2.2 Using Workstation Clusters

SOFTWARE TOOLS

- Several workstation packages for workstation cluster parallel programming:
  - PVM (Parallel Virtual Machine) - Oak Ridge National Lab
    - Homogeneous & heterogeneous workstations
    - C and FORTRAN support
  - MPI (Message Passing Interface) - MPI Forum
    - Provides a message-passing standard
    - Several public-domain implementations: [http://www.osc.edu/mpi/](http://www.osc.edu/mpi/)
  - LAM (Ohio Supercomputing Center)
  - MPICH (Argonne National Lab and Mississippi State University)
  - CHIMP (Edinburgh Parallel Computing Center)
  - UNIFY (Mississippi State University)
  - Vendor-Specific proprietary message-passing packages:
    - MPL (IBM) for SP-2, etc.

Programmer is responsible for:

- Decomposing the problem into separate (C or FORTRAN) programs.
- Compiling each program for the specific workstation type.
- Homogeneous or heterogeneous workstations
- Common file system support or not
- Defining the set of computers used on a problem
  - Use a `hostfile` containing the names of computers
  - Start one machine and add others from the PVM console

What if (# programs) > (# processes) & why do we prefer this?

Message routing in PVM:

- Routing is done by using daemon processes installed by PVM on the computers that form the virtual machine.
- Each PVM daemon keeps sufficient information that helps to select the routing path.

2.3 Evaluating Parallel Programs

**PARALLEL EXECUTION TIME**

For a parallel algorithm: \( t_P = t_{comp} + t_{comm} \)

Calculation of \( t_{comp} \):

- Computation steps of the most complex process.
- Assumption: all processors are the same and operating at the same speed.
- What if not? Load balancing required.

Calculation of \( t_{comm} \):

- Depends upon size of message, underlying interconnect, and transfer mode.
- In a cluster, communication time depends upon many factors, including network structure and network contention.

**Message Latency** is the time to send a message with no data (including pack & unpack times.)
We have ignored the following factors:
• contention of the communication network
• not having source and destination directly linked

And assumed that:
• the overhead incurred by including information other
  than data in the packet is a constant and can be part
  of \( t_{\text{startup}} \).

**Important Note on Interpretation of Equations**

We will make many simplifying assumptions in subsequent chapters:
• \( t_{\text{comp}} \) & \( t_{\text{comm}} \) is measured in units of an arithmetic operation (system dependent)
• system is homogeneous (identical processors with equal speed)
• all arithmetic operations require the same time
• we count the number of computation steps
  – on one processor when all processes perform the same operation
  – in longest process in other situations.
• \( t_{\text{communication}} \) for sending an integer or a real takes the same time.

Sending \( q \) messages of \( n \) data items takes

\[
\text{time} = q (t_{\text{startup}} + n t_{\text{data}})
\]

\( t_{\text{p}} = t_{\text{comp}} + t_{\text{comm}} \)

**Example**

• Suppose that a computer can operate at 200 MFLOPs (200 million floating-point operations per second) and the startup time is 1 \( \mu \)s.

• Then the computer could execute 200 f-p operations in the time taken in the message startup.

**Latency Hiding**

The deleterious effect on the execution time as shown in previous example is known as the Achilles' heel of message-passing computers.

\( t_{\text{startup}} \) is one/two orders of magnitude greater than \( t_{\text{comp}} \).

will dominate the communication time in many cases, unless \( n \) is quite large.

**Time Complexity of a Parallel Algorithm**

If we use time complexity analysis, which hides lower terms, \( t_{\text{communication}} \) will have a time complexity of \( O(n) \).

Complexity of \( t_{\text{p}} \) will be the sum of the computation and communication.

**Example Problem:**

Suppose that we were to add \( n \) numbers on two computers, where each computer adds \( n/2 \) numbers together.

• The numbers are initially held by the first computer.
• The second computer submits its result to the first computer for adding the two partial sums together.
**Computation/Communication Ratio**

Communication is very costly.

If both $t_{comp}$ and $t_{comm}$ has the same complexity $\Rightarrow$ ?

Ideally $t_{comp} \gg > t_{comm}$ and $n$ will improve performance.

**Example:** N-body problem

$t_{comm}$ is $O(N)$ and $t_{comp}$ is $O(N^2)$

we can find an $N$ where $t_{comm}$ will dominate the $t_{comp}$.

---

**Cost-Optimal Algorithms**

The cost to solve a problem is proportional to the execution time on a single processor system (using the fastest known sequential algorithm.)

$$Cost = t_p x n = k x t_s$$

where $k$ is a constant.

A parallel algorithm is cost-optimal if

$(\text{Parallel time complexity}) \times (\# \text{ processors}) = \text{sequential time complexity}$

**Example:**

Suppose the best known sequential algorithm for a problem has time complexity of $O(n \log n)$. Are the following cost optimal?

- A parallel algorithm for the same problem that uses $n$ processes and has a time complexity of $O(\log n)$.
- A parallel algorithm that uses $n^2$ processors and has time complexity of $O(1)$.

---

**Comments on Asymptotic Analysis**

Time complexity is

- often used for:
  - sequential program analysis
  - theoretical analysis of parallel programs
- much less useful for evaluating the potential performance of parallel programs:
  - Big-Oh and other complexity notations use asymptotic methods (the variable under consideration to tend to infinity) however
    - often the # processors are constrained and
    - data sizes are finite and manageable.
  - Analysis often ignores lower terms that could be important (e.g., $t_{comm}$ dominates overall communication time when $n$ is large)
  - Analysis also ignores other factors that appear in real computers, such as communication contention.