Shared-Memory Programming APIs

Pthreads

Diego Fabregat-Traver and Prof. Paolo Bientinesi

HPAC, RWTH Aachen
fabregat@aices.rwth-aachen.de

WS15/16

Outline

Shared-memory parallel programming APIs

- POSIX threads (pthreads)
  - Focus: task parallelism
  - Relatively low level
  - Explicit management of threads
  - Available on Unix systems. In Windows: Windows threads.
- Threading Building Blocks (TBB)
  - Based on C++ templates
  - Focus: task parallelism
  - Available on Windows, Linux, OSX, Android, ...
- OpenMP
  - Higher level of abstraction
  - Objective: high performance
  - Focus on data parallelism
  - Support for task parallelism

Pthreads

- Threading interface (IEEE POSIX standard)
- Available on UNIX-like systems (e.g., Linux, MacOSX)
- Library based:
  - Thread creation, management, termination
  - Thread synchronization, mutual exclusion
- Powerful and flexible
- Explicit, *low level*
- Task parallelism

Pthreads API

- Over 100 functions:
  - `#include <pthread.h>`
- Thread management
  - `pthread_create( ... )`
  - `pthread_join( ... )`
  - `pthread_exit( ... )`
  - `pthread_self( ... )`
- Thread synchronization
  - `pthread_barrier_*( ... )`
- Thread exclusive access
  - `pthread_mutex_init( ... )`
  - `pthread_mutex_destroy( ... )`
  - `pthread_mutex_lock( ... )`
  - `pthread_mutex_unlock( ... )`
- And many more...

Examples

See code samples in the course's webpage

- Basic scheme
  - `01.pthread_skeleton.c`
- Passing arguments
  - `02.pthread_pass_id.c`
  - `03.pthread_axpy.c`
- Critical sections and mutexes
  - `06.pthread_dot-v2.c`
- Task parallelism
  - `07.task-server.c` (pseudocode/ideas)

Key points to remember

- Thread creation: pass the function to be executed
- Passing data: data structures passed as arguments
  - Explicitly pack and unpack
- Loop parallelism: explicit calculation of loop bounds
- Use of mutexes and other objects to synchronize and ensure exclusive access:
  - Reductions
  - Access to queues