Lecture 9

Tasks and Load Balancing

Institute of Computer Science & Information Technology,
Faculty of Management & Computer Sciences,
The University of Agriculture, Peshawar, Pakistan.

Characteristics of Tasks

- We have discussed various parallelization strategies, which allow us to identify the concurrency and decompose it into tasks that can be executed in parallel.

- The next step in the process of designing a parallel algorithm is to take these tasks and assign(i.e., map) them onto the available processes.

- While formulating a mapping scheme to construct a good parallel algorithm, we often need some **characteristics of tasks** and their **interactions**.

Characteristics of Tasks(1)

1. Task Generation
2. Task Sizes
3. Knowledge of Task Sizes
4. Size of Data Associated with Tasks

Task Generation

- The tasks that constitute a parallel algorithm may be generated either statically or dynamically.
- Static task generation refers to the scenario where all the tasks are known before the algorithm starts execution.
- Data decomposition usually leads to static task generation.
- For example: matrix multiplication

Task Generation (1)

- Certain decompositions lead to a dynamic task generations during the execution of the algorithm.
- In such decompositions, the actual tasks are not explicitly available a priori although the high level rules or guidelines governing task generation are known as a part of the algorithm.
- Recursive decomposition can lead to dynamic task generation.
- For example, recursive decomposition in quicksort.

Task Sizes

- The size of a task is the relative amount of time required to complete it.
- The complexity of mapping schemes often depends on whether or not the task are uniform; i.e., whether or not they require roughly the same amount of time.
- If the amount of time required by the tasks varies significantly, then they are said to be non-uniform.

- For example, the tasks in the decompositions for matrix multiplication considered uniform.

- On the other hand, the tasks in quicksort are non-uniform.

Knowledge of the Task sizes

- This characteristic that influences the choice of mapping scheme is knowledge of the task size.
- If the size of all the tasks is known, then this information can often be used in mapping of tasks to processes.
- In the various decompositions, the computation time for each task is known before the parallel program starts or the opposite case is also possible.

Size of Data Associated with Tasks

- It is an important consideration for mapping as the data associated with a task must be available to the process performing that task, and the size and the location of these data may determine the process that can perform the task without incurring excessive data-movement overheads.

Characteristics of Inter-Task Interactions

- In any nontrivial parallel algorithm, tasks need to interact with each other to share data, work or synchronization information.

- Different parallel algorithms require different types of interaction among concurrent task.

- The inter-task interactions can be categorized as below:

1. Static versus Dynamic
2. Regular versus Irregular
3. Read-only versus Read-Write
4. One-way versus Two way

Static versus Dynamic

- One way of classifying the type of interactions that take place among concurrent tasks is to consider whether or not these interactions have a static or dynamic patterns.

- An interaction patter is static if for each task, the interactions happens at predetermined times, and set of tasks to interact at these times is known prior to the execution of the algorithm.

- An interaction patter is dynamic if the timing of interactions or the set of tasks to interact with cannot be determined prior to the execution of the algorithm.

Regular vs. Irregular

- On the other hand, an interaction pattern is called irregular if no such regular pattern exists.

- Irregular and dynamic communications are harder to handle particularly in the message-passing programming paradigm.

Read only vs. Read Write

- Data sharing interactions can be categorized as either read only or read write interactions.

- As the name suggests, in read only interactions, task require only a read access to the data shared among many concurrent tasks.

One way vs. Two way

- In some interactions, the data or work needed by a task or a subset of tasks is explicitly supplied by another task.

- Such interactions are called two-way interactions and usually involve predefined producer and consumer tasks.

One way vs. Two way (1)

- In other interactions, only one of a pair of communicating tasks initiates the interaction and completes it without interrupting the other one.
- Such an interaction is called a one-way interaction.
- All read-only interactions can be formulated as one-way interactions.
- Read-write interactions can be either one-way or two way.

Load Balancing

- Once a computation has been decomposed into tasks, these tasks are mapped onto processes with the objective that all tasks complete in the shortest amount of elapsed time.

- In order to achieve a small execution time, the overheads of executing the tasks in parallel must be minimized.

Load Balancing(1)

- For a given decomposition, there are two key sources of overhead:

1. The time spent in inter-process interaction is one source of overhead.
2. Another important source of overhead is the time that some processes may spend being idle.

- Some processes can be idle even before the overall computation is finished for a variety of reasons.
- Uneven load distribution may cause some processes for finish earlier than others.

Load Balancing (2)

- Therefore, a good mapping of tasks onto processes must strive to achieve the twin objectives of

1. Reducing the amount of time processes spend in interacting with each other.
2. Reducing the total amount of time some processes are idle while the others are engaged in performing some task.

Load Balancing (3)

- These two objectives often conflict with each other.
- For example, the objective of minimizing the interactions can be easily achieved by assigning sets of tasks that need to interact with each other onto the same process.
- As a result, the processes with a lighter load will be idle when those with a heavier load are trying to finish their tasks.
- Similarly, to balance the load among processes, it may be necessary to assign tasks that interact heavily to different processes.
- Due to the conflicts between these objectives, finding a good mapping is a nontrivial problem.

Load Balancing (4)

- Mapping techniques used in parallel algorithms can be broadly classified into two categories: static and dynamic.

- The parallel programming paradigm and the characteristics of tasks and the interactions among them determine whether a static or a dynamic mapping is more suitable.

Static Mapping

- Static mapping techniques distribute the tasks among processes prior to the execution of the algorithm.

- For statically generated tasks, either static or dynamic mapping can be used.

- The choice of a good mapping in this case depends on several factors:

1. The knowledge of task sizes
2. The size of data associated with tasks
3. The characteristics of inter-task interactions
4. The parallel programming paradigm

- Algorithm that make use of static mapping are in general easier to design and program.

Dynamic Mapping

- Dynamic mapping techniques distribute the work among processes during the execution of the algorithm.

- Some simple guidelines can assist you to choose best mapping:

1. If tasks are generated dynamically, then they must be mapped dynamically too.
2. If task sizes are unknown, then a static mapping can potentially lead to serious load-imbalances and dynamic mapping are usually more effective.
3. If the amount of data associated with tasks is large relative to the computation, then a dynamic mapping may entail moving this data among processes.

Dynamic Mapping (I)

- The cost of this data movement may outweigh some other advantages of dynamic mapping and may render a static mapping more suitable.
- However, in a shared-address paradigm, dynamic mapping may work well even with large data associated with tasks if the interactions is read-only.
- The reader should be aware that the shared-address-space programming paradigm does not automatically provide immunity against data-movement cost.

- Now based on the static and dynamic mapping, two broader categories of mapping constituted:

1. Scheme of Static Mapping
2. Scheme of Dynamic Mapping.