GLT API proposal

Adrián Castelló 02/23/2016

GLT (Generic Lightweight Threads)

Common API for Qthreads, MassiveThreads and Argobots libraries

Two implementations

- Static Inline functions (needs recompilation)
- Shared library

Three approaches (to be discussed)

1. Only functionality present in all libraries
    - Pros: Perfect for the library with minimum functionality
    - Cons: Lot of functionality is lost
2. All functionality for all libraries
    - Pros: All functionality is offered
    - Cons: Functionality that is not offered in a library
        - Try to reproduce the functionality with library functions
        - Error Message if it can not be supported
3. Just common functionality and the one that can be replaced with current features (e.g. Tasklets function replaced with ULT functions)
    - Pros: Most of the overall functionality
    - Cons: Specific library functions are discarded

GLT Objects

GLT object/ LWT	Qthreads	MassiveThreads	Argobots
GLT_ult	aligned_t	myth_thread_t	ABT_thread
GLT_tasklet	aligned_t	myth_thread_t	ABT_task
GLT_thread	aligned_t	myth_thread_t	ABT_xstream
GLT_mutex	aligned_t	myth_mutex_t	ABT_mutex
GLT_barrier	qt_barrier_t	myth_barrier_t	ABT_barrier
GLT_cond	aligned_t	myth_cond_t	ABT_cond
GLT_event	X*	X*	ABT_event
GLT_future	X*	X*	ABT_eventual
GLT_promise	X*	X*	ABT_future
GLT_timer	qtimer_t	timeval	ABT_timer

*Not supported
Should be included?

GLT Core

GLT function/ LWT	Qthreads	MassiveThreads	Argobots
glt_init	✓	✓	✓
glt_finalize	✓	✓	✓
glt_ult_malloc	✓	✓	✓
glt_ult_creation	✓	✓	✓
glt_ult_creation_to	✓	✓	✓
glt_yield	✓	✓	✓
glt_ult_join	✓	✓	✓
glt_ult_migrate	X	X	✓+
glt_ult_migrate_to	✓*	X	✓+

* Qthreads migrates the current thread to a specific shepherd
+ Argobots migrates a specific thread to a specific destination

GLT Core II

GLT function/ LWT	Qthreads	MassiveThreads	Argobots
glt_ult_get_id	✓*	X	✓+
glt_get_thread_num	✓	✓	✓
glt_get_num_threads	✓	✓	✓
glt_get_wtime	✓'	✓'	✓
glt_timer_create/free	✓	✓'	✓
glt_timer_start/stop	✓	✓'	✓
glt_timer_get_secs	✓	✓'	✓

* return current thread id
+ return a given thread id
'Implemented with gettimeofday

GLT Synchronization

GLT function/ LWT	Qthreads	MassiveThreads	Argobots
glt_mutex_create/free	aligned_t*	✓	✓
glt_mutex_lock/unlock	✓	✓	✓
glt_barrier_create/free	✓	✓	✓
glt_barrier_wait	✓	✓	✓
glt_cond_create/free	aligned_t*	✓	✓
glt_cond_signal	qthread_empty	✓	✓
glt_cond_wait	qthread_writeEF	✓	✓
glt_cond_broadcast	qthread_empty+	✓	✓

*Locks affect to memory address
+Act as the signal function

GLT Extended

GLT function/ LWT	Qthreads	MassiveThreads	Argobots
glt_tasklet_malloc	glt_ult_malloc	glt_ult_malloc	✓
glt_tasklet_creation	glt_ult_creation	glt_ult_creation	✓
glt_tasklet_creation_to	glt_ult_creation_to	glt_ult_creation_to	✓
glt_tasklet_join	glt_ult_join	glt_ult_join	✓

As tasklets are not supported by Qthreads and MassiveThreads, ULTs are managed by these functions.

GLT Extended II (to be discussed)

GLT function/ LWT	Qthreads	MassiveThreads	Argobots
glt_event_XX	X	X	ABT_event_XX
glt_future_XX	X	X	ABT_eventual_XX
glt_promise_XX	X	X	ABT_future_XX

As events, futures and promise are not supported by qthreads and massivethreads. Should this functionality not to be in the common API?

Environment variables

- GLT_NUM_THREADS: Controls the number of OS threads (Execution Streams, Shepherds and Workers) (default 1)
- Argobots
  - GLT_NUM_POOLS: Controls the number of pools. If less pools than threads are created, they are assigned using a round robin mechanism (default 1)
- Qthreads
  - GLT_NUM_WORKERS_PER_THREAD: Controls the number of workers per thread. If just one thread and more than 1 worker are created, the thread is bound to a node and each worker to a cpu. If more than one thread is created it is bounded to a cpu as well as the workers. (default 1)
- MassiveThreads
  - There are no specific environment variables

If these environment variables are not set, selected library ones need to be specified