Arquitecturas de Alto Desempenho

Introduction to High-Performance Computing

António Rui Borges

Summary

- *High-performance computing*
- *Architectural basics of a parallel machine*
- *Parallel decomposition*
- *Tools to be used to code parallel applications*
- *Suggested reading*

High-performance computing - 1

The area of *High-performance computing* (HPC) is always changing as new technologies and processes become established. In general, it pertains to the use of multiple tightly-coupled processors or computer clusters to run concurrently computational-intensive tasks with high throughput and efficiency. It is common to include in the HPC concept not only the computer architecture, but also a set of elements such as hardware systems, software tools, programming platforms and parallel programming paradigms.

Over the last decade, HPC has evolved significantly, namely due to the emergence of CPU-GPU heterogeneous architectures, which has led to a fundamental paradigm shift in parallel programming.

High-performance computing - 2

Top supercomputer sites (as in November 2021)

Adapted from: https://www.top500.org/lists/2021/11/

System name	*Node characteristics / Interconnect*	*Location*	*Number of nodes / cores Total memory*	*High performance Linpack mark* **PFlops**	*Power* **(MW)**	*Operating System*
Supercomputer Fugaku Fujitsu A64FX 48C	A64FX 48C 2.2GHz Tofu Interconnect D	RIKEN Center for Computational Science Kobe, Japan https://www.r-ccs.riken.jp/en/	158 976 / 7 630 848 48 + 2 core / node 5,09 PB	442,01	29,90	RedHat Linux
Summit IBM Power System AC922	IBM POWER9 22C 3.07GHz NVIDIA Volta GV100 Dual-rail Mellanox EDR Infiniband	Oak Ridge National Laboratory Oak Ridge, United States https://www.ornl.gov	4 608 / 2 397 824 2 (22 core) + 6 GPU / node 2,80 PB	143,50	9,78	RedHat Linux
Sierra IBM Power System S922LC	IBM POWER9 22C 3.07GHz NVIDIA Volta GV100 Dual-rail Mellanox EDR Infiniband	Lawrence Livermore National Laboratory Livermore, United States http://www.llnl.gov	3 022 / 1 572 480 2 (22 core) + 4 GPU / node 1,38 PB	94,64	7,44	RedHat Linux
Sunway TaihuLight (Divine Power, the light of Taihu Lake) Sunway MPP	Sunway SW26010 260C 1.45GHz Sunway	National Supercomputing Center Wuxi, China http://www.nsccwx.cn	40 960 / 10 649 600 4 (1+64 core) / node 1,31 PB	93,01	15,37	Proprietary Linux-based
Perlmutter HPE Cray EX235n	AMD EPYC 7763 64C 2.45GHz NVIDIA A100 SXM4 Slingshot-10	Lawrence Berkeley National Laboratory Berkeley, United States https://www.nersc.gov/systems/perlmutter/	4500 / 761 856 1 / 2 core + 4 GPU / node 0,42 PB	70,87	2,59	HPE Cray OS

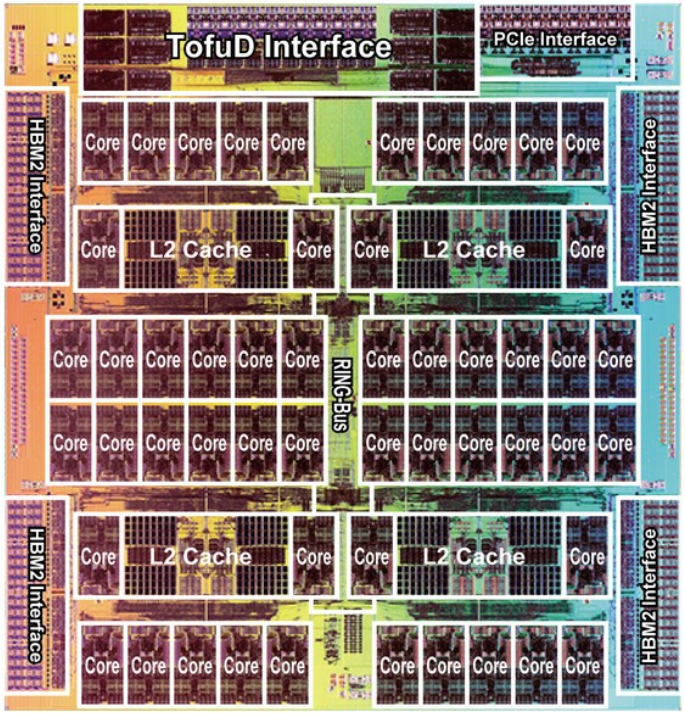
High-performance computing - 3

Supercomputer Fugaku

From: https://www.r-ccs.riken.jp/en/

Fujitsu A64FX 48C

From: https://www.r-ccs.riken.jp/en/fugaku/about/



CPU-Die (Image courtesy of Fujitsu)

High-performance computing - 5

SI prefixes for orders of magnitude in units

Prefix		***Order of magnitude***		***Computing Performance* (FLOPS)**	***Memory Size* (B)**	
K (kilo)	Ki	10^3	2^{10}	KFLOPS	KB	KiB
M (mega)	Mi	10^6	2^{20}	MFLOPS	MB	MiB
G (giga)	Gi	10^9	2^{30}	GFLOPS	GB	GiB
T (tera)	Ti	10^{12}	2^{40}	TFLOPS	TB	TiB
P (peta)	Pi	10^{15}	2^{50}	PFLOPS	PB	PiB
E (exa)	Eo	10^{18}	2^{60}	EFLOPS	EB	EoB
Z (zeta)	Zo	10^{21}	2^{70}	ZFLOPS	ZB	ZoB
Y (yota)	Yi	10^{24}	2^{80}	YFLOPS	YB	YiB

Supercomputing community was aiming to reach EFlops by 2020 and is aiming to reach ZFLOPS by 2030.

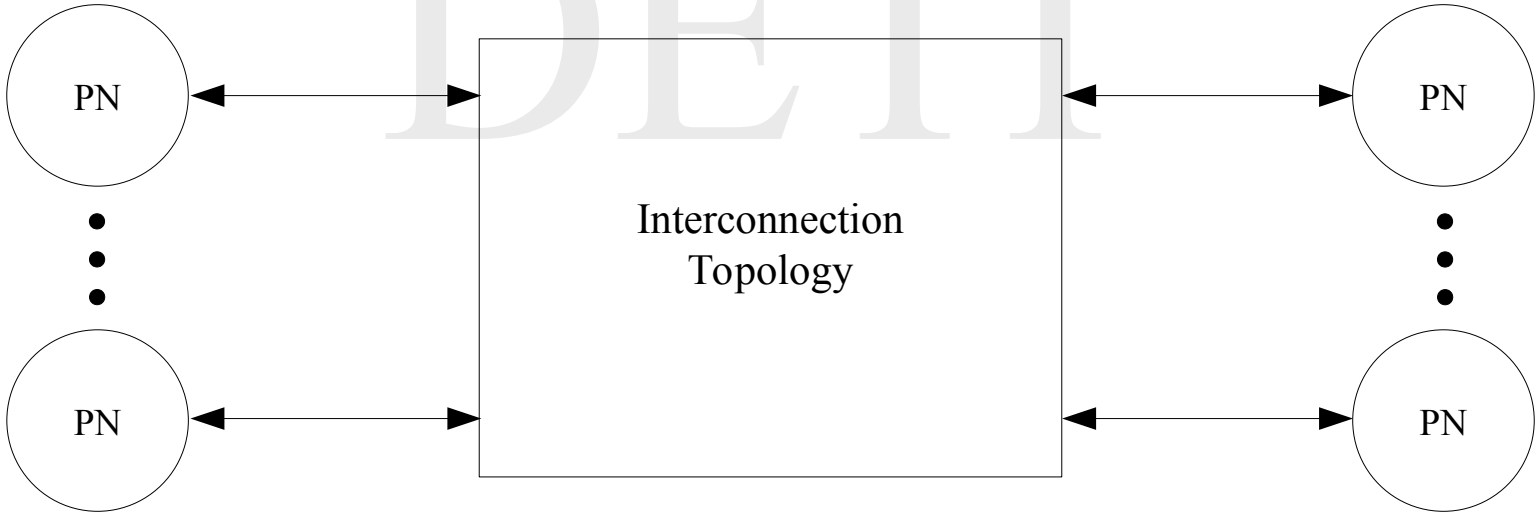
High-performance computing - 6

Major supercomputing application areas

- cosmology, astrophysics and astronomy
- computational chemistry, biology and engineering
- computer science
- earth sciences and materials
- weather forecasting
- geographic information science and technology
- global security
- nuclear fusion
- weapons and complex integration

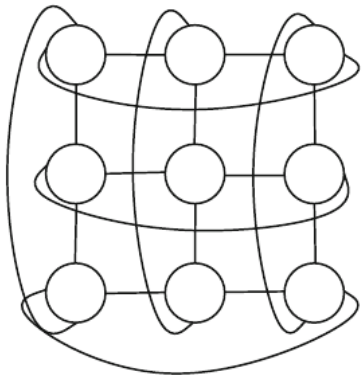
Architectural basics of a parallel machine - 1

Present day high performance computers are at the top level distributed memory parallel machines. They may be thought of as vast clusters of processing nodes (PNs) interconnected by some network topology. The reason for the fact is *scalabilty*, the ability for system performance to increase as new nodes are attached to it.

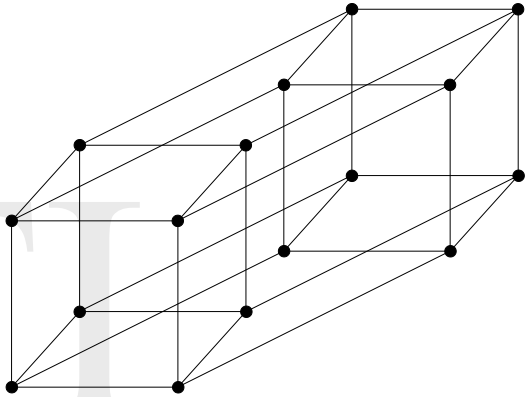


Architectural basics of a parallel machine - 2

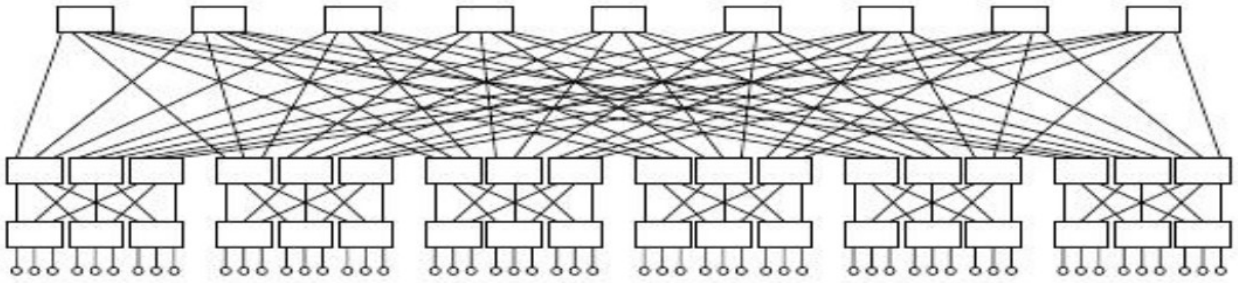
Common interconnection topologies



torus mesh



hipercube



fat tree

Architectural basics of a parallel machine - 3

The key concerns on the interconnection topology are twofold

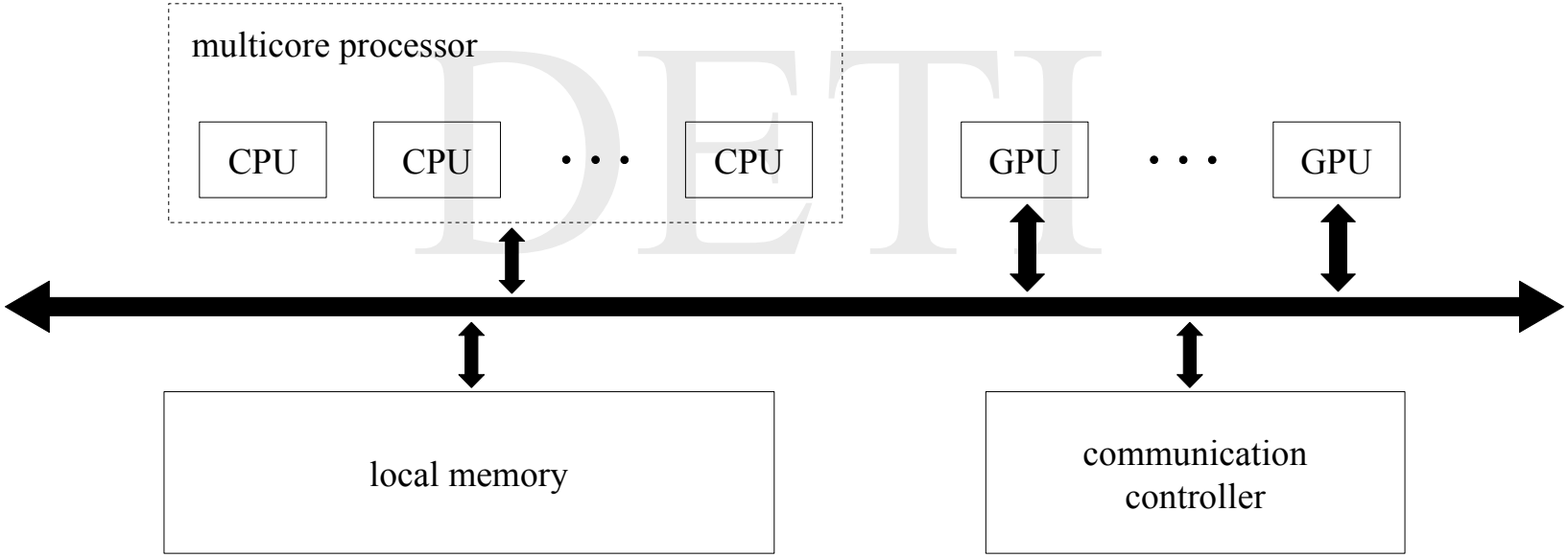
- to keep the number of connections per node small as the number of processing nodes in the cluster increase
- to keep communication time and bandwidth constant as the number of processing nodes in the cluster increase.

Both in the torus mesh and the hipercube, all connections are point-to-point and, as such, have a fixed bandwidth. The number of connections per node is always four in the former case and $\log_2 n$ in the latter case, where $n = 2^k$ is the number of processing nodes in the cluster. The communication time, however, depends on the location of the communication nodes, being at the most $\sqrt{n}$ and $\log_2 n$, respectively, of the equivalent communication time between two adjacent nodes.

A fat tree, on the other hand, is a hierarchical network that tries to keep the same bandwidth at all bisections. All processing nodes transmit at the line speed if the packets are uniformly distributed along the available paths. Since a single connection per node is required and, by using k-port switches, $k^3/4$ processing nodes may be attached to it, it presents good scalability properties.

Architectural basics of a parallel machine - 4

Processing node



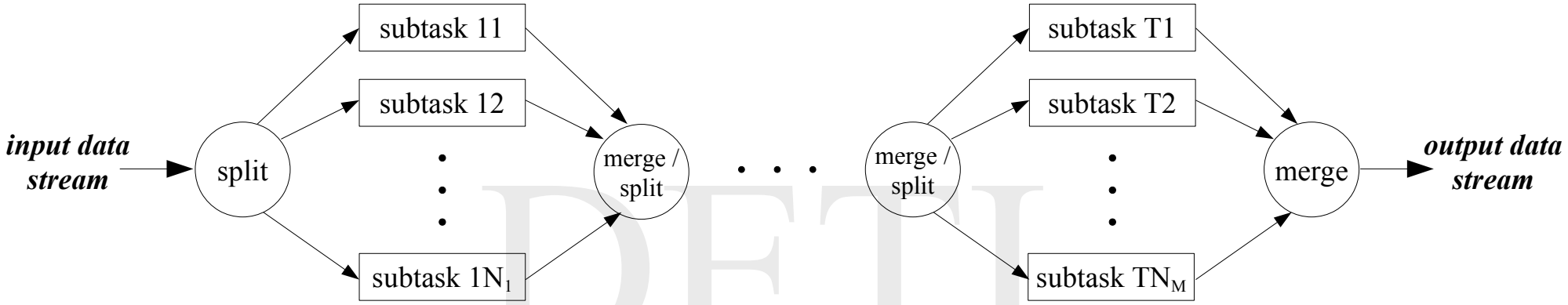
Architectural basics of a parallel machine - 5

A typical processing node consists of one or two multicore CPU sockets and two or more many-core GPUs which gave rise to the name *heterogeneous computing* when referring to this kind of arrangement.

The CPU code is responsible in this context to manage the environment, the code and the data for the GPU, before loading the computation-intensive tasks on the device. GPU computing is not meant to replace CPU computing. CPUs are optimized for dynamic workloads, marked by short sequences of computational operations and unpredictable flow control. On the other hand, GPUs aim at the other hand of the spectrum: workloads that are dominated by computational tasks with simple flow control.

Thus, CPU+GPU heterogeneous parallel computing architectures evolved because the CPU and the GPU have complementary attributes that enable applications to perform best using both types of processors.

Parallel decomposition - 1



Typically, parallel decomposition is data-driven.

Chunks of data of the input stream are fed to a T-stage pipeline of operations. At each stage, data is further split so that operations may be carried out independently in mutual exclusive parts of the chunk being processed. In between the stages, data chunks may undergo reshuffling.

Parallel decomposition - 2

Parallel algorithms may be designed with various degrees of granularity. *Granularity* can be defined as the way how parallel operations are expressed. In this sense, it is not possible to express them without thinking about the hardware platform where the code is going to run.

Parallelism is, thus, organized in three main categories

- *fine-grained parallelism* – parallel operations are expressed at the variable level, it assumes an instruction is executed simultaneously on multiple data sets, a SIMD (single instruction – multiple data) architecture is considered
- *medium-grained parallelism* – parallel operations are expressed at the thread level within a process, a MIMD (multiple instruction – multiple data) architecture of the shared memory type is considered
- *coarse-grained parallelism* – parallel operations are expressed at the process level, a MIMD (multiple instruction – multiple data) architecture of the distributed memory type is considered.

In high performance computing, all three categories of granularity are blended on algorithmic design.

Tools to be used to code parallel applications

Parallel applications that are going to be developed, will be written in C Language. Three specific libraries / APIs will be used to implement parallel granularity presented by the algorithms

- *pthread library* – to create multithreaded applications to be run in shared memory architectures (medium-grained parallelism)
- *MPI* (*message passing interface*) – to create multiprocess applications to be run in distributed memory architectures (coarse-grained parallelism)
- CUDA C – to create applications where parallelism is expressed at the variable level, intended to be run in CPU-GPU heterogeneous architectures (fine-grained parallelism).

Suggested reading

- *Introduction to HPC with MPI for Data Science*, Nielsson F., Springer International, 2016
  - Chapter 1: *A Glance at High Performance Computing (HPC)*
- *Programming Massively Parallel Processors: A Hands-on Approach*, Kirk D.B., Hwu W.W., 3rd Edition, Morgan Kaufmann, 2017
  - Chapter 1: *Introduction*