_WRITEBACK_ASYNC**(memnode)
- #define **_STARPU_TRACE_START_PROGRESS**(memnode)
- #define **_STARPU_TRACE_END_PROGRESS**(memnode)
- #define **_STARPU_TRACE_USER_EVENT**(code)
- #define **_STARPU_TRACE_SET_PROFILING**(status)
- #define **_STARPU_TRACE_TASK_WAIT_FOR_ALL**()
- #define **_STARPU_TRACE_EVENT**(S)
- #define **_STARPU_TRACE_THREAD_EVENT**(S)
- #define **_STARPU_TRACE_LOCKING_MUTEX**()
- #define **_STARPU_TRACE_MUTEX_LOCKED**()
- #define **_STARPU_TRACE_UNLOCKING_MUTEX**()
- #define **_STARPU_TRACE_MUTEX_UNLOCKED**()
- #define **_STARPU_TRACE_TRYLOCK_MUTEX**()
- #define **_STARPU_TRACE_RDLOCKING_RWLOCK**()
- #define **_STARPU_TRACE_RWLOCK_RDLOCKED**()
- #define **_STARPU_TRACE_WRLOCKING_RWLOCK**()
- #define **_STARPU_TRACE_RWLOCK_WRLOCKED**()
- #define **_STARPU_TRACE_UNLOCKING_RWLOCK**()
- #define **_STARPU_TRACE_RWLOCK_UNLOCKED**()
- #define **_STARPU_TRACE_LOCKING_SPINLOCK**(file, line)
- #define **_STARPU_TRACE_SPINLOCK_LOCKED**(file, line)
- #define **_STARPU_TRACE_UNLOCKING_SPINLOCK**(file, line)
- #define **_STARPU_TRACE_SPINLOCK_UNLOCKED**(file, line)
- #define **_STARPU_TRACE_TRYLOCK_SPINLOCK**(file, line)
- #define **_STARPU_TRACE_COND_WAIT_BEGIN**()
- #define **_STARPU_TRACE_COND_WAIT_END**()
- #define **_STARPU_TRACE_BARRIER_WAIT_BEGIN**()
- #define **_STARPU_TRACE_BARRIER_WAIT_END**()
- #define **_STARPU_TRACE_MEMORY_FULL**(size)
- #define **_STARPU_TRACE_DATA_LOAD**(workerid, size)
- #define **_STARPU_TRACE_START_UNPARTITION**(handle, memnode)
- #define **_STARPU_TRACE_END_UNPARTITION**(handle, memnode)
- #define **_STARPU_TRACE_SCHED_COMPONENT_PUSH_PRIO**(workerid, ntasks, exp_len)
- #define **_STARPU_TRACE_SCHED_COMPONENT_POP_PRIO**(workerid, ntasks, exp_len)
- #define **_STARPU_TRACE_HYPERVISOR_BEGIN**()
- #define **_STARPU_TRACE_HYPERVISOR_END**()
- #define **_STARPU_TRACE_SCHED_COMPONENT_NEW**(component)
- #define **_STARPU_TRACE_SCHED_COMPONENT_CONNECT**(parent, child)
- #define **_STARPU_TRACE_SCHED_COMPONENT_PUSH**(from, to, task)
- #define **_STARPU_TRACE_SCHED_COMPONENT_PULL**(from, to, task)
- #define **_STARPU_TRACE_HANDLE_DATA_REGISTER**(handle)
- #define **_STARPU_TRACE_HANDLE_DATA_UNREGISTER**(handle)
- #define **_STARPU_TRACE_WORKER_START_FETCH_INPUT**(job, id)
- #define **_STARPU_TRACE_WORKER_END_FETCH_INPUT**(job, id)
- #define **_STARPU_TRACE_DATA_STATE_INVALID**(handle, node)
- #define **_STARPU_TRACE_DATA_STATE_OWNER**(handle, node)
- #define **_STARPU_TRACE_DATA_STATE_SHARED**(handle, node)
- #define **_STARPU_TRACE_DATA_REQUEST_CREATED**(handle, orig, dest, prio, is_pre)

**Functions**

- static unsigned long **_starpu_fxt_get_job_id** (void)

**Variables**

- unsigned long **_starpu_job_cnt**

## 6.36 graph.h File Reference

```
#include <common/list.h>
```

**Data Structures**

- struct _starpu_graph_node

**Functions**

- void **_starpu_graph_init** (void)
- void **_starpu_graph_wrlock** (void)
- void **_starpu_graph_rdlock** (void)
- void **_starpu_graph_wrunlock** (void)
- void **_starpu_graph_rdunlock** (void)
- void _starpu_graph_add_job (struct _starpu_job ∗job)
- void _starpu_graph_add_job_dep (struct _starpu_job ∗job, struct _starpu_job ∗prev_job)
- void _starpu_graph_drop_job (struct _starpu_job ∗job)
- void _starpu_graph_drop_dropped_nodes (void)
- void _starpu_graph_compute_depths (void)
- void _starpu_graph_compute_descendants (void)
- void _starpu_graph_foreach (void(∗func)(void ∗data, struct _starpu_graph_node ∗node), void ∗data)

**Variables**

- int **_starpu_graph_record**

### 6.36.1 Data Structure Documentation

#### 6.36.1.1 struct _starpu_graph_node

**Data Fields**

| starpu_pthread_mutex_t | mutex | protects access to the job |
|---|---|---|
| struct _starpu_job ∗ | job | pointer to the job, if it is still alive, NULL otherwise |
| struct _starpu_graph_node_multilist_top | top | Fields for graph analysis for scheduling heuristicsMember of list of all jobs without incoming dependency |
| struct _starpu_graph_node_multilist_bottom | bottom | Member of list of all jobs without outgoing dependency |
| struct _starpu_graph_node_multilist_all | all | Member of list of all jobs |
| struct _starpu_graph_node_multilist_dropped | dropped | Member of list of dropped jobs |

**Data Fields**

| | | |
|---:|---|---|
| struct _starpu_graph_node ∗∗ | incoming | set of incoming dependencies May contain NULLs for terminated jobs |
| unsigned ∗ | incoming_slot | Index within corresponding outgoing array |
| unsigned | n_incoming | Number of slots used |
| unsigned | alloc_incoming | Size of incoming |
| struct _starpu_graph_node ∗∗ | outgoing | set of outgoing dependencies |
| unsigned ∗ | outgoing_slot | Index within corresponding incoming array |
| unsigned | n_outgoing | Number of slots used |
| unsigned | alloc_outgoing | Size of outgoing |
| unsigned | depth | Rank from bottom, in number of jobs Only available if _starpu_graph_compute_depths was called |
| unsigned | descendants | Number of children, grand-children, etc. Only available if _starpu_graph_compute_descendants was called |
| int | graph_n | Variable available for graph flow |

## 6.36.2 Function Documentation

### 6.36.2.1 _starpu_graph_add_job()

```
void _starpu_graph_add_job (
            struct _starpu_job * job )
```
Add a job to the graph, called before any _starpu_graph_add_job_dep call

### 6.36.2.2 _starpu_graph_add_job_dep()

```
void _starpu_graph_add_job_dep (
            struct _starpu_job * job,
            struct _starpu_job * prev_job )
```
Add a dependency between jobs

### 6.36.2.3 _starpu_graph_drop_job()

```
void _starpu_graph_drop_job (
            struct _starpu_job * job )
```
Remove a job from the graph

### 6.36.2.4 _starpu_graph_drop_dropped_nodes()

```
void _starpu_graph_drop_dropped_nodes (
            void  )
```
Really drop the nodes from the graph now

### 6.36.2.5 _starpu_graph_compute_depths()

```
void _starpu_graph_compute_depths (
            void  )
```
This make StarPU compute for each task the depth, i.e. the length of the longest path to a task without outgoing dependencies. This does not take job duration into account, just the number

**6.36.2.6 _starpu_graph_compute_descendants()**

```
void _starpu_graph_compute_descendants (
            void  )
```
Compute the descendants of jobs in the graph

**6.36.2.7 _starpu_graph_foreach()**

```
void _starpu_graph_foreach (
            void(*)(void *data, struct _starpu_graph_node *node) func,
            void * data )
```
This calls *func* for each node of the task graph, passing also *data* as it Apply func on each job of the graph

## 6.37 helper_mct.h File Reference

**Data Structures**

- struct _starpu_mct_data

**Functions**

- struct _starpu_mct_data ∗ **starpu_mct_init_parameters** (struct starpu_sched_component_mct_data ∗params)
- unsigned **starpu_mct_compute_execution_times** (struct starpu_sched_component ∗component, struct starpu_task ∗task, double ∗estimated_lengths, double ∗estimated_transfer_length, unsigned ∗suitable_↩ components)
- void **starpu_mct_compute_expected_times** (struct starpu_sched_component ∗component, struct starpu_task ∗task, double ∗estimated_lengths, double ∗estimated_transfer_length, double ∗estimated↩ _ends_with_task, double ∗min_exp_end_with_task, double ∗max_exp_end_with_task, unsigned ∗suitable↩ _components, unsigned nsuitable_components)
- double **starpu_mct_compute_fitness** (struct _starpu_mct_data ∗d, double exp_end, double min_exp_end, double max_exp_end, double transfer_len, double local_energy)
- int **starpu_mct_get_best_component** (struct _starpu_mct_data ∗d, struct starpu_task ∗task, double ∗estimated_lengths, double ∗estimated_transfer_length, double ∗estimated_ends_with_task, double min_↩ exp_end_with_task, double max_exp_end_with_task, unsigned ∗suitable_components, unsigned nsuitable↩ _components)

### 6.37.1 Data Structure Documentation

**6.37.1.1 struct _starpu_mct_data**

**Data Fields**

| | | |
|---|---|---|
| double | alpha | |
| double | beta | |
| double | _gamma | |
| double | idle_power | |
| starpu_pthread_mutex_t | scheduling_mutex | |

## 6.38 idle_hook.h File Reference

**Functions**

- void **_starpu_init_idle_hooks** (void)
- unsigned **_starpu_execute_registered_idle_hooks** (void)

## 6.39 implicit_data_deps.h File Reference

```
#include <starpu.h>
#include <common/config.h>
```

**Functions**

- struct starpu_task * **_starpu_detect_implicit_data_deps_with_handle** (struct starpu_task *pre_↩
  sync_task, struct starpu_task *post_sync_task, struct _starpu_task_wrapper_dlist *post_sync_task_↩
  dependency_slot, starpu_data_handle_t handle, enum starpu_data_access_mode mode, unsigned task_↩
  handle_sequential_consistency)
- int **_starpu_test_implicit_data_deps_with_handle** (starpu_data_handle_t handle, enum starpu_data_↩
  access_mode mode)
- void **_starpu_detect_implicit_data_deps** (struct starpu_task *task)
- void **_starpu_release_data_enforce_sequential_consistency** (struct starpu_task *task, struct _starpu↩
  _task_wrapper_dlist *task_dependency_slot, starpu_data_handle_t handle)
- void **_starpu_release_task_enforce_sequential_consistency** (struct _starpu_job *j)
- void **_starpu_add_post_sync_tasks** (struct starpu_task *post_sync_task, starpu_data_handle_t handle)
- void **_starpu_unlock_post_sync_tasks** (starpu_data_handle_t handle)
- void _starpu_implicit_data_deps_write_hook (void(*func)(starpu_data_handle_t))
- int _starpu_data_wait_until_available (starpu_data_handle_t handle, enum starpu_data_access_mode
  mode, const char *sync_name)
- void **_starpu_data_clear_implicit** (starpu_data_handle_t handle)

### 6.39.1 Function Documentation

#### 6.39.1.1 _starpu_implicit_data_deps_write_hook()

```
void _starpu_implicit_data_deps_write_hook (
            void(*)(starpu_data_handle_t) func )
```
Register a hook to be called when a write is submitted

#### 6.39.1.2 _starpu_data_wait_until_available()

```
int _starpu_data_wait_until_available (
            starpu_data_handle_t handle,
            enum starpu_data_access_mode mode,
            const char * sync_name )
```
This function blocks until the handle is available in the requested mode

## 6.40 jobs.h File Reference

```
#include <starpu.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <stdarg.h>
#include <common/config.h>
#include <common/timing.h>
#include <common/list.h>
#include <common/fxt.h>
#include <core/dependencies/tags.h>
```

```
#include <datawizard/datawizard.h>
#include <core/perfmodel/perfmodel.h>
#include <core/errorcheck.h>
#include <common/barrier.h>
#include <common/utils.h>
#include <cuda.h>
```

## Data Structures

- struct _starpu_data_descr
- struct _starpu_job

## Macros

- #define **_STARPU_CPU_MAY_PERFORM**(j)
- #define **_STARPU_CUDA_MAY_PERFORM**(j)
- #define **_STARPU_OPENCL_MAY_PERFORM**(j)
- #define **_STARPU_MIC_MAY_PERFORM**(j)
- #define **_STARPU_JOB_GET_ORDERED_BUFFER_INDEX**(job, i)
- #define **_STARPU_JOB_GET_ORDERED_BUFFER_HANDLE**(job, i)
- #define **_STARPU_JOB_GET_ORDERED_BUFFER_MODE**(job, i)
- #define **_STARPU_JOB_GET_ORDERED_BUFFER_NODE**(job, i)
- #define **_STARPU_JOB_SET_ORDERED_BUFFER_INDEX**(job, __index, i)
- #define **_STARPU_JOB_SET_ORDERED_BUFFER_HANDLE**(job, __handle, i)
- #define **_STARPU_JOB_SET_ORDERED_BUFFER_MODE**(job, __mode, i)
- #define **_STARPU_JOB_SET_ORDERED_BUFFER_NODE**(job, __node, i)
- #define **_STARPU_JOB_SET_ORDERED_BUFFER**(job, buffer, i)
- #define **_STARPU_JOB_GET_ORDERED_BUFFERS**(job)
- #define **_STARPU_JOB_GET_DEP_SLOTS**(job)

## Typedefs

- typedef void(∗ _starpu_cl_func_t) (void ∗∗, void ∗)

## Functions

- void **_starpu_job_init** (void)
- void **_starpu_job_fini** (void)
- struct _starpu_job ∗ _starpu_job_create (struct starpu_task ∗task) STARPU_ATTRIBUTE_MALLOC
- void _starpu_job_destroy (struct _starpu_job ∗j)
- int _starpu_job_finished (struct _starpu_job ∗j)
- void _starpu_wait_job (struct _starpu_job ∗j)
- int _starpu_test_job_termination (struct _starpu_job ∗j)
- void _starpu_job_prepare_for_continuation_ext (struct _starpu_job ∗j, unsigned continuation_resubmit, void(∗continuation_callback_on_sleep)(void ∗arg), void ∗continuation_callback_on_sleep_arg)
- void **_starpu_job_prepare_for_continuation** (struct _starpu_job ∗j)
- void **_starpu_job_set_omp_cleanup_callback** (struct _starpu_job ∗j, void(∗omp_cleanup_callback)(void ∗arg), void ∗omp_cleanup_callback_arg)
- void _starpu_exclude_task_from_dag (struct starpu_task ∗task)
- unsigned _starpu_enforce_deps_and_schedule (struct _starpu_job ∗j)
- unsigned **_starpu_enforce_deps_starting_from_task** (struct _starpu_job ∗j)
- unsigned _starpu_reenforce_task_deps_and_schedule (struct _starpu_job ∗j)
- void **_starpu_enforce_deps_notify_job_ready_soon** (struct _starpu_job ∗j, _starpu_notify_job_start_data ∗data, int tag)
- void _starpu_handle_job_submission (struct _starpu_job ∗j)

- void _starpu_handle_job_termination (struct _starpu_job ∗j)
- size_t _starpu_job_get_data_size (struct starpu_perfmodel ∗model, struct starpu_perfmodel_arch ∗arch, unsigned nimpl, struct _starpu_job ∗j)
- struct starpu_task ∗ _starpu_pop_local_task (struct _starpu_worker ∗worker)
- int _starpu_push_local_task (struct _starpu_worker ∗worker, struct starpu_task ∗task, int prio)

### 6.40.1 Data Structure Documentation

#### 6.40.1.1 struct _starpu_data_descr

**Data Fields**

| | | |
|---:|---|---|
| starpu_data_handle_t | handle | |
| enum starpu_data_access_mode | mode | |
| int | node | |
| int | index | This is the value actually chosen, only set by _starpu_fetch_task_input for coherency with __starpu_push_task_output |
| int | orderedindex | |

### 6.40.2 Typedef Documentation

#### 6.40.2.1 _starpu_cl_func_t

```
typedef void(* _starpu_cl_func_t) (void **, void *)
```
codelet function

### 6.40.3 Function Documentation

#### 6.40.3.1 _starpu_job_create()

```
struct _starpu_job* _starpu_job_create (
            struct starpu_task * task )
```
Create an internal struct _starpu_job ∗structure to encapsulate the task.

#### 6.40.3.2 _starpu_job_destroy()

```
void _starpu_job_destroy (
            struct _starpu_job * j )
```
Destroy the data structure associated to the job structure

#### 6.40.3.3 _starpu_job_finished()

```
int _starpu_job_finished (
            struct _starpu_job * j )
```
Test for the termination of the job

#### 6.40.3.4 _starpu_wait_job()

```
void _starpu_wait_job (
            struct _starpu_job * j )
```
Wait for the termination of the job

**6.40.3.5 _starpu_test_job_termination()**

```
int _starpu_test_job_termination (
            struct _starpu_job * j )
```
Test for the termination of the job

**6.40.3.6 _starpu_job_prepare_for_continuation_ext()**

```
void _starpu_job_prepare_for_continuation_ext (
            struct _starpu_job * j,
            unsigned continuation_resubmit,
            void(*)(void *arg) continuation_callback_on_sleep,
            void * continuation_callback_on_sleep_arg )
```
Prepare the job for accepting new dependencies before becoming a continuation.

**6.40.3.7 _starpu_exclude_task_from_dag()**

```
void _starpu_exclude_task_from_dag (
            struct starpu_task * task )
```
Specify that the task should not appear in the DAG generated by debug tools.

**6.40.3.8 _starpu_enforce_deps_and_schedule()**

```
unsigned _starpu_enforce_deps_and_schedule (
            struct _starpu_job * j )
```
try to submit job j, enqueue it if it's not schedulable yet. The job's sync mutex is supposed to be held already

**6.40.3.9 _starpu_reenforce_task_deps_and_schedule()**

```
unsigned _starpu_reenforce_task_deps_and_schedule (
            struct _starpu_job * j )
```
When waking up a continuation, we only enforce new task dependencies

**6.40.3.10 _starpu_handle_job_submission()**

```
void _starpu_handle_job_submission (
            struct _starpu_job * j )
```
Called at the submission of the job

**6.40.3.11 _starpu_handle_job_termination()**

```
void _starpu_handle_job_termination (
            struct _starpu_job * j )
```
This function must be called after the execution of a job, this triggers all job's dependencies and perform the callback function if any.

**6.40.3.12 _starpu_job_get_data_size()**

```
size_t _starpu_job_get_data_size (
            struct starpu_perfmodel * model,
            struct starpu_perfmodel_arch * arch,
            unsigned nimpl,
            struct _starpu_job * j )
```
Get the sum of the size of the data accessed by the job.

**6.40.3.13 _starpu_pop_local_task()**

```
struct starpu_task* _starpu_pop_local_task (
            struct _starpu_worker * worker )
```
Get a task from the local pool of tasks that were explicitly attributed to that worker.

**6.40.3.14 _starpu_push_local_task()**

```
int _starpu_push_local_task (
            struct _starpu_worker * worker,
            struct starpu_task * task,
            int prio )
```
Put a task into the pool of tasks that are explicitly attributed to the specified worker. If "back" is set, the task is put at the back of the list. Considering the tasks are popped from the back, this value should be 0 to enforce a FIFO ordering.

## 6.41 malloc.h File Reference

**Functions**

- void **_starpu_malloc_init** (unsigned dst_node)
- void **_starpu_malloc_shutdown** (unsigned dst_node)
- void **_starpu_free_on_node** (unsigned dst_node, uintptr_t addr, size_t size)
- int **_starpu_malloc_flags_on_node** (unsigned dst_node, void ∗∗A, size_t dim, int flags)
- int **_starpu_free_flags_on_node** (unsigned dst_node, void ∗A, size_t dim, int flags)

## 6.42 memalloc.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/list.h>
#include <datawizard/interfaces/data_interface.h>
#include <datawizard/coherency.h>
#include <datawizard/copy_driver.h>
#include <datawizard/data_request.h>
```

## 6.43 memory_manager.h File Reference

```
#include <starpu.h>
```

**Functions**

- int _starpu_memory_manager_init ()
- void _starpu_memory_manager_set_global_memory_size (unsigned node, size_t size)
- size_t _starpu_memory_manager_get_global_memory_size (unsigned node)
- int **_starpu_memory_manager_test_allocate_size** (unsigned node, size_t size)

### 6.43.1 Function Documentation

**6.43.1.1 _starpu_memory_manager_init()**

```
int _starpu_memory_manager_init ( )
```
Initialises the memory manager

### 6.43.1.2 _starpu_memory_manager_set_global_memory_size()

```
void _starpu_memory_manager_set_global_memory_size (
            unsigned node,
            size_t size )
```
Initialises the global memory size for the given node

### 6.43.1.3 _starpu_memory_manager_get_global_memory_size()

```
size_t _starpu_memory_manager_get_global_memory_size (
            unsigned node )
```
Gets the global memory size for the given node

## 6.44 memory_nodes.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <datawizard/coherency.h>
#include <datawizard/memalloc.h>
#include <datawizard/node_ops.h>
#include <common/utils.h>
#include <core/workers.h>
#include <core/simgrid.h>
```

**Data Structures**

- struct _starpu_cond_and_worker
- struct _starpu_memory_node_descr

**Macros**

- #define **starpu_node_get_kind**
- #define **starpu_memory_nodes_get_count**
- #define **starpu_worker_get_memory_node**

**Functions**

- void **_starpu_memory_nodes_init** (void)
- void **_starpu_memory_nodes_deinit** (void)
- static void **_starpu_memory_node_add_nworkers** (unsigned node)
- void _starpu_worker_drives_memory_node (struct _starpu_worker ∗worker, unsigned memnode)
- static struct _starpu_node_ops ∗ **_starpu_memory_node_get_node_ops** (unsigned node)
- static unsigned **_starpu_memory_node_get_nworkers** (unsigned node)
- static void **_starpu_simgrid_memory_node_set_host** (unsigned node, starpu_sg_host_t host)
- static starpu_sg_host_t **_starpu_simgrid_memory_node_get_host** (unsigned node)
- unsigned **_starpu_memory_node_register** (enum starpu_node_kind kind, int devid, struct _starpu_node↩
  _ops ∗node_ops)
- void **_starpu_memory_node_register_condition** (struct _starpu_worker ∗worker, starpu_pthread_cond↩
  _t ∗cond, unsigned nodeid)
- static struct _starpu_memory_node_descr ∗ **_starpu_memory_node_get_description** (void)
- static enum starpu_node_kind **_starpu_node_get_kind** (unsigned node)
- static unsigned **_starpu_memory_nodes_get_count** (void)
- static unsigned _starpu_worker_get_memory_node (unsigned workerid)

**Variables**

- char **_starpu_worker_drives_memory** [STARPU_NMAXWORKERS][STARPU_MAXNODES]
- struct _starpu_memory_node_descr **_starpu_descr**

### 6.44.1 Data Structure Documentation

#### 6.44.1.1 struct _starpu_cond_and_worker

**Data Fields**

| starpu_pthread_cond_t ∗ | cond | |
|---|---|---|
| struct _starpu_worker ∗ | worker | |

#### 6.44.1.2 struct _starpu_memory_node_descr

**Data Fields**

| | | |
|---|---|---|
| unsigned | nnodes | |
| enum starpu_node_kind | nodes[STARPU_MAXNODES] | |
| struct _starpu_node_ops ∗ | node_ops[STARPU_MAXNODES] | |
| int | devid[STARPU_MAXNODES] | Get the device id associated to this node, or -1 if not applicable |
| unsigned | nworkers[STARPU_MAXNODES] | |
| starpu_sg_host_t | host[STARPU_MAXNODES] | |
| starpu_pthread_rwlock_t | conditions_rwlock | Every worker is associated to a condition variable on which the worker waits when there is task available. It is possible that multiple worker share the same condition variable, so we maintain a list of all these condition variables so that we can wake up all worker attached to a memory node that are waiting on a task. |
| struct _starpu_cond_and_worker | conditions_attached_to_node[STARPU_MAXNODES][STARPU_NMAXWORKERS] | |
| struct _starpu_cond_and_worker | conditions_all[STARPU_MAXNODES ∗STARPU_NMAXWORKERS] | |
| unsigned | total_condition_count | the number of queues attached to each node |
| unsigned | condition_count[STARPU_MAXNODES] | |

### 6.44.2 Function Documentation

#### 6.44.2.1 _starpu_worker_drives_memory_node()

```
void _starpu_worker_drives_memory_node (
            struct _starpu_worker * worker,
            unsigned memnode )
```
same utility as _starpu_memory_node_add_nworkers

**6.44.2.2 _starpu_worker_get_memory_node()**

```
static unsigned _starpu_worker_get_memory_node (
            unsigned workerid ) [inline], [static]
```
This workerid may either be a basic worker or a combined worker
We have a combined worker

## 6.45 memstats.h File Reference

```
#include <starpu.h>
#include <common/config.h>
```

**Typedefs**

- typedef void ∗ **_starpu_memory_stats_t**

**Functions**

- void **_starpu_memory_stats_init** (starpu_data_handle_t handle)
- void **_starpu_memory_stats_init_per_node** (starpu_data_handle_t handle, unsigned node)
- void **_starpu_memory_stats_free** (starpu_data_handle_t handle)
- void **_starpu_memory_display_handle_stats** (FILE ∗stream, starpu_data_handle_t handle)
- void **_starpu_memory_handle_stats_cache_hit** (starpu_data_handle_t handle, unsigned node)
- void **_starpu_memory_handle_stats_loaded_shared** (starpu_data_handle_t handle, unsigned node)
- void **_starpu_memory_handle_stats_loaded_owner** (starpu_data_handle_t handle, unsigned node)
- void **_starpu_memory_handle_stats_shared_to_owner** (starpu_data_handle_t handle, unsigned node)
- void **_starpu_memory_handle_stats_invalidated** (starpu_data_handle_t handle, unsigned node)

## 6.46 mp_common.h File Reference

```
#include <semaphore.h>
#include <starpu.h>
#include <common/config.h>
#include <common/list.h>
#include <common/barrier.h>
#include <common/thread.h>
#include <datawizard/interfaces/data_interface.h>
#include <datawizard/copy_driver.h>
```

## 6.47 multiple_regression.h File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <core/perfmodel/perfmodel.h>
#include <starpu.h>
```

**Functions**

- int **_starpu_multiple_regression** (struct starpu_perfmodel_history_list ∗ptr, double ∗coeff, unsigned ncoeff, unsigned nparameters, const char ∗∗parameters_names, unsigned ∗∗combinations, const char ∗codelet_↩ name)

## 6.48 node_ops.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <datawizard/copy_driver.h>
```

**Data Structures**

- struct _starpu_node_ops

**Typedefs**

- typedef int(∗ **copy_interface_func_t**) (starpu_data_handle_t handle, void ∗src_interface, unsigned src_↵
  node, void ∗dst_interface, unsigned dst_node, struct _starpu_data_request ∗req)
- typedef int(∗ **copy_data_t**) (uintptr_t src_ptr, size_t src_offset, unsigned src_node, uintptr_t dst_ptr, size_t
  dst_offset, unsigned dst_node, size_t ssize, struct _starpu_async_channel ∗async_channel)
- typedef int(∗ **copy2d_data_t**) (uintptr_t src_ptr, size_t src_offset, unsigned src_node, uintptr_t dst_ptr, size↵
  _t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks, size_t ld_src, size_t ld_dst, struct _↵
  starpu_async_channel ∗async_channel)
- typedef int(∗ **copy3d_data_t**) (uintptr_t src_ptr, size_t src_offset, unsigned src_node, uintptr_t dst_ptr, size↵
  _t dst_offset, unsigned dst_node, size_t blocksize, size_t numblocks_1, size_t ld1_src, size_t ld1_dst, size_t
  numblocks_2, size_t ld2_src, size_t ld2_dst, struct _starpu_async_channel ∗async_channel)

**Functions**

- const char ∗ **_starpu_node_get_prefix** (enum starpu_node_kind kind)

## 6.49 openmp_runtime_support.h File Reference

```
#include <starpu.h>
#include <common/list.h>
#include <common/starpu_spinlock.h>
#include <common/uthash.h>
#include <ucontext.h>
```

**Data Structures**

- struct starpu_omp_numeric_place
- struct starpu_omp_place
- struct starpu_omp_data_environment_icvs
- struct starpu_omp_device_icvs
- struct starpu_omp_implicit_task_icvs
- struct starpu_omp_global_icvs
- struct starpu_omp_initial_icv_values
- struct starpu_omp_task_group
- struct starpu_omp_task_link
- struct starpu_omp_condition
- struct starpu_omp_critical

**Macros**

- #define _XOPEN_SOURCE
- #define STARPU_OMP_MAX_ACTIVE_LEVELS

**Enumerations**

- enum starpu_omp_place_name {
  **starpu_omp_place_undefined**, **starpu_omp_place_threads**, **starpu_omp_place_cores**, **starpu_omp↩_place_sockets**,
  **starpu_omp_place_numerical** }
- enum starpu_omp_task_state {
  **starpu_omp_task_state_clear**, **starpu_omp_task_state_preempted**, **starpu_omp_task_state_↩terminated**, **starpu_omp_task_state_zombie**,
  starpu_omp_task_state_target }
- enum **starpu_omp_task_wait_on** {
  **starpu_omp_task_wait_on_task_childs**, **starpu_omp_task_wait_on_region_tasks**, **starpu_omp_↩task_wait_on_barrier**, **starpu_omp_task_wait_on_group**,
  **starpu_omp_task_wait_on_critical**, **starpu_omp_task_wait_on_ordered**, **starpu_omp_task_wait_on↩_lock**, **starpu_omp_task_wait_on_nest_lock** }
- enum **starpu_omp_task_flags** { **STARPU_OMP_TASK_FLAGS_IMPLICIT**, **STARPU_OMP_TASK_FL↩AGS_UNDEFERRED**, **STARPU_OMP_TASK_FLAGS_FINAL**, **STARPU_OMP_TASK_FLAGS_UNTIED** }

**Variables**

- starpu_pthread_key_t **omp_thread_key**
- starpu_pthread_key_t **omp_task_key**

### 6.49.1 Data Structure Documentation

#### 6.49.1.1 struct starpu_omp_numeric_place

**Data Fields**

| | | |
|---|---|---|
| int | excluded_place | |
| int ∗ | included_numeric_items | |
| int | nb_included_numeric_items | |
| int ∗ | excluded_numeric_items | |
| int | nb_excluded_numeric_items | |

#### 6.49.1.2 struct starpu_omp_place

OpenMP place for thread afinity, defined by the OpenMP spec

**Data Fields**

| | | |
|---|---|---|
| int | abstract_name | |
| int | abstract_excluded | |
| int | abstract_length | |
| struct starpu_omp_numeric_place ∗ | numeric_places | |
| int | nb_numeric_places | |

#### 6.49.1.3 struct starpu_omp_data_environment_icvs

Internal Control Variables (ICVs) declared following OpenMP 4.0.0 spec section 2.3.1

**Data Fields**

| | | |
|---|---|---|
| int | dyn_var | parallel region icvs |
| int | nest_var | |
| int ∗ | nthreads_var | |

**Data Fields**

| | | |
|---:|:---|:---|
| int | thread_limit_var | nthreads_var ICV is a list |
| int | active_levels_var | |
| int | levels_var | |
| int ∗ | bind_var | |
| int | run_sched_var | bind_var ICV is a list loop region icvs |
| unsigned long long | run_sched_chunk_var | |
| int | default_device_var | program execution icvs |
| int | max_task_priority_var | |

### 6.49.1.4 struct starpu_omp_device_icvs

**Data Fields**

| | | |
|---:|:---|:---|
| int | max_active_levels_var | parallel region icvs |
| int | def_sched_var | loop region icvs |
| unsigned long long | def_sched_chunk_var | |
| int | stacksize_var | program execution icvs |
| int | wait_policy_var | |

### 6.49.1.5 struct starpu_omp_implicit_task_icvs

**Data Fields**

| | | |
|---:|:---|:---|
| int | place_partition_var | parallel region icvs |

### 6.49.1.6 struct starpu_omp_global_icvs

**Data Fields**

| | | |
|---:|:---|:---|
| int | cancel_var | program execution icvs |

### 6.49.1.7 struct starpu_omp_initial_icv_values

**Data Fields**

| | | |
|---:|:---|:---|
| int | dyn_var | |
| int | nest_var | |
| int ∗ | nthreads_var | |
| int | run_sched_var | |
| unsigned long long | run_sched_chunk_var | |
| int | def_sched_var | |
| unsigned long long | def_sched_chunk_var | |
| int ∗ | bind_var | |
| int | stacksize_var | |
| int | wait_policy_var | |
| int | thread_limit_var | |
| int | max_active_levels_var | |
| int | active_levels_var | |

**Data Fields**

| | | |
|---:|---|---|
| int | levels_var | |
| int | place_partition_var | |
| int | cancel_var | |
| int | default_device_var | |
| int | max_task_priority_var | |
| struct starpu_omp_place | places | not a real ICV, but needed to store the contents of OMP_PLACES |

**6.49.1.8   struct starpu_omp_task_group**

**Data Fields**

| | | |
|---:|---|---|
| int | descendent_task_count | |
| struct starpu_omp_task ∗ | leader_task | |
| struct starpu_omp_task_group ∗ | p_previous_task_group | |

**6.49.1.9   struct starpu_omp_task_link**

**Data Fields**

| | | |
|---:|---|---|
| struct starpu_omp_task ∗ | task | |
| struct starpu_omp_task_link ∗ | next | |

**6.49.1.10   struct starpu_omp_condition**

**Data Fields**

| | | |
|---:|---|---|
| struct starpu_omp_task_link ∗ | contention_list_head | |

**6.49.1.11   struct starpu_omp_critical**

**Data Fields**

| | | |
|---:|---|---|
| UT_hash_handle | hh | |
| struct _starpu_spinlock | lock | |
| unsigned | state | |
| struct starpu_omp_task_link ∗ | contention_list_head | |
| const char ∗ | name | |

## 6.49.2   Macro Definition Documentation

### 6.49.2.1   _XOPEN_SOURCE

`#define _XOPEN_SOURCE`
ucontexts have been deprecated as of POSIX 1-2004 _XOPEN_SOURCE required at least on OS/X
TODO: add detection in configure.ac

**6.49.2.2   STARPU_OMP_MAX_ACTIVE_LEVELS**

```
#define STARPU_OMP_MAX_ACTIVE_LEVELS
```
Arbitrary limit on the number of nested parallel sections

### 6.49.3   Enumeration Type Documentation

**6.49.3.1   starpu_omp_place_name**

```
enum starpu_omp_place_name
```
Possible abstract names for OpenMP places

**6.49.3.2   starpu_omp_task_state**

```
enum starpu_omp_task_state
```

**Enumerator**

| | |
|---|---|
| starpu_omp_task_state_target | target tasks are non-preemptible tasks, without dedicated stack and OpenMP Runtime Support context |

## 6.50   perfmodel.h File Reference

```
#include <common/config.h>
#include <starpu.h>
#include <core/task_bundle.h>
#include <stdio.h>
```

**Data Structures**

- struct _starpu_perfmodel_state

**Macros**

- #define _STARPU_PERFMODEL_VERSION

**Functions**

- char ∗ **_starpu_get_perf_model_dir_codelet** ()
- char ∗ **_starpu_get_perf_model_dir_bus** ()
- char ∗ **_starpu_get_perf_model_dir_debug** ()
- double **_starpu_history_based_job_expected_perf** (struct starpu_perfmodel ∗model, struct starpu_↵ perfmodel_arch ∗arch, struct _starpu_job ∗j, unsigned nimpl)
- void **_starpu_load_history_based_model** (struct starpu_perfmodel ∗model, unsigned scan_history)
- void **_starpu_init_and_load_perfmodel** (struct starpu_perfmodel ∗model)
- void **_starpu_initialize_registered_performance_models** (void)
- void **_starpu_deinitialize_registered_performance_models** (void)
- void **_starpu_deinitialize_performance_model** (struct starpu_perfmodel ∗model)
- double **_starpu_regression_based_job_expected_perf** (struct starpu_perfmodel ∗model, struct starpu↵ _perfmodel_arch ∗arch, struct _starpu_job ∗j, unsigned nimpl)
- double **_starpu_non_linear_regression_based_job_expected_perf** (struct starpu_perfmodel ∗model, struct starpu_perfmodel_arch ∗arch, struct _starpu_job ∗j, unsigned nimpl)

- double **_starpu_multiple_regression_based_job_expected_perf** (struct starpu_perfmodel ∗model, struct starpu_perfmodel_arch ∗arch, struct _starpu_job ∗j, unsigned nimpl)
- void **_starpu_update_perfmodel_history** (struct _starpu_job ∗j, struct starpu_perfmodel ∗model, struct starpu_perfmodel_arch ∗arch, unsigned cpuid, double measured, unsigned nimpl)
- int **_starpu_perfmodel_create_comb_if_needed** (struct starpu_perfmodel_arch ∗arch)
- void **_starpu_create_sampling_directory_if_needed** (void)
- void **_starpu_load_bus_performance_files** (void)
- void **_starpu_set_calibrate_flag** (unsigned val)
- unsigned **_starpu_get_calibrate_flag** (void)
- unsigned ∗ **_starpu_get_cuda_affinity_vector** (unsigned gpuid)
- unsigned ∗ **_starpu_get_opencl_affinity_vector** (unsigned gpuid)
- void **_starpu_save_bandwidth_and_latency_disk** (double bandwidth_write, double bandwidth_read, double latency_write, double latency_read, unsigned node, const char ∗name)
- void **_starpu_write_double** (FILE ∗f, const char ∗format, double val)
- int **_starpu_read_double** (FILE ∗f, char ∗format, double ∗val)
- void **_starpu_simgrid_get_platform_path** (int version, char ∗path, size_t maxlen)
- void **_starpu_perfmodel_realloc** (struct starpu_perfmodel ∗model, int nb)
- void **_starpu_free_arch_combs** (void)
- hwloc_topology_t **_starpu_perfmodel_get_hwtopology** ()

## Variables

- unsigned **_starpu_calibration_minimum**

### 6.50.1 Data Structure Documentation

#### 6.50.1.1 struct _starpu_perfmodel_state

**Data Fields**

| | | |
|---|---|---|
| struct starpu_perfmodel_per_arch ∗∗ | per_arch | |
| int ∗∗ | per_arch_is_set | |
| starpu_pthread_rwlock_t | model_rwlock | |
| int ∗ | nimpls | |
| int ∗ | nimpls_set | |
| int | ncombs | The number of combinations currently used by the model |
| int | ncombs_set | The number of combinations allocated in the array nimpls and ncombs |
| int ∗ | combs | |

### 6.50.2 Macro Definition Documentation

#### 6.50.2.1 _STARPU_PERFMODEL_VERSION

```
#define _STARPU_PERFMODEL_VERSION
```
Performance models files are stored in a directory whose name include the version of the performance model format. The version number is also written in the file itself. When updating the format, the variable _STARPU_PE↩ RFMODEL_VERSION should be updated. It is then possible to switch easily between differents versions of StarPU having different performance model formats.

## 6.51 prio_deque.h File Reference

```
#include <starpu.h>
#include <starpu_scheduler.h>
#include <core/task.h>
```

**Data Structures**

- struct _starpu_prio_deque

**Functions**

- static void **_starpu_prio_deque_init** (struct _starpu_prio_deque *pdeque)
- static void **_starpu_prio_deque_destroy** (struct _starpu_prio_deque *pdeque)
- static int _starpu_prio_deque_is_empty (struct _starpu_prio_deque *pdeque)
- static void **_starpu_prio_deque_erase** (struct _starpu_prio_deque *pdeque, struct starpu_task *task)
- static int _starpu_prio_deque_push_front_task (struct _starpu_prio_deque *pdeque, struct starpu_task *task)
- static int **_starpu_prio_deque_push_back_task** (struct _starpu_prio_deque *pdeque, struct starpu_task *task)
- static struct starpu_task * **_starpu_prio_deque_highest_task** (struct _starpu_prio_deque *pdeque)
- static struct starpu_task * _starpu_prio_deque_pop_task (struct _starpu_prio_deque *pdeque)
- static struct starpu_task * **_starpu_prio_deque_pop_back_task** (struct _starpu_prio_deque *pdeque)
- static int **_starpu_prio_deque_pop_this_task** (struct _starpu_prio_deque *pdeque, int workerid, struct starpu_task *task)
- struct starpu_task * _starpu_prio_deque_pop_task_for_worker (struct _starpu_prio_deque *, int workerid, int *skipped)
- struct starpu_task * _starpu_prio_deque_deque_task_for_worker (struct _starpu_prio_deque *, int workerid, int *skipped)
- struct starpu_task * **_starpu_prio_deque_deque_first_ready_task** (struct _starpu_prio_deque *, unsigned workerid)

### 6.51.1 Data Structure Documentation

#### 6.51.1.1 struct _starpu_prio_deque

**Data Fields**

| struct starpu_task_prio_list | list | |
|---:|---|---|
| unsigned | ntasks | |
| unsigned | nprocessed | |
| double | exp_start | |
| double | exp_end | |
| double | exp_len | |

### 6.51.2 Function Documentation

#### 6.51.2.1 _starpu_prio_deque_is_empty()

```
static int _starpu_prio_deque_is_empty (
            struct _starpu_prio_deque * pdeque ) [inline], [static]
```

return 0 iff the struct _starpu_prio_deque is not empty

**6.51.2.2 _starpu_prio_deque_push_front_task()**

```
static int _starpu_prio_deque_push_front_task (
            struct _starpu_prio_deque * pdeque,
            struct starpu_task * task ) [inline], [static]
```
push a task in O(lg(nb priorities))

**6.51.2.3 _starpu_prio_deque_pop_task()**

```
static struct starpu_task* _starpu_prio_deque_pop_task (
            struct _starpu_prio_deque * pdeque ) [static]
```
all _starpu_prio_deque_pop/deque_task function return a task or a NULL pointer if none are available in O(lg(nb priorities))

**6.51.2.4 _starpu_prio_deque_pop_task_for_worker()**

```
struct starpu_task* _starpu_prio_deque_pop_task_for_worker (
            struct _starpu_prio_deque * ,
            int workerid,
            int * skipped )
```
return a task that can be executed by workerid

**6.51.2.5 _starpu_prio_deque_deque_task_for_worker()**

```
struct starpu_task* _starpu_prio_deque_deque_task_for_worker (
            struct _starpu_prio_deque * ,
            int workerid,
            int * skipped )
```
return a task that can be executed by workerid

## 6.52 prio_list.h File Reference

```
#include <common/rbtree.h>
```

**Macros**

- #define **PRIO_LIST_INLINE**
- #define **PRIO_struct**
- #define **PRIO_LIST_CREATE_TYPE**(ENAME, PRIOFIELD)

## 6.53 profiling.h File Reference

```
#include <starpu.h>
#include <starpu_profiling.h>
#include <starpu_util.h>
#include <common/config.h>
```

**Functions**

- struct starpu_profiling_task_info ∗ _starpu_allocate_profiling_info_if_needed (struct starpu_task ∗task)
- void _starpu_worker_update_profiling_info_executing (int workerid, struct timespec ∗executing_time, int executed_tasks, uint64_t used_cycles, uint64_t stall_cycles, double consumed_energy, double flops)
- void _starpu_worker_restart_sleeping (int workerid)
- void _starpu_worker_stop_sleeping (int workerid)
- void _starpu_worker_register_executing_start_date (int workerid, struct timespec ∗executing_start)

- void _starpu_worker_register_executing_end (int workerid)
- void _starpu_initialize_busid_matrix (void)
- int _starpu_register_bus (int src_node, int dst_node)
- void _starpu_bus_update_profiling_info (int src_node, int dst_node, size_t size)
- void **_starpu_profiling_set_task_push_start_time** (struct starpu_task ∗task)
- void **_starpu_profiling_set_task_push_end_time** (struct starpu_task ∗task)
- void _starpu_profiling_init (void)
- void _starpu_profiling_start (void)
- void **_starpu_profiling_terminate** (void)

### 6.53.1 Function Documentation

#### 6.53.1.1 _starpu_allocate_profiling_info_if_needed()

```
struct starpu_profiling_task_info* _starpu_allocate_profiling_info_if_needed (
            struct starpu_task * task )
```

Create a task profiling info structure (with the proper time stamps) in case profiling is enabled.

#### 6.53.1.2 _starpu_worker_update_profiling_info_executing()

```
void _starpu_worker_update_profiling_info_executing (
            int workerid,
            struct timespec * executing_time,
            int executed_tasks,
            uint64_t used_cycles,
            uint64_t stall_cycles,
            double consumed_energy,
            double flops )
```

Update the per-worker profiling info after a task (or more) was executed. This tells StarPU how much time was spent doing computation.

#### 6.53.1.3 _starpu_worker_restart_sleeping()

```
void _starpu_worker_restart_sleeping (
            int workerid )
```

Record the date when the worker started to sleep. This permits to measure how much time was spent sleeping.

#### 6.53.1.4 _starpu_worker_stop_sleeping()

```
void _starpu_worker_stop_sleeping (
            int workerid )
```

Record the date when the worker stopped sleeping. This permits to measure how much time was spent sleeping.

#### 6.53.1.5 _starpu_worker_register_executing_start_date()

```
void _starpu_worker_register_executing_start_date (
            int workerid,
            struct timespec * executing_start )
```

Record the date when the worker started to execute a piece of code. This permits to measure how much time was really spent doing computation at the end of the codelet.

#### 6.53.1.6 _starpu_worker_register_executing_end()

```
void _starpu_worker_register_executing_end (
            int workerid )
```

Record that the worker is not executing any more.

### 6.53.1.7 _starpu_initialize_busid_matrix()

```
void _starpu_initialize_busid_matrix (
            void )
```
When StarPU is initialized, a matrix describing all the bus between memory nodes is created: it indicates whether there is a physical link between two memory nodes or not. This matrix should contain the identifier of the bus between two nodes or -1 in case there is no link.

### 6.53.1.8 _starpu_register_bus()

```
int _starpu_register_bus (
            int src_node,
            int dst_node )
```
Tell StarPU that there exists a link between the two memory nodes. This function returns the identifier associated to the bus which can be used to retrieve profiling information about the bus activity later on.

### 6.53.1.9 _starpu_bus_update_profiling_info()

```
void _starpu_bus_update_profiling_info (
            int src_node,
            int dst_node,
            size_t size )
```
Tell StarPU that "size" bytes were transferred between the two specified memory nodes.

### 6.53.1.10 _starpu_profiling_init()

```
void _starpu_profiling_init (
            void )
```
This function needs to be called before other starpu_profile_∗ functions

### 6.53.1.11 _starpu_profiling_start()

```
void _starpu_profiling_start (
            void )
```
This function starts profiling if the STARPU_PROFILING environment variable was set

## 6.54 progress_hook.h File Reference

**Functions**

- void **_starpu_init_progression_hooks** (void)
- unsigned **_starpu_execute_registered_progression_hooks** (void)

## 6.55 rbtree.h File Reference

```
#include <stddef.h>
#include <assert.h>
#include <stdint.h>
#include <sys/types.h>
#include "rbtree_i.h"
```

**Macros**

- #define **MACRO_BEGIN**
- #define **MACRO_END**
- #define **STARPU_RBTREE_LEFT**
- #define **STARPU_RBTREE_RIGHT**

- #define [STARPU_RBTREE_INITIALIZER](#)
- #define [starpu_rbtree_entry](#)(node, type, member)
- #define [starpu_rbtree_lookup](#)(tree, key, cmp_fn)
- #define [starpu_rbtree_lookup_nearest](#)(tree, key, cmp_fn, dir)
- #define [starpu_rbtree_insert](#)(tree, node, cmp_fn)
- #define [starpu_rbtree_lookup_slot](#)(tree, key, cmp_fn, slot)
- #define [starpu_rbtree_first](#)(tree)
- #define [starpu_rbtree_last](#)(tree)
- #define [starpu_rbtree_prev](#)(node)
- #define [starpu_rbtree_next](#)(node)
- #define [starpu_rbtree_for_each_remove](#)(tree, node, tmp)

## Functions

- static void [starpu_rbtree_init](#) ([struct starpu_rbtree](#) ∗tree)
- static void [starpu_rbtree_node_init](#) ([struct starpu_rbtree_node](#) ∗node)
- static int **starpu_rbtree_node_unlinked** (const [struct starpu_rbtree_node](#) ∗node)
- static int [starpu_rbtree_empty](#) (const [struct starpu_rbtree](#) ∗tree)
- static void [starpu_rbtree_insert_slot](#) ([struct starpu_rbtree](#) ∗tree, uintptr_t slot, [struct starpu_rbtree_node](#) ∗node)
- void [starpu_rbtree_remove](#) ([struct starpu_rbtree](#) ∗tree, [struct starpu_rbtree_node](#) ∗node)

### 6.55.1 Macro Definition Documentation

#### 6.55.1.1 STARPU_RBTREE_INITIALIZER

```
#define STARPU_RBTREE_INITIALIZER
```
Static tree initializer.

#### 6.55.1.2 starpu_rbtree_entry

```
#define starpu_rbtree_entry(
            node,
            type,
            member )
```
Macro that evaluates to the address of the structure containing the given node based on the given type and member.

#### 6.55.1.3 starpu_rbtree_lookup

```
#define starpu_rbtree_lookup(
            tree,
            key,
            cmp_fn )
```
Look up a node in a tree.
Note that implementing the lookup algorithm as a macro gives two benefits: First, it avoids the overhead of a callback function. Next, the type of the cmp_fn parameter isn't rigid. The only guarantee offered by this implementation is that the key parameter is the first parameter given to cmp_fn. This way, users can pass only the value they need for comparison instead of e.g. allocating a full structure on the stack.
See [starpu_rbtree_insert()](#).

### 6.55.1.4 starpu_rbtree_lookup_nearest

```
#define starpu_rbtree_lookup_nearest(
            tree,
            key,
            cmp_fn,
            dir )
```
Look up a node or one of its nearest nodes in a tree.

This macro essentially acts as starpu_rbtree_lookup() but if no entry matched the key, an additional step is performed to obtain the next or previous node, depending on the direction (left or right).

The constraints that apply to the key parameter are the same as for starpu_rbtree_lookup().

### 6.55.1.5 starpu_rbtree_insert

```
#define starpu_rbtree_insert(
            tree,
            node,
            cmp_fn )
```
Insert a node in a tree.

This macro performs a standard lookup to obtain the insertion point of the given node in the tree (it is assumed that the inserted node never compares equal to any other entry in the tree) and links the node. It then checks red-black rules violations, and rebalances the tree if necessary.

Unlike starpu_rbtree_lookup(), the cmp_fn parameter must compare two complete entries, so it is suggested to use two different comparison inline functions, such as myobj_cmp_lookup() and myobj_cmp_insert(). There is no guarantee about the order of the nodes given to the comparison function.

See starpu_rbtree_lookup().

### 6.55.1.6 starpu_rbtree_lookup_slot

```
#define starpu_rbtree_lookup_slot(
            tree,
            key,
            cmp_fn,
            slot )
```
Look up a node/slot pair in a tree.

This macro essentially acts as starpu_rbtree_lookup() but in addition to a node, it also returns a slot, which identifies an insertion point in the tree. If the returned node is null, the slot can be used by starpu_rbtree_insert_slot() to insert without the overhead of an additional lookup. The slot is a simple uintptr_t integer.

The constraints that apply to the key parameter are the same as for starpu_rbtree_lookup().

### 6.55.1.7 starpu_rbtree_first

```
#define starpu_rbtree_first(
            tree )
```
Return the first node of a tree.

### 6.55.1.8 starpu_rbtree_last

```
#define starpu_rbtree_last(
            tree )
```
Return the last node of a tree.

### 6.55.1.9 starpu_rbtree_prev

```
#define starpu_rbtree_prev(
            node )
```
Return the node previous to the given node.

**6.55.1.10 starpu_rbtree_next**

```
#define starpu_rbtree_next(
            node )
```
Return the node next to the given node.

**6.55.1.11 starpu_rbtree_for_each_remove**

```
#define starpu_rbtree_for_each_remove(
            tree,
            node,
            tmp )
```
Forge a loop to process all nodes of a tree, removing them when visited.

This macro can only be used to destroy a tree, so that the resources used by the entries can be released by the user. It basically removes all nodes without doing any color checking.

After completion, all nodes and the tree root member are stale.

**6.55.2 Function Documentation**

**6.55.2.1 starpu_rbtree_init()**

```
static void starpu_rbtree_init (
            struct starpu_rbtree * tree )  [inline], [static]
```
Initialize a tree.

**6.55.2.2 starpu_rbtree_node_init()**

```
static void starpu_rbtree_node_init (
            struct starpu_rbtree_node * node )  [inline], [static]
```
Initialize a node.

A node is in no tree when its parent points to itself.

**6.55.2.3 starpu_rbtree_empty()**

```
static int starpu_rbtree_empty (
            const struct starpu_rbtree * tree )  [inline], [static]
```
Return true if tree is empty.

**6.55.2.4 starpu_rbtree_insert_slot()**

```
static void starpu_rbtree_insert_slot (
            struct starpu_rbtree * tree,
            uintptr_t slot,
            struct starpu_rbtree_node * node )  [inline], [static]
```
Insert a node at an insertion point in a tree.

This macro essentially acts as starpu_rbtree_insert() except that it doesn't obtain the insertion point with a standard lookup. The insertion point is obtained by calling starpu_rbtree_lookup_slot(). In addition, the new node must not compare equal to an existing node in the tree (i.e. the slot must denote a null node).

**6.55.2.5 starpu_rbtree_remove()**

```
void starpu_rbtree_remove (
            struct starpu_rbtree * tree,
            struct starpu_rbtree_node * node )
```
Remove a node from a tree.

After completion, the node is stale.

## 6.56 rbtree_i.h File Reference

```
#include <assert.h>
```

**Data Structures**

- struct starpu_rbtree_node
- struct starpu_rbtree

**Macros**

- #define STARPU_RBTREE_COLOR_MASK
- #define **STARPU_RBTREE_PARENT_MASK**
- #define STARPU_RBTREE_COLOR_RED
- #define **STARPU_RBTREE_COLOR_BLACK**
- #define STARPU_RBTREE_SLOT_INDEX_MASK
- #define **STARPU_RBTREE_SLOT_PARENT_MASK**

**Functions**

- static int starpu_rbtree_check_alignment (const struct starpu_rbtree_node ∗node)
- static int starpu_rbtree_check_index (int index)
- static int starpu_rbtree_d2i (int diff)
- static struct starpu_rbtree_node ∗ starpu_rbtree_parent (const struct starpu_rbtree_node ∗node)
- static uintptr_t starpu_rbtree_slot (struct starpu_rbtree_node ∗parent, int index)
- static struct starpu_rbtree_node ∗ starpu_rbtree_slot_parent (uintptr_t slot)
- static int starpu_rbtree_slot_index (uintptr_t slot)
- void starpu_rbtree_insert_rebalance (struct starpu_rbtree ∗tree, struct starpu_rbtree_node ∗parent, int index, struct starpu_rbtree_node ∗node)
- struct starpu_rbtree_node ∗ starpu_rbtree_nearest (struct starpu_rbtree_node ∗parent, int index, int direction)
- struct starpu_rbtree_node ∗ starpu_rbtree_firstlast (const struct starpu_rbtree ∗tree, int direction)
- struct starpu_rbtree_node ∗ starpu_rbtree_walk (struct starpu_rbtree_node ∗node, int direction)
- struct starpu_rbtree_node ∗ starpu_rbtree_postwalk_deepest (const struct starpu_rbtree ∗tree)
- struct starpu_rbtree_node ∗ starpu_rbtree_postwalk_unlink (struct starpu_rbtree_node ∗node)

### 6.56.1 Data Structure Documentation

#### 6.56.1.1 struct starpu_rbtree_node

Red-black node structure.
To reduce the number of branches and the instruction cache footprint, the left and right child pointers are stored in an array, and the symmetry of most tree operations is exploited by using left/right variables when referring to children.
In addition, this implementation assumes that all nodes are 4-byte aligned, so that the least significant bit of the parent member can be used to store the color of the node. This is true for all modern 32 and 64 bits architectures, as long as the nodes aren't embedded in structures with special alignment constraints such as member packing.

**Data Fields**

| | | |
|---|---|---|
| uintptr_t | parent | |
| struct starpu_rbtree_node ∗ | children[2] | |

**6.56.1.2 struct starpu_rbtree**

Red-black tree structure.

**Data Fields**

| struct starpu_rbtree_node * | root | |
| --- | --- | --- |

## 6.56.2 Macro Definition Documentation

**6.56.2.1 STARPU_RBTREE_COLOR_MASK**

```
#define STARPU_RBTREE_COLOR_MASK
```
Masks applied on the parent member of a node to obtain either the color or the parent address.

**6.56.2.2 STARPU_RBTREE_COLOR_RED**

```
#define STARPU_RBTREE_COLOR_RED
```
Node colors.

**6.56.2.3 STARPU_RBTREE_SLOT_INDEX_MASK**

```
#define STARPU_RBTREE_SLOT_INDEX_MASK
```
Masks applied on slots to obtain either the child index or the parent address.

## 6.56.3 Function Documentation

**6.56.3.1 starpu_rbtree_check_alignment()**

```
static int starpu_rbtree_check_alignment (
            const struct starpu_rbtree_node * node ) [inline], [static]
```
Return true if the given pointer is suitably aligned.

**6.56.3.2 starpu_rbtree_check_index()**

```
static int starpu_rbtree_check_index (
            int index ) [inline], [static]
```
Return true if the given index is a valid child index.

**6.56.3.3 starpu_rbtree_d2i()**

```
static int starpu_rbtree_d2i (
            int diff ) [inline], [static]
```
Convert the result of a comparison into an index in the children array (0 or 1).
This function is mostly used when looking up a node.

**6.56.3.4 starpu_rbtree_parent()**

```
static struct starpu_rbtree_node* starpu_rbtree_parent (
            const struct starpu_rbtree_node * node ) [static]
```
Return the parent of a node.

**6.56.3.5 starpu_rbtree_slot()**

```
static uintptr_t starpu_rbtree_slot (
            struct starpu_rbtree_node * parent,
            int index ) [inline], [static]
```
Translate an insertion point into a slot.

**6.56.3.6 starpu_rbtree_slot_parent()**

```
static struct starpu_rbtree_node* starpu_rbtree_slot_parent (
            uintptr_t slot ) [static]
```
Extract the parent address from a slot.

**6.56.3.7 starpu_rbtree_slot_index()**

```
static int starpu_rbtree_slot_index (
            uintptr_t slot ) [inline], [static]
```
Extract the index from a slot.

**6.56.3.8 starpu_rbtree_insert_rebalance()**

```
void starpu_rbtree_insert_rebalance (
            struct starpu_rbtree * tree,
            struct starpu_rbtree_node * parent,
            int index,
            struct starpu_rbtree_node * node )
```
Insert a node in a tree, rebalancing it if necessary.

The index parameter is the index in the children array of the parent where the new node is to be inserted. It is ignored if the parent is null.

This function is intended to be used by the starpu_rbtree_insert() macro only.

**6.56.3.9 starpu_rbtree_nearest()**

```
struct starpu_rbtree_node* starpu_rbtree_nearest (
            struct starpu_rbtree_node * parent,
            int index,
            int direction )
```
Return the previous or next node relative to a location in a tree.

The parent and index parameters define the location, which can be empty. The direction parameter is either S↩
TARPU_RBTREE_LEFT (to obtain the previous node) or STARPU_RBTREE_RIGHT (to obtain the next one).

**6.56.3.10 starpu_rbtree_firstlast()**

```
struct starpu_rbtree_node* starpu_rbtree_firstlast (
            const struct starpu_rbtree * tree,
            int direction )
```
Return the first or last node of a tree.

The direction parameter is either STARPU_RBTREE_LEFT (to obtain the first node) or STARPU_RBTREE_RIGHT (to obtain the last one).

**6.56.3.11 starpu_rbtree_walk()**

```
struct starpu_rbtree_node* starpu_rbtree_walk (
            struct starpu_rbtree_node * node,
            int direction )
```
Return the node next to, or previous to the given node.

The direction parameter is either STARPU_RBTREE_LEFT (to obtain the previous node) or STARPU_RBTREE↩
_RIGHT (to obtain the next one).

**6.56.3.12 starpu_rbtree_postwalk_deepest()**

struct starpu_rbtree_node* starpu_rbtree_postwalk_deepest (

    const struct starpu_rbtree * *tree* )

Return the left-most deepest node of a tree, which is the starting point of the postorder traversal performed by starpu_rbtree_for_each_remove().

**6.56.3.13 starpu_rbtree_postwalk_unlink()**

struct starpu_rbtree_node* starpu_rbtree_postwalk_unlink (

    struct starpu_rbtree_node * *node* )

Unlink a node from its tree and return the next (right) node in postorder.

## 6.57 regression.h File Reference

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <core/perfmodel/perfmodel.h>
#include <starpu.h>
```

**Functions**

- int **_starpu_regression_non_linear_power** (struct starpu_perfmodel_history_list ∗ptr, double ∗a, double ∗b, double ∗c)

## 6.58 rwlock.h File Reference

```
#include <stdint.h>
#include <starpu.h>
```

**Data Structures**

- struct _starpu_rw_lock

**Functions**

- void _starpu_init_rw_lock (struct _starpu_rw_lock ∗lock)
- void _starpu_take_rw_lock_write (struct _starpu_rw_lock ∗lock)
- void _starpu_take_rw_lock_read (struct _starpu_rw_lock ∗lock)
- int _starpu_take_rw_lock_write_try (struct _starpu_rw_lock ∗lock)
- int _starpu_take_rw_lock_read_try (struct _starpu_rw_lock ∗lock)
- void _starpu_release_rw_lock (struct _starpu_rw_lock ∗lock)

**6.58.1 Data Structure Documentation**

**6.58.1.1 struct _starpu_rw_lock**

Dummy implementation of a RW-lock using a spinlock.

**Data Fields**

| uint32_t | busy | |
|---|---|---|
| uint8_t | writer | |
| uint16_t | readercnt | |

**6.58.2   Function Documentation**

**6.58.2.1   _starpu_init_rw_lock()**

```
void _starpu_init_rw_lock (
            struct _starpu_rw_lock * lock )
```
Initialize the RW-lock

**6.58.2.2   _starpu_take_rw_lock_write()**

```
void _starpu_take_rw_lock_write (
            struct _starpu_rw_lock * lock )
```
Grab the RW-lock in a write mode

**6.58.2.3   _starpu_take_rw_lock_read()**

```
void _starpu_take_rw_lock_read (
            struct _starpu_rw_lock * lock )
```
Grab the RW-lock in a read mode

**6.58.2.4   _starpu_take_rw_lock_write_try()**

```
int _starpu_take_rw_lock_write_try (
            struct _starpu_rw_lock * lock )
```
Try to grab the RW-lock in a write mode. Returns 0 in case of success, -1 otherwise.

**6.58.2.5   _starpu_take_rw_lock_read_try()**

```
int _starpu_take_rw_lock_read_try (
            struct _starpu_rw_lock * lock )
```
Try to grab the RW-lock in a read mode. Returns 0 in case of success, -1 otherwise.

**6.58.2.6   _starpu_release_rw_lock()**

```
void _starpu_release_rw_lock (
            struct _starpu_rw_lock * lock )
```
Unlock the RW-lock.

## 6.59   sched_component.h File Reference

```
#include <starpu_sched_component.h>
```

**Functions**

- void _starpu_sched_component_lock_all_workers (void)
- void **_starpu_sched_component_unlock_all_workers** (void)
- void **_starpu_sched_component_workers_destroy** (void)
- struct _starpu_worker ∗ **_starpu_sched_component_worker_get_worker** (struct starpu_sched_↩
  component ∗)
- struct starpu_bitmap ∗ **_starpu_get_worker_mask** (unsigned sched_ctx_id)

**6.59.1   Function Documentation**

**6.59.1.1 _starpu_sched_component_lock_all_workers()**

```
void _starpu_sched_component_lock_all_workers (
            void  )
```

lock and unlock drivers for modifying schedulers

## 6.60 sched_ctx.h File Reference

```
#include <starpu.h>
#include <starpu_sched_ctx.h>
#include <starpu_sched_ctx_hypervisor.h>
#include <starpu_scheduler.h>
#include <common/config.h>
#include <common/barrier_counter.h>
#include <common/utils.h>
#include <profiling/profiling.h>
#include <semaphore.h>
#include <core/task.h>
#include "sched_ctx_list.h"
#include <hwloc.h>
```

**Data Structures**

- struct _starpu_sched_ctx
- struct _starpu_ctx_change

**Macros**

- #define **NO_RESIZE**
- #define **REQ_RESIZE**
- #define **DO_RESIZE**
- #define **STARPU_GLOBAL_SCHED_CTX**
- #define **STARPU_NMAXSMS**
- #define **_starpu_sched_ctx_get_sched_ctx_for_worker_and_job**(w, j)
- #define **STARPU_SCHED_CTX_CHECK_LOCK**(sched_ctx_id)

**Functions**

- void _starpu_init_all_sched_ctxs (struct _starpu_machine_config ∗config)
- struct _starpu_sched_ctx ∗ _starpu_create_sched_ctx (struct starpu_sched_policy ∗policy, int ∗workerid, int nworkerids, unsigned is_init_sched, const char ∗sched_name, int min_prio_set, int min_prio, int max_←prio_set, int max_prio, unsigned awake_workers, void(∗sched_policy_init)(unsigned), void ∗user_data, int nsub_ctxs, int ∗sub_ctxs, int nsms)
- void _starpu_delete_all_sched_ctxs ()
- int _starpu_wait_for_all_tasks_of_sched_ctx (unsigned sched_ctx_id)
- int _starpu_wait_for_n_submitted_tasks_of_sched_ctx (unsigned sched_ctx_id, unsigned n)
- void _starpu_decrement_nsubmitted_tasks_of_sched_ctx (unsigned sched_ctx_id)
- void **_starpu_increment_nsubmitted_tasks_of_sched_ctx** (unsigned sched_ctx_id)
- int **_starpu_get_nsubmitted_tasks_of_sched_ctx** (unsigned sched_ctx_id)
- int **_starpu_check_nsubmitted_tasks_of_sched_ctx** (unsigned sched_ctx_id)
- void **_starpu_decrement_nready_tasks_of_sched_ctx** (unsigned sched_ctx_id, double ready_flops)
- unsigned **_starpu_increment_nready_tasks_of_sched_ctx** (unsigned sched_ctx_id, double ready_flops, struct starpu_task ∗task)
- int **_starpu_wait_for_no_ready_of_sched_ctx** (unsigned sched_ctx_id)
- int _starpu_get_index_in_ctx_of_workerid (unsigned sched_ctx, unsigned workerid)

- starpu_pthread_mutex_t ∗ _starpu_get_sched_mutex (struct _starpu_sched_ctx ∗sched_ctx, int worker)
- int _starpu_get_workers_of_sched_ctx (unsigned sched_ctx_id, int ∗pus, enum starpu_worker_archtype arch)
- void _starpu_worker_gets_out_of_ctx (unsigned sched_ctx_id, struct _starpu_worker ∗worker)
- unsigned _starpu_worker_belongs_to_a_sched_ctx (int workerid, unsigned sched_ctx_id)
- unsigned _starpu_sched_ctx_last_worker_awake (struct _starpu_worker ∗worker)
- unsigned _starpu_sched_ctx_get_current_context ()
- int _starpu_workers_able_to_execute_task (struct starpu_task ∗task, struct _starpu_sched_ctx ∗sched_ctx)
- void **_starpu_fetch_tasks_from_empty_ctx_list** (struct _starpu_sched_ctx ∗sched_ctx)
- unsigned **_starpu_sched_ctx_allow_hypervisor** (unsigned sched_ctx_id)
- struct starpu_perfmodel_arch ∗ **_starpu_sched_ctx_get_perf_archtype** (unsigned sched_ctx)
- void _starpu_sched_ctx_post_exec_task_cb (int workerid, struct starpu_task ∗task, size_t data_size, uint32_t footprint)
- void **starpu_sched_ctx_add_combined_workers** (int ∗combined_workers_to_add, unsigned n_↩ combined_workers_to_add, unsigned sched_ctx_id)
- struct _starpu_sched_ctx ∗ __starpu_sched_ctx_get_sched_ctx_for_worker_and_job (struct _starpu_worker ∗worker, struct _starpu_job ∗j)
- static struct _starpu_sched_ctx ∗ **_starpu_get_sched_ctx_struct** (unsigned id)
- static int **_starpu_sched_ctx_check_write_locked** (unsigned sched_ctx_id)
- static void **_starpu_sched_ctx_lock_write** (unsigned sched_ctx_id)
- static void **_starpu_sched_ctx_unlock_write** (unsigned sched_ctx_id)
- static void **_starpu_sched_ctx_lock_read** (unsigned sched_ctx_id)
- static void **_starpu_sched_ctx_unlock_read** (unsigned sched_ctx_id)
- static unsigned **_starpu_sched_ctx_worker_is_master_for_child_ctx** (unsigned sched_ctx_id, unsigned workerid, struct starpu_task ∗task)
- void _starpu_worker_apply_deferred_ctx_changes (void)

### 6.60.1 Data Structure Documentation

#### 6.60.1.1 struct _starpu_ctx_change

per-worker list of deferred ctx_change ops

**Data Fields**

| int | sched_ctx_id | |
|---|---|---|
| int | op | |
| int | nworkers_to_notify | |
| int ∗ | workerids_to_notify | |
| int | nworkers_to_change | |
| int ∗ | workerids_to_change | |

### 6.60.2 Function Documentation

#### 6.60.2.1 _starpu_init_all_sched_ctxs()

```
void _starpu_init_all_sched_ctxs (
            struct _starpu_machine_config * config )
```

init sched_ctx_id of all contextes

#### 6.60.2.2 _starpu_create_sched_ctx()

```
struct _starpu_sched_ctx* _starpu_create_sched_ctx (
```

```
                struct starpu_sched_policy * policy,
                int * workerid,
                int nworkerids,
                unsigned is_init_sched,
                const char * sched_name,
                int min_prio_set,
                int min_prio,
                int max_prio_set,
                int max_prio,
                unsigned awake_workers,
                void(*)(unsigned) sched_policy_init,
                void * user_data,
                int nsub_ctxs,
                int * sub_ctxs,
                int nsms )
```
allocate all structures belonging to a context

### 6.60.2.3 _starpu_delete_all_sched_ctxs()

```
void _starpu_delete_all_sched_ctxs ( )
```
delete all sched_ctx

### 6.60.2.4 _starpu_wait_for_all_tasks_of_sched_ctx()

```
int _starpu_wait_for_all_tasks_of_sched_ctx (
                unsigned sched_ctx_id )
```
This function waits until all the tasks that were already submitted to a specific context have been executed.

### 6.60.2.5 _starpu_wait_for_n_submitted_tasks_of_sched_ctx()

```
int _starpu_wait_for_n_submitted_tasks_of_sched_ctx (
                unsigned sched_ctx_id,
                unsigned n )
```
This function waits until at most n tasks are still submitted.

### 6.60.2.6 _starpu_decrement_nsubmitted_tasks_of_sched_ctx()

```
void _starpu_decrement_nsubmitted_tasks_of_sched_ctx (
                unsigned sched_ctx_id )
```
In order to implement starpu_wait_for_all_tasks_of_ctx, we keep track of the number of task currently submitted to the context

### 6.60.2.7 _starpu_get_index_in_ctx_of_workerid()

```
int _starpu_get_index_in_ctx_of_workerid (
                unsigned sched_ctx,
                unsigned workerid )
```
Return the corresponding index of the workerid in the ctx table

### 6.60.2.8 _starpu_get_sched_mutex()

```
starpu_pthread_mutex_t* _starpu_get_sched_mutex (
                struct _starpu_sched_ctx * sched_ctx,
                int worker )
```
Get the mutex corresponding to the global workerid

### 6.60.2.9 _starpu_get_workers_of_sched_ctx()

```
int _starpu_get_workers_of_sched_ctx (
                unsigned sched_ctx_id,
```

```
            int * pus,
            enum starpu_worker_archtype arch )
```
Get workers belonging to a certain context, it returns the number of workers take care: no mutex taken, the list of workers might not be updated

**6.60.2.10 _starpu_worker_gets_out_of_ctx()**

```
void _starpu_worker_gets_out_of_ctx (
            unsigned sched_ctx_id,
            struct _starpu_worker * worker )
```
Let the worker know it does not belong to the context and that it should stop poping from it

**6.60.2.11 _starpu_worker_belongs_to_a_sched_ctx()**

```
unsigned _starpu_worker_belongs_to_a_sched_ctx (
            int workerid,
            unsigned sched_ctx_id )
```
Check if the worker belongs to another sched_ctx

**6.60.2.12 _starpu_sched_ctx_last_worker_awake()**

```
unsigned _starpu_sched_ctx_last_worker_awake (
            struct _starpu_worker * worker )
```
indicates wheather this worker should go to sleep or not (if it is the last one awake in a context he should better keep awake)

**6.60.2.13 _starpu_sched_ctx_get_current_context()**

```
unsigned _starpu_sched_ctx_get_current_context ( )
```
If starpu_sched_ctx_set_context() has been called, returns the context id set by its last call, or the id of the initial context

**6.60.2.14 _starpu_workers_able_to_execute_task()**

```
int _starpu_workers_able_to_execute_task (
            struct starpu_task * task,
            struct _starpu_sched_ctx * sched_ctx )
```
verify that some worker can execute a certain task

**6.60.2.15 _starpu_sched_ctx_post_exec_task_cb()**

```
void _starpu_sched_ctx_post_exec_task_cb (
            int workerid,
            struct starpu_task * task,
            size_t data_size,
            uint32_t footprint )
```
Notifies the hypervisor that a tasks was poped from the workers' list

**6.60.2.16 __starpu_sched_ctx_get_sched_ctx_for_worker_and_job()**

```
struct _starpu_sched_ctx* __starpu_sched_ctx_get_sched_ctx_for_worker_and_job (
            struct _starpu_worker * worker,
            struct _starpu_job * j )
```
if the worker is the master of a parallel context, and the job is meant to be executed on this parallel context, return a pointer to the context

**6.60.2.17 _starpu_worker_apply_deferred_ctx_changes()**

```
void _starpu_worker_apply_deferred_ctx_changes (
            void  )
```

Go through the list of deferred ctx changes of the current worker and apply any ctx change operation found until the list is empty

## 6.61 sched_ctx_list.h File Reference

**Data Structures**

- struct _starpu_sched_ctx_list
- struct _starpu_sched_ctx_elt
- struct _starpu_sched_ctx_list_iterator

**Functions**

- struct _starpu_sched_ctx_elt ∗ _starpu_sched_ctx_elt_find (struct _starpu_sched_ctx_list ∗list, unsigned sched_ctx)
- void **_starpu_sched_ctx_elt_ensure_consistency** (struct _starpu_sched_ctx_list ∗list, unsigned sched←-_ctx)
- void **_starpu_sched_ctx_elt_init** (struct _starpu_sched_ctx_elt ∗elt, unsigned sched_ctx)
- struct _starpu_sched_ctx_elt ∗ **_starpu_sched_ctx_elt_add_after** (struct _starpu_sched_ctx_list ∗list, un-signed sched_ctx)
- struct _starpu_sched_ctx_elt ∗ **_starpu_sched_ctx_elt_add_before** (struct _starpu_sched_ctx_list ∗list, unsigned sched_ctx)
- struct _starpu_sched_ctx_elt ∗ **_starpu_sched_ctx_elt_add** (struct _starpu_sched_ctx_list ∗list, unsigned sched_ctx)
- void **_starpu_sched_ctx_elt_remove** (struct _starpu_sched_ctx_list ∗list, struct _starpu_sched_ctx_elt ∗elt)
- int **_starpu_sched_ctx_elt_exists** (struct _starpu_sched_ctx_list ∗list, unsigned sched_ctx)
- int **_starpu_sched_ctx_elt_get_priority** (struct _starpu_sched_ctx_list ∗list, unsigned sched_ctx)
- struct _starpu_sched_ctx_list ∗ _starpu_sched_ctx_list_find (struct _starpu_sched_ctx_list ∗list, unsigned prio)
- struct _starpu_sched_ctx_elt ∗ **_starpu_sched_ctx_list_add_prio** (struct _starpu_sched_ctx_list ∗∗list, un-signed prio, unsigned sched_ctx)
- int **_starpu_sched_ctx_list_add** (struct _starpu_sched_ctx_list ∗∗list, unsigned sched_ctx)
- void **_starpu_sched_ctx_list_remove_elt** (struct _starpu_sched_ctx_list ∗∗list, struct _starpu_sched_ctx←-_elt ∗rm)
- int **_starpu_sched_ctx_list_remove** (struct _starpu_sched_ctx_list ∗∗list, unsigned sched_ctx)
- int **_starpu_sched_ctx_list_move** (struct _starpu_sched_ctx_list ∗∗list, unsigned sched_ctx, unsigned prio_to)
- int **_starpu_sched_ctx_list_exists** (struct _starpu_sched_ctx_list ∗list, unsigned prio)
- void **_starpu_sched_ctx_list_remove_all** (struct _starpu_sched_ctx_list ∗list)
- void **_starpu_sched_ctx_list_delete** (struct _starpu_sched_ctx_list ∗∗list)
- int _starpu_sched_ctx_list_push_event (struct _starpu_sched_ctx_list ∗list, unsigned sched_ctx)
- int **_starpu_sched_ctx_list_pop_event** (struct _starpu_sched_ctx_list ∗list, unsigned sched_ctx)
- int **_starpu_sched_ctx_list_pop_all_event** (struct _starpu_sched_ctx_list ∗list, unsigned sched_ctx)
- int _starpu_sched_ctx_list_iterator_init (struct _starpu_sched_ctx_list ∗list, struct _starpu_sched_ctx_list_←-iterator ∗it)
- int **_starpu_sched_ctx_list_iterator_has_next** (struct _starpu_sched_ctx_list_iterator ∗it)
- struct _starpu_sched_ctx_elt ∗ **_starpu_sched_ctx_list_iterator_get_next** (struct _starpu_sched_ctx_←-list_iterator ∗it)

### 6.61.1 Data Structure Documentation

**6.61.1.1 struct _starpu_sched_ctx_list**

**Data Fields**

| | |
|---|---|
| struct _starpu_sched_ctx_list * | prev |
| struct _starpu_sched_ctx_list * | next |
| struct _starpu_sched_ctx_elt * | head |
| unsigned | priority |

**6.61.1.2   struct _starpu_sched_ctx_elt**

Represents a circular list of sched context.

**Data Fields**

| | |
|---|---|
| struct _starpu_sched_ctx_elt * | prev |
| struct _starpu_sched_ctx_elt * | next |
| struct _starpu_sched_ctx_list * | parent |
| unsigned | sched_ctx |
| long | task_number |
| unsigned | last_poped |

**6.61.1.3   struct _starpu_sched_ctx_list_iterator**

**Data Fields**

| | |
|---|---|
| struct _starpu_sched_ctx_list * | list_head |
| struct _starpu_sched_ctx_elt * | cursor |

**6.61.2   Function Documentation**

**6.61.2.1   _starpu_sched_ctx_elt_find()**

```
struct _starpu_sched_ctx_elt* _starpu_sched_ctx_elt_find (
            struct _starpu_sched_ctx_list * list,
            unsigned sched_ctx )
```
Element (sched_ctx) level operations

**6.61.2.2   _starpu_sched_ctx_list_find()**

```
struct _starpu_sched_ctx_list* _starpu_sched_ctx_list_find (
            struct _starpu_sched_ctx_list * list,
            unsigned prio )
```
List (priority) level operations

**6.61.2.3   _starpu_sched_ctx_list_push_event()**

```
int _starpu_sched_ctx_list_push_event (
            struct _starpu_sched_ctx_list * list,
            unsigned sched_ctx )
```
Task number management

**6.61.2.4   _starpu_sched_ctx_list_iterator_init()**

```
int _starpu_sched_ctx_list_iterator_init (
```

        struct _starpu_sched_ctx_list * *list,*
        struct _starpu_sched_ctx_list_iterator * *it* )
Iterator operations


## 6.62    sched_policy.h File Reference

```
#include <starpu.h>
#include <signal.h>
#include <core/workers.h>
#include <core/sched_ctx.h>
#include <starpu_scheduler.h>
#include <core/simgrid.h>
```

**Macros**

- #define **_STARPU_SCHED_BEGIN**
- #define **_STARPU_SCHED_END**
- #define **_STARPU_TASK_BREAK_ON**(task, what)

**Functions**

- void **_starpu_sched_init** (void)
- struct starpu_sched_policy ∗ **_starpu_get_sched_policy** (struct _starpu_sched_ctx ∗sched_ctx)
- void **_starpu_init_sched_policy** (struct _starpu_machine_config ∗config, struct _starpu_sched_ctx ∗sched_ctx, struct starpu_sched_policy ∗policy)
- void **_starpu_deinit_sched_policy** (struct _starpu_sched_ctx ∗sched_ctx)
- struct starpu_sched_policy ∗ **_starpu_select_sched_policy** (struct _starpu_machine_config ∗config, const char ∗required_policy)
- void **_starpu_sched_task_submit** (struct starpu_task ∗task)
- void **_starpu_sched_do_schedule** (unsigned sched_ctx_id)
- int **_starpu_push_task** (struct _starpu_job ∗task)
- int **_starpu_repush_task** (struct _starpu_job ∗task)
- int **_starpu_push_task_to_workers** (struct starpu_task ∗task)
- struct starpu_task ∗ **_starpu_pop_task** (struct _starpu_worker ∗worker)
- struct starpu_task ∗ **_starpu_pop_every_task** (struct _starpu_sched_ctx ∗sched_ctx)
- void **_starpu_sched_post_exec_hook** (struct starpu_task ∗task)
- int **_starpu_pop_task_end** (struct starpu_task ∗task)
- void **_starpu_wait_on_sched_event** (void)
- struct starpu_task ∗ **_starpu_create_conversion_task** (starpu_data_handle_t handle, unsigned int node) STARPU_ATTRIBUTE_MALLOC
- struct starpu_task ∗ **_starpu_create_conversion_task_for_arch** (starpu_data_handle_t handle, enum starpu_node_kind node_kind) STARPU_ATTRIBUTE_MALLOC
- void **_starpu_sched_pre_exec_hook** (struct starpu_task ∗task)
- void **_starpu_print_idle_time** ()

**Variables**

- struct starpu_sched_policy **_starpu_sched_lws_policy**
- struct starpu_sched_policy **_starpu_sched_ws_policy**
- struct starpu_sched_policy **_starpu_sched_prio_policy**
- struct starpu_sched_policy **_starpu_sched_random_policy**
- struct starpu_sched_policy **_starpu_sched_dm_policy**
- struct starpu_sched_policy **_starpu_sched_dmda_policy**
- struct starpu_sched_policy **_starpu_sched_dmda_prio_policy**
- struct starpu_sched_policy **_starpu_sched_dmda_ready_policy**

- struct starpu_sched_policy **_starpu_sched_dmda_sorted_policy**
- struct starpu_sched_policy **_starpu_sched_dmda_sorted_decision_policy**
- struct starpu_sched_policy **_starpu_sched_eager_policy**
- struct starpu_sched_policy **_starpu_sched_parallel_heft_policy**
- struct starpu_sched_policy **_starpu_sched_peager_policy**
- struct starpu_sched_policy **_starpu_sched_heteroprio_policy**
- struct starpu_sched_policy **_starpu_sched_modular_eager_policy**
- struct starpu_sched_policy **_starpu_sched_modular_eager_prefetching_policy**
- struct starpu_sched_policy **_starpu_sched_modular_eager_prio_policy**
- struct starpu_sched_policy **_starpu_sched_modular_gemm_policy**
- struct starpu_sched_policy **_starpu_sched_modular_prio_policy**
- struct starpu_sched_policy **_starpu_sched_modular_prio_prefetching_policy**
- struct starpu_sched_policy **_starpu_sched_modular_random_policy**
- struct starpu_sched_policy **_starpu_sched_modular_random_prio_policy**
- struct starpu_sched_policy **_starpu_sched_modular_random_prefetching_policy**
- struct starpu_sched_policy **_starpu_sched_modular_random_prio_prefetching_policy**
- struct starpu_sched_policy **_starpu_sched_modular_parallel_random_policy**
- struct starpu_sched_policy **_starpu_sched_modular_parallel_random_prio_policy**
- struct starpu_sched_policy **_starpu_sched_modular_ws_policy**
- struct starpu_sched_policy **_starpu_sched_modular_heft_policy**
- struct starpu_sched_policy **_starpu_sched_modular_heft_prio_policy**
- struct starpu_sched_policy **_starpu_sched_modular_heft2_policy**
- struct starpu_sched_policy **_starpu_sched_modular_heteroprio_policy**
- struct starpu_sched_policy **_starpu_sched_modular_heteroprio_heft_policy**
- struct starpu_sched_policy **_starpu_sched_modular_parallel_heft_policy**
- struct starpu_sched_policy **_starpu_sched_graph_test_policy**
- struct starpu_sched_policy **_starpu_sched_tree_heft_hierarchical_policy**
- long **_starpu_task_break_on_push**
- long **_starpu_task_break_on_sched**
- long **_starpu_task_break_on_pop**
- long **_starpu_task_break_on_exec**

## 6.62.1 Function Documentation

### 6.62.1.1 _starpu_push_task_to_workers()

```
int _starpu_push_task_to_workers (
            struct starpu_task * task )
```
actually pushes the tasks to the specific worker or to the scheduler

### 6.62.1.2 _starpu_pop_task()

```
struct starpu_task* _starpu_pop_task (
            struct _starpu_worker * worker )
```
pop a task that can be executed on the worker

### 6.62.1.3 _starpu_pop_every_task()

```
struct starpu_task* _starpu_pop_every_task (
            struct _starpu_sched_ctx * sched_ctx )
```
pop every task that can be executed on the worker

## 6.63 simgrid.h File Reference

```
#include <xbt/xbt_os_time.h>
```

**Data Structures**

- struct _starpu_pthread_args

**Macros**

- #define **MAX_TSD**
- #define **STARPU_MPI_AS_PREFIX**
- #define **_starpu_simgrid_running_smpi**()
- #define **_starpu_simgrid_cuda_malloc_cost**()
- #define **_starpu_simgrid_queue_malloc_cost**()
- #define **_starpu_simgrid_task_submit_cost**()
- #define **_starpu_simgrid_fetching_input_cost**()
- #define **_starpu_simgrid_sched_cost**()
- #define **_SIMGRID_TIMER_BEGIN**(cond)
- #define **_SIMGRID_TIMER_END**
- #define _starpu_simgrid_data_new(size)
- #define **_starpu_simgrid_data_increase**(size)
- #define **_starpu_simgrid_data_alloc**(size)
- #define **_starpu_simgrid_data_free**(size)
- #define **_starpu_simgrid_data_transfer**(size, src_node, dst_node)

**Functions**

- void **_starpu_start_simgrid** (int ∗argc, char ∗∗argv)
- void **_starpu_simgrid_init_early** (int ∗argc, char ∗∗∗argv)
- void **_starpu_simgrid_init** (void)
- void **_starpu_simgrid_deinit** (void)
- void **_starpu_simgrid_deinit_late** (void)
- void **_starpu_simgrid_actor_setup** (void)
- void **_starpu_simgrid_wait_tasks** (int workerid)
- void **_starpu_simgrid_submit_job** (int workerid, struct _starpu_job ∗job, struct starpu_perfmodel_arch ∗perf_arch, double length, unsigned ∗finished)
- int **_starpu_simgrid_transfer** (size_t size, unsigned src_node, unsigned dst_node, struct _starpu_data_↩ request ∗req)
- int **_starpu_simgrid_wait_transfer_event** (union _starpu_async_channel_event ∗event)
- int **_starpu_simgrid_test_transfer_event** (union _starpu_async_channel_event ∗event)
- void **_starpu_simgrid_sync_gpus** (void)
- int _starpu_simgrid_get_nbhosts (const char ∗prefix)
- unsigned long long **_starpu_simgrid_get_memsize** (const char ∗prefix, unsigned devid)
- starpu_sg_host_t **_starpu_simgrid_get_host_by_name** (const char ∗name)
- starpu_sg_host_t **_starpu_simgrid_get_memnode_host** (unsigned node)
- starpu_sg_host_t **_starpu_simgrid_get_host_by_worker** (struct _starpu_worker ∗worker)
- void **_starpu_simgrid_get_platform_path** (int version, char ∗path, size_t maxlen)
- msg_as_t **_starpu_simgrid_get_as_by_name** (const char ∗name)
- int **starpu_mpi_world_rank** (void)
- int **_starpu_mpi_simgrid_init** (int argc, char ∗argv[ ])
- void _starpu_simgrid_count_ngpus (void)
- void **_starpu_simgrid_xbt_thread_create** (const char ∗name, void_f_pvoid_t code, void ∗param)

**Variables**

- starpu_pthread_queue_t **_starpu_simgrid_transfer_queue** [STARPU_MAXNODES]
- starpu_pthread_queue_t **_starpu_simgrid_task_queue** [STARPU_NMAXWORKERS]

## 6.63.1 Macro Definition Documentation

### 6.63.1.1 _starpu_simgrid_data_new

```
#define _starpu_simgrid_data_new(
            size )
```
Experimental functions for OOC stochastic analysis

## 6.63.2 Function Documentation

### 6.63.2.1 _starpu_simgrid_get_nbhosts()

```
int _starpu_simgrid_get_nbhosts (
            const char * prefix )
```
Return the number of hosts prefixed by PREFIX

### 6.63.2.2 _starpu_simgrid_count_ngpus()

```
void _starpu_simgrid_count_ngpus (
            void )
```
Called at initialization to count how many GPUs are interfering with each bus

## 6.64 sink_common.h File Reference

```
#include <common/config.h>
```

## 6.65 sort_data_handles.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <stdlib.h>
#include <stdarg.h>
#include <core/jobs.h>
#include <datawizard/coherency.h>
#include <datawizard/memalloc.h>
```

**Functions**

- void _starpu_sort_task_handles (struct _starpu_data_descr descr[ ], unsigned nbuffers)

## 6.65.1 Function Documentation

**6.65.1.1 _starpu_sort_task_handles()**

```
void _starpu_sort_task_handles (
            struct _starpu_data_descr descr[],
            unsigned nbuffers )
```

To avoid deadlocks, we reorder the different buffers accessed to by the task so that we always grab the rw-lock associated to the handles in the same order.

## 6.66 source_common.h File Reference

## 6.67 starpu_clusters_create.h File Reference

```
#include <starpu.h>
#include <core/workers.h>
#include <common/list.h>
#include <string.h>
#include <omp.h>
```

## 6.68 starpu_data_cpy.h File Reference

```
#include <starpu.h>
```

**Functions**

- int **_starpu_data_cpy** (starpu_data_handle_t dst_handle, starpu_data_handle_t src_handle, int asynchronous, void(∗callback_func)(void ∗), void ∗callback_arg, int reduction, struct starpu_task ∗reduction_↩
dep_task)

## 6.69 starpu_debug_helpers.h File Reference

```
#include <starpu.h>
#include <starpu_config.h>
#include <starpu_util.h>
```

**Functions**

- void _starpu_benchmark_ping_pong (starpu_data_handle_t handle, unsigned node0, unsigned node1, unsigned niter)
- void _starpu_debug_display_structures_size (FILE ∗stream)

### 6.69.1 Function Documentation

**6.69.1.1 _starpu_benchmark_ping_pong()**

```
void _starpu_benchmark_ping_pong (
            starpu_data_handle_t handle,
            unsigned node0,
            unsigned node1,
            unsigned niter )
```

Perform a ping pong between the two memory nodes

**6.69.1.2 _starpu_debug_display_structures_size()**

```
void _starpu_debug_display_structures_size (
            FILE * stream )
```
Display the size of different data structures

## 6.70 starpu_fxt.h File Reference

```
#include <starpu.h>
#include <starpu_config.h>
#include <common/config.h>
```

## 6.71 starpu_parameters.h File Reference

**Macros**

- #define **_STARPU_CPU_ALPHA**
- #define **_STARPU_CUDA_ALPHA**
- #define **_STARPU_OPENCL_ALPHA**
- #define **_STARPU_MIC_ALPHA**
- #define **_STARPU_MPI_MS_ALPHA**

## 6.72 starpu_spinlock.h File Reference

```
#include <errno.h>
#include <stdint.h>
#include <common/config.h>
#include <common/fxt.h>
#include <common/thread.h>
#include <starpu.h>
```

**Data Structures**

- struct _starpu_spinlock

**Macros**

- #define **_starpu_spin_destroy**(_lock)
- #define **_starpu_spin_checklocked**(_lock)
- #define **_starpu_spin_lock**(lock)
- #define **_starpu_spin_trylock**(lock)
- #define **_starpu_spin_unlock**(lock)
- #define **STARPU_SPIN_MAXTRY**

**Functions**

- static int **_starpu_spin_init** (struct _starpu_spinlock *lock)
- static int **__starpu_spin_lock** (struct _starpu_spinlock *lock, const char *file STARPU_ATTRIBUTE_UN←
  USED, int line STARPU_ATTRIBUTE_UNUSED, const char *func STARPU_ATTRIBUTE_UNUSED)
- static int **__starpu_spin_trylock** (struct _starpu_spinlock *lock, const char *file STARPU_ATTRIBUTE_←
  UNUSED, int line STARPU_ATTRIBUTE_UNUSED, const char *func STARPU_ATTRIBUTE_UNUSED)
- static int **__starpu_spin_unlock** (struct _starpu_spinlock *lock, const char *file STARPU_ATTRIBUTE_←
  UNUSED, int line STARPU_ATTRIBUTE_UNUSED, const char *func STARPU_ATTRIBUTE_UNUSED)

**6.72.1 Data Structure Documentation**

**6.72.1.1 struct _starpu_spinlock**

**Data Fields**

| starpu_pthread_spinlock_t | lock | |
|---|---|---|

# 6.73 starpu_task_insert_utils.h File Reference

```
#include <stdlib.h>
#include <stdarg.h>
#include <starpu.h>
```

**Typedefs**

- typedef void(∗ **_starpu_callback_func_t**) (void ∗)

**Functions**

- int **_starpu_codelet_pack_args** (void ∗∗arg_buffer, size_t ∗arg_buffer_size, va_list varg_list)
- int **_starpu_task_insert_create** (struct starpu_codelet ∗cl, struct starpu_task ∗task, va_list varg_list)
- int **_fstarpu_task_insert_create** (struct starpu_codelet ∗cl, struct starpu_task ∗task, void ∗∗arglist)

# 6.74 tags.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/starpu_spinlock.h>
#include <core/dependencies/cg.h>
```

**Data Structures**

- struct _starpu_tag

**Macros**

- #define **_STARPU_TAG_SIZE**

**Enumerations**

- enum _starpu_tag_state {
  STARPU_INVALID_STATE, STARPU_ASSOCIATED, STARPU_BLOCKED, STARPU_READY,
  STARPU_DONE }

**Functions**

- void **_starpu_init_tags** (void)
- void **_starpu_notify_tag_dependencies** (struct _starpu_tag ∗tag)
- void **_starpu_notify_job_start_tag_dependencies** (struct _starpu_tag ∗tag, _starpu_notify_job_start_data ∗data)
- void **_starpu_tag_declare** (starpu_tag_t id, struct _starpu_job ∗job)
- void _starpu_tag_set_ready (struct _starpu_tag ∗tag)
- unsigned **_starpu_submit_job_enforce_task_deps** (struct _starpu_job ∗j)
- void **_starpu_tag_clear** (void)

### 6.74.1 Data Structure Documentation

#### 6.74.1.1 struct _starpu_tag

**Data Fields**

| struct _starpu_spinlock | lock | Lock for this structure. Locking order is in dependency order: a tag must not be locked before locking a tag it depends on |
|---|---|---|
| starpu_tag_t | id | an identifier for the task |
| enum _starpu_tag_state | state | |
| struct _starpu_cg_list | tag_successors | |
| struct _starpu_job ∗ | job | which job is associated to the tag if any ? |
| unsigned | is_assigned | |
| unsigned | is_submitted | |

### 6.74.2 Enumeration Type Documentation

#### 6.74.2.1 _starpu_tag_state

enum _starpu_tag_state

**Enumerator**

| STARPU_INVALID_STATE | this tag is not declared by any task |
|---|---|
| STARPU_ASSOCIATED | _starpu_tag_declare was called to associate the tag to a task |
| STARPU_BLOCKED | some task dependencies are not fulfilled yet |
| STARPU_READY | the task can be (or has been) submitted to the scheduler (all deps fulfilled) |
| STARPU_DONE | the task has been performed |

### 6.74.3 Function Documentation

#### 6.74.3.1 _starpu_tag_set_ready()

```
void _starpu_tag_set_ready (
            struct _starpu_tag ∗ tag )
```
lock should be taken, and this releases it

## 6.75 task.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <core/jobs.h>
```

**Macros**

- #define **_STARPU_TASK_SET_INTERFACE**(task, interface, i)
- #define **_STARPU_TASK_GET_INTERFACES**(task)

**Functions**

- void _starpu_task_destroy (struct starpu_task ∗task)
- int _starpu_task_test_termination (struct starpu_task ∗task)
- void _starpu_task_init (void)
- void **_starpu_task_deinit** (void)
- void **_starpu_set_current_task** (struct starpu_task ∗task)
- int **_starpu_submit_job** (struct _starpu_job ∗j)
- int **_starpu_task_submit_nodeps** (struct starpu_task ∗task)
- void **_starpu_task_declare_deps_array** (struct starpu_task ∗task, unsigned ndeps, struct starpu_task ∗task_array[ ], int check)
- static struct _starpu_job ∗ _starpu_get_job_associated_to_task (struct starpu_task ∗task)
- int _starpu_task_submit_internally (struct starpu_task ∗task)
- int **_starpu_handle_needs_conversion_task** (starpu_data_handle_t handle, unsigned int node)
- int **_starpu_handle_needs_conversion_task_for_arch** (starpu_data_handle_t handle, enum starpu_↩ node_kind node_kind)
- void _starpu_task_prepare_for_continuation_ext (unsigned continuation_resubmit, void(∗continuation_↩ callback_on_sleep)(void ∗arg), void ∗continuation_callback_on_sleep_arg)
- void **_starpu_task_prepare_for_continuation** (void)
- void **_starpu_task_set_omp_cleanup_callback** (struct starpu_task ∗task, void(∗omp_cleanup_↩ callback)(void ∗arg), void ∗omp_cleanup_callback_arg)
- int **_starpu_task_uses_multiformat_handles** (struct starpu_task ∗task)
- int **_starpu_task_submit_conversion_task** (struct starpu_task ∗task, unsigned int workerid)
- void **_starpu_task_check_deprecated_fields** (struct starpu_task ∗task)
- void **_starpu_codelet_check_deprecated_fields** (struct starpu_codelet ∗cl)
- static starpu_cpu_func_t **_starpu_task_get_cpu_nth_implementation** (struct starpu_codelet ∗cl, unsigned nimpl)
- static starpu_cuda_func_t **_starpu_task_get_cuda_nth_implementation** (struct starpu_codelet ∗cl, unsigned nimpl)
- static starpu_opencl_func_t **_starpu_task_get_opencl_nth_implementation** (struct starpu_codelet ∗cl, unsigned nimpl)
- static starpu_mic_func_t **_starpu_task_get_mic_nth_implementation** (struct starpu_codelet ∗cl, unsigned nimpl)
- static starpu_mpi_ms_func_t **_starpu_task_get_mpi_ms_nth_implementation** (struct starpu_codelet ∗cl, unsigned nimpl)
- static const char ∗ **_starpu_task_get_cpu_name_nth_implementation** (struct starpu_codelet ∗cl, unsigned nimpl)
- void **_starpu_watchdog_init** (void)
- void **_starpu_watchdog_shutdown** (void)
- int **_starpu_task_wait_for_all_and_return_nb_waited_tasks** (void)
- int **_starpu_task_wait_for_all_in_ctx_and_return_nb_waited_tasks** (unsigned sched_ctx)

### 6.75.1 Function Documentation

#### 6.75.1.1 _starpu_task_destroy()

```
void _starpu_task_destroy (
            struct starpu_task * task )
```
Internal version of starpu_task_destroy: don't check task->destroy flag

#### 6.75.1.2 _starpu_task_test_termination()

```
int _starpu_task_test_termination (
            struct starpu_task * task )
```
Test for the termination of the task. Call starpu_task_destroy if required and the task is terminated.

**6.75.1.3 _starpu_task_init()**

```
void _starpu_task_init (
            void  )
```
A pthread key is used to store the task currently executed on the thread. _starpu_task_init initializes this pthread key and _starpu_set_current_task updates its current value.


**6.75.1.4 _starpu_get_job_associated_to_task()**

```
static struct _starpu_job* _starpu_get_job_associated_to_task (
            struct starpu_task * task )  [static]
```
Returns the job structure (which is the internal data structure associated to a task).


**6.75.1.5 _starpu_task_submit_internally()**

```
int _starpu_task_submit_internally (
            struct starpu_task * task )
```
Submits starpu internal tasks to the initial context


**6.75.1.6 _starpu_task_prepare_for_continuation_ext()**

```
void _starpu_task_prepare_for_continuation_ext (
            unsigned continuation_resubmit,
            void(*)(void *arg) continuation_callback_on_sleep,
            void * continuation_callback_on_sleep_arg )
```
Prepare the current task for accepting new dependencies before becoming a continuation.


## 6.76 task_bundle.h File Reference

```
#include <starpu_thread.h>
```


**Data Structures**

- struct _starpu_task_bundle_entry
- struct _starpu_task_bundle
- struct _starpu_handle_list


**Functions**

- void _starpu_task_bundle_destroy (starpu_task_bundle_t bundle)
- void _insertion_handle_sorted (struct _starpu_handle_list **listp, starpu_data_handle_t handle, enum starpu_data_access_mode mode)


### 6.76.1 Data Structure Documentation

**6.76.1.1 struct _starpu_task_bundle_entry**

struct _starpu_task_bundle_entry ================================= Purpose ======= Structure used to describe a linked list containing tasks in _starpu_task_bundle.
Fields ====== task Pointer to the task structure.
next Pointer to the next element in the linked list.

**Data Fields**

| | | |
|---|---|---|
| struct starpu_task ∗ | task | |
| struct _starpu_task_bundle_entry ∗ | next | |

### 6.76.1.2 struct _starpu_task_bundle

struct _starpu_task_bundle ========================== Purpose ======= Structure describing a list of tasks that should be scheduled on the same worker whenever it's possible. It must be considered as a hint given to the scheduler as there is no guarantee that they will be executed on the same worker.

Fields ====== mutex Mutex protecting the structure.

list Array of tasks included in the bundle.

closed Used to know if the user is still willing to add/remove some tasks in the bundle. Especially useful for the runtime to know whether it is safe to destroy a bundle.

**Data Fields**

| | | |
|---:|---|---|
| starpu_pthread_mutex_t | mutex | Mutex protecting the bundle |
| struct _starpu_task_bundle_entry ∗ | list | |
| int | closed | |

### 6.76.1.3 struct _starpu_handle_list

struct _starpu_handle_list =========================== Purpose ======= Structure describing a list of handles sorted by address to speed-up when looking for an element. The list cannot containes duplicate handles.

Fields ====== handle Pointer to the handle structure.

access_mode Total access mode over the whole bundle.

next Pointer to the next element in the linked list.

**Data Fields**

| | | |
|---:|---|---|
| starpu_data_handle_t | handle | |
| enum starpu_data_access_mode | mode | |
| struct _starpu_handle_list ∗ | next | |

## 6.76.2 Function Documentation

### 6.76.2.1 _starpu_task_bundle_destroy()

```
void _starpu_task_bundle_destroy (
            starpu_task_bundle_t bundle )
```

_starpu_task_bundle_destroy =========================== Purpose ======= Destroy and deinitialize a bundle, memory previoulsy allocated is freed.

Arguments ========= bundle (input) Bundle to destroy.

### 6.76.2.2 _insertion_handle_sorted()

```
void _insertion_handle_sorted (
            struct _starpu_handle_list ∗∗ listp,
            starpu_data_handle_t handle,
            enum starpu_data_access_mode mode )
```

_insertion_handle_sorted ======================== Purpose ======= Insert an handle in a _starpu_↩ handle_list, elements are sorted in increasing order, considering their physical address. As the list doesn't accept duplicate elements, a handle with the same address as an handle contained in the list is not inserted, but its mode access is merged with the one of the latter.

Arguments ========= listp (input, output) Pointer to the first element of the list. In the case of an empty list or an inserted handle with small address, it should have changed when the call returns.

handle (input) Handle to insert in the list.

mode (input) Access mode of the handle.

## 6.77 thread.h File Reference

```
#include <common/utils.h>
```

**Macros**

- #define **starpu_pthread_spin_init**
- #define **starpu_pthread_spin_destroy**
- #define **starpu_pthread_spin_lock**
- #define **starpu_pthread_spin_trylock**
- #define **starpu_pthread_spin_unlock**

**Functions**

- static int **_starpu_pthread_spin_init** (starpu_pthread_spinlock_t ∗lock, int pshared STARPU_ATTRIBUT←
  E_UNUSED)
- static int **_starpu_pthread_spin_destroy** (starpu_pthread_spinlock_t ∗lock STARPU_ATTRIBUTE_UNU←
  SED)
- static int **_starpu_pthread_spin_lock** (starpu_pthread_spinlock_t ∗lock)
- static void **_starpu_pthread_spin_checklocked** (starpu_pthread_spinlock_t ∗lock STARPU_ATTRIBUT←
  E_UNUSED)
- static int **_starpu_pthread_spin_trylock** (starpu_pthread_spinlock_t ∗lock)
- static int **_starpu_pthread_spin_unlock** (starpu_pthread_spinlock_t ∗lock)

## 6.78 timing.h File Reference

```
#include <stdint.h>
#include <common/config.h>
#include <starpu.h>
#include <starpu_util.h>
```

**Functions**

- void _starpu_timing_init (void)
- void **_starpu_clock_gettime** (struct timespec ∗ts)

### 6.78.1 Function Documentation

#### 6.78.1.1 _starpu_timing_init()

```
void _starpu_timing_init (
            void )
```
_starpu_timing_init must be called prior to using any of these timing functions.

## 6.79 topology.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/list.h>
#include <common/fxt.h>
```

**Macros**

- #define **STARPU_NOWORKERID**
- #define **STARPU_ACTIVETHREAD**
- #define **STARPU_NONACTIVETHREAD**

**Functions**

- int _starpu_build_topology (struct _starpu_machine_config ∗config, int no_mp_config)
- void _starpu_destroy_machine_config (struct _starpu_machine_config ∗config)
- void _starpu_destroy_topology (struct _starpu_machine_config ∗config)
- unsigned _starpu_topology_get_nhwcpu (struct _starpu_machine_config ∗config)
- unsigned _starpu_topology_get_nhwpu (struct _starpu_machine_config ∗config)
- unsigned _starpu_topology_get_nnumanodes (struct _starpu_machine_config ∗config)
- void _starpu_topology_filter (hwloc_topology_t topology)
- int _starpu_bind_thread_on_cpu (int cpuid, int workerid, const char ∗name)
- void _starpu_bind_thread_on_cpus (struct _starpu_combined_worker ∗combined_worker)
- struct _starpu_worker ∗ **_starpu_get_worker_from_driver** (struct starpu_driver ∗d)
- int **starpu_memory_nodes_get_numa_count** (void)
- int **starpu_memory_nodes_numa_id_to_hwloclogid** (unsigned id)
- int _starpu_task_data_get_node_on_node (struct starpu_task ∗task, unsigned index, unsigned target_node)
- int **_starpu_task_data_get_node_on_worker** (struct starpu_task ∗task, unsigned index, unsigned worker)

### 6.79.1 Function Documentation

#### 6.79.1.1 _starpu_build_topology()

```
int _starpu_build_topology (
            struct _starpu_machine_config * config,
            int no_mp_config )
```
Detect the number of memory nodes and where to bind the different workers.

#### 6.79.1.2 _starpu_destroy_machine_config()

```
void _starpu_destroy_machine_config (
            struct _starpu_machine_config * config )
```
Should be called instead of _starpu_destroy_topology when _starpu_build_topology returns a non zero value.

#### 6.79.1.3 _starpu_destroy_topology()

```
void _starpu_destroy_topology (
            struct _starpu_machine_config * config )
```
Destroy all resources used to store the topology of the machine.

#### 6.79.1.4 _starpu_topology_get_nhwcpu()

```
unsigned _starpu_topology_get_nhwcpu (
            struct _starpu_machine_config * config )
```
returns the number of physical cpus

#### 6.79.1.5 _starpu_topology_get_nhwpu()

```
unsigned _starpu_topology_get_nhwpu (
            struct _starpu_machine_config * config )
```
returns the number of logical cpus

### 6.79.1.6 _starpu_topology_get_nnumanodes()

```
unsigned _starpu_topology_get_nnumanodes (
            struct _starpu_machine_config * config )
```
returns the number of NUMA nodes

### 6.79.1.7 _starpu_topology_filter()

```
void _starpu_topology_filter (
            hwloc_topology_t topology )
```
Small convenient function to filter hwloc topology depending on HWLOC API version

### 6.79.1.8 _starpu_bind_thread_on_cpu()

```
int _starpu_bind_thread_on_cpu (
            int cpuid,
            int workerid,
            const char * name )
```
Bind the current thread on the CPU logically identified by "cpuid". The logical ordering of the processors is either that of hwloc (if available), or the ordering exposed by the OS.

### 6.79.1.9 _starpu_bind_thread_on_cpus()

```
void _starpu_bind_thread_on_cpus (
            struct _starpu_combined_worker * combined_worker )
```
Bind the current thread on the set of CPUs for the given combined worker.

### 6.79.1.10 _starpu_task_data_get_node_on_node()

```
int _starpu_task_data_get_node_on_node (
            struct starpu_task * task,
            unsigned index,
            unsigned target_node )
```
Get the memory node for data number i when task is to be executed on memory node target_node

## 6.80 utils.h File Reference

```
#include <common/config.h>
#include <starpu.h>
#include <sys/stat.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
```

**Macros**

- #define **DO_CREQ_v_WW**(_creqF, _ty1F, _arg1F, _ty2F, _arg2F)
- #define **DO_CREQ_v_W**(_creqF, _ty1F, _arg1F)
- #define **ANNOTATE_HAPPENS_BEFORE**(obj)
- #define **ANNOTATE_HAPPENS_BEFORE_FORGET_ALL**(obj)
- #define **ANNOTATE_HAPPENS_AFTER**(obj)
- #define **VALGRIND_HG_DISABLE_CHECKING**(start, len)
- #define **VALGRIND_HG_ENABLE_CHECKING**(start, len)
- #define **VALGRIND_STACK_REGISTER**(stackbottom, stacktop)
- #define **VALGRIND_STACK_DEREGISTER**(id)
- #define **RUNNING_ON_VALGRIND**
- #define **STARPU_RUNNING_ON_VALGRIND**

- #define **STARPU_HG_DISABLE_CHECKING**(variable)
- #define **STARPU_HG_ENABLE_CHECKING**(variable)
- #define **STARPU_DEBUG_PREFIX**
- #define **_STARPU_UYIELD**()
- #define **STARPU_VALGRIND_YIELD**()
- #define **STARPU_UYIELD**()
- #define **_STARPU_DEBUG**(fmt, ...)
- #define **_STARPU_DEBUG_NO_HEADER**(fmt, ...)
- #define **_STARPU_EXTRA_DEBUG**(fmt, ...)
- #define **_STARPU_LOG_IN**()
- #define **_STARPU_LOG_OUT**()
- #define **_STARPU_LOG_OUT_TAG**(outtag)
- #define **_STARPU_MSG**(fmt, ...)
- #define **_STARPU_DISP**(fmt, ...)
- #define **_STARPU_ERROR**(fmt, ...)
- #define **_STARPU_DECLTYPE**(x)
- #define **_STARPU_MALLOC**(ptr, size)
- #define **_STARPU_CALLOC**(ptr, nmemb, size)
- #define **_STARPU_REALLOC**(ptr, size)
- #define **_STARPU_IS_ZERO**(a)

**Functions**

- char ∗ **_starpu_mkdtemp_internal** (char ∗tmpl)
- char ∗ **_starpu_mkdtemp** (char ∗tmpl)
- int **_starpu_mkpath** (const char ∗s, mode_t mode)
- void **_starpu_mkpath_and_check** (const char ∗s, mode_t mode)
- char ∗ **_starpu_mktemp** (const char ∗directory, int flags, int ∗fd)
- char ∗ _starpu_mktemp_many (const char ∗directory, int depth, int flags, int ∗fd)
- void **_starpu_rmtemp_many** (char ∗path, int depth)
- void **_starpu_rmdir_many** (char ∗path, int depth)
- int **_starpu_fftruncate** (FILE ∗file, size_t length)
- int **_starpu_ftruncate** (int fd, size_t length)
- int **_starpu_frdlock** (FILE ∗file)
- int **_starpu_frdunlock** (FILE ∗file)
- int **_starpu_fwrlock** (FILE ∗file)
- int **_starpu_fwrunlock** (FILE ∗file)
- char ∗ **_starpu_get_home_path** (void)
- void **_starpu_gethostname** (char ∗hostname, size_t size)
- void _starpu_drop_comments (FILE ∗f)
- const char ∗ _starpu_job_get_model_name (struct _starpu_job ∗j)
- const char ∗ _starpu_job_get_task_name (struct _starpu_job ∗j)
- const char ∗ _starpu_codelet_get_model_name (struct starpu_codelet ∗cl)
- int **_starpu_check_mutex_deadlock** (starpu_pthread_mutex_t ∗mutex)
- void **_starpu_util_init** (void)

**6.80.1 Function Documentation**

**6.80.1.1 _starpu_mktemp_many()**

```
char* _starpu_mktemp_many (
            const char * directory,
            int depth,
            int flags,
            int * fd )
```
This version creates a hierarchy of n temporary directories, useful when creating a lot of temporary files to be stored in the same place

**6.80.1.2 _starpu_drop_comments()**

```
void _starpu_drop_comments (
            FILE * f )
```
If FILE is currently on a comment line, eat it.

**6.80.1.3 _starpu_job_get_model_name()**

```
const char* _starpu_job_get_model_name (
            struct _starpu_job * j )
```
Returns the symbol associated to that job if any.

**6.80.1.4 _starpu_job_get_task_name()**

```
const char* _starpu_job_get_task_name (
            struct _starpu_job * j )
```
Returns the name associated to that job if any.

**6.80.1.5 _starpu_codelet_get_model_name()**

```
const char* _starpu_codelet_get_model_name (
            struct starpu_codelet * cl )
```
Returns the symbol associated to that job if any.

## 6.81 uthash.h File Reference

```
#include <string.h>
#include <stddef.h>
#include <inttypes.h>
```

**Data Structures**

- struct UT_hash_bucket
- struct UT_hash_table
- struct UT_hash_handle

**Macros**

- #define **DECLTYPE**(x)
- #define **DECLTYPE_ASSIGN**(dst, src)
- #define **UTHASH_VERSION**
- #define **uthash_fatal**(msg)
- #define **uthash_malloc**(sz)
- #define **uthash_free**(ptr, sz)
- #define **uthash_noexpand_fyi**(tbl)
- #define **uthash_expand_fyi**(tbl)
- #define **HASH_INITIAL_NUM_BUCKETS**

- #define **HASH_INITIAL_NUM_BUCKETS_LOG2**
- #define **HASH_BKT_CAPACITY_THRESH**
- #define **ELMT_FROM_HH**(tbl, hhp)
- #define **HASH_FIND**(hh, head, keyptr, keylen, out)
- #define **HASH_BLOOM_MAKE**(tbl)
- #define **HASH_BLOOM_FREE**(tbl)
- #define **HASH_BLOOM_ADD**(tbl, hashv)
- #define **HASH_BLOOM_TEST**(tbl, hashv)
- #define **HASH_MAKE_TABLE**(hh, head)
- #define **HASH_ADD**(hh, head, fieldname, keylen_in, add)
- #define **HASH_CHECK_KEY**(hh, head, keyptr, keylen, out)
- #define **HASH_ADD_KEYPTR**(hh, head, keyptr, keylen_in, add)
- #define **HASH_TO_BKT**(hashv, num_bkts, bkt)
- #define **HASH_DELETE**(hh, head, delptr)
- #define **HASH_FIND_STR**(head, findstr, out)
- #define **HASH_ADD_STR**(head, strfield, add)
- #define **HASH_FIND_INT**(head, findint, out)
- #define **HASH_ADD_INT**(head, intfield, add)
- #define **HASH_FIND_PTR**(head, findptr, out)
- #define **HASH_ADD_PTR**(head, ptrfield, add)
- #define **HASH_DEL**(head, delptr)
- #define **HASH_FSCK**(hh, head)
- #define **HASH_EMIT_KEY**(hh, head, keyptr, fieldlen)
- #define **HASH_FCN**
- #define **HASH_BER**(key, keylen, num_bkts, hashv, bkt)
- #define **HASH_SAX**(key, keylen, num_bkts, hashv, bkt)
- #define **HASH_FNV**(key, keylen, num_bkts, hashv, bkt)
- #define **HASH_OAT**(key, keylen, num_bkts, hashv, bkt)
- #define **HASH_JEN_MIX**(a, b, c)
- #define **HASH_JEN**(key, keylen, num_bkts, hashv, bkt)
- #define **get16bits**(d)
- #define **HASH_SFH**(key, keylen, num_bkts, hashv, bkt)
- #define **HASH_KEYCMP**(a, b, len)
- #define **HASH_FIND_IN_BKT**(tbl, hh, head, keyptr, keylen_in, out)
- #define **HASH_ADD_TO_BKT**(head, addhh)
- #define **HASH_DEL_IN_BKT**(hh, head, hh_del)
- #define **HASH_EXPAND_BUCKETS**(tbl)
- #define **HASH_SORT**(head, cmpfcn)
- #define **HASH_SRT**(hh, head, cmpfcn)
- #define **HASH_SELECT**(hh_dst, dst, hh_src, src, cond)
- #define **HASH_CLEAR**(hh, head)
- #define **HASH_ITER**(hh, head, el, tmp)
- #define **HASH_COUNT**(head)
- #define **HASH_CNT**(hh, head)
- #define **HASH_SIGNATURE**
- #define **HASH_BLOOM_SIGNATURE**

**Typedefs**

- typedef struct UT_hash_bucket **UT_hash_bucket**
- typedef struct UT_hash_table **UT_hash_table**
- typedef struct UT_hash_handle **UT_hash_handle**

### 6.81.1 Data Structure Documentation

#### 6.81.1.1 struct UT_hash_bucket

**Data Fields**

| | | |
|---:|:---|:---|
| struct UT_hash_handle * | hh_head | |
| unsigned | count | |
| unsigned | expand_mult | |

**6.81.1.2 struct UT_hash_table**

**Data Fields**

| | | |
|---:|:---|:---|
| UT_hash_bucket * | buckets | |
| unsigned | num_buckets | |
| unsigned | log2_num_buckets | |
| unsigned | num_items | |
| struct UT_hash_handle * | tail | |
| ptrdiff_t | hho | |
| unsigned | ideal_chain_maxlen | |
| unsigned | nonideal_items | |
| unsigned | ineff_expands | |
| unsigned | noexpand | |
| uint32_t | signature | |

**6.81.1.3 struct UT_hash_handle**

**Data Fields**

| | | |
|---:|:---|:---|
| struct UT_hash_table * | tbl | |
| void * | prev | |
| void * | next | |
| struct UT_hash_handle * | hh_prev | |
| struct UT_hash_handle * | hh_next | |
| void * | key | |
| unsigned | keylen | |
| unsigned | hashv | |

# 6.82 write_back.h File Reference

```
#include <starpu.h>
#include <datawizard/coherency.h>
```

## Functions

- void _starpu_write_through_data (starpu_data_handle_t handle, unsigned requesting_node, uint32_t write↩
  _through_mask)

## 6.82.1 Function Documentation

**6.82.1.1 _starpu_write_through_data()**

```
void _starpu_write_through_data (
            starpu_data_handle_t handle,
            unsigned requesting_node,
            uint32_t write_through_mask )
```
If a write-through mask is associated to that data handle, this propagates the the current value of the data onto the different memory nodes in the write_through_mask.

**6.82.1.1 _starpu_write_through_data()**

```
void _starpu_write_through_data (
```

# Chapter 7

# StarPU MPI File Documentation

## 7.1  starpu_mpi_cache.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
```

### Functions

- void **_starpu_mpi_cache_init** (MPI_Comm comm)
- void **_starpu_mpi_cache_shutdown** ()
- void **_starpu_mpi_cache_data_init** (starpu_data_handle_t data_handle)
- void **_starpu_mpi_cache_data_clear** (starpu_data_handle_t data_handle)
- void **_starpu_mpi_cache_flush** (starpu_data_handle_t data_handle)

### Variables

- int **_starpu_cache_enabled**

## 7.2  starpu_mpi_driver.h File Reference

```
#include <starpu.h>
```

### Functions

- void **_starpu_mpi_driver_init** (struct starpu_conf ∗conf)
- void **_starpu_mpi_driver_shutdown** ()

## 7.3  starpu_mpi_init.h File Reference

```
#include <starpu.h>
#include <starpu_mpi.h>
```

### Functions

- void **_starpu_mpi_do_initialize** (struct _starpu_mpi_argc_argv ∗argc_argv)

## 7.4 starpu_mpi_nmad_backend.h File Reference

```
#include <common/config.h>
#include <nm_sendrecv_interface.h>
#include <nm_session_interface.h>
#include <nm_mpi_nmad.h>
```

### Data Structures

- struct _starpu_mpi_req_backend

### 7.4.1 Data Structure Documentation

#### 7.4.1.1 struct _starpu_mpi_req_backend

**Data Fields**

| | | |
|---:|---|---|
| MPI_Request | data_request | |
| starpu_pthread_mutex_t | req_mutex | |
| starpu_pthread_cond_t | req_cond | |
| starpu_pthread_mutex_t | posted_mutex | |
| starpu_pthread_cond_t | posted_cond | |
| struct _starpu_mpi_req ∗ | other_request | In the case of a Wait/Test request, we are going to post a request to test the completion of another request |
| MPI_Request | size_req | |
| struct _starpu_mpi_envelope ∗ | envelope | |
| unsigned | is_internal_req:1 | |
| unsigned | to_destroy:1 | |
| struct _starpu_mpi_req ∗ | internal_req | |
| struct _starpu_mpi_early_data_handle ∗ | early_data_handle | |
| UT_hash_handle | hh | |
| nm_gate_t | gate | |
| nm_session_t | session | |
| nm_sr_request_t | data_request | |
| int | waited | |
| piom_cond_t | req_cond | |
| nm_sr_request_t | size_req | |

## 7.5 starpu_mpi_stats.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
```

### Functions

- void **_starpu_mpi_comm_amounts_init** (MPI_Comm comm)
- void **_starpu_mpi_comm_amounts_shutdown** ()
- void **_starpu_mpi_comm_amounts_inc** (MPI_Comm comm, unsigned dst, MPI_Datatype datatype, int count)

- void **_starpu_mpi_comm_amounts_display** (FILE ∗stream, int node)

## 7.6  starpu_mpi_cache_stats.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
```

**Macros**

- #define **_starpu_mpi_cache_stats_inc**(dst, data_handle)
- #define **_starpu_mpi_cache_stats_dec**(dst, data_handle)

**Functions**

- void **_starpu_mpi_cache_stats_init** ()
- void **_starpu_mpi_cache_stats_shutdown** ()
- void **_starpu_mpi_cache_stats_update** (unsigned dst, starpu_data_handle_t data_handle, int count)

## 7.7  starpu_mpi_early_data.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
#include <common/uthash.h>
#include <starpu_mpi_private.h>
```

**Data Structures**

- struct _starpu_mpi_early_data_handle
- struct _starpu_mpi_early_data_handle_tag_hashlist

**Functions**

- void **_starpu_mpi_early_data_init** (void)
- void **_starpu_mpi_early_data_check_termination** (void)
- void **_starpu_mpi_early_data_shutdown** (void)
- struct _starpu_mpi_early_data_handle ∗ **_starpu_mpi_early_data_create** (struct _starpu_mpi_envelope ∗envelope, int source, MPI_Comm comm) STARPU_ATTRIBUTE_MALLOC
- struct _starpu_mpi_early_data_handle ∗ **_starpu_mpi_early_data_find** (struct _starpu_mpi_node_tag ∗node_tag)
- void **_starpu_mpi_early_data_add** (struct _starpu_mpi_early_data_handle ∗early_data_handle)
- struct _starpu_mpi_early_data_handle_tag_hashlist ∗ **_starpu_mpi_early_data_extract** (struct _starpu_← mpi_node_tag ∗node_tag)

### 7.7.1  Data Structure Documentation

#### 7.7.1.1  struct _starpu_mpi_early_data_handle

**Data Fields**

| starpu_data_handle_t | handle | |
|---|---|---|

**Data Fields**

| | | |
|---:|:---|:---|
| struct _starpu_mpi_req ∗ | req | |
| void ∗ | buffer | |
| size_t | size | |
| int | req_ready | |
| struct _starpu_mpi_node_tag | node_tag | |
| starpu_pthread_mutex_t | req_mutex | |
| starpu_pthread_cond_t | req_cond | |

**7.7.1.2 struct _starpu_mpi_early_data_handle_tag_hashlist**

**Data Fields**

| | | |
|---:|:---|:---|
| struct _starpu_mpi_early_data_handle_list | list | |
| UT_hash_handle | hh | |
| starpu_mpi_tag_t | data_tag | |

## 7.8 starpu_mpi_sync_data.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
```

**Functions**

- void **_starpu_mpi_sync_data_init** (void)
- void **_starpu_mpi_sync_data_check_termination** (void)
- void **_starpu_mpi_sync_data_shutdown** (void)
- struct _starpu_mpi_req ∗ **_starpu_mpi_sync_data_find** (starpu_mpi_tag_t data_tag, int source, MPI_↩
  Comm comm)
- void **_starpu_mpi_sync_data_add** (struct _starpu_mpi_req ∗req)
- int **_starpu_mpi_sync_data_count** ()

## 7.9 starpu_mpi_comm.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <mpi/starpu_mpi_mpi_backend.h>
```

**Functions**

- void **_starpu_mpi_comm_init** (MPI_Comm comm)
- void **_starpu_mpi_comm_shutdown** ()
- void **_starpu_mpi_comm_register** (MPI_Comm comm)
- void **_starpu_mpi_comm_post_recv** ()
- int **_starpu_mpi_comm_test_recv** (MPI_Status ∗status, struct _starpu_mpi_envelope ∗∗envelope, MPI_↩
  Comm ∗comm)
- void **_starpu_mpi_comm_cancel_recv** ()

## 7.10 starpu_mpi_early_request.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
#include <common/config.h>
#include <common/list.h>
```

**Data Structures**

- struct _starpu_mpi_early_request_tag_hashlist

**Functions**

- void **_starpu_mpi_early_request_init** (void)
- void **_starpu_mpi_early_request_shutdown** (void)
- int **_starpu_mpi_early_request_count** (void)
- void **_starpu_mpi_early_request_check_termination** (void)
- void **_starpu_mpi_early_request_enqueue** (struct _starpu_mpi_req ∗req)
- struct _starpu_mpi_req ∗ **_starpu_mpi_early_request_dequeue** (starpu_mpi_tag_t data_tag, int source, MPI_Comm comm)
- struct _starpu_mpi_early_request_tag_hashlist ∗ **_starpu_mpi_early_request_extract** (starpu_mpi_tag_t data_tag, int source, MPI_Comm comm)

### 7.10.1 Data Structure Documentation

#### 7.10.1.1 struct _starpu_mpi_early_request_tag_hashlist

**Data Fields**

| struct _starpu_mpi_req_list | list | |
| --- | --- | --- |
| UT_hash_handle | hh | |
| starpu_mpi_tag_t | data_tag | |

## 7.11 starpu_mpi_mpi_backend.h File Reference

```
#include <common/config.h>
#include <common/uthash.h>
```

**Data Structures**

- struct _starpu_mpi_envelope
- struct _starpu_mpi_req_backend

**Macros**

- #define **_STARPU_MPI_TAG_ENVELOPE**
- #define **_STARPU_MPI_TAG_DATA**
- #define **_STARPU_MPI_TAG_SYNC_DATA**

**Enumerations**

- enum **_starpu_envelope_mode** { **_STARPU_MPI_ENVELOPE_DATA**, **_STARPU_MPI_ENVELOPE_S**↩ **YNC_READY** }

**Variables**

- int **_starpu_mpi_tag**

### 7.11.1 Data Structure Documentation

#### 7.11.1.1 struct _starpu_mpi_envelope

**Data Fields**

| | | |
|---:|---|---|
| enum _starpu_envelope_mode | mode | |
| starpu_ssize_t | size | |
| starpu_mpi_tag_t | data_tag | |
| unsigned | sync | |

#### 7.11.1.2 struct _starpu_mpi_req_backend

**Data Fields**

| | | |
|---:|---|---|
| MPI_Request | data_request | |
| starpu_pthread_mutex_t | req_mutex | |
| starpu_pthread_cond_t | req_cond | |
| starpu_pthread_mutex_t | posted_mutex | |
| starpu_pthread_cond_t | posted_cond | |
| struct _starpu_mpi_req * | other_request | In the case of a Wait/Test request, we are going to post a request to test the completion of another request |
| MPI_Request | size_req | |
| struct _starpu_mpi_envelope * | envelope | |
| unsigned | is_internal_req:1 | |
| unsigned | to_destroy:1 | |
| struct _starpu_mpi_req * | internal_req | |
| struct _starpu_mpi_early_data_handle * | early_data_handle | |
| UT_hash_handle | hh | |
| nm_gate_t | gate | |
| nm_session_t | session | |
| nm_sr_request_t | data_request | |
| int | waited | |
| piom_cond_t | req_cond | |
| nm_sr_request_t | size_req | |

## 7.12 starpu_mpi_private.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/uthash.h>
#include <starpu_mpi.h>
#include <starpu_mpi_fxt.h>
#include <common/list.h>
#include <common/prio_list.h>
#include <common/starpu_spinlock.h>
#include <core/simgrid.h>
```

**Data Structures**

- struct _starpu_simgrid_mpi_req
- struct _starpu_mpi_node
- struct _starpu_mpi_node_tag
- struct _starpu_mpi_coop_sends
- struct _starpu_mpi_data
- struct _starpu_mpi_req
- struct _starpu_mpi_argc_argv
- struct _starpu_mpi_backend

**Macros**

- #define **STARPU_MPI_ASSERT_MSG**(x, msg, ...)
- #define **_STARPU_MPI_MALLOC**(ptr, size)
- #define **_STARPU_MPI_CALLOC**(ptr, nmemb, size)
- #define **_STARPU_MPI_REALLOC**(ptr, size)
- #define **_STARPU_MPI_COMM_DEBUG**(ptr, count, datatype, node, tag, utag, comm, way)
- #define **_STARPU_MPI_COMM_TO_DEBUG**(ptr, count, datatype, dest, tag, utag, comm)
- #define **_STARPU_MPI_COMM_FROM_DEBUG**(ptr, count, datatype, source, tag, utag, comm)
- #define **_STARPU_MPI_DEBUG**(level, fmt, ...)
- #define **_STARPU_MPI_DISP**(fmt, ...)
- #define **_STARPU_MPI_MSG**(fmt, ...)
- #define **_STARPU_MPI_LOG_IN**()
- #define **_STARPU_MPI_LOG_OUT**()

**Enumerations**

- enum **_starpu_mpi_request_type** {
  **SEND_REQ**, **RECV_REQ**, **WAIT_REQ**, **TEST_REQ**,
  **BARRIER_REQ**, **PROBE_REQ**, **UNKNOWN_REQ** }

**Functions**

- int **_starpu_mpi_simgrid_mpi_test** (unsigned ∗done, int ∗flag)
- void **_starpu_mpi_simgrid_wait_req** (MPI_Request ∗request, MPI_Status ∗status, starpu_pthread_↩ queue_t ∗queue, unsigned ∗done)
- char ∗ **_starpu_mpi_get_mpi_error_code** (int code)
- void **_starpu_mpi_env_init** (void)
- struct _starpu_mpi_data ∗ **_starpu_mpi_data_get** (starpu_data_handle_t data_handle)
- void **_starpu_mpi_submit_ready_request** (void ∗arg)
- void **_starpu_mpi_release_req_data** (struct _starpu_mpi_req ∗req)
- void **_starpu_mpi_coop_sends_build_tree** (struct _starpu_mpi_coop_sends ∗coop_sends)
- void **_starpu_mpi_coop_send** (starpu_data_handle_t data_handle, struct _starpu_mpi_req ∗req, enum starpu_data_access_mode mode, int sequential_consistency)
- void **_starpu_mpi_submit_coop_sends** (struct _starpu_mpi_coop_sends ∗coop_sends, int submit_control, int submit_data)
- void **_starpu_mpi_submit_ready_request_inc** (struct _starpu_mpi_req ∗req)
- void **_starpu_mpi_request_init** (struct _starpu_mpi_req ∗∗req)
- struct _starpu_mpi_req ∗ **_starpu_mpi_request_fill** (starpu_data_handle_t data_handle, int srcdst, starpu_mpi_tag_t data_tag, MPI_Comm comm, unsigned detached, unsigned sync, int prio, void(∗callback)(void ∗), void ∗arg, enum _starpu_mpi_request_type request_type, void(∗func)(struct _↩ starpu_mpi_req ∗), int sequential_consistency, int is_internal_req, starpu_ssize_t count)
- void **_starpu_mpi_request_destroy** (struct _starpu_mpi_req ∗req)

- void **_starpu_mpi_isend_size_func** (struct _starpu_mpi_req ∗req)
- void **_starpu_mpi_irecv_size_func** (struct _starpu_mpi_req ∗req)
- int **_starpu_mpi_wait** (starpu_mpi_req ∗public_req, MPI_Status ∗status)
- int **_starpu_mpi_test** (starpu_mpi_req ∗public_req, int ∗flag, MPI_Status ∗status)
- int **_starpu_mpi_barrier** (MPI_Comm comm)
- void **_starpu_mpi_progress_shutdown** (void ∗∗value)
- int **_starpu_mpi_progress_init** (struct _starpu_mpi_argc_argv ∗argc_argv)
- void **_starpu_mpi_wait_for_initialization** ()
- void **_starpu_mpi_data_flush** (starpu_data_handle_t data_handle)

## Variables

- starpu_pthread_wait_t **_starpu_mpi_thread_wait**
- starpu_pthread_queue_t **_starpu_mpi_thread_dontsleep**
- int **_starpu_debug_rank**
- int **_starpu_mpi_comm_debug**
- int **_starpu_mpi_fake_world_size**
- int **_starpu_mpi_fake_world_rank**
- int **_starpu_mpi_use_prio**
- int **_starpu_mpi_nobind**
- int **_starpu_mpi_thread_cpuid**
- int **_starpu_mpi_use_coop_sends**
- PRIO_struct **_starpu_mpi_req**
- struct _starpu_mpi_backend **_mpi_backend**

### 7.12.1 Data Structure Documentation

#### 7.12.1.1 struct _starpu_simgrid_mpi_req

**Data Fields**

| | | |
|---:|---|---|
| MPI_Request ∗ | request | |
| MPI_Status ∗ | status | |
| starpu_pthread_queue_t ∗ | queue | |
| unsigned ∗ | done | |

#### 7.12.1.2 struct _starpu_mpi_node

**Data Fields**

| | | |
|---|---|---|
| MPI_Comm | comm | |
| int | rank | |

#### 7.12.1.3 struct _starpu_mpi_node_tag

**Data Fields**

| | | |
|---|---|---|
| struct _starpu_mpi_node | node | |
| starpu_mpi_tag_t | data_tag | |

#### 7.12.1.4 struct _starpu_mpi_coop_sends

**Data Fields**

| struct _starpu_mpi_req_multilist_coop_sends | reqs | |
|---:|---|---|
| struct _starpu_mpi_data * | mpi_data | |
| struct _starpu_spinlock | lock | |
| struct _starpu_mpi_req ** | reqs_array | |
| unsigned | n | |
| unsigned | redirects_sent | |

**7.12.1.5  struct _starpu_mpi_data**

**Data Fields**

| int | magic | |
|---:|---|---|
| struct _starpu_mpi_node_tag | node_tag | |
| char * | cache_sent | |
| int | cache_received | |
| struct _starpu_spinlock | coop_lock | |
| struct _starpu_mpi_coop_sends * | coop_sends | |

**7.12.1.6  struct _starpu_mpi_argc_argv**

**Data Fields**

| int | initialize_mpi | |
|---:|---|---|
| int * | argc | |
| char *** | argv | |
| MPI_Comm | comm | |
| int | fargc | Fortran argc |
| char ** | fargv | Fortran argv |
| int | rank | |
| int | world_size | |

# 7.13   starpu_mpi_tag.h File Reference

```
#include <starpu.h>
#include <stdlib.h>
#include <mpi.h>
```

**Functions**

- void **_starpu_mpi_tag_init** (void)
- void **_starpu_mpi_tag_shutdown** (void)
- void **_starpu_mpi_tag_data_register** (starpu_data_handle_t handle, starpu_mpi_tag_t data_tag)
- int **_starpu_mpi_tag_data_release** (starpu_data_handle_t handle)
- starpu_data_handle_t **_starpu_mpi_tag_get_data_handle_from_tag** (starpu_mpi_tag_t data_tag)

# 7.14   starpu_mpi_datatype.h File Reference

```
#include <starpu_mpi.h>
```

```
#include <starpu_mpi_private.h>
```

**Functions**

- void **_starpu_mpi_datatype_init** (void)
- void **_starpu_mpi_datatype_shutdown** (void)
- void **_starpu_mpi_datatype_allocate** (starpu_data_handle_t data_handle, struct _starpu_mpi_req ∗req)
- void **_starpu_mpi_datatype_free** (starpu_data_handle_t data_handle, MPI_Datatype ∗datatype)
- MPI_Datatype **_starpu_mpi_datatype_get_user_defined_datatype** (starpu_data_handle_t data_handle)

## 7.15 starpu_mpi_fxt.h File Reference

```
#include <starpu.h>
#include <common/config.h>
#include <common/fxt.h>
```

**Macros**

- #define **_STARPU_MPI_FUT_START**
- #define **_STARPU_MPI_FUT_STOP**
- #define **_STARPU_MPI_FUT_BARRIER**
- #define **_STARPU_MPI_FUT_ISEND_SUBMIT_BEGIN**
- #define **_STARPU_MPI_FUT_ISEND_SUBMIT_END**
- #define **_STARPU_MPI_FUT_IRECV_SUBMIT_BEGIN**
- #define **_STARPU_MPI_FUT_IRECV_SUBMIT_END**
- #define **_STARPU_MPI_FUT_ISEND_COMPLETE_BEGIN**
- #define **_STARPU_MPI_FUT_ISEND_COMPLETE_END**
- #define **_STARPU_MPI_FUT_DATA_SET_RANK**
- #define **_STARPU_MPI_FUT_IRECV_TERMINATED**
- #define **_STARPU_MPI_FUT_ISEND_TERMINATED**
- #define **_STARPU_MPI_FUT_TESTING_DETACHED_BEGIN**
- #define **_STARPU_MPI_FUT_TESTING_DETACHED_END**
- #define **_STARPU_MPI_FUT_TEST_BEGIN**
- #define **_STARPU_MPI_FUT_TEST_END**
- #define **_STARPU_MPI_FUT_IRECV_COMPLETE_BEGIN**
- #define **_STARPU_MPI_FUT_IRECV_COMPLETE_END**
- #define **_STARPU_MPI_FUT_SLEEP_BEGIN**
- #define **_STARPU_MPI_FUT_SLEEP_END**
- #define **_STARPU_MPI_FUT_DTESTING_BEGIN**
- #define **_STARPU_MPI_FUT_DTESTING_END**
- #define **_STARPU_MPI_FUT_UTESTING_BEGIN**
- #define **_STARPU_MPI_FUT_UTESTING_END**
- #define **_STARPU_MPI_FUT_UWAIT_BEGIN**
- #define **_STARPU_MPI_FUT_UWAIT_END**
- #define **_STARPU_MPI_FUT_POLLING_BEGIN**
- #define **_STARPU_MPI_FUT_POLLING_END**
- #define **_STARPU_MPI_FUT_DRIVER_RUN_BEGIN**
- #define **_STARPU_MPI_FUT_DRIVER_RUN_END**
- #define **_STARPU_MPI_FUT_DATA_SET_TAG**
- #define **_STARPU_MPI_TRACE_START**(a, b)
- #define **_STARPU_MPI_TRACE_STOP**(a, b)
- #define **_STARPU_MPI_TRACE_BARRIER**(a, b, c)
- #define **_STARPU_MPI_TRACE_ISEND_SUBMIT_BEGIN**(a, b, c)

- #define **_STARPU_MPI_TRACE_ISEND_SUBMIT_END**(a, b, c, d)
- #define **_STARPU_MPI_TRACE_IRECV_SUBMIT_BEGIN**(a, b)
- #define **_STARPU_MPI_TRACE_IRECV_SUBMIT_END**(a, b)
- #define **_STARPU_MPI_TRACE_ISEND_COMPLETE_BEGIN**(a, b, c)
- #define **_STARPU_MPI_TRACE_COMPLETE_BEGIN**(a, b, c)
- #define **_STARPU_MPI_TRACE_COMPLETE_END**(a, b, c)
- #define **_STARPU_MPI_TRACE_TERMINATED**(a, b, c)
- #define **_STARPU_MPI_TRACE_ISEND_COMPLETE_END**(a, b, c)
- #define **_STARPU_MPI_TRACE_IRECV_COMPLETE_BEGIN**(a, b)
- #define **_STARPU_MPI_TRACE_IRECV_COMPLETE_END**(a, b)
- #define **_STARPU_MPI_TRACE_SLEEP_BEGIN**()
- #define **_STARPU_MPI_TRACE_SLEEP_END**()
- #define **_STARPU_MPI_TRACE_DTESTING_BEGIN**()
- #define **_STARPU_MPI_TRACE_DTESTING_END**()
- #define **_STARPU_MPI_TRACE_UTESTING_BEGIN**(a, b)
- #define **_STARPU_MPI_TRACE_UTESTING_END**(a, b)
- #define **_STARPU_MPI_TRACE_UWAIT_BEGIN**(a, b)
- #define **_STARPU_MPI_TRACE_UWAIT_END**(a, b)
- #define **_STARPU_MPI_TRACE_DATA_SET_RANK**(a, b)
- #define **_STARPU_MPI_TRACE_DATA_SET_TAG**(a, b)
- #define **_STARPU_MPI_TRACE_TESTING_DETACHED_BEGIN**()
- #define **_STARPU_MPI_TRACE_TESTING_DETACHED_END**()
- #define **_STARPU_MPI_TRACE_TEST_BEGIN**(peer, data_tag)
- #define **_STARPU_MPI_TRACE_TEST_END**(peer, data_tag)
- #define **_STARPU_MPI_TRACE_POLLING_BEGIN**()
- #define **_STARPU_MPI_TRACE_POLLING_END**()
- #define **_STARPU_MPI_TRACE_DRIVER_RUN_BEGIN**()
- #define **_STARPU_MPI_TRACE_DRIVER_RUN_END**()

## 7.16 starpu_mpi_select_node.h File Reference

```
#include <mpi.h>
```

**Macros**

- #define **_STARPU_MPI_NODE_SELECTION_MAX_POLICY**

**Functions**

- void **_starpu_mpi_select_node_init** ()
- int **_starpu_mpi_select_node** (int me, int nb_nodes, struct starpu_data_descr ∗descr, int nb_data, int policy)

## 7.17 starpu_mpi_task_insert.h File Reference

**Functions**

- int **_starpu_mpi_find_executee_node** (starpu_data_handle_t data, enum starpu_data_access_mode mode, int me, int ∗do_execute, int ∗inconsistent_execute, int ∗xrank)
- void **_starpu_mpi_exchange_data_before_execution** (starpu_data_handle_t data, enum starpu_data_↩ access_mode mode, int me, int xrank, int do_execute, int prio, MPI_Comm comm)
- int **_starpu_mpi_task_postbuild_v** (MPI_Comm comm, int xrank, int do_execute, struct starpu_data_descr ∗descrs, int nb_data, int prio)

## 7.18 load_balancer_policy.h File Reference

```
#include <starpu_mpi_lb.h>
```

**Data Structures**

- struct load_balancer_policy

**Variables**

- struct load_balancer_policy **load_heat_propagation_policy**

## 7.19 load_data_interface.h File Reference

```
#include <starpu.h>
```

**Data Structures**

- struct load_data_interface

**Macros**

- #define **LOAD_DATA_GET_NSUBMITTED_TASKS**(interface)
- #define **LOAD_DATA_GET_SLEEP_THRESHOLD**(interface)
- #define **LOAD_DATA_GET_WAKEUP_THRESHOLD**(interface)

**Functions**

- void **load_data_data_register** (starpu_data_handle_t *handle, unsigned home_node, int sleep_task_↩ threshold, double wakeup_ratio)
- int **load_data_get_sleep_threshold** (starpu_data_handle_t handle)
- int **load_data_get_wakeup_threshold** (starpu_data_handle_t handle)
- int **load_data_get_current_phase** (starpu_data_handle_t handle)
- int **load_data_get_nsubmitted_tasks** (starpu_data_handle_t handle)
- int **load_data_get_nfinished_tasks** (starpu_data_handle_t handle)
- int **load_data_inc_nsubmitted_tasks** (starpu_data_handle_t handle)
- int **load_data_inc_nfinished_tasks** (starpu_data_handle_t handle)
- int **load_data_next_phase** (starpu_data_handle_t handle)
- int **load_data_update_elapsed_time** (starpu_data_handle_t handle)
- double **load_data_get_elapsed_time** (starpu_data_handle_t handle)
- int **load_data_update_wakeup_cond** (starpu_data_handle_t handle)
- int **load_data_wakeup_cond** (starpu_data_handle_t handle)

### 7.19.1 Data Structure Documentation

#### 7.19.1.1 struct load_data_interface

interface for load_data

**Data Fields**

| double | start | Starting time of the execution |
|--------|-------|-------------------------------|
| double | elapsed_time | Elapsed time until the start time and the time when event "launch a load balancing phase" is triggered |

**Data Fields**

| | | |
|---:|---|---|
| int | phase | Current submission phase, i.e how many balanced steps have already happened so far. |
| int | nsubmitted_tasks | Number of currently submitted tasks |
| int | nfinished_tasks | Number of currently finished tasks |
| int | sleep_task_threshold | Task threshold to sleep the submission thread |
| int | wakeup_task_threshold | Task threshold to wake-up the submission thread |
| double | wakeup_ratio | Ratio of submitted tasks to wait for completion before waking up the submission thread |

# 7.20 data_movements_interface.h File Reference

```
#include <starpu.h>
```

**Data Structures**

- struct data_movements_interface

**Macros**

- #define **DATA_MOVEMENTS_GET_SIZE_TABLES**(interface)
- #define **DATA_MOVEMENTS_GET_TAGS_TABLE**(interface)
- #define **DATA_MOVEMENTS_GET_RANKS_TABLE**(interface)

**Functions**

- void **data_movements_data_register** (starpu_data_handle_t *handle, unsigned home_node, int *ranks, starpu_mpi_tag_t *tags, int size)
- starpu_mpi_tag_t ** **data_movements_get_ref_tags_table** (starpu_data_handle_t handle)
- int ** **data_movements_get_ref_ranks_table** (starpu_data_handle_t handle)
- int **data_movements_reallocate_tables** (starpu_data_handle_t handle, int size)
- starpu_mpi_tag_t * **data_movements_get_tags_table** (starpu_data_handle_t handle)
- int * **data_movements_get_ranks_table** (starpu_data_handle_t handle)
- int **data_movements_get_size_tables** (starpu_data_handle_t handle)

## 7.20.1 Data Structure Documentation

### 7.20.1.1 struct data_movements_interface

interface for data_movements

**Data Fields**

| | | |
|---:|---|---|
| starpu_mpi_tag_t * | tags | Data tags table |
| int * | ranks | Ranks table (where to move the corresponding data) |
| int | size | Size of the tables |

# Chapter 8

# StarPU Resource Manager File Documentation

## 8.1 starpurm_private.h File Reference

**Data Structures**

- struct s_starpurm

**Enumerations**

- enum **e_state** { **state_uninitialized**, **state_init** }
- enum **e_starpurm_unit_type** {
  **starpurm_unit_cpu**, **starpurm_unit_opencl**, **starpurm_unit_cuda**, **starpurm_unit_mic**,
  **starpurm_unit_ntypes** }

### 8.1.1 Data Structure Documentation

#### 8.1.1.1 struct s_starpurm

**Data Fields**

| | | |
|---:|---|---|
| hwloc_topology_t | topology | Machine topology as detected by hwloc. |
| unsigned | max_ncpus | Current upper bound on the number of CPU cores selectable for computing with the runtime system. |
| unsigned | selected_ncpus | Number of currently selected CPU workers |
| unsigned | selected_nworkers | Number of currently selected workers (CPU+devices) |
| int | state | Initialization state of the RM instance. |
| int | dynamic_resource_sharing | Boolean indicating the state of the dynamic resource sharing layer. !0 indicates that dynamic resource sharing is enabled. 0 indicates that dynamic resource sharing is disabled. |
| unsigned | sched_ctx_id | Id of the StarPU's sched_ctx used by the RM instance. |
| int | unit_ntypes | Number of unit types supported by this RM instance. |
| int ∗ | nunits_by_type | Number of unitss available for each type. |
| int | nunits | Number of units. |
| int ∗ | unit_offsets_by_type | Offset of unit numbering for each type. |
| struct s_starpurm_unit ∗ | units | Array of units. |

**Data Fields**

| | | |
|---:|---|---|
| hwloc_cpuset_t | global_cpuset | Cpuset of all the StarPU's workers (CPU+devices. |
| hwloc_cpuset_t | all_cpu_workers_cpuset | Cpuset of all StarPU CPU workers. |
| hwloc_cpuset_t | all_opencl_device_workers_cpuset | Cpuset of all StarPU OpenCL workers. |
| hwloc_cpuset_t | all_cuda_device_workers_cpuset | Cpuset of all StarPU CUDA workers. |
| hwloc_cpuset_t | all_mic_device_workers_cpuset | Cpuset of all StarPU MIC workers. |
| hwloc_cpuset_t | all_device_workers_cpuset | Cpuset of all StarPU device workers. |
| hwloc_cpuset_t | selected_cpuset | Cpuset of all selected workers (CPU+devices). |
| hwloc_cpuset_t | initially_owned_cpuset_mask | Cpuset mask of initially owned cpuset or full if not used. |
| int | max_worker_id | maximum value among worker ids |
| int ∗ | worker_unit_ids | worker id to unit id table |
| unsigned int | max_temporary_ctxs | Temporary contexts accounting. |
| unsigned int | avail_temporary_ctxs | |
| pthread_mutex_t | temporary_ctxs_mutex | |
| pthread_cond_t | temporary_ctxs_cond | |
| int | starpu_in_pause | Global StarPU pause state |
| pthread_t | event_thread | Event list. |
| pthread_mutex_t | event_list_mutex | |
| pthread_cond_t | event_list_cond | |
| pthread_cond_t | event_processing_cond | |
| int | event_processing_enabled | |
| int | event_processing_ended | |
| struct s_starpurm_event ∗ | event_list_head | |
| struct s_starpurm_event ∗ | event_list_tail | |