Parallel Algorithms for Data Mining

Peter Christen
Computer Sciences Laboratory, RSISE
Australian National University

peter.christen@anu.edu.au
http://csl.anu.edu.au/~peter

17 May 2000

Funded by grants from the Swiss National Science Foundation and Novartis Stiftung

Why Parallel Computing?

• Large data sets and long processing times
  (e.g. simulations in physics and chemistry, weather forecast, etc.)

• Limitations of sequential computers
  Processor speed, I/O- and memory bandwidth

• Many applications and algorithms contain parallelism
  (e.g. pipelining, domain decomposition)

• Data Mining: Data sets from Giga-Bytes to Peta-Bytes, several scans over data set needed, complex algorithms
Parallel Architectures

Parallel Architectures: Two Examples

<table>
<thead>
<tr>
<th>CSIRO Capricorn</th>
<th>ANU DCS Bunyip</th>
</tr>
</thead>
<tbody>
<tr>
<td>– Sun Enterprise 4500 Server</td>
<td>– Beowulf Linux Cluster</td>
</tr>
<tr>
<td>– Shared memory (SMP)</td>
<td>– Distributed memory</td>
</tr>
<tr>
<td>– 12 UltraSPARC 400 MHz RISC (8 Mega-Bytes cache each)</td>
<td>– 96 Dual Pentium III 550 MHz</td>
</tr>
<tr>
<td>– 6,912 Mega-Bytes main memory</td>
<td>– 36,864 Mega-Bytes main memory</td>
</tr>
<tr>
<td>– 250 Giga-Bytes disk storage array (RAID)</td>
<td>(384 Mega-Bytes per node)</td>
</tr>
<tr>
<td></td>
<td>- 1,305.6 Giga-Bytes disk</td>
</tr>
<tr>
<td></td>
<td>(13.6 Giga-Bytes per node)</td>
</tr>
<tr>
<td></td>
<td>– 100 Mega-Bit Ethernet network</td>
</tr>
</tbody>
</table>

According to the last www.top500.org (Nov'99) the ANU Beowulf is Australia's fastest supercomputer.
Parallel Architectures: ANU Beowulf Topology

Parallel Architectures: ANU Beowulf Implementation
Different Kinds of Parallelism

- **Functional Parallelism:** Each processor runs a sub-job, the result is passed to the next processor (pipeline principle).

- **Data Parallelism:** All processors do the same job on different parts of the data (domain decomposition).

- **Master-Worker:** Master process distributes tasks to worker processes which return result back. Good if workers can operate independently.

- **Single-Program Multiple-Data (SPMD):** The same program runs on all processors, but on different sub-sets of the data.

Parallel Programming

- **Message Passing:** Calls to communication routines, e.g. 
  \( \text{SEND(data, P2)} \) or \( \text{BROADCAST(vector, P0)} \) 
  Mainly on distributed memory architectures → PVM, MPI

- **Threads:** Program parts that can run independently. 
  Mainly on shared memory architectures → OpenMP, Pthreads

- **Parallel Compilers:** Extensions of languages with parallel statements, e.g. 
  \( \text{DISTRIBUTE A (BLOCK) ONTO P} \) 
  \( \text{DO IN PARALLEL ...} \) 
  → High Performance Fortran (HPF)
Parallel Performance

- **Speedup**: Sequential time divided by parallel time: \( Sp(p) = T_1/T_p \)
  Desired: \( Sp(p) = p \) (hard to achieve).

- **Efficiency**: Speedup divided by the number of processes: \( Ef(p) = \frac{Sp(p)}{p} \leq 1 \)
  Sometimes: Super-linear speedup \( Sp(p) > p \rightarrow Ef(p) > 1 \)
  (Cache, memory and I/O effects, etc.)

- **Scalability**: Efficiency often drops as the number of processes is increased.
  Scalability gives a measure how much the data size has to be increased to get
  the same efficiency on more processes.

Parallel Obstacles

- **Amdahl's Law**: Most algorithms and programs contain sequential parts,
  which limit maximal speedup and inhibit scalability, e.g. 10\% sequential code
  → Maximal speedup 10!

- Balancing the load (distributing work onto processors) can be hard to achieve.

- Data distribution can become a bottleneck (e.g. if all processors are connected
  to only one I/O system).

- Parallel programs often have to be adapted to a given architecture to get
  maximum performance.
Data Mining Cycle

Example: Assembly

- Each data record adds some values into a matrix.
- The whole data set has to be read only once.
- The size of the matrix is independent of the number of data records.
- Parallelism is easy to achieve: Each processor only reads a part of the data set and adds into a local matrix.
- Reading and assembling is done in blocks of a given size.
## Parallel Assembly - Two Implementations

<table>
<thead>
<tr>
<th>Master-Worker</th>
<th>SPMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master process controls assembly</td>
<td>All processes compute distribution</td>
</tr>
<tr>
<td>Master sends messages to workers with start position in file and number of records to assemble</td>
<td>Each process reads and assembles (\frac{n}{p}) data records</td>
</tr>
<tr>
<td>After assembling a block, worker sends \textit{ready} message back to master and gets next task</td>
<td>No communication needed during assembly</td>
</tr>
</tbody>
</table>

After the assembly is finished, the local matrices are collected and summed on the host-processor (Reduce operation).

---

### Assembly on Sun Enterprise

<table>
<thead>
<tr>
<th>Block length = 1000</th>
<th>Block length = 100,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of processes</td>
<td>Number of processes</td>
</tr>
<tr>
<td>Time in seconds</td>
<td>Time in seconds</td>
</tr>
</tbody>
</table>

![Graph 1](#)  
![Graph 2](#)
Assembly on Beowulf

Assembly on Beowulf: Speedup and Efficiency

Paralleln Algorithms for Data Mining

14
17 May 2000

Peter Christen
Outlook

- Parallel computing can help to solve bigger and more complex problems.
- It can speed up existing applications.
- Not all applications parallelise well or yield in good speedup and scalability.
- Good parallel programs should be scalable both in data size (number of records) and number of processors.
- Parallel programming is still complicated and cumbersome (run-time effects).