Module 1a
Introduction to Parallel Data Mining

VPAC Education Grant Round 2
EDPNMO006/2001
David Taniar and Kate Smith
Monash University
Parallel Data Mining

- Data mining - process of discovering interesting knowledge, such as patterns, associations, changes, anomalies and significant structures, from large amount of data stored in databases, data warehouses, or other information repositories.

- Data mining or knowledge discovery process steps:
  - data cleaning
  - data integration
  - data transformation
  - data mining
  - pattern evaluation
  - knowledge presentation

• data cleaning - handles noisy, erroneous, missing, or irrelevant data.
• data integration - where multiple, heterogeneous data sources may be integrated into one.
• data selection - where data relevant to the analysis task are retrieved from the database.
• data transformation - where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.
• data mining - process where intelligent methods are applied in order to extract data patterns.
• pattern evaluation - identify the truly interesting patterns representing knowledge based on some interestingness measures.
• knowledge presentation - visualization and knowledge representation techniques are used to present the mined knowledge to user.
Data mining tasks can be classified into two categories, that is descriptive data mining and predictive data mining. Descriptive data mining describes the data set in a concise and summary manner and presents interesting general properties of the data; whereas in predictive data mining, one or a set of models is constructed and performs inference on the available set of data, and attempts to predict the behavior of new data sets.

- **Class description** - provides a concise and summarization of a collection of data and distinguishes it from others. The summarization of a collection of data is called class characterization; whereas the comparison between two or more collections of data is called class comparison or discrimination.

- **Association** - discovery of association relationships or correlation among a set of items. Often describes as a set of rules called association rules. Association analysis is widely used in transaction data analysis for directed marketing, catalog design etc.

- **Classification** - analyzes a set of training data and constructs a model for each class based on the features in the data. A decision tree or a set of classification rule is generated by such a classification process, which can be used for better understanding of each class in the database and for classification of new data.

- **Prediction** - predicts the possible values of some missing data or the value distribution of certain attributes in a set of objects. It involves the finding of the set of attributes relevant to the attribute of interest and predicting the value distribution based on the set of data similar to the selected object(s).
• **Clustering** - identify clusters embedded in the data, where a cluster is a collection of data objects that are similar to one another. Similarity can be expressed by distance functions, specified by users or experts. The main objective of clustering algorithm is to ensure that the inter-cluster similarity is low and the intra-cluster similarity is high.

• **Time series analysis** - analyze large set of time series data to find certain regularities and interesting characteristics, including searching for similar sequences or subsequences, and mining sequential patterns, periodictics, trends and deviation. For example to predict the trend of the market stock values.
Motivation & Objectives

Parallel data mining

Motivations
- high dimensionality of data (wide) and large volumes of data (deep)
- more complex data - images, geographical data, scientific data
- availability of multiple processors or parallel computers

Objectives
- reduces data mining time (speed up and scale up)
- simplifying the tasks
- sometimes better solutions
Parallelism (1)

- Two main types
  - Data parallelism
  - Task/Control parallelism

- Data parallelism - data parallelism refers to the execution of the same or rather similar operation or instruction on multiple large data subsets at the same time. All processors execute the same program, but the program can use the control structure if-then-else to determine which instruction to execute in which processor. So, that are some part of the same program that is not executed by program in other processors.

- In data parallelism it is necessary to partition the data into subsets (subset 1.1, subset 1.2 etc). The speedup is achieved by reducing the amount of data that needs to be handled by each processor.
• In control parallelism, each processor execute different instruction set. The programs are coordinate together to achieve the final same objective. The idea of control parallelism is to reduce a complex task to smaller subtasks. This paradigm did not necessary implemented in multi-processor architecture.

• The data set operated each program does not necessary the same.
## Parallelism (3)

<table>
<thead>
<tr>
<th>Data parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td>• easier implementations</td>
</tr>
<tr>
<td>• less architecture dependent</td>
</tr>
<tr>
<td>• more scalable or productive</td>
</tr>
<tr>
<td><strong>Disadvantages</strong></td>
</tr>
<tr>
<td>• load balancing from data skew</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td>• reduce the complexity of serial algorithms</td>
</tr>
</tbody>
</table>

### Data parallelism vs control parallelism

- the parallelism version of algorithm is very similar to that of the serial version. So, the serial version is easier to port it to the parallel version. The development time can be saved without total new implementations. Whereas in the control parallelism, a totally new program needs to be developed to accommodate the parallelism.

- there is not needs to change the control flow of the serial algorithm in order to execute in the parallel architecture. The control flow of parallel algorithm still the same as the serial version, just the handling of data change. But it does not totally independent from the machine architecture. Some instruction may be specific to a machine architecture. On the other hand, a new program needs to be tailor to the machine architecture in the deploying machine in control parallelism. This is because it make use of the architecture specific design to achieve parallelism.

- data parallelism has better scalability for large databases than control parallelism. We can add more processor when the data are increase. This will result in more processor operate of smaller data subset. On the other hand, in control parallelism the sub-tasks which handle the data will have more data as the data volume increase. It is not possible to increase the number of subtask since it is fixed.

- since each processor in the data parallelism implementations execute the same set of instruction, we can say they are redundancy and the complexity of the program is similar to the serial algorithm. Unlike data parallelism, the control parallelism
### Parallelism (4)

<table>
<thead>
<tr>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• machine dependent</td>
</tr>
<tr>
<td>• specialized implementations</td>
</tr>
</tbody>
</table>

try to reduce the complexity of the task in serial algorithm by dividing it into subtask. Although this may result in high coordination cost.

- Load balancing or data skew problem is difficult to handle in data parallelism concept. It is not sure which type of partitioning is more suitable to one data mining algorithm. Conventional method is to divide equally the data without looking into the data properties itself. In some data mining algorithm, improper data partitioning will result in one processor being idle due to the irrelevant of the data at that partition.
## Examples of parallel algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Control Parallelism</th>
<th>Data Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>Each branch of the tree is formed into a task</td>
<td>Training set is partitioned across the processors. Processors evaluate a node of tree in parallel.</td>
</tr>
<tr>
<td>Neural Network</td>
<td>Network divided into subnets. The sub-nets are allocated across the processors.</td>
<td>Network is duplicated onto each processor. Training set is split across the processors.</td>
</tr>
<tr>
<td>Association rules</td>
<td>Itemset is divided across the processors with selective copying such that each processor can independently count and choose candidate.</td>
<td>Data is distributed across the processors and each node counts and chooses candidates synchronously.</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td></td>
<td>The initial population is partitioned into sub-populations and allocated to the processors following the island model.</td>
</tr>
</tbody>
</table>
Parallel Architectures

- Consist of three building blocks
  - processors (PE),
  - memory modules, and
  - interconnection network - connects the processors to each other and sometimes to memory modules as well.
- Parallel architecture difference in the way they arrange those three elements.
In SIMD, a single processor executes a single instruction stream, but broadcasts each instruction to be executed to a number of data processors (ALU). These data processors interpret the instruction’s addresses either as local addresses in their own local memories or as global addresses.

At given instant of time, a given PE is either “active” and doing exactly the same thing as all the other active processes, or it is idle.
• In MIMD processors is a full-fledged CPU with both a control unit and an ALU. Thus each PE is capable of executing its own program at its own pace.
• The executions of MIMD are asynchronous.
• There are two distinction of MIMD, that is the distributed memory or shared memory.
Parallel Architectures (1)

- MIMD-Shared Memory Processors

In shared memory MIMD the main memory is shared by several PE’s connected via bus line/network.

Some of the problems of this architecture are

- Cache coherence - variables in the cache are shared by processes, thus it is hard to determine whether the values did modified by other processor or not.
- Mainlining memory consistency - the programmer-visible ordering effects of memory references both within a processor element and between different processor elements.
- difficult to programs
- excessive memory management required for concurrent access
• Each PE has its own memory space. The PE elements are interconnected via communication network.
• The main problem with this architecture is routing and communication overhead.
• If there are more than one route to communicate with other PE’s, how to determine which shortest route to take.
• There are also problems if intermediate nodes are involved between two communication processors.
Parallel Architectures (3)

- Clusters of shared memory

SMP  SMP  ...  SMP

communications network
Problems of Parallel Architectures (1)

- Start-up cost-start up cost associated with initiating a single process
- Interference-parallel processors often shared some resources (disk, memory etc), a slowdown may result from the interference of each new process as it competes with existing processes for commonly held resources.
- Data skew - unbalance workload is divided among processors
- Communications - often process may have to communicate with other processes. In a synchronized environment, one process may wait for another process before can proceed execution.
Problems of Parallel Architectures (2)

- Consolidation - in certain phase of the parallel algorithms, the results from difference processors needs to be gather together into a single processor. This result in under utilization of other processing power and will slow down the speed up.
Distributed data mining

- **Motivations**
  - Data are physically separated
  - Heterogeneous sources of data
  - Different representations or format
  - Decomposition
End of Module 1a
Introduction