Assignment - 2

Group - 2

1. **Raman Sharma – 22114076**
Contribution: Described control and data parallelism in PDC.
Contact No: +91 75218 29455
Email ID: raman_s@cs.iitr.ac.in

2. **Boda Yashwanth – 22114022**
Contribution: Described the effects of control and data parallelism on traditional software process models.
Contact No: +91 89779 25111
Email ID: boda_y@cs.iitr.ac.in

3. **Ayush Ranjan – 22114018**
Contribution: Described distributed computing from system, programming and architecture perspectives.
Contact No: +91 74829 58551
Email ID: ayush_r@cs.iitr.ac.in

4. **Souvik Karmakar – 22114096**
Contribution: Described parallel and distributed computing from system, programming and architecture perspectives.
Contact No: +91 86429 24659
Email ID: souvik_k@cs.iitr.ac.in

5. **Sarvasva Gupta – 22114086**
Contribution: Described control and data parallelism in PDC.
Contact No: +91 79899 04811
Email ID: sarvasva_g@cs.iitr.ac.in

6. **Anvit Gupta – 22114009**
Contribution: Described parallel computing from system, programming and architecture perspectives.
Contact No: +91 94620 11044
Email ID: anvit_g@cs.iitr.ac.in

7. Vineet Kumar – 22114107
Contribution: Contributed by analysing the effects of control and data parallelism in traditional software process models.
Contact No: +91 92634 36403
Email ID: vineet_k@cs.iitr.ac.in

Q. Discuss parallel and distributed computing from the following perspective: System, Programming, and Architecture with appropriate examples.

Ans:

System Perspective:

- **Parallel Computing:** In parallel computing, multiple processors perform multiple tasks assigned to them simultaneously. Memory in parallel systems can either be shared or distributed. Parallel computing provides concurrency and saves time and money.

  Example: Supercomputers use parallel computing to perform large-scale simulations and data processing tasks.

- **Distributed Computing:** In distributed computing, we have multiple autonomous computers which seem to the user as a single system. In distributed systems, there is no shared memory and computers communicate with each other through message passing. A single task is divided among different computers.

  Example: The World Wide Web is a massive distributed computing network.

Programming Perspective:

- **Parallel Programming:** Parallel programming focuses on arranging multiple activities to co-occur. It involves writing programs that can execute multiple operations at once. These programs are typically executed within the shared-memory paradigm (e.g., multi-threading or SIMD instructions) or in a distributed-memory paradigm (e.g., message-passing interface).

- **Distributed Programming:** Distributed programming focuses on arranging that activities occur in different places. It involves writing programs that run on multiple connected computers at the same time. The computers communicate and coordinate their actions by passing messages to each other.

Architecture Perspective:

- **Parallel Architecture:** Parallel computer architectures can be classified into two categories: shared-memory systems and distributed-memory systems. Shared-memory systems have multiple processors that share a single, global memory space.

In contrast, distributed-memory systems consist of processors with their own local memory.

Example: Multi-core processors in most personal computers are an example of shared-memory parallel architecture.

- **Distributed Architecture:** In distributed architectures, each node in the system is an independent computer with its own memory and CPU. These nodes communicate with each other through a network.

  Example: Cloud computing platforms like Amazon Web Services (AWS) or Google Cloud Platform (GCP) are examples of distributed architectures.

The choice between parallel and distributed computing depends on the problem at hand and the resources available. Some problems are best solved with a large number of lightweight, distributed processes, while others require the raw power of parallel computation.

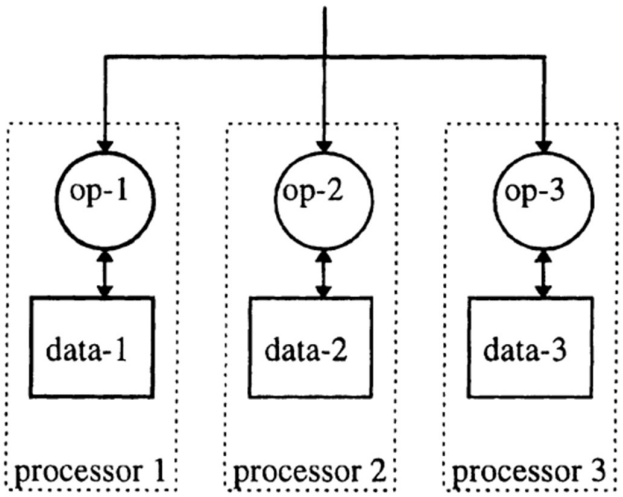
Q. What do you mean by control and data parallelism in PDC? Discuss with examples.

Ans:

Control Parallelism:

Explanation:

Control parallelism involves dividing a task into smaller, independent units of work that can be executed concurrently. These work units may represent different branches of a program or iterations of a loop. Control parallelism enables multiple program parts to execute simultaneously, improving overall performance and efficiency.



Control Parallelism

Example: Task Scheduling in a Multithreaded Scheduler.

Consider a multithreaded task scheduler responsible for managing the execution of various tasks within an operating system or application. The scheduler needs to allocate tasks to available threads efficiently to maximize system throughput and responsiveness.

- **Task Queue**: The scheduler maintains a queue of tasks waiting to be executed.
- **Multiple Threads**: There are multiple worker threads available for task execution.
- **Task Allocation**: The scheduler assigns tasks from the queue to available threads based on priority or other scheduling policies.
- **Task Execution**: Each thread executes its assigned task independently of other threads.

Benefits:

- **Improved Throughput**: By parallelizing task allocation and execution, the scheduler can utilize available resources more effectively, leading to improved system throughput.
- **Enhanced Responsiveness**: Parallel task execution ensures that tasks are processed promptly, enhancing system responsiveness and user experience.

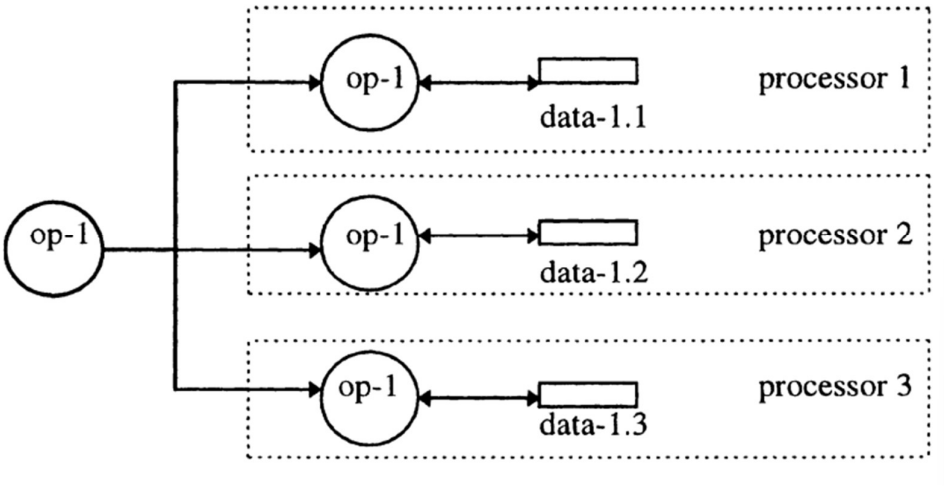
Applications of Control Parallelism:

- **Web Server Processing**: Handles multiple client requests simultaneously, improving overall throughput and responsiveness.
- **Real-Time Systems**: Ensures timely execution of critical tasks by parallelizing independent operations within strict time constraints.

Data Parallelism:

Explanation:

Data parallelism aims to exploit parallelism by dividing a large dataset into smaller segments and processing each segment concurrently across multiple processing units. The primary characteristics of data parallelism include:



Data Parallelism

- **Uniformity of Operations**: The same operation or task is applied to each subset of the data.
- **Independent Data Segments**: Data segments can be processed independently of each other, without requiring synchronization or coordination between segments.

Example: Parallel Image Processing using Data Parallelism.

Data parallelism can be applied to image processing tasks where operations need to be performed on individual pixels or segments of an image simultaneously. By distributing image data across multiple processing units, such as CPU cores or GPUs, and applying the same operation to each segment concurrently, data parallelism can significantly accelerate image processing tasks.

- **Image Segmentation**: The image is partitioned into smaller segments, with each segment containing a subset of pixels.
- **Parallel Processing**: Each segment of the image is processed independently using parallel processing units.
- **Uniform Operations**: The same image processing operation (e.g., filtering, blurring, edge detection) is applied to each segment concurrently.
- **Result Integration**: Processed segments are combined to reconstruct the final output image.

Benefits:

- **High Throughput:** Parallel processing of image segments allows for efficient utilization of processing resources, resulting in higher throughput.
- **Reduced Processing Time**: By processing image segments concurrently, the overall processing time is significantly reduced, leading to faster results.

- **Scalability**: The approach is scalable and can accommodate images of varying sizes without compromising performance.

Applications of Data Parallelism:

- **Big Data Analytics**: Distributes large datasets across a cluster for parallel processing in frameworks like Hadoop and Spark.
- **Machine Learning and Deep Learning**: Accelerates training and inference tasks by distributing workload across multiple GPUs or CPUs.
- **Scientific Computing**: Speeds up computations in simulations and analyses of large datasets in fields like climate modelling.

Q. Study and analyse the effects of data and control parallelism on traditional software process models.

Ans: Analysing the impact of control and data parallelism on different software development life cycle models requires understanding how each model works and where parallelism could be applied. Here's a breakdown for each model:

I. **Classical Waterfall Model:**

- Control Parallelism: Not applicable. The sequential nature of the waterfall model, where each phase (Requirement, Design, Implementation, Testing, etc.) must be completed before the next begins, prevents meaningful control parallelism.
- Data Parallelism: Limited applicability. While independent components or modules within a phase could potentially be developed in parallel, the rigid dependencies across phases hinder large-scale data parallelism benefits.

II. **Iterative Waterfall Model:**

- Control Parallelism: Somewhat applicable. Iterations introduce feedback loops allowing parallel work on different iterations' phases for certain activities. However, dependencies within each iteration still limit broad application.
- Data Parallelism: Increased applicability. With independent functionalities being defined and developed iteratively, opportunities for data parallelism within each iteration grow. However, careful phase management is crucial to avoid issues.

III. **Spiral Model:**

- Control Parallelism: Partially applicable. The risk-driven, cyclical nature of the spiral model allows some activities to be parallelized within and across cycles. However, risk mitigation and overall project control often limit extensive parallelism.

- Data Parallelism: Moderate applicability. Similar to the iterative model, independent functionalities within each cycle can benefit from data parallelism. However, managing dependencies and risk mitigation requirements remains crucial.

IV. **V-Model:**

- Control Parallelism: Limited applicability. Similar to the waterfall model, the sequential nature of phases (requirements-design and testing-implementation) restricts opportunities for effective control parallelism.

- Data Parallelism: Limited applicability. While independent modules within each phase could be developed in parallel, the strong dependencies across phases hinder significant data parallelism benefits.

Conclusion: Control parallelism faces significant challenges in all models due to their inherent sequentially. Data parallelism holds more promise, especially in iterative and spiral models with opportunities for independent development within specific phases or iterations. It's important to remember that parallelism should be carefully evaluated and implemented considering project complexity, dependencies, and potential risks. It's important to note that these are general observations, and the specific effects of parallelism can vary depending on the project characteristics and implementation details.