Toward HPC

Chapter 1 – A first take on parallelism

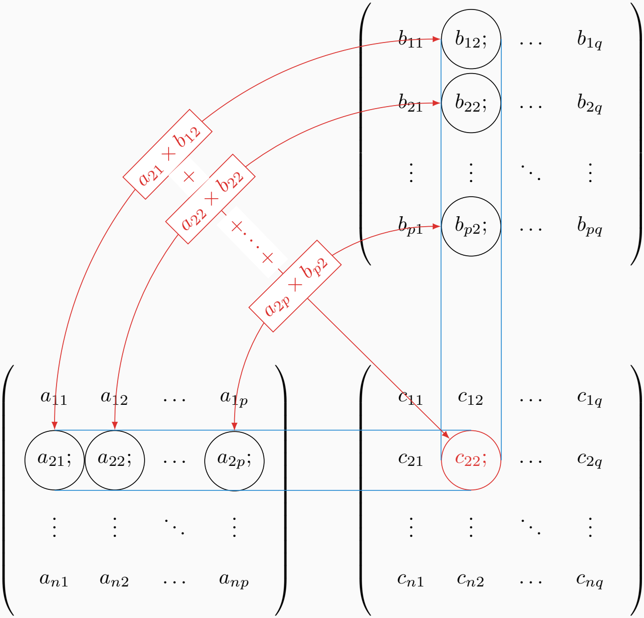
M1 – MSIAM

March 18, 2019

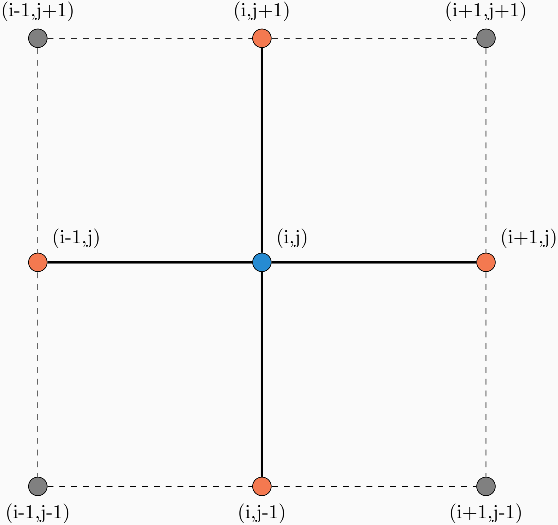
Top 500

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	200,794.9	9,783
2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384

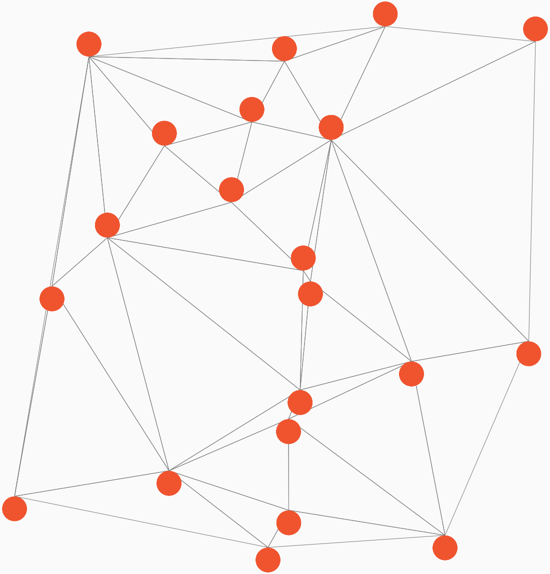
Matrix Multiplication



Images



Graph



Organization

General information

Contact (course + labs)
Jean-Baptiste Keck: jean-baptiste.keck@univ-grenoble-alpes.fr
Office 124 – Imag Building – 1st floor
Email headers: [HPC]

Contact (labs)
Christophe Picard: christophe.picard@imag.fr
Office 174 – Imag Building – 1st floor

Class organization

- Lectures and labs
- Only one written exam, no project

Context

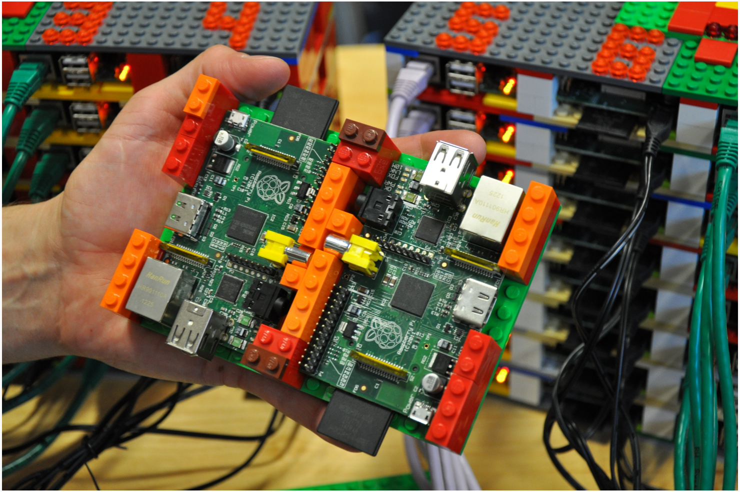
- Decrease time to solution.
- Solve larger problem.
- Combine resources of several processing units: gain access to more memory and more processing power.
- Harness the processing power of modern architectures.
- Use idle computer to perform embarrassing parallelism computation (SETI@home).
- Improve the precision of computations in a limited time (weather forecast).

Objectives

- Understand the different level of parallelism.
- Apprehend the concepts required for real-life applications.
- Experiment with different tools of parallel computing.
- Put into practice the theoretical concepts through an application.

Introduction

Legoland

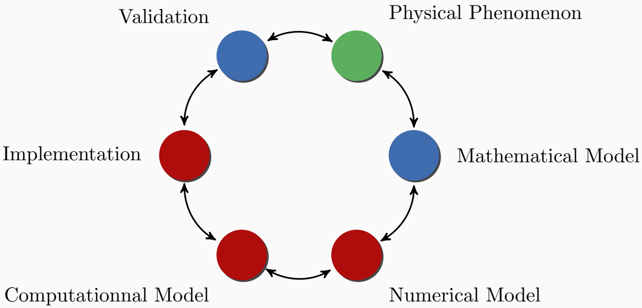


Curie

Why HPC?

- Sequential programming is limited
  - applications are parallel.
  - access to memory is limited.
  - engineering/cost limitations: it is easier to increase the number of unit that the frequency of a processor.
- Expectations for the applications are increasing
  - weather forecast
  - fluid-structure
  - CAD
  - big-data
  - cryptology
- Performances are evolving
  - Hardware is faster.
  - Algorithms are more efficients.

Modeling and computation



Goals of HPC

Intensive computation

- Solve a problem faster.
- Models are more sophisticated.
- Increase resolution of models.
- Increase interactivity.

Example

- Improve the rate: compute N problems simultaneously
  $\Longrightarrow$ Run the same sequential program N times using different inputs.
- Decrease response time: solve a problem with N times faster.
  $\Longrightarrow$ The program is executed only once using N processes.
- Increase the size of the problem: compute a problem N times larger.
  $\Longrightarrow$ The program is executed only once combining N memory resources.

How to increase performances?

HPC attempts to speed solution by dividing task into sub-tasks and executing simultaneously on different processing units.

- Identify where parallelism will be the most effective.
- Know the set of technological constraints.
- Design solution adapted to the problem and the constraints.

Limitations of sequential solutions

Clock speed limitation

- Current leakage.
- Power consumption.
- Heat dissipation.

} Not compatible with mobile devices

Standard optimisations

- Instruction prefetching
- Instruction reordering
- Pipelined functions units
- Branch prediction
- Functional unit allocation
- Hyperthreading

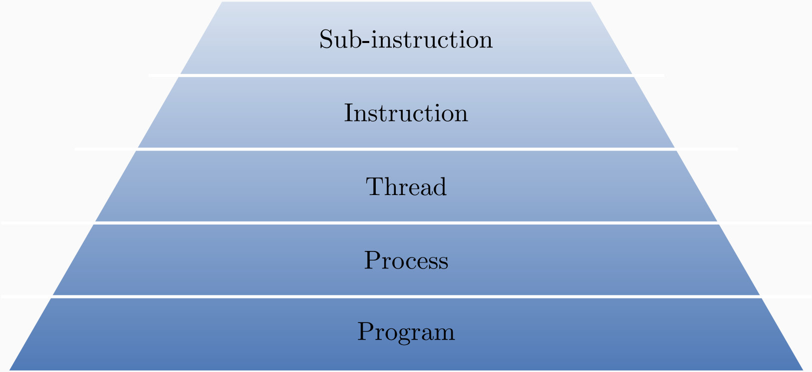
} No control of the programmer

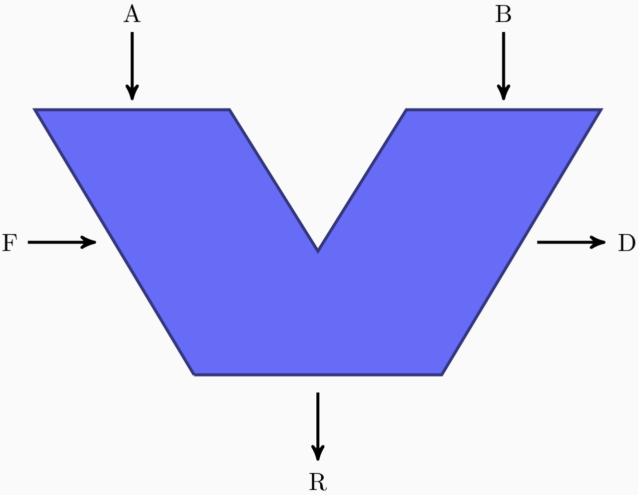
How to achieve parallelism

- Processors: multicore, memory, network, accelerators, instructions.
- Compilers: dedicated library, automatic parallelism.
- Algorithms: tailored algorithms.
- Mathematics: adapted numerical methods, evolutionary methods.

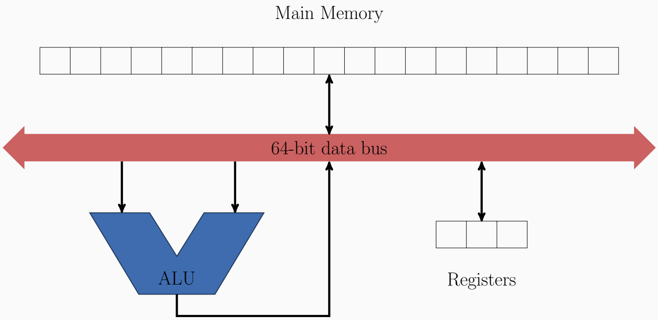
Parallel platforms

Parallel hierarchy

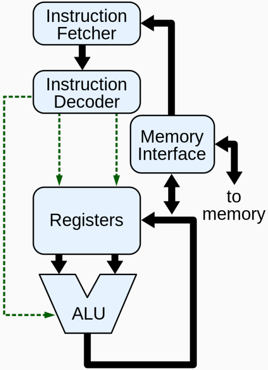


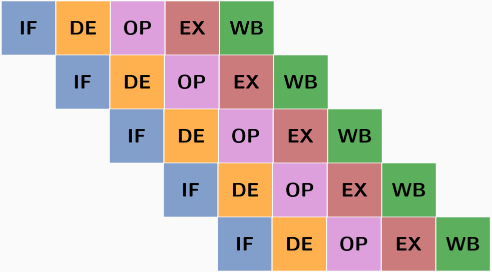

A
B
F
D
R

Abstract processor

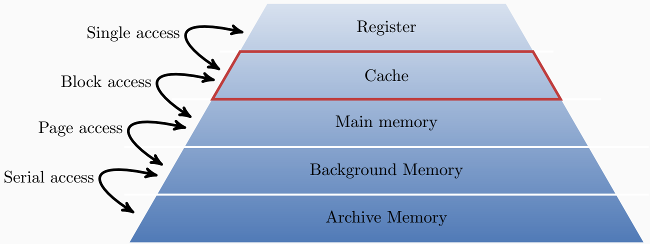


Pipeline architecture





Memory hierarchy



Flynn classification

- SISD – Single Instruction, Single Data
- SIMD – Single Instruction, Multiple Data
- MISD – Multiple Instruction, Single Data
- MIMD – Multiple Instruction, Multiple Data

Most modern architectures are based on MIMD principles.

Main architectures (1)

- Multiple processing units: all the processing unit shares the same global memory.
    - Scaling is complex from the algorithm point of view but also from the technical point of view.
    - Intuitive programming since most of modern programming tools manage memory accesses automatically.
    - Local programming
- Cluster: aggregation of processing unit link through a high speed network. Memories are locals and each unit has its own memory. There is no global memory access.
    - Scaling only depends on the number of resources that are allocated to the cluster.
    - Specific communication protocol must be used for the processors to interact.
    - Optimization of the ratio computation/communication requires careful design.

Main architectures (2)

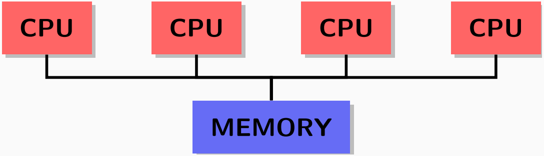
- Hybrids: heterogeneous collection of processing unit with a level of distributed memory. Each node can itself be a shared memory or a distributed memory architecture.
  - Programming is hard: at least two parallel paradigms must be used
  - Optimisation is complex.
  - Performance gain is better.
  - Some resources must be virtualized.

Emerging architectures

- Grid: heterogeneous processing unit link through low speed network (LAN/WAN).
  - Low network
  - No administration required.
  - Only for applications having low network requirements (SETI@HOME)
- Cloud: virtualization of hardware resources.
  - Low performances compared to standard architectures.
  - Cost effective for low usage.

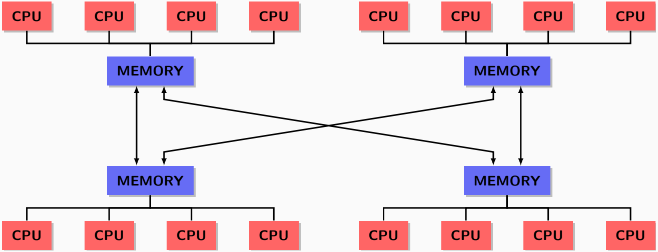
Shared memory (1)

- All the processors shares the same memory space. They communicate using reading and writing shared variables.
- Each processing unit carry out its task independently but modification of shared variables are instantaneous.
- Two kind of shared memories
  - SMP (Symmetric MultiProcessor) – All the processors share a link to the memory. Access to the memory is uniform.



Shared memory (2)

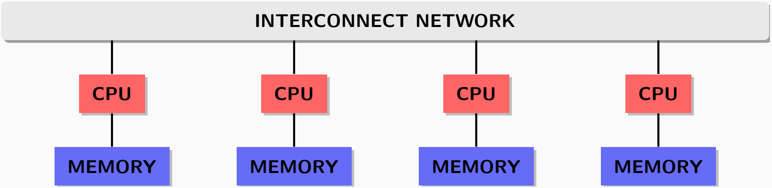
- NUMA (NonUniform Memory Access) – All the processors can access to the memory but not uniformly. Each processor has a preferred access to some memory part.



  - Decrease the risk of bottleneck to memory access.
  - Local memory cache on each processor to mitigate the effect of non-uniform access.

Distributed memory

- Each processor has its own memory. There is no global memory space.
- Each processor communicate with the others using messages.
  - Modification of variables are local and only the processor managing the memory can access it.
  - Each processor work independently on its own set of variables.
  - The speed of the resolution depends on the architecture: network, topology, processors.
  - Can scale easily.



Top 500

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,397,824	143,500.0	200,794.9	9,783
2	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384

OpenMP

- OpenMP (Open Multi-Processing) is an application programming interface that supports shared memory multiprocessing programming
  - It is available for C, C++, and Fortran on most platforms
  - Provides a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications
  - Works for platforms ranging from the standard desktop computer to the multi-socket workstation / compute server.
- OpenMP is an implementation of multithreading
  - A master thread forks a specified number of slave threads and the system divides a task among them.
  - Parallel and critical sections are described by using compiler pragmas directly in the code.
  - Requires specific compiler support (gcc 4.9 has support for OpenMP 4.0), compile with -fopenmp flag.
- We will use this standard for the labs.

Examples on shared memory