IE 495 Lecture 8

September 21, 2000

Reading for This Lecture

- Primary
  - AHU, Chapter 2
- Secondary
  - Horowitz and Sahni, Chapter 2, Section 1

Parallel Algorithm Design

Review from last lecture

Design Issues

- Platform/Architecture
- Task Decomposition
- Task Mapping/Scheduling
- Communication Protocol

Platforms

- High Performance Parallel Computers
  - Massively parallel
  - Distributed
- "Off the shelf" Parallel Computers
  - Small shared memory servers
  - Virtual parallel computers

Approaches to Task Decomposition

- Fine-grained parallelism
  - Suited for massively parallel systems (many small processors)
  - These are the algorithms we've primarily talked about so far .
- Course-grained parallelism
  - Suited to small numbers of more powerful processors.
  - Data decomposition
    - Recursion/Divide and Conquer
    - Domain Decomposition
  - Functional parallelism
    - Data Dependency Analysis
  - These algorithms are more common and easier to implement.

Approaches to Mapping

- Concurrency
  - Data dependency analysis
- Locality
  - Interconnection network
  - Communication pattern
- Mapping is an optimization problem.
- These are very difficult to solve in general.

Communication Protocols

Message-passing

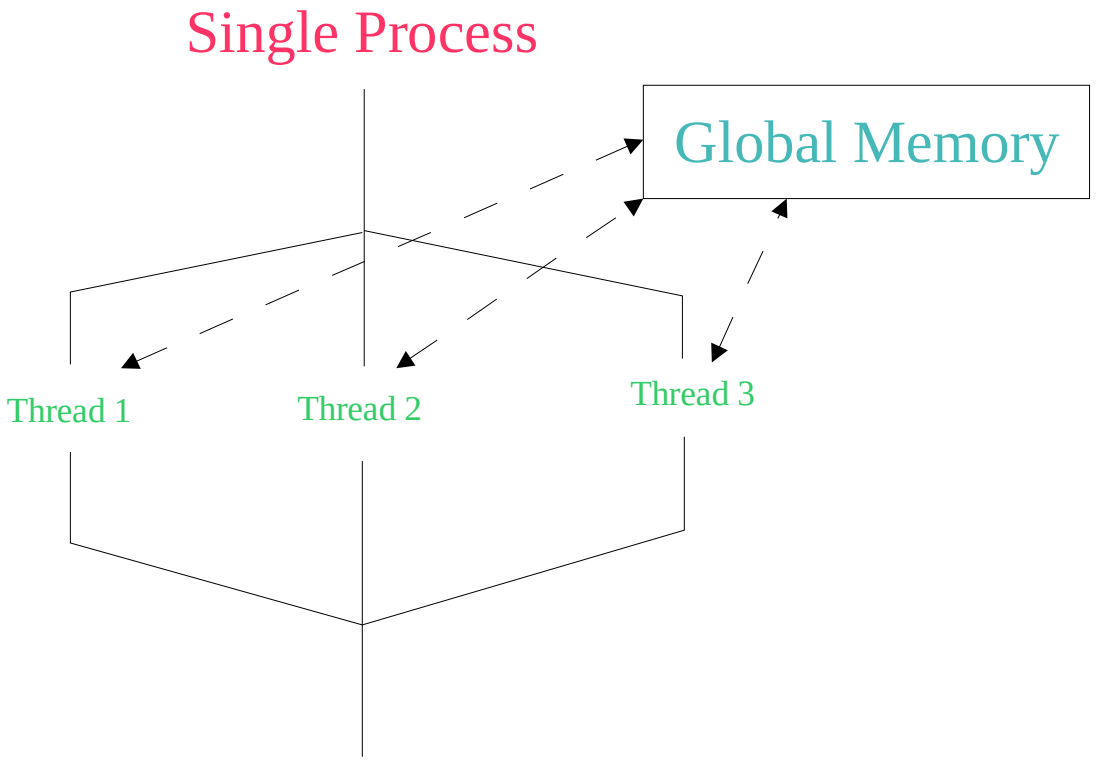
- Used primarily in distributed-memory or "hybrid" environments.
- Data is passed through explicit send and receive function calls.
- There is no explicit synchronization.
- In general, this is the most flexible and portable protocol.
- PVM and MPI are the established standards.

Comunication Protocols

OpenMP/Threads

- Used in shared-memory environments.
- Parallelism through "threading".
- Threads communicate through global memory.
- Can have explicit synchronization.
- OpenMP is the emerging standard.

OpenMP/Threads



OpenMP Implementation

- OpenMP is implemented through compiler directives.
- User is responsible for indicating what code segments should be performed in parallel.
- The user is also responsible for eliminating potential memory conflicts, etc.
- The compiler is responsible for inserting platform-specific function calls, etc.

OpenMP Features

- Capabilities are dependent on the compiler.
  - Primarily used on shared-memory architectures
  - Can work in distributed-memory environments (TreadMarks)
- Explicit synchronization
- Locking functions
- Critical regions
- Private and shared variables

Using OpenMP

- Compiler directives
  - *parallel*
  - *parallel for*
  - *parallel sections*
  - *barrier*
  - *private*
  - *critical*
- Shared library functions
  - omp_get_num_threads()
  - omp_set_lock()

OpenMP Example

OpenMP Concepts and Issues

- Race Conditions
  - Conflicts between processes in updating data.
- Deadlocks
- Critical regions
- Lock functions

And Now For Something Completely Different...

Basic Data Structures

What is a data structure?

- Data structures are schemes for organizing and storing sets.
- Data structures make it easy to perform certain set operations.
- Examples of set operations.
  - add
  - delete
  - find_min
  - delete_min
  - union

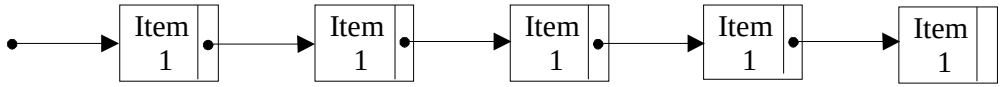
Choosing the right data structure

- Data structures consist of
  - a scheme for storing the set(s), and
  - algorithms for performing the desired operations
- Hence, each set operation has an associated complexity
- To choose a data structure, you should know
  - something about the elements of the set, and
  - what operations you will want to perform on the set.

Example: Lists

- A list is a finite sequence of elements drawn from a set
- List operations
  - *`insert()`*
  - *`delete()`*
  - *`concatenate()`*
  - *`split()`*
- List storage
  - array
  - linked list

Linked Lists



	NAME	NEXT
0	-	1
1	Item 1	3
2	Item 2	0
3	Item 3	4
4	Item 4	2
5	Empty	0