2.6 Parallel programming

2.6.1 Thread parallelism

More complicated than sequential programming, expose the parallelism to the programmer and let the programmer manage everything explicitly.

What is thread?

A unix process has in memory: the program code, a heap, a stack with quick-changing information.

Process can have multiple threads, they see the same program code and heap, but their own stack.

The fork-join mechanism

A program has one master thread, other threads are created by thread spawning and the master thread can wait for their completion.

The totality of all data that a thread can access is called its context.

A processor switching between the execution of different threads is called context switch.

2.6.2 Open MP

It is not a language but an extension to the existing C and Fortran languages.

Open MP is the way to program shared memory.

2.6.3 Distributed memory programming through message passing

Blocking communication instructions: a send instruction does not finish until the send item is actually received, and a receive instruction waits for the corresponding send.

Non-blocking communication instructions, the user needs to allocate a buffer for each send and check when it is safe to overwrite the buffer.

Message Passing Interface (MPI) is the standard solution for programming distributed memory.

MPI routines: a) process management; point-to-point communication; collective calls.

Collective operations: reduction, broadcast, gather, scatter, all-to-all

2.6.4 Hybrid shared/distributed memory computing

Using MPI to communicate between the nodes (inter-node communication) and Open MP for parallelism on the node (intra-node communication).

The hybrid approach offers some advantages since it bundles messages.

2.6.5 Parallel languages

Some languages reflect the fact that many operations in scientific computing are data parallel.

Partitional Global Address Space model: there is only one address space but this address space is partitioned and each partition has affinity with a thread or process.

Parallel language: Unified Parallel C, High performance Fortran, Co-array Fortran, Chapel, Fortress, X10, Linda.

2.6.6 OS-based approaches

All data was considered to be cached on a processor and moved through the network on demand.

2.6.7 Active messages

With active messages, one processor can send data to another without that second processor doing an explicit receive operation.

Remote method invocation

2.6.8 Bulk synchronous parallelism

Here the programmer needs to spell out communications but not their ordering. The BSP model orders the program into a sequence of supersteps, each of which ends with a barrier synchronization.

2.6.9 Data dependencies

If two statements refer to the same data item, we say that there is a data dependency between the statements.

Types of data dependencies: a) flow dependencies, or ‘read-after-write’; b) anti dependencies, or ‘write-after-read’; c) output dependencies, or ‘write-after-write’.

2.6.10 Program design for parallelism

Parallel data structures, using Array-Of-Structure vs Structure-Of-Array.

Latency hiding. There has to be enough work per processor to offset the communication. Or called overlapping computation with communication.

Questions

1. Why will it give the wrong result when we increase the time for the update? P96.

2. Why would blocking communication occur?