Parallel Programming Environments

Presented By: Anand Saoji
Yogesh Patel
Outline

- Introduction
- How ?
- Parallel Architectures
- Parallel Programming Models
- Conclusion
- References
Introduction

- Recent advancements in scientific and commercial applications demand faster and efficient computers.
- Physical limitations and high costs make it difficult to increase processor speed beyond certain limits.
- 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.
How?

- Recent advances in high speed networks
- Rapid improvements in microprocessor design
- Availability of high performance software implementations
- Cost effective high performance parallel computing
Parallel Programming Architectures

- **Shared Memory**
  - Multiple processors can operate independently but share the same memory resources.

- **Distributed Memory**
  - Processors have their own local memory.

- **Hybrid**
  - Shared memory component can be graphics processing units (GPU).
  - Distributed memory component is the networking of multiple SMP/GPU machines.
Parallel Programming Architectures

- Shared Memory

```
   P_1  M_1  P_2  M_2  P_n  M_n

   ----------- Interconnection network  -----------

   Memory  Memory  Memory
```
Parallel Programming Architectures

- Shared Memory

Diagram:

- Processors (P₁, P₂, ..., Pₙ)
- Memories (M₁, M₂, ..., Mₙ)
- Interconnection network

Connections:

- P₁ ↔ M₁ ↔ P₂ ↔ M₂ ↔ Pₙ ↔ Mₙ
Parallel Programming Architectures

- Distributed Memory

Diagram showing a communication network with multiple processors (P1, P2, Pn) and memories connected through a communication network.
Parallel Programming Models

- Distributed Memory / Message passing
- Data Parallel
- Shared Memory
- Hybrid
Distributed Memory

- Also known as message passing model.
- Tasks use their own memory for computation.
- Multiple tasks can work on same physical machine.
- Exchange takes place by sending and receiving messages.
- Data transfers usually require cooperative operations.
- E.g. – A send requires a corresponding matching receive.
- Divided as - MPI and PVM
Message Passing Interface (MPI) is a standardized and portable message-passing system.

Designed by a group of researchers to be used on a wide variety of parallel computers.

It defines the syntax and semantics of a core of library routines.

Useful for programmers writing codes in C and Fortran 77.

Several free versions are available in public domain.

This has helped in development of a parallel software industry to a great extent.
History

- 1992 - MPI forum was formed.
- 1994 – MPI 1 was released.
- 1996 – MPI 2 was released.
- Today MPI is the de-facto standard for message passing.
- MPI implementations exists for all popular parallel computing platforms.
- MPI forum is drafting MPI -3 standard.
Advantages

- **Standardization** – It is the only library to be considered as a standard.
- **Portability** – Can run code on any platform supporting MPI.
- **Functionality** – Numerous routines are defined.
- **Availability** – Variety of implementations are available.
- **Performance** – Can exploit hardware features to optimize performance.
Available Environments

- FM-MPI – A version of MPICH built on fast messages.
- WMPI – A full MPI implementation for Microsoft WIN 32 platforms.
  Based on MPICH and a P4 API and P4 device standard.
- MPIPro – Based on Win-MPICH on Windows NT.
- PaTENT – High performance MPI on windows NT.
  Robust and high performance.
Parallel Virtual Machine (PVM) is a software tool for parallel networking of computers.

Designed to allow set of heterogeneous computers to be used as a single distributed parallel computer.

Large computational problems can be solved more cost effectively.

PVM enables users to exploit the existing computer hardware to solve large problems at minimal additional cost.

PVM consists of a run-time environment and library for message-passing, task and resource management, and fault notification.
History

- Developed by the University of Tennessee, Oak Ridge National Laboratory and Emory University.
- 1989 - The first version was written at ORNL.
- 1993 - Version 3
  Supports fault tolerance
  Better portability.
Many environments exist and some of them were modified and improved versions were produced for specific work.

- **CUMULVS** - (Collaborative User Migration, User Library for Visualization and Steering)
- **HP-PVM** - (High Performance Parallel Virtual Machine)
- **DAMPVM** - (Dynamic Allocation and Migration Parallel Virtual Machine)
- **WPVM** - (Windows Parallel Virtual Machine)
## Comparison

<table>
<thead>
<tr>
<th>MPI</th>
<th>PVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rich set of communication functions.</td>
<td>Heterogeneity</td>
</tr>
<tr>
<td>Higher communication performance.</td>
<td>Fault tolerance</td>
</tr>
<tr>
<td>Better portability</td>
<td>Almost supports exclusively interoperability</td>
</tr>
</tbody>
</table>
Parallel works focuses on performing operations on data set.

- Set of task work collectively on same data structures.
- Tasks perform same operation on their partition of work.

- Shared memory architectures - All tasks may have access to the data structure through global memory.
- Distributed memory architectures - The data structure is split up and resides as "chunks" in the local memory of each task.
Data Parallel

- Programming is usually accomplished by writing a program with data parallel constructs.
- Constructs can be calls to:
  - Data parallel subroutine library.
  - Compiler directives recognized by a data parallel compiler.
- Implemented in Fortran 77 / Fortran 90 or High performance Fortran (HPF).
According to shared memory model multiple programs can simultaneously access a common memory location by multiple programs with an intent to provide communication among them without creating redundant copies.

Shared memory is an efficient means of passing data between programs.
Models

- Shared Memory Model (without threads)
- Threads Model
- Directive based model
- other Shared Memory Models: CILK, TBB
In this memory model, read and write is done at same memory space.

Locks/ semaphores are used to control the read and write of shared memory space.

Advantage - No explicit communication required.

Disadvantage - becomes more difficult to understand and manage data locality.

It can be solved by keeping the data local to the processor that works on it. But data locality is tough to understand.
Implementations

- Native compilers or hardware translators turn program variables into memory addresses which can be accessed globally.
- On distributed shared memory machines, memory is physically distributed across a network of machines, but made global through specialized hardware and software such as SGI Origin.
Threading Model

- Threading model is a type of shared memory programming.
- A single process has multiple execution concurrent paths.
- These model is based on the thread library routines to parallelize the application.
- Condition variables and locks are used to establish communication and synchronization between threads.
Threading Model

### Advantages
- Most suitable for applications having/based on the multiplicity of data
- High flexibility for the programmer
- Widely used, support available, and related tools can be easily found.

### Disadvantages
- Difficult to synchronize, causing communication overheads.
- Precautions should be taken while using global data otherwise it may cause deadlock and false sharing.
- Since threading model stands at a low level of abstraction, it is not good for a programming model.
Directive Based Model

- Based on thread based model.
- It has a high level compiler.
- Takes care of parallelizing the application.
- Manages low level features such as -
  - Partitioning.
  - Work management.
  - Synchronization between threads.
  - Communication between threads.
Directive based model

Advantages
- Easy to write and maintain parallel applications.
- Implicitly Deals with low level abstraction.
- No need to worry about deadlocks and false sharing.

Disadvantages
- Less flexibility provided to programmer.
- Not widely used.
Based on threading model.

Requires thread library.

Developed as a standard, the IEEE POSIX 1003.1c standard (1995).

Supports only C language.

Present in all POSIX based system such as LINUX, QNX and many more.
OpenMP

- Stands for Open Message Passing.
- It is based on compiler directive model.
- Available in Fortran and C/C++ implementation. The OpenMP Fortran API was released October 28, 1997 and C/C++ API was released in late 1998.
- Easy and simple to use.
- Best suited for task based applications.
TBB

- TBB (Threading Building block)
- Developed by Intel.
- Supports task based parallelism.
- Replaces the threading libraries.
- Also hides the details about the threading mechanisms for performance and scalability.
- Works at high level abstraction.
- Supports ISO C++.
Development is done in terms of modules rather than threads.

Developer has to specify tasks instead of threads.

Compatible with other threading packages such as Pthread and OpenMP.

Targets performance and emphasizes scalable data parallel programming.

First to implement the concepts like recursive splitting algorithm and task stealing algorithm.

Task stealing algorithm is a popular and dominant model to achieve greater speed up for task based application.
Cilk++

- Founded by a student of MIT, later over taken by Intel.
- Based on task based parallel library.
- First to implement Hyper Object library.
- Hyper Object library is used to solve race conditions.
- Easy to code and comes with a inbuilt debugger.
- Race detector decreases the scope of errors during development phase and guarantees a race-free application.
 Benchmark

- It was found in the research of Srikar Chowdary Ravela[1], that for matrix multiplication TBB model is the best as it gives the best speedup and least development time.
- For Jacobi Interation, Pthread is considered as best because it requires global computation.
- Similarly for Laplace heat transfer, pthreads prove the best model.
Conclusion

- Many programming models are available.
- All of them have particular advantages and limitations.
- Depending on the problem specification a particular programming model can be considered as best.
- Also MPI has emerged as a de-facto standard and is very popular for distributed applications.
- Pthreads has been developed as a standard in POSIX systems.
- Pthreads is very popular for concurrent applications and is provided by almost all hardware vendors.
References


- “Introduction to parallel computing”
  https://computing.llnl.gov/tutorials/parallel_comp/#MemoryArch.

- Wikipedia, “Message passing Interface “
  http://en.wikipedia.org/wiki/Message_Passing_Interface”
Thank You