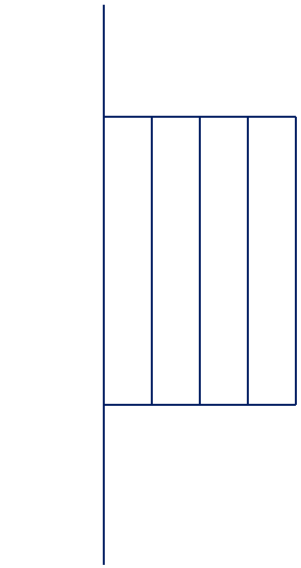

Advanced OpenMP
Lecture 10: Alternatives to OpenMP

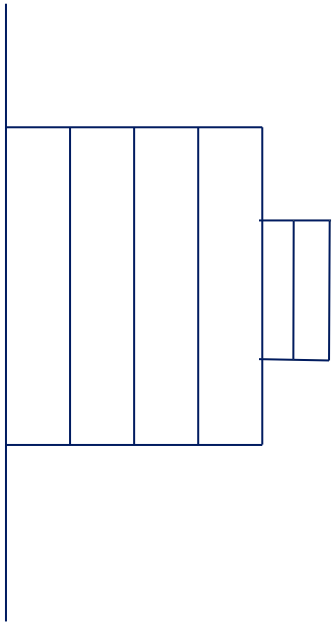
What's wrong with OpenMP?

- OpenMP is designed for programs where you want a fixed number of threads, and you always want the threads to be consuming CPU cycles.
  - cannot arbitrarily start/stop threads
  - cannot put threads to sleep and wake them up later
- OpenMP is good for programs where each thread is doing (more-or-less) the same thing.
- Although OpenMP supports C++, it's not especially OO friendly
  - though it is gradually getting better.
- OpenMP doesn't support Java
- OpenMP programs don't run on distributed memory architectures (e.g. clusters of PCs)

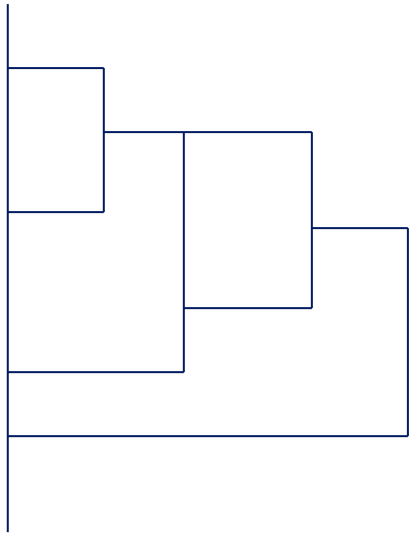
What's wrong with OpenMP? (cont.)



Can do this



Can do this



Can't do this

What are the alternatives?

- Threading libraries
  - POSIX threads
  - Boost threads
  - Intel TBB
  - Java threads

- Other programming models
  - MPI
  - PGAS languages
  - Novelties

Threading libraries

- Pure library interfaces (no directives)
- Lower level of abstraction than OpenMP
  - requires more changes to code
  - harder to maintain
  - some are less portable than OpenMP
- Free software

Programming model

- Similar to OpenMP in many respects
  - shared and private data
  - similar synchronisation issues
- Routine to create a new thread
  - pass as arguments a function for the new thread to execute, plus some arguments
  - thread dies when the passed function exits
- Routines for synchronising threads
  - wait for a thread to exit
  - locks, barriers, semaphores, condition variables,….
- Some libraries have the notion of tasks
  - pass a function to a thread pool: task gets executed by one of the threads sometime in the future

POSIX threads

- POSIX threads (or Pthreads) is a standard library for shared memory programming without directives.
  - Part of the ANSI/IEEE 1003.1 standard (1996)
- Interface is a C library
  - no standard Fortran interface
  - can be used with C++, but not OO friendly
- Widely available
  - even for Windows
  - typically installed as part of OS
  - code is pretty portable
- Lots of low-level control over behaviour of threads

Thread forking

```
#include <pthread.h>


int pthread_create(
          pthread_t *thread,
          const pthread_attr_t *attr,
          void*(*start_routine, void*),
          void *arg)
```

- Creates a new thread:
  - first argument returns a pointer to a thread descriptor.
  - can set attributes.
  - new thread will execute `start_routine(arg)`
  - return value is error code.

Thread joining

```
#include <pthread.h>

int pthread_join(
        pthread_t thread,
        void **value_ptr)
```

- Waits for the specified thread to finish.
  - thread finishes when **`start_routine`** exits
  - second argument holds return value from **`start_routine`**

Synchronisation

- Barriers
- Mutex locks
  - Behaviour is essentially the same as the OpenMP lock routines.
- Condition variables
  - Behaviour is essentially the same as wait/notify in Java

Hello World

```
#include <pthread.h>
#define NTHREADS 5
int i, threadnum[NTHREADS];
pthread_t tid[NTHREADS];

for (i=0; i<NTHREADS; i++) {
  threadnum[i]=i;
  pthread_create(&tid[i], NULL, hello, &threadnum[i]);
}

for (i=0; i<NTHREADS; i++)
   pthread_join(tid[i], NULL);
```

Hello World (cont.)

```
void* hello (void *arg) {
    int myid;

    myid = *(int *)arg;
    printf("Hello world from thread %d\n", myid);

    return (0);
}
```

BOOST threads

- C++ library for multithreaded programming
- Similar functionality to POSIX threads
  - but with a proper OO interface
- Closest thing to a C++ standard
  - new version of C++ standard (C++11) contains a standard thread library which is similar to BOOST threads.
  - will take some time for stable compilers to appear
- Reasonably portable
  - need to install and build library
  - should be OK for most common C++ compilers

- **`http://www.boost.org/doc/libs/1_40_0/doc/html/thread.html`**

Hello world

```
#include <boost/thread/thread.hpp>
#include <iostream>

void hello()
{
  std::cout <<  "Hello world" << std::endl;
}

int main(int argc, char* argv[])
{
  boost::thread thrd(&hello);
  thrd.join();
  return 0;
}
```

Intel Thread Building Blocks (TBB)

- C++ library for multithreaded programming
- Offers somewhat higher level of abstraction that POSIX/ BOOST threads
  - notion of tasks rather that explicit threads
  - support for parallel loops and reductions
  - support for concurrency on containers
- Moderately portable
  - support for Intel and gcc compilers on Linux and Mac OS X, Intel and Visual C++ on Windows
  - no build required to install

- **`http://www.threadingbuildingblocks.org`**

Hello World

```
#include <iostream>
#include <tbb/parallel_for.h>

using namespace tbb;

class Hello
{
public:
void operator()(int x) const {
std::cout << "Hello world\n";
}
};

int main()
{
// parallelizing:
// for(int i = 0; i < 2; ++i) { ... }
parallel_for(0, 2, 1, Hello());

return 0;
}
```

Java threads

- Threads are an inbuilt part of the Java language
- Very portable (available in every Java VM)
- Java has lots of nice properties as a programming language
  - high performance isn't necessarily one of them!
- Well integrated into Java's OO model
- Both explicit thread creation and task models
- Synchronisation methods
  - every object contains a mutex lock
  - condition variables, barriers, explicit lock objects

- `http://java.sun.com/docs/books/tutorial/essential/concurrency/`

MPI

- Message Passing Interface (MPI)
- Principal method used for programming distributed memory architectures
  - library interface for Fortran, C, C++
- MPI programs will run just fine on multicore systems
- Need to install an MPI library and configure it to use shared memory to pass messages
- MPI programs typically take much longer to develop than OpenMP programs (months not weeks)
  - extremely portable and scalable solution
- Hybrid OpenMP/MPI becoming a common solution for very large systems

PGAS languages

- Partitioned Global Address Space
- Genuine language extensions
  - Co-array Fortran
  - Unified Parallel C
- Designed to work on distributed memory systems and be easier to write than MPI
  - will also work on shared memory
- Lack of mature, robust implementations that actually deliver good performance
- Not terribly portable at the moment

Novelties

- As multicore programming becomes mainstream, a lot of new programming languages/APIs have appeared
  - some commercial, some academic
  - Intel ArBB, Cilk, OpenCL, HMPP, OpenACC, StarSs, X10, Chapel,….
- Some are designed to address accelerator devices such as GPUs as well as “traditional” multicore.
- Unclear which (if any) will become popular/mature/widely implemented.
- Probably best avoided in the short term, unless you wish to experience a lot of pain!

What to do?

- OpenMP is a good solution for many programs in the scientific computing area.
- However, there may be very sound reasons for not using OpenMP.
- If you are going to use one of the alternatives, you should convince yourself carefully of why you are not using OpenMP!