Parallelization Strategies & Granularity

Lecture-13-14

Parallel & Distributed Computing

Course Outlines

Course Name: Parallel and Distributed Computing
Credit Hours: 3(3-0)
Prerequisites: Data Communications and Computer Networks

Course Outlines:

Why use parallel and distributed systems? Why not use them? Speedup and Amdahl's Law, Hardware architectures: multiprocessors (shared memory), networks of workstations (distributed memory), clusters (latest variation). Software architectures: threads and shared memory, processes and message passing, distributed shared memory (DSM), distributed shared data (DSD). Possible research and project topics, Parallel Algorithms, Concurrency and synchronization, Data and work partitioning, Common parallelization strategies, Granularity, Load balancing, Examples: parallel search, parallel sorting, etc. Shared-Memory Programming: Threads, Pthreads, Locks and semaphores, Distributed-Memory Programming: Message Passing, MPI, PVM. Other Parallel Programming Systems, Distributed shared memory, Aurora: Scoped behaviour and abstract data types, Enterprise: Process templates. Research Topics.

Parallelization Strategies: challenge

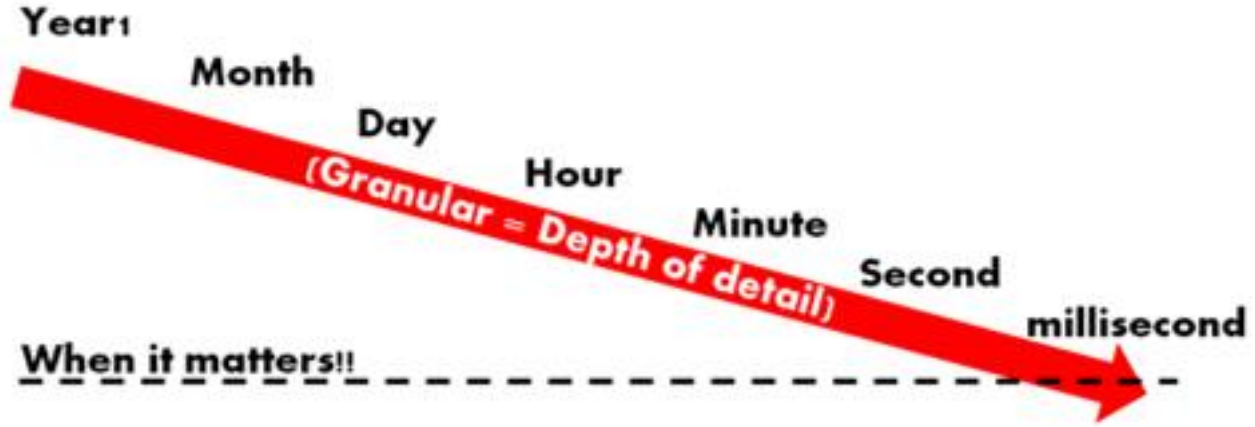
- Selecting the best parallelization strategy: using proper parallelization strategy is ***crucial*** in order to achieve the ***optimal performance***, such that the calculation takes the ***shortest possible time***.
- It is very difficult to give a ***universal method*** for finding the ***optimal strategy***, since it depends not only on the ***platform***, but also on the ***application*** itself.

Parallelization Strategies

- Parallelization Strategies define ***how to implement parallelization opportunities***.
- How much levels?
- How much granular?

Parallelization Strategies cont… Granularity

- Parallelization granularity refers to the ***size of parallel entities*** at which we can divide an application.
- Granularity is concerned with depth or level of details.



Parallelization Strategies cont… Granularity

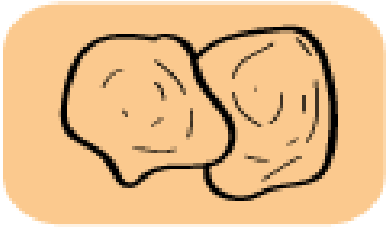
- For both levels we may define mainly two granularities:
- ***Fine-grain***
- **Coarse-grain**

Parallelization Strategies cont…
Granularity

Coarse-grain

Less granular

Understand the idea/concept

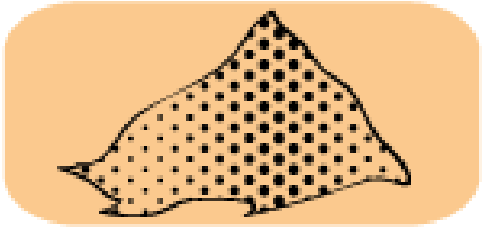


Able to describe the idea/concept with some detail

Able to describe and explain the idea/concept with some detail

Fine-grain

More granular



Able to describe and explain the idea/concept with full detail

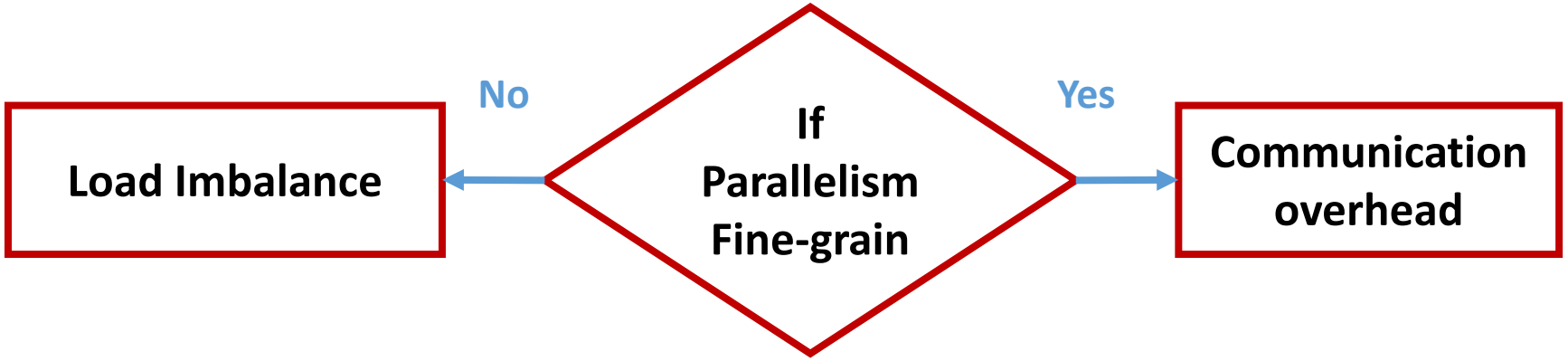
Parallelization Strategies cont…
Coarse vs. Fine Grain Parallelism

- ***Fine-grained parallelism*** means ***individual tasks*** are ***relatively small*** in terms of **code size** and **execution time**.
- The **data** is ***transferred among processors*** **frequently** in amounts of one or a few memory words.
- ***Coarse-grained*** is the opposite: **data** is ***communicated infrequently***, after ***larger amounts*** of **computation**.

Parallelization Strategies cont…
Coarse vs. Fine Grain Parallelism

- The ***finer the granularity***, the ***greater the potential for parallelism*** and hence **speed-up**, but the ***greater the overheads*** of ***synchronization*** and ***communication***.
- On the other side, if the ***granularity is too coarse***, the performance can suffer from ***load imbalance***.
- In order to attain the best **parallel performance**, the best **balance** between ***load*** and ***communication overhead*** needs to be found.

Parallelization Strategies cont…
Coarse vs. Fine Grain Parallelism



Parallelization Strategies cont…
Coarse vs. Fine Grain Parallelism

- In order to attain the best **parallel performance**, the best **balance** between ***load*** and ***communication overhead*** needs to be found.

Parallelization Strategies cont… Granularity

1. Task-level parallelization

2. Data-level parallelization

- For both levels we may define mainly two granularities:

    1. Fine-grain

    2. Coarse-grain.

Parallelization Strategies cont…
Task-level Granularity

- Task-level granularity is directly related to the ***program decomposition*** into **independent tasks**.

- It has two types:

1. **Fine-grain tasking**

2. **Coarse-grain tasking**

Parallelization Strategies cont…
Task-level Granularity (Fine-grain tasking)

- **Fine-grain tasking** consists of dividing the program in fundamental separate tasks and each single task is parallelized apart (separately).
- This process is called **fission**.

Parallelization Strategies cont…
Task-level Granularity (Fine-grain tasking)

- The benefit of the fine grain parallelization strategy is the ***high reusability*** since each task may be ***found in more than one algorithm*** which is the case of most **image processing algorithms**.
- This means that a ***parallel version*** of this operation can be ***reused more than once*** in different applications ***without any modification*** to ensure high ***portability*** among different applications.

Parallelization Strategies cont…
Task-level Granularity (Fine-grain tasking)

- However, this strategy may suffer from:

1. The overhead introduced by successive threads launches

2. Poor temporal data locality

Parallelization Strategies cont…
Task-level Granularity (Coarse-grain tasking)

- **Coarse-grain tasking** consists in packing a sequence of tasks into a ***macro task***.
- This process is called **fusion**.
- Each macro task is ***assigned to a single thread*** which processes a part of data array.

Parallelization Strategies cont…
Task-level Granularity (Coarse-grain tasking)

- The implementation of this parallelization strategy needs ***additional programming effort*** to manage dependencies between neighbors data in order to minimize the ***synchronization barriers*** and ***data communication***.
- When the implementation of the coarse-grain strategy is optimized, runtime performances may be improved by increasing ***temporal data locality*** and by ***avoiding overhead*** caused by threads launches and data transfers.

Parallelization Strategies cont…
Task-level Granularity (Coarse-grain tasking)

- Temporal locality refers **to the reuse of specific data and/or resources** within a relatively small time duration.

Parallelization Strategies cont…
Data-level Granularity

- Data-level granularity defines the degree of decomposition of initial data into data subsets.
- It also has two types:
    1. Fine-grain Tiling
    2. Coarse-grain Tiling

Parallelization Strategies cont…
Data-level Granularity (Fine-grain Tiling)

- **Fine-grain Tiling** consists in decomposing the ***data*** into ***small subsets*** (small tiles in the case of image processing).
- These ***small tiles*** are assigned to ***small groups of threads***.
- This strategy exposes a **high degree** of ***concurrency*** and takes advantage of ***architectures*** supporting a huge number of ***threads***.

Parallelization Strategies cont…
Data-level Granularity (Fine-grain Tiling)

- In addition, this strategy is usually ***not constrained*** by the ***hardware resources limitations***.
- However, it suffers from a significant overhead in processing ***replicated data*** at borders to handle **boundary dependencies**.
- Boundaries are created as per purposes, each boundary must create objects related to a single area of concern, like a Layer, a Feature, etc.

Parallelization Strategies cont…
Data-level Granularity (Coarse-grain Tiling)

- **Coarse-grain Tiling** consists in decomposing data into ***large data subsets*** and involving ***large groups of threads***.
- This strategy **reduces** the ***data replication*** at borders and offers ***high space data locality***.
- However large tiles may not **fit well** with available resources, ***cache*** or ***local memory size***, which may ***degrade performance***.

Parallelization Strategies cont…
Types of Parallelization

- Types of parallelism or, also known as parallelism models, define the way to organize ***independent workflows***.
- We can distinguish three main types of parallelism:

1. Data parallelism
2. Task parallelism
3. Pipeline parallelism

Parallelization Strategies cont…
Types of Parallelization (Data Parallelism)

- Data parallelism refers to ***work units*** executing the ***same operations*** on a **set of data**.
- The data is typically organized into a **common structure** such as ***arrays***.
- Each work unit performs the ***same operations*** as other work units but on ***different elements*** of the **data structure**.

Parallelization Strategies cont…
Types of Parallelization (Data Parallelism)

- The first concern of this parallelism type is how to distribute data on work units while keeping them independent.
- Data parallelism is generally easier to exploit due to the simpler computational model involved.

Parallelization Strategies cont…
Types of Parallelization (Task Parallelism)

- Task parallelism is known also as ***functional parallelism*** or control parallelism.
- This type of parallelism considers the case when work units are executing on different ***control flow paths***.
- Work units may execute different operations on either the **same data** or **different data**.

Parallelization Strategies cont…
Types of Parallelization (Task Parallelism)

- In task parallelism, work units can be **known** at the ***beginning of execution*** or can be ***generated at runtime***.
- Task parallelism could be expressed in the GPU context in two ways.

  In a **multi-GPU environment**, each task could be processed by a **separate GPU**.

Parallelization Strategies cont…
Types of Parallelization (Task Parallelism)

- In a **single GPU environment**, the task parallelism is expressed as ***independent kernel queues*** called respectively ***streams*** and command queues in **CUDA** (**Compute Unified Device Architecture**) and **OpenCL**.
- kernel queues are executed concurrently where each kernel queue performs its workload in a data-parallel fashion.

Parallelization Strategies cont…
Types of Parallelization (Pipeline Parallelism)

- Pipeline parallelism is also known as **temporal parallelism**.
- This type of parallelism is applied to chains of **producers** and **consumers** that are directly connected.
- Each task is divided in a number of ***successive phases***.
- The result of each work unit is delivered to the next for processing.

Parallelization Strategies cont…
Types of Parallelization (Pipeline Parallelism)

- At the same time, the producer work unit starts to process a given phase of a new task.
- Compared to data parallelism, this approach offers ***reduced latency***, ***reduced buffering***, and ***good locality***.

Parallelization Strategies cont…
Types of Parallelization (Pipeline Parallelism)

- However, this form of pipelining introduces ***extra synchronization***, as producers and consumers must stay ***tightly coupled*** in their execution.