Introducing Parallel Processing into the Curriculum of the Hungarian Higher Educational Institutions of Technology Tempus SJEP No. 08333-94

Activity No 1: Parallel Programming Mobility Grant and Partner No. 5

Parallel Programming Course

The main subjects are:

- Introduction to parallel processing. Classification of models.
- Concurrent, cooperative processes: basic concepts, problems.
- Cooperation via shared memory.
- Cooperation via message passing.
- Object based and object oriented models.
- Data parallel models.
- **Language and system tools for parallel programming.**
- Problems of real-time and distributed systems.
- Execution models of declarative languages.
- Programming in parallel C.
- Mapping, debugging, performance analysis.

Overview of Parallel Programming

Why is parallel processing important ?

- To achieve higher performance ? Just for performance ?
  - parallelism is just a technological fix
  - not relevant to the applications engineer
  - hide it (like other details of hardware e.g. registers, cache,...)
- It's a common and natural mechanism for abstracting 'real-world' components
- It simplifies our engineering

Parallel processing myths

- harder than serial processing
- fragile (implicitly non-deterministic semantic)
- deadlock problems
- resource connection problems
- difficult to debug

Common experiences ... but not necessary ones !

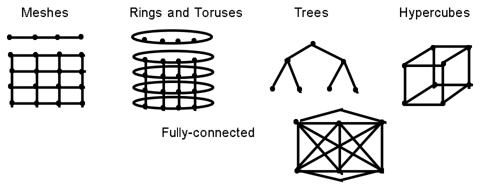
Architectures

Parallel architectures: Flynn's taxonomy

INSTRUCTIONS \ DATA	Single	Multiple
Single	Single Instruction Single Data **SISD** (serial machines)	Single Instruction Multiple Data **SIMD**
Multiple	Multiple Instruction Single Data **MISD**	Multiple Instruction Multiple Data **MIMD**

Characteristics of architectures

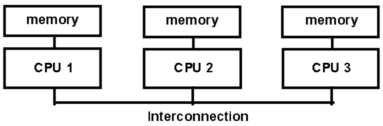
- Processor granularity
- Homogenous or heterogeneous
- Latency and bandwidth of interconnection
- Topology



Basic Concepts of Cooperation

- **Cooperation via shared memory**
- **Cooperation via virtual shared memory (paging)**
- **Cooperation via message passing**

Distributed Memory Concept



- Multiple processors work on a same problem
- Memory and address space are local to each processor
- Data is shared by passing messages
- Can be loosely or tightly coupled

  WAN → LAN → multiproc. computer

Why use Distributed Memory Computers ?

Benefits of parallel programming

- Decreased wallclock time

Benefits of distributed memory computers

- Scalable
- Cost effective
- Increased reliability through redundancy
- Can incorporate specialized processors

Disadvantages of distributed memory computers

- Communication is expensive
- Not all algorithms are appropriate
- Existing serial and shared memory applications may need to be restructured
- Effort is required to achieve real speed-up
- Programs can be difficult to debug

Why use Distributed Memory Computers ? (cont)

Suitable applications

- Applications distributed across networks
- Large grain size applications

```
high ratio of computation to communication
for each iteration
    for j data points
        do k calculations
    share bytes of data
        with y other processors
end iteration
```


Language and System Tools for Distributed Memory Computers

Several tools are available:

PVM, MPI, Express, Linda, ISIS, ParaScope, EUI, NX, P4, Para++, HeNCE, etc.

Common concepts are:

- Process-level parallelism
- Parallelism is provided through subroutine libraries for C, C++, FORTRAN, LISP, Ada, etc.

PVM

Parallel Virtual Machine (PVM)

- Basic model: message passing

  sending process packs data into message buffer,specifies a receiver, sends the message

  receiving process receives the message, unpacks the buffer

  The sender must to known whom to send the message
- Provides communication and synchronization primitives
- Uses daemons to communicate between different processors in a network
- Supports Heterogeneous run-time environment (uses XDR)
- Public domain
- Tools: XPVM, HENCE

Express

Express

- Basic model: message passing

  The low-level message passing is similar to the PVM paradigm

  The high-level message passing:

  Sequential program is distributed across N nodes

  The host node coordinates the work across the other nodes
- Express also provides higher level routines to simplify coordination of processes, CUBIX I/O, data decomposition, averaging, etc.
- Plotix library provides parallel graphics system

Linda

Linda

- Basic model: logical shared memory
  shared variable = tuple
  logical shared memory = tuple space
  operations:

  out → adds tuple to tuple space
  read → reads a tuple
  in → reads a tuple and deletes it from tuple space

  any process can access any tuple

  i.e the "sender" does not have to identify a "receiver"

  although each tuple is physically located on specific processor, the programmer does not need to know where it is stored

ISIS, ParaScope

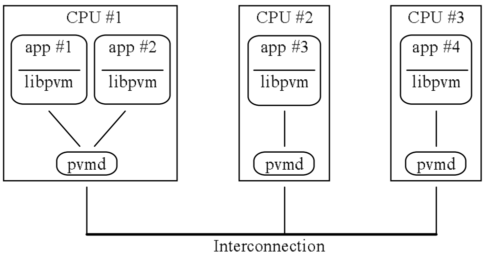
ISIS

- Basic model: virtual synchrony
  event driven process group
  replicated data
  uses RPC calls

ParaScope

- NOT an API
  Editor for generating FORTRAN D code

Basic Concepts of PVM



app # → Part of the application

libpvm → Language if.

pvmd → PVM daemon

Parallelizing for PVM

Strategies for parallelization:

"Embarrassing" parallelism

- Running several instances of same task with different parameters
- Suitable only if serial execution time of each instance is acceptable

Parallelize on a loop

- The computation is independent of the previous iteration

Parcel out tasks (master / slave)

- Supervisor task runs on one node
- Useful when work of main thread is much less than work of independent tasks
- Easy to run on arbitrary number of processors, if there are many independent tasks

Parallelizing for PVM (cont)

Strategies for parallelization (cont);

"Rolling " execution

- Each node hands off data to next at end of its serial section
- Useful when serial portion of code is not trivial but is still significantly sorter than parallel part
- All nodes run identical code - good load balance
- Especially suited to ring topology

"Regional"parallelism

- Data dependences are localized in portions of arrays
- Useful when serial time is greater than parallel time
- Requires internode communication
- Most difficult

Debugging and Tracing Tools for PVM

Several tools are available:

PraGraph, Xab, abmon, topg

Common concepts are:

- Intercepts the PVM calls, and generates trace records.
- Replays the trace records.

ParaGraph

- Graphical display package for visualizing the behavior and performance of parallel programs on message passing multiprocessor architectures.
- Uses PICL (Portable Instrumented Communications Library) traces as input.

Debugging and Tracing Tools for PVM (cont)

Xab

- Set of tools for analyzing and debugging PVM programs
- Major components:

  Xab libraries, abmon, xab, topg

- Provides visual and textual feedback about program execution bay using tracing information
- Not really a "debugger"

HeNCE (Heterogeneous Network Computing Environment)

Main characteristics:

- Graphical parallel programming environment
- Interface for creating, compiling, and debugging parallel programs
- The programmer does not need to use any parallel programming primitives
- Relies on the PVM system