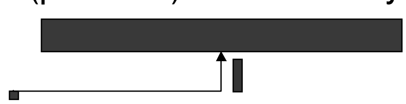
MPI-2 and Threads

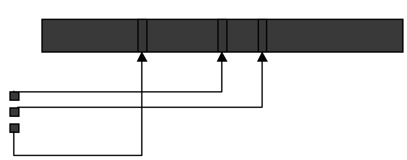
1

What are Threads?

- Executing program (process) is defined by
  - » Address space
  - » Program Counter



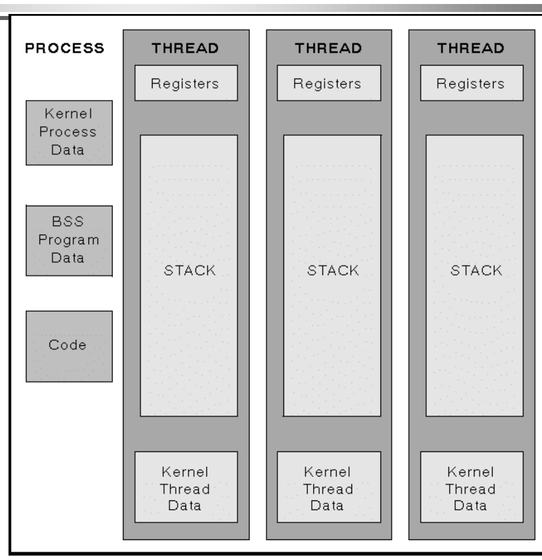
- Threads are multiple program counters



2

Inside a Thread

- http://www.spc.ibm.com/spcdocs/aixdocs/aix41gthr.html#threads



3

Kinds of Threads

- Almost a process
  - » Kernel (Operating System) schedules
  - » each thread can make independent system calls
- Co-routines
  - » User schedules (sort of…)
- Memory references
  - » Hardware schedules

4

Kernel Threads

- System calls (e.g., read, accept) block calling thread but not process
- Alternative to “nonblocking” or “asynchronous” I/O:
  - » create_thread
    thread calls blocking read
- Can be expensive

5

User Threads

- System calls (may) block all threads in process
- Allows multiple processors to cooperate on data operations
  - » loop: create # threads = # processors - 1
    each thread does part of loop
- Cheaper than kernel threads
  - » Still must save registers (if in same processor)
  - » Parallelism requires OS to schedule threads on different processors

6

Hardware Threads

- Hardware controls threads
- Allows single processor to interleave memory references and operations
  - » Unsatisfied memory ref changes thread
  - » Separate registers for each thread
- Single cycle thread switch with appropriate hardware
  - » basis of Tera MTA computer http://www.tera.com
  - » like kernel threads, replaces nonblocking hardware operations - multiple pending loads
  - » Even lighter weight—just change PC

Why Use Threads?

- Manage multiple points of interaction
  - » Low overhead steering/probing
  - » Background checkpoint save
- Alternate method for nonblocking operations
  - » CORBA method invocation (no funky nonblocking calls)
- Hiding memory latency
- Fine-grain parallelism
  - » Compiler parallelism

Latency Hiding

Thread Interfaces

- POSIX "pthreads"
- Windows
  - » Kernel threads
  - » User threads called "fibers"

Library-based

Invoke a routine in a separate thread

- Java
  - » First major language with threads
  - » Provides memory synchronization model: methods (procedures) declared "synchronized" executed by one thread at a time
  - » (don't mention Ada, which had tasks)
- OpenMP (Fortran only for now)
  - » Mostly directive-based parallel loops
  - » Some thread features (lock/unlock)
  - » http://www.openmp.org

9

Thread Issues

- Synchronization
  - » Avoiding conflicting operations
- Variable Name Space
  - » Interaction between threads and the language
- Scheduling
  - » Will the OS do what you want?

10

Synchronization of Access

- Read/write model

```
a = 1;                b = 1;
barrier();            barrier();
b = 2;                while (a==1) ;
a = 2;                printf( "%d\n", b );
```

  What does thread 2 print?
- Need lock/unlock to synchroni 2 order
  - » OpenMP has `FLUSH`, possibly worse
  - » `volatile` in C
  - » Fortran has no corresponding concept
- Java has "synchronized" methods (procedures)

Variable Names

- Each thread can access all of a processes memory (except for the thread's stack)
  - » Named variables refer to the address space—thus visible to all threads
  - » Compiler doesn't distinguish `A` in one thread from `A` in another
  - » No modularity
  - » Like using Fortran blank COMMON for all variables
- NEC has a variant where all variables names refer to different variables unless specified
  - » All variables are on thread stack by default (even globals)
  - » More modular

Scheduling Threads

- If threads used for latency hiding
  - » Schedule on the same processor
    - – Provides better data locality, cache usage
- If threads used for parallel execution
  - » Schedule on different processors using different memory pathways

13

The Changing Computing Model

- More interaction
  - » Threads allow low-overhead agents on any compution
    - – OS schedules if necessary; no overhead if nothing happens (almost…)
  - » Changes the interaction model from batch (give commands, wait for results) to constant interaction
- Fine-grain parallelism
  - » Simpler SMP programming model
- Lowering the Memory Wall
  - » CPU speeds increasing much faster than memory
  - » hardware threads hide memory latency

14

Threads and MPI

- `MPI_Init_thread(&argc,&argv,required,&provided)`
  - » Thread modes:
    - – MPI_THREAD_SINGLE — One thread (MPI_Init)
    - – MPI_THREAD_FUNNELED — One thread making MPI calls
    - – MPI_THREAD_SERIALIZED — One thread at a time making MPI calls
    - – MPI_THREAD_MULTIPLE — Free for all
- Coexist with compiler (thread) parallelism for SMPs
- MPI could have defined the same modes on a communicator basis (more natural, and MPICH will do this through attributes)

15

Using Threads with MPI

- MPI defines what it means to support threads but does not require that support
  - » Some vendors (such as IBM and Sun) support multi-threaded MPI processes
  - » Others (such as SGI) do not
    - – Interoperation with other thread systems (essentially MPI_THREAD_FUNNELED) may be supported
- Active messages, interrupt receives, etc. are essentially MPI calls, such as a blocking receive, in a separate thread

16