Parallel Programming Environment And Methods

Mayur Hadpe
Nikhilesh Sahani
INTRODUCTION:

- The environment within which parallel programs are constructed is called the *parallel programming environment*.

- To implement a parallel algorithm a parallel program is required.

- New architectures have emerged introducing parallelism in computing systems.

- Currently, various kinds of high performance machines based on the integration of many processors are available.
Key Parallel Programming Steps:

- To find the concurrency in the given problem.
- To structure the algorithm so that concurrency can be exploited.
- To implement the algorithm in a suitable programming environment.
- To execute and tune the performance of the code on a parallel system.
What Is Parallel Computing?

In the simplest sense, parallel computing is the simultaneous use of multiple computing resources to solve a computational problem.

- To be run using multiple CPUs
- A problem is broken into discrete parts that can be solved concurrently.
- Each part is further broken down to a series of instructions.

- Instructions from each part execute simultaneously on different CPUs.
Sequential to Parallel Conversion
Why Parallel Computing?

➢ This is a legitimate question. Parallel computing is complex in any aspect.

➢ The primary reasons for using parallel computing:
  ❖ Save time - wall clock time
  ❖ Solve larger problems
  ❖ Provide concurrency (do multiple things at the same time)

➢ Other reasons might include:
  ❖ Taking advantage of non-local resources - using available compute resources on a wide area network, or even the Internet when local compute resources are scarce.
PARALLEL PROGRAMMING ARCHITECTURES:

- Shared Memory
- Distributed Memory
- Hybrid Distributed-Shared Memory
Shared Memory:

- Multiple processors can operate independently but share the same memory resources.
- Changes in a memory location made by one processor are visible to all other processors.
- Shared memory machines can be divided into two main classes based upon memory access times: UMA and NUMA.
Distributed Memory:

- Processors have their own local memory.
- It operates independently, changes it makes to its local memory have no effect on the memory of other processors.
MPI v/s PVM

- MPI is a standardized and portable message passing system.
- MPI is language independent. Can be implemented using Java and Matlab.

- PVM is a software tool that allows connection of heterogeneous machines that can be viewed as a single distributed parallel processor.
- PVM uses existing hardware. Cost effective in term of hardware requirement.
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>MPI</th>
<th>PVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>It is an interface and can be implemented by most of the programming languages.</td>
<td>It is a software tool for networking of parallel computers.</td>
</tr>
<tr>
<td></td>
<td>Dynamic task creation is not supported.</td>
<td>Supports Dynamic Task Creation.</td>
</tr>
<tr>
<td></td>
<td>Wide range of abilities.</td>
<td>Limited abilities.</td>
</tr>
<tr>
<td></td>
<td>Can not control task creation and task termination</td>
<td>Can control task initiation and task termination.</td>
</tr>
<tr>
<td></td>
<td>Poor Resource Management.</td>
<td>Better resource Management</td>
</tr>
<tr>
<td></td>
<td>Used widely as can be implemented by numerous programming languages.</td>
<td>Used in a select few applications.</td>
</tr>
</tbody>
</table>
Hybrid Distributed-Shared Memory:

- The largest and fastest computers in the world today employ both shared and distributed memory architectures.

- Advantages and Disadvantages: whatever is common to both shared and distributed memory architectures.
Overview:

- There are several parallel programming models in common use:
  - Shared Memory
  - Threads
  - Message Passing
  - Data Parallel
  - Hybrid

- Parallel programming models exist as an abstraction above hardware and memory architectures.

- Which model to use is often a combination of what is available and best suited to a problem. There is no "best" model, although there are certainly better implementations of some models over others.
Shared Memory Model:

- In the shared-memory programming model, tasks share a common address space, which they read and write asynchronously.

- Various mechanisms such as locks / semaphores may be used to control access to the shared memory.

- An important disadvantage in terms of performance is that it becomes more difficult to understand and manage data locality.
Message Passing Model:

- The message passing model demonstrates the following characteristics:
  - A set of tasks that use their own local memory during computation. Multiple tasks can reside on the same physical machine as well across an arbitrary number of machines.
  - Tasks exchange data through communications via sending and receiving messages.
  - Data transfer usually requires cooperative operations to be performed by each process. For example, a send operation must have a matching receive operation.
 THREAD MODELS:

- In the threads model of parallel programming, a single process can have multiple, concurrent execution paths.
  - Any thread can execute any subroutine at the same time as other threads.
  - **Threads communicate** with each other through global memory (updating address locations). This requires synchronization constructs to ensure that more than one thread is not updating the same global address at any time.
- Threads are commonly associated with shared memory architectures and operating systems.
HYBRID MODEL:

- In this model, any two of the parallel programming models are combined.

- Currently, a common example of a hybrid model is the combination of the message passing model (MPI) with either the threads model (POSIX threads) or the shared memory model (OpenMP).

- Another common example of a hybrid model is combining data parallel with message passing.
Types of Parallelism:

There are two basic types of parallelism that can be explored in the development of parallel programs:

1. Data parallelism
2. Functional parallelism

In the case of data parallelism, the same instruction set is applied to multiple items of a data structure. This programming model can be easily implemented for shared memory machines.

In the case of functional parallelism, the program is partitioned into cooperative tasks. Each task can execute a different set of functions/code and all tasks can run asynchronously.
A Parallel Programming Environment Based on Message Passing

- Parallel computer systems with multiprocessors have powerful computing ability, on the contrary, it makes the parallel programming very difficult.
- So it is very important to design an efficient and flexible parallel programming environment, in which the users can do parallel computing more easily.
- This parallel programming environment on SPARC network workstations and Transputer-based parallel accelerator.
Parallel Programming Environment Architecture:

- The parallel programming environment is constructed in a multi-layer architecture.
- On the lowest level of hardware/operating system, we form a parallel Runtime System Kernel which controls:
  1. the management of processor hardware
  2. division of parallel computing tasks
  3. processor allocation
  4. Message passing between the processors.
- The upper layer is a parallel library which provides a lot of parallel functions for the users to do processor allocation, message passing, application downloading and runtime monitoring.
Architecture Cont...:

- On the highest level of the parallel programming environment, the system provides a set of toolkits.

- Through these tools, users can:
  - monitor the running applications
  - Evaluate and enhance the performance of parallel programs.

- The NDB (Network Debug) tool provides the ability to do parallel debug to the parallel applications.
Parallel Programming Environment Mechanism:

- When doing parallel programming, it contains the following main steps:
  - processor allocation.
  - application downloading.
  - data exchanging through message passing.
  - gathering results from the parallel processing nodes.
Communication Modes:

- The parallel programming environment is based on the asynchronous point-to-point message passing system.
- The system provides two kinds of communication modes, synchronous and asynchronous message passing.
- The asynchronous message passing "read" function call blocks till the matching message has arrived.
- In the message passing model, processors communicate by sending and receiving messages in blocking or non-blocking mode.
Blocking and Non-Blocking Mode:

- **Sending mode:**
  - **Blocking mode:**
    A task sending a message stops its processing and waits until the destination task receives it.
  - **Non-blocking mode:**
    It sends the message and continues its execution.

- **Receiving mode:**
  - **Blocking mode:**
    A task stops its processing until a specific message arrives.
  - **Non-blocking mode:**
    It just checks whether there is a message for it and carries on its processing if not.
THE TWO BASIC FUNCTIONS IN THE PARALLEL PROGRAMMING ENVIRONMENT ARE "EXREAD" AND "EXWRITE".

THEY ARE USED ON THE HOST AND NODES FREELY. THE CALLING SYNTAX IS AS FOLLOW:

```
status = exwrite(*buffer, length, &dest, &type);
```

WHEN RECEIVING THE MESSAGES IT IS CALLED AS:

```
status = exread(*buffer, length, &src, &type);
```
NON-BLOCKING COMMUNICATION:

- Some applications, such as real-time control, will be not easily met by the blocking functions.

- In these applications, it is important to react quickly and flexibly to input data and "pipelined" operations in which one may wish to process one set of data while waiting for another to arrive.
The non-blocking function `exhandle` provides a mechanism for "handling" messages as soon as they arrived at a processor. The calling sequence:

```c
status = exhandle(*func, &src, &type);
```

Another form of non-blocking processing is nonblocking read and write:

```c
status = exreceive(*buffer, length, &src, &type, &state);
status = exsend("buffer, length, &dest, &type, &state);
```
Cluster Environment In Parallel Computing

- Cluster of workstation or personal computers connected to high speed network.

- Cost Effective as compared to parallel computers.

- An alternative distributed memory parallel computer with special hardware.
Problems in Cluster Environment

- No major improvement in performance of software on cluster environment. Mainly due to lack of standard mechanism.
- Design and implement portable Distributed Shared Memory System.
- Shared Memory model makes programming easy.
- No hardware support to implement distributed shared memory.
- For remote memory access high performance is required.
- Easier to use standard language and standard message passing libraries.
Portable Distributed Memory System

- Functions that can handle request for memory access and transfer of data.

- Requires hardware and OS support.

- Performance improvement requires a mechanism similar to DMA to access remote memory.

- Address resolution for data located in remote processor memory.
Hurdles in Implementation

- No Direct Memory Access (DMA) to remote memory.

- Requires implementation of memory access method using software interrupt and user level event queue.

- Newest architecture have DMA like interface between network and memory.
Solution

- Data transfer is done using ports on distributed systems.
- Requires a mechanism for address resolution.
- Single Program Multiple Data (SPMD) programming style is used on cluster environment.
- Use the same binary file on all node processors. The memory layout of remote memory is known at compile time thus solving the problem of address resolution.
Portability

- One of the major concern in programming.

- For a program to be portable it is strongly preferred that program be written in standard programming language that is widely used and use of standard message passing function.

- PVM v/s MPI : powerful implementation of distributed Shared memory system.
Design and Implementation

Problem Definition

“Safe and reliable implementation of active messages and address resolution of remote data.”

Active Messages

• Messaging objects capable of processing on its own.
• Optimizes network communication by reducing latency and providing application direct user-level access to network hardware.
• Unnecessary acknowledgements are avoided between communication request and data send/receive action.
Active Messages on PVM & It’s Enhancement

- Use of PVM signal handling functions in implementing handler invocations of active messages.

- `pvm_sendsig` is send with signal number and data to be sent/received on UDP.

  - `Pvm_safesigsend(tid, signum, mid)`*

  - `Pvm_safesignsendrecv(tid, signum, mid, bufid)`*
Global Address Space

- Use of SPMD simplifies address resolution.
- Data in remote processor is placed at an identical address as that of the local processor.
- The loader and the OS are responsible for keeping the correspondence.
- Address of global memory at each processing node is separated into base address and offset.
- When a node receives a request, it returns data located at address which is base address plus the address offset of the request.
Memory Access Sequence

1. Issue a remote memory access request.
   \[ \text{Offset Address} = \text{address of local memory} - \text{base address of local memory} \]

2. Send the request to the task of the remote process using \textit{pvm\_safesignedsendrecv}

3. Receive the request and invoke the handler.

4. Calculate the address and access the data of address in signal handler.
   \[ \text{Address accessed} = \text{address offset in request} + \text{base address of local memory} \]

5. Return the value of data.
Sequence of Memory Access

1. offset

2. pvm_safesigsend + offset

3. handler invocation

4. access(bss' + offset)

5. pvm_recv

Processor

In Handler

Processor'

bss

task

bss'

task'
CONCLUSION

- Many parallel programming environments are available.
- Each of it has its own advantage and disadvantage.
- A best fit is the one based on the need of application.
- PVM is portable with support of software.
- Heterogeneous hardware can be exploited by parallel program using the portability of PVM.
QUESTIONS?
THANK YOU