Hybrid Parallel Computing

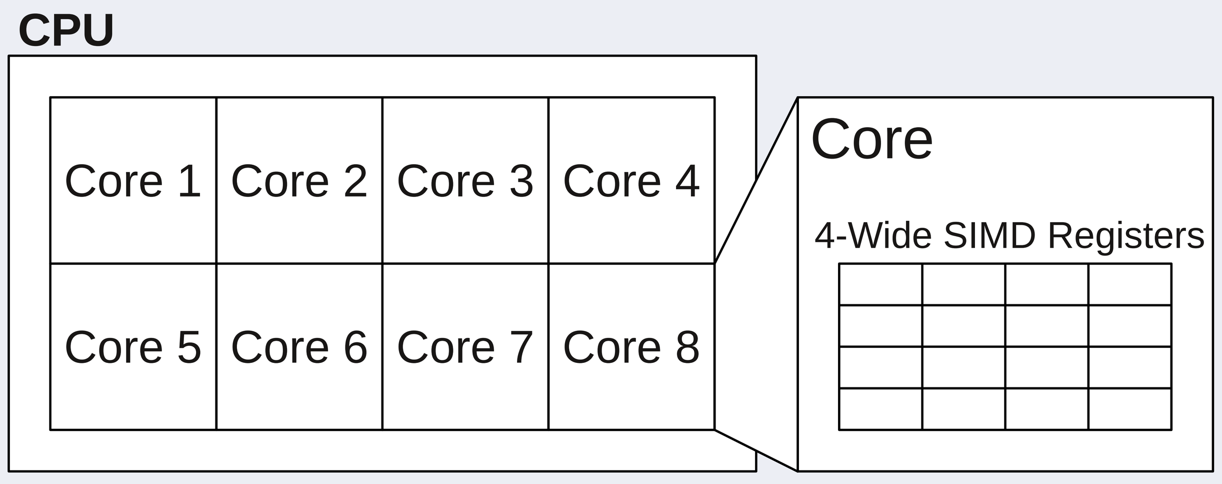
Sujin Philip, Brian Summa, Peer-Timo Bremer, Valerio Pascucci

SCI X 2012

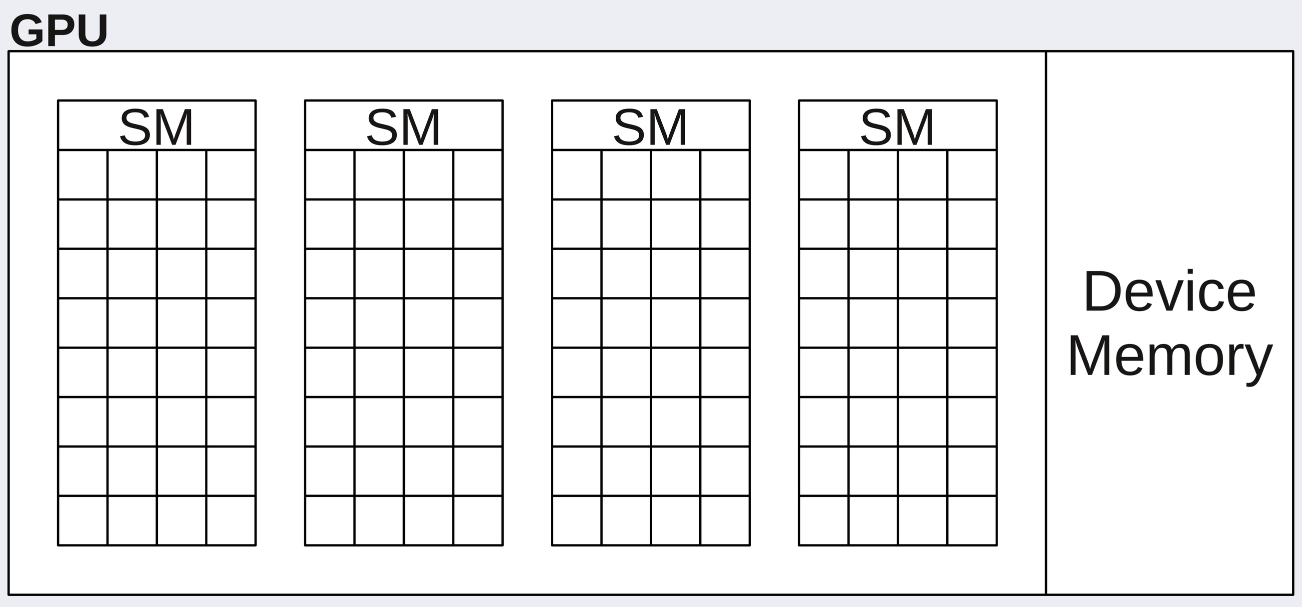
Motivation:

- Clusters of commodity hardware are the most ubiquitous form of computing power available today.
- The processing power of such clusters has been consistently increasing due to:
  - Increase in the number of connected nodes.
  - Increase in available parallelism within the nodes in the form of multicore CPUs and GPUs.
- Hybrid parallel computing is the efficient use of all these different types of resources to achieve high performance.

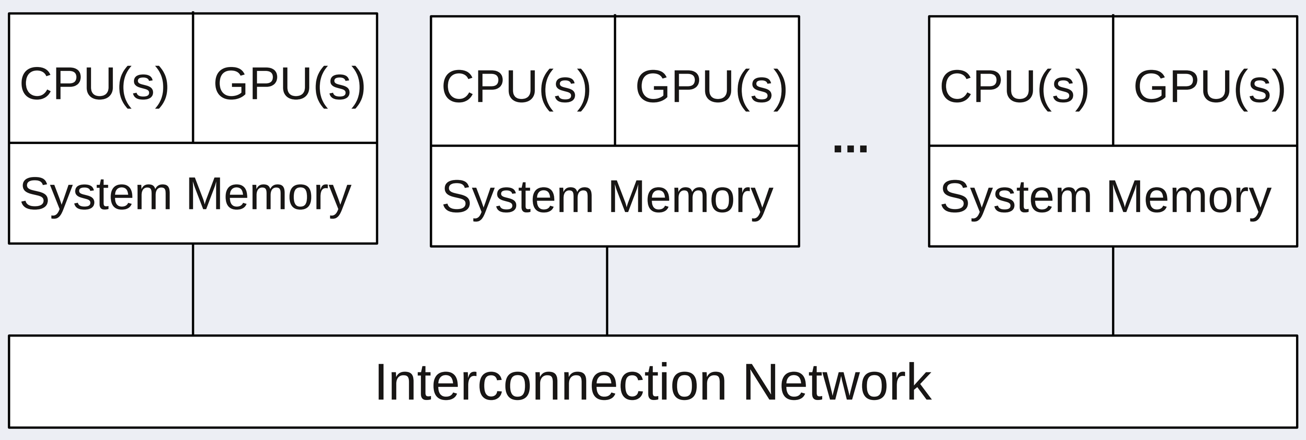
Hardware Architectures:



- A multicore CPU consists of many large, powerful cores.
- Within these cores, SIMD instructions and registers provide further parallelism.



- A cuda enabled GPU:
  - Contains multiple independent processors called Streaming Multiprocessors (SM).
  - Each SM contains many smaller cores that execute in a SIMD fashion.



Cluster

- A typical cluster contains multiple nodes of computers working in parallel and communicating via a fast interconnection network.

Shared and Distributed Memory Systems:

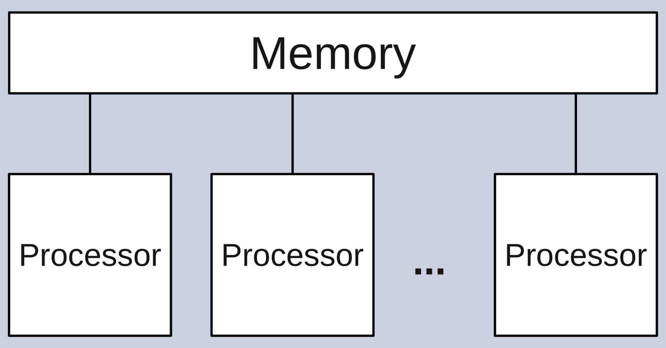


Fig: Shared Memory System

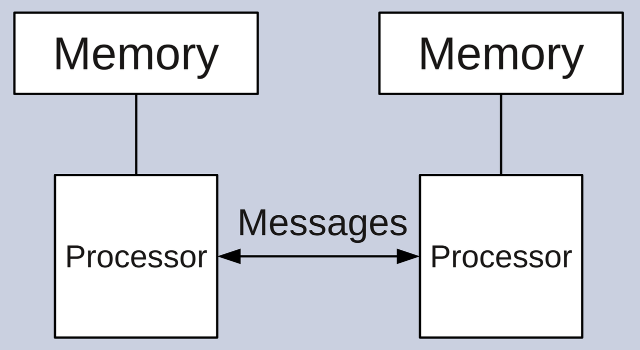


Fig: Distributed Memory System

The programming paradigms for parallelism differ based on the architecture.

Our Approach:

- Divide the computation into a set of tasks.
- Divide large input data into smaller chunks.
- Create a *graph* with:
  - The tasks as the nodes.
  - Dependencies as the edges.
- Compilers may be able to analyze code and create such graphs automatically.

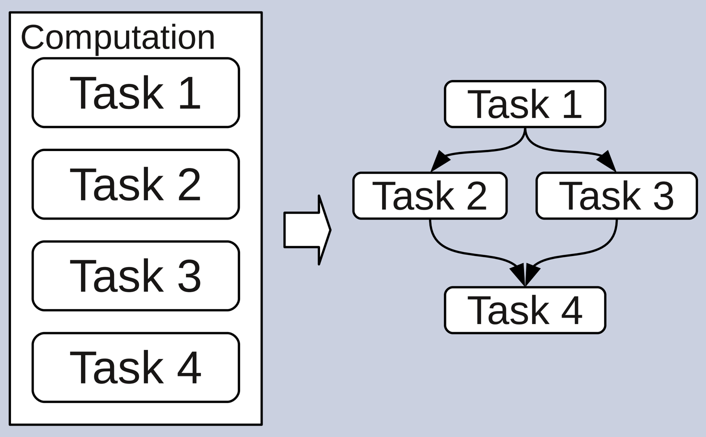


Fig: Computation and Task Graph

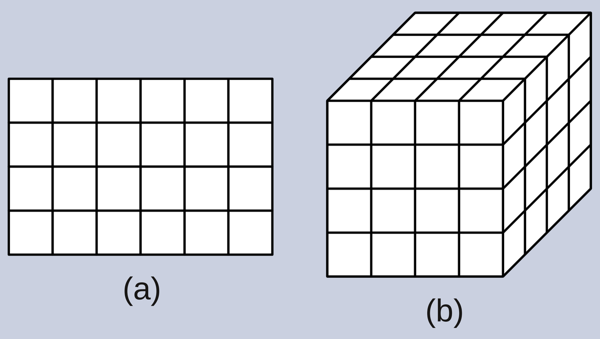


Fig: Data divided into chunks

- A scheduler [3] analyzes the graph and schedules the tasks and data chunks so as to:
  - Achieve high efficiency.
  - Achieve balanced load among the resources.
  - Minimize communication between the nodes.

Types of Parallelism:

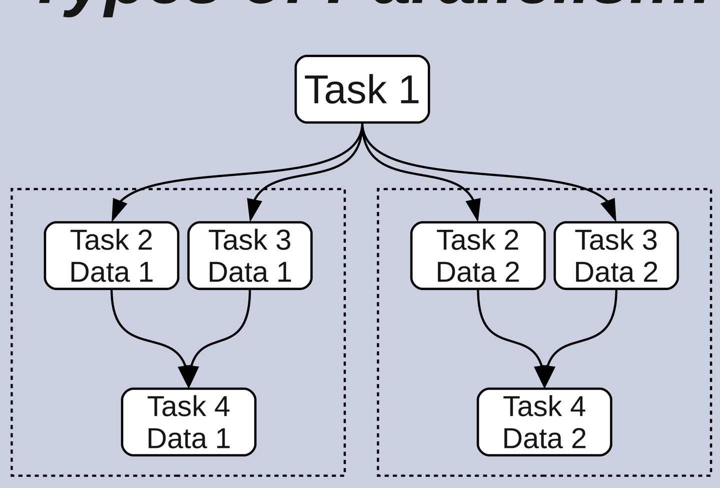


Fig: Data Parallelism

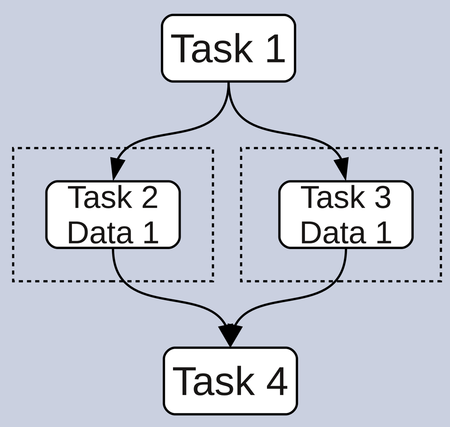


Fig: Task Parallelism

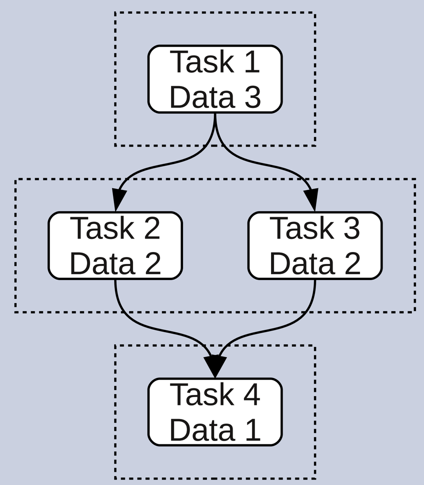
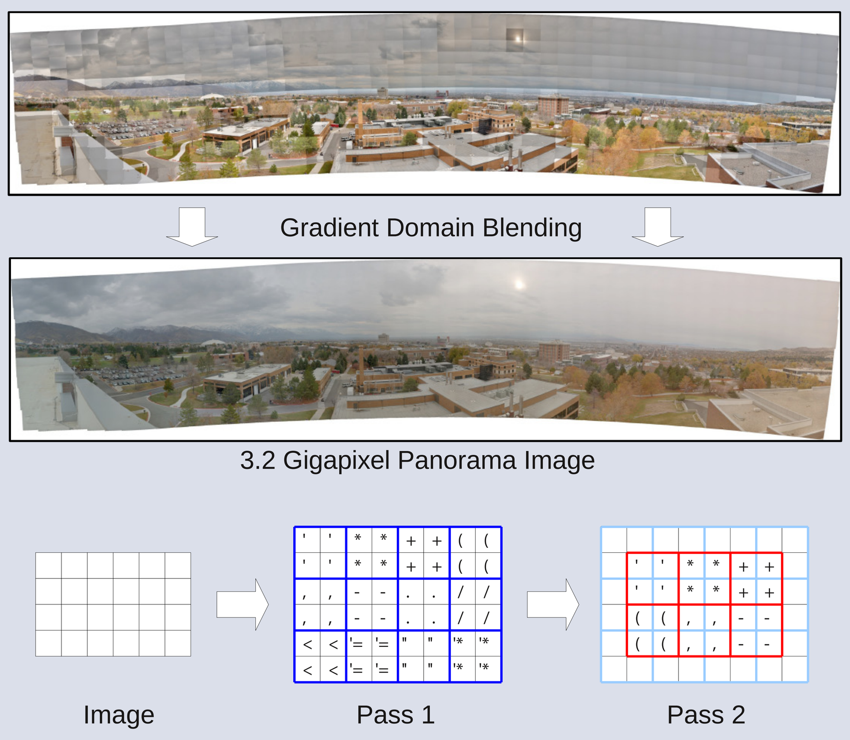


Fig: Pipeline Parallelism

With this design we are able to take advantage of three fundam-ental types of parallelism: data, task and pipeline parallelism.

Example:



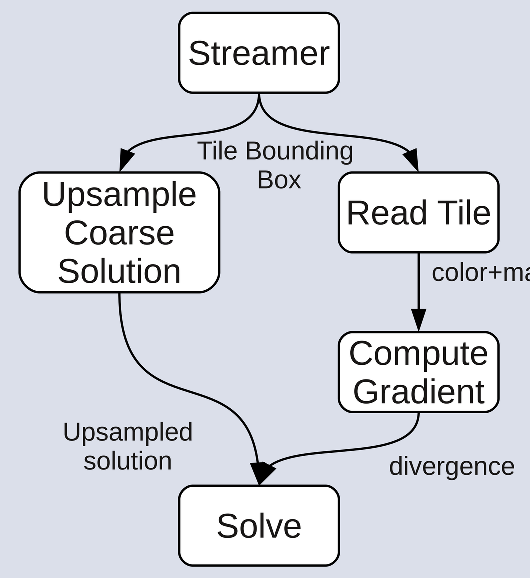


Fig: Task Graph

We have implemented a Hybrid solver for Gradient Domain Blending of massive panoramic images [1][2].

References:

[1] Philip S., Summa B., Bremer P. T., Pascucci V.: **Parallel Gradient Domain Processing of Massive Images**. In proceedings of the Eurographics Symposium on Parallel Graphics and Visualization 2011, pp. 11-19.

[2] Philip S., Summa B., Bremer P. T., Pascucci V.: **Hybrid CPU-GPU Solver for Gradient Domain Processing of Massive Images**. In proceedings of the International Conference on Parallel and Distributed Systems 2011, pp. 244-251.

[3] Vo H. T., Osmari D. K., Summa B., Comba J. L. D., Pascucci V., Silva C. T.: **Streaming-enabled parallel dataflow architecture for multicore systems**. *Comput. Graph. Forum* 29, 3 (2010), 1073-1082.