Chapter 5

Distributed Memory Systems: Part I

Distributed Memory Hierarchy

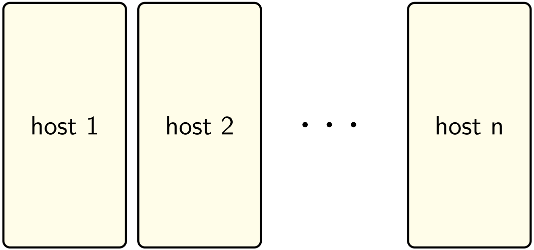

Figure: Distributed memory computer schematic

Distributed Memory Hierarchy

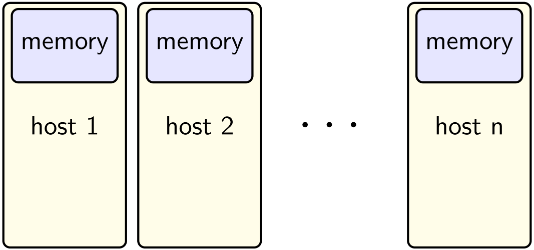

Figure: Distributed memory computer schematic

Distributed Memory Hierarchy

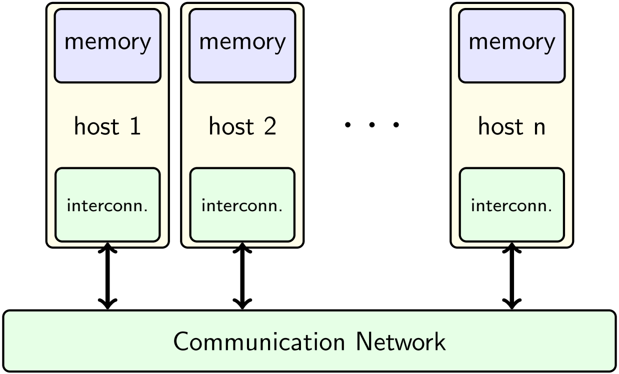

Figure: Distributed memory computer schematic

Comparison of Distributed Memory Systems

Rankings

1. **TOP500[6] :**
   List of the 500 fastest HPC machines in the world sorted by their maximal LINPACK[7] performance (in TFlops) achieved.

[6] http://www.top500.org/
[7] http://www.netlib.org/benchmark/hpl/
[8] http://green.graph500.org/

Comparison of Distributed Memory Systems

Rankings

1. **TOP500 :**
   List of the 500 fastest HPC machines in the world sorted by their maximal LINPACK performance (in TFlops) achieved.
2. **Green500[6] :**
   Taking into account the energy consumption the Green500 is basically a resorting of the TOP500 according to TFlops/Watt as the ranking measure.

[6] http://www.green500.org/
[7] http://green.graph500.org/

Comparison of Distributed Memory Systems

Rankings

1. **TOP500 :**
   List of the 500 fastest HPC machines in the world sorted by their maximal LINPACK performance (in TFlops) achieved.
2. **Green500 :**
   Taking into account the energy consumption the Green500 is basically a resorting of the TOP500 according to TFlops/Watt as the ranking measure.
3. **(Green) Graph500[6] :**
   Designed for data intensive computations it uses a graph algorithm based benchmark to rank the supercomputers with respect to GTEPS (10^9 Traversed edges per second). As for the TOP500 a resorting of the systems by an energy measure is provided, as the Green Graph 500 list[7].

[6] http://www.graph500.org/
[7] http://green.graph500.org/

Comparison of Distributed Memory Systems

Architectural Streams Currently Pursued

The three leading systems in the TOP500 list are currently[8] of three different types representing the main streams pursued in increasing the performance of distributed HPC systems.

Mainly all HPC systems today consist of single hosts of one of the following three types. The performance boost is achieved by connecting ever increasing numbers of those hosts in large clusters.

Comparison of Distributed Memory Systems

Architectural Streams Currently Pursued

1. **Hybrid accelerator/CPU hosts**,
   Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x at DOE/SC/Oak Ridge National Laboratory United States

Comparison of Distributed Memory Systems

Architectural Streams Currently Pursued

1. **Hybrid accelerator/CPU hosts**,

   Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x at DOE/SC/Oak Ridge National Laboratory United States

2. **Manycore and embedded hosts**

   Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz at DOE/NNSA/LLNL United States

Comparison of Distributed Memory Systems

Architectural Streams Currently Pursued

1. **Hybrid accelerator/CPU hosts**,
   `Titan` - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x at DOE/SC/Oak Ridge National Laboratory United States
2. **Manycore and embedded hosts**
   `Sequoia` - BlueGene/Q, Power BQC 16C 1.60 GHz at DOE/NNSA/LLNL United States
3. **Multicore CPU powered hosts**,
   `K computer`, SPARC64 VIIIfx 2.0GHz, Tofu interconnect at RIKEN Advanced Institute for Computational Science Japan

Comparison of Distributed Memory Systems

Hybrid Accelerator/CPU Hosts

We have elaborately studied these hosts in the previous chapter.

Compared to a standard desktop (as treated there) in the cluster version the interconnect plays a more important role. Espacially Multi-GPU features may use GPUs on remote hosts (as compared to remote NUMA nodes) more efficiently due to the high speed interconnect.

Compared to CPU-only hosts, these systems usually benefit from the large number of cores generating high flop-rates at comparably low energy costs.

Comparison of Distributed Memory Systems

Manycore and Embedded Hosts

Manycore and embedded systems are designed to use low power processors to get a good flop per Watt ratio. They make up for the lower per core flop counts by using enormous numbers of cores.

BlueGene/Q

- Base chip IBM PowerPC 64Bit based, 16(+2) cores, 1.6GHz
- each core has a SIMD Quad-vector double precision FPU
- 16 user cores, 1 system assist core, 1 spare core
- cores connected to 32MB eDRAM L2Cache (half core speed) via crossbar switch
- crates of 512 chips arranged in 5d torus ($4 \times 4 \times 4 \times 4 \times 2$)
- chip-to-chip communication at 2Gbit/s using on-chip logic
- 2 crates per rack $\rightsquigarrow$ 1024 compute nodes = 16,384 user cores
- interconnect added in 2 drawers with 8 PCIe slots (e.g. for Infiniband, or 10Gig Ethernet.)

Comparison of Distributed Memory Systems

Multicore CPU Hosts

Basically these clusters are a collection of standard processors. The actual multicore processors, however, are not necessarily of x86 or amd64 type, e.g. the K computer uses SPARC VIII processors and other employ IBM Power 7 processors.

Standard x86 or amd64 provide the obvious advantage of easy usability, since software developed for standard desktops can be ported easily. The SPARC and POWER processors overcome some of the x86 disadvantages (e.g. expensive task switches) and thus often provide increased performance due to reduced latencies.

Comparison of Distributed Memory Systems

The 2020 vision: Exascale Computing

difference	name (symbol)	meaning
	Kilobyte (kB)	10^3 Byte = 1 000 Byte
2,40%	Kibibyte (KiB)	2^{10} Byte = 1 024 Byte
	Megabyte (MB)	10^6 Byte = 1 000 000 Byte
4,86%	Mebibyte (MiB)	2^{20} Byte = 1 048 576 Byte
	Gigabyte (GB)	10^9 Byte = 1 000 000 000 Byte
7,37%	Gibibyte (GiB)	2^{30} Byte = 1 073 741 824 Byte
	Terabyte (TB)	10^{12} Byte = 1 000 000 000 000 Byte
9,95%	Tebibyte (TiB)	2^{40} Byte = 1 099 511 627 776 Byte
	Petabyte (PB)	10^{15} Byte = 1 000 000 000 000 000 Byte
12,6%	Pebibyte (PiB)	2^{50} Byte = 1 125 899 906 842 624 Byte
	Exabyte (EB)	10^{18} Byte = 1 000 000 000 000 000 000 Byte
15,3%	Exbibyte (EiB)	2^{60} Byte = 1 152 921 504 606 846 976 Byte

Table: decimal and binary prefixes

Comparison of Distributed Memory Systems

The 2020 vision: Exascale Computing

The two standard prefixes in decimal and binary representations of memory sizes are given in Table 7. The decimal prefixes are also used for displaying numbers of floating point operations per second (flops) executed by a certain machine.

name	LINPACK Perfomance	Memory Size
Titan	17 590.0 TFlop/s	710 144 GB
Sequoia	16 324.8 TFlop/s	1 572 864 GB
K computer	10 510.0 TFlop/s	1 410 048 GB

Table: Petascale systems available

Comparison of Distributed Memory Systems

The 2020 vision: Exascale Computing

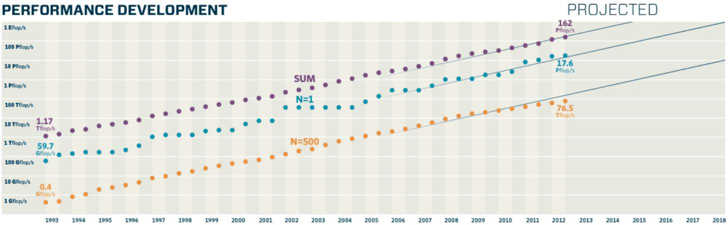

Figure: Performance development of TOP500 HPC machines taken from TOP500 poster November 2012

Comparison of Distributed Memory Systems

State of the art (statistics)

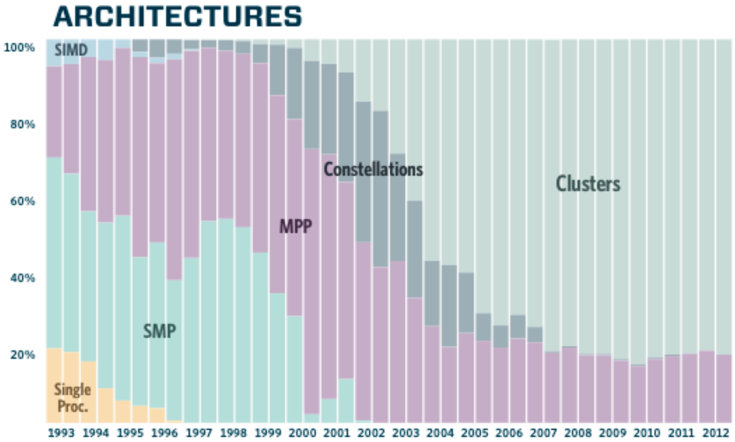

Figure: TOP500 architectures taken from TOP500 poster November 2012

Comparison of Distributed Memory Systems

State of the art (statistics)

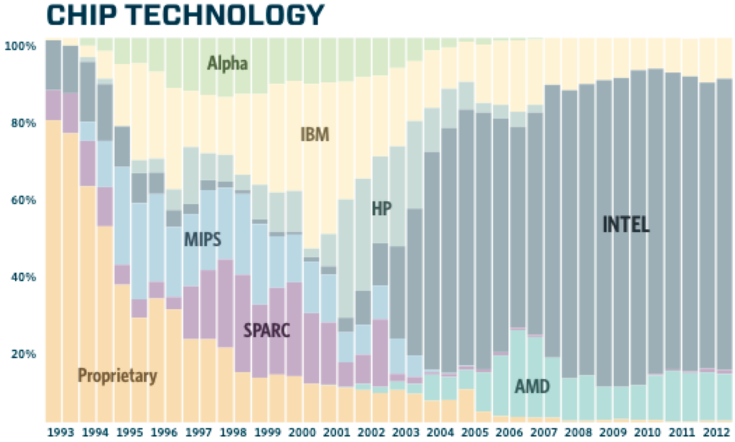

Figure: Chip technologies of TOP500 HPC machines taken from TOP500 poster November 2012

Comparison of Distributed Memory Systems

State of the art (statistics)

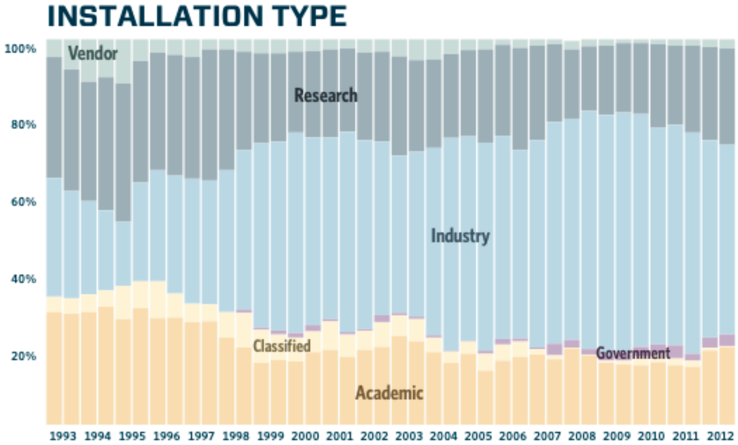

Figure: Installation types of TOP500 HPC machines taken from TOP500 poster November 2012

Comparison of Distributed Memory Systems

State of the art (statistics)

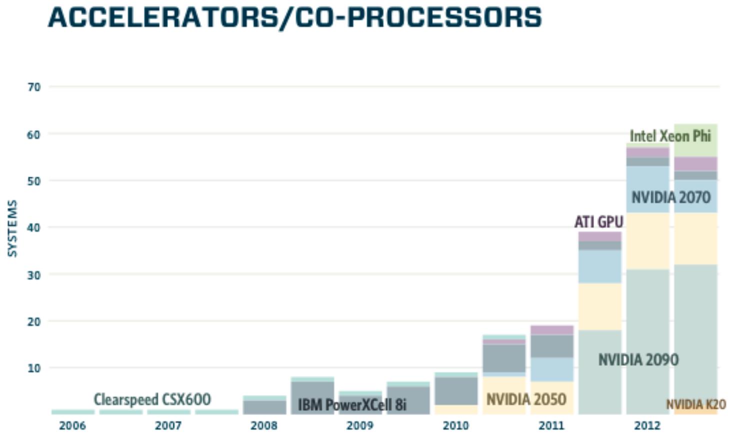

Figure: Accelerators and Co-Processors employed in TOP500 HPC machines taken from TOP500 poster November 2012

Communication of Data

Communication Operations via Message Passing

Message passing

is the programming model commonly used for distributed memory systems, where each node has its own exclusive memory and we have an overall distributed address space. Exchange of data between the local memories of separate hosts is realized by sending messages between the hosts.

Usually the communication is (network) socket based, although the basic principles can also be applied to multicore machines, e.g. by using shared memory blocks to implement the communication.

Communication of Data

Communication Operations via Message Passing: Blocking vs. Non-blocking

Communication operations in the Message Passing Interface (MPI) are belonging to 2 global classes categorized by their local (process on host) behavior.

Definition (blocking operation)

A communication operation is called blocking if the return of the process control to the calling process means that the operation has completed the entire transfer.

Definition (non-blocking operation)

In a non-blocking operation the process control is returned to the calling process as soon as the communication has been initiated. The communication may be ongoing while the calling process continues its program.

Communication of Data

Communication Operations via Message Passing: Synchronous vs. Asynchronous

Looking at the same operations from a global perspective, i.e., not looking at the local message but the global communication, they determine the two classes of

Definition (synchronous communication)

The synchronous communication between a sending an a receiving process is implemented such that sending operations do not complete (i.e. return control to the calling process) before the receiving counterpart has at least started the execution.

Definition (asynchronous communication)

In asynchronous communication the sending and receiving process are not coordinated, i.e., the sender can execute its operation without the receiving counterpart waiting in its operation.

Communication of Data

Communication Operations via Message Passing: Synchronous vs. Asynchronous

Example

- oral or telephone chats are synchronous communications, since all partners are engaged in the communication simultaneously.
- classic mail or electronic mail are asynchronous communication, where the sender never knows if, or when the message was actually received.

Communication of Data

Communication Operations via Message Passing

Communication between MPI processes can not only be classified via their influence on global or local process flow, but also with respect to the number of partners involved. MPI is distinguishing between

- **point-to-point communication**, where both ends are occupied by a single process, and
- **collective communication** where a single process sends out messages to multiple receiving processes, or collects messages from several sending processes.

Communication of Data

Communication Operations via Message Passing: Point-to-Point Communication

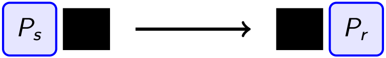

Figure: Point-to-Point Communication

Communication of Data

Communication Operations via Message Passing: Collective Communication

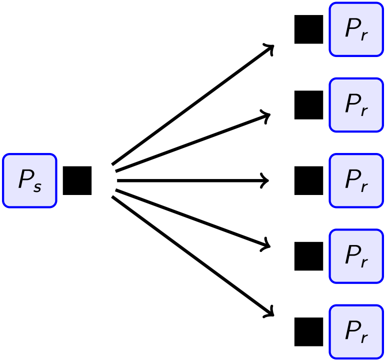

Figure: Broadcast Operation

Communication of Data

Communication Operations via Message Passing: Collective Communication

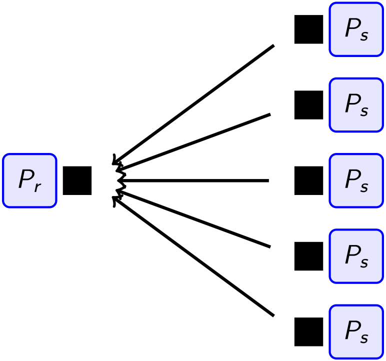

Figure: Reduction Operation

Communication of Data

Communication Operations via Message Passing: Collective Communication

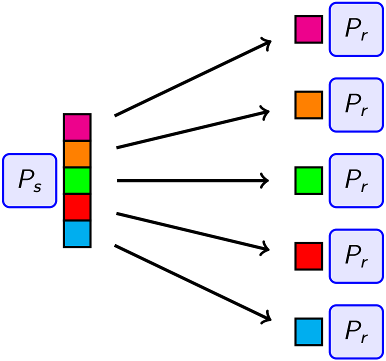

Figure: Scatter Operation

Communication of Data

Communication Operations via Message Passing: Collective Communication

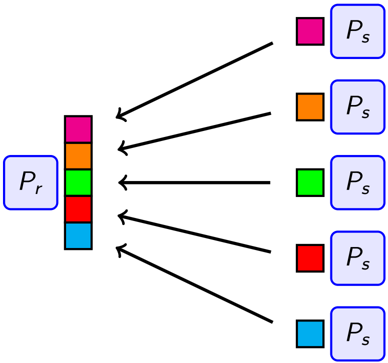

Figure: Gather Operation