Parallel Linux Clusters

prof. Ing. Pavel Tvrdík CSc.

Katedra počítačových systémů
Fakulta informačních technologií
České vysoké učení technické v Praze

InstallFest, 5.3.2011, ČVUT

Parallel Computer Systems

Definition 1

A **parallel computer** is a group of interconnected **computing elements** that cooperate in order to solve faster computationally and data-intensive problems.

Levels of parallelism in computer systems

- **ILP** CPU cores: superscalar, VLIW, multithreaded VLSI computing nodes.
- **Multicore** processors: several (2, 4, 8, 16, ...) CPU cores on 1 VLSI chip.
- **GPU farms, clusters**: more $(10^2 - 10^5)$ GPUs as coprocessors of a CPU.
- **SMP** (Symmetric MultiProcessor) **servers**: units or tens of processors with shared memory.
- **Clusters** of computers: clusters of $10^2 - 10^5$ computers (typically Linux).
- **Tightly-coupled massively parallel supercomputers**: $10^2 - 10^6$ computing nodes interconnected by several special networks.
- **Computing grids**: group of computing systems, specialized for various phases of a larger computation interconnected by Internet.
- **Cloud computing** infrastructure, **data centers**, ...

Embarrassingly parallel applications

- Purely data parallel (embarrassingly parallel) tasks: computations on various data parts are independent and do not require data exchanges or synchronization.
- Examples:
  - Mandelbrot set calculation.
  - Ray-tracing and other rendering in pixel computer graphics.
  - Brute-force searches in cryptography, genomics, other databases.
- Implementation examples:
  - SETI@home.
  - condor (`http://www.cs.wisc.edu/condor/description.html`).
  - SNOW package (`http://www.stat.uiowa.edu/ luke/R/cluster/cluster.html`).

Typical parallel applications

- Parallel linear algebra algorithms:
  - parallel matrix-matrix, matrix-vector multiplications,
  - parallel solvers of a set of linear equations,
  - parallel matrix-matrix multiplications,
  - finite volume method,
  - finite element method,
- parallel combinatorial space search.

Příklad náročné aplikace: předpověď počasí

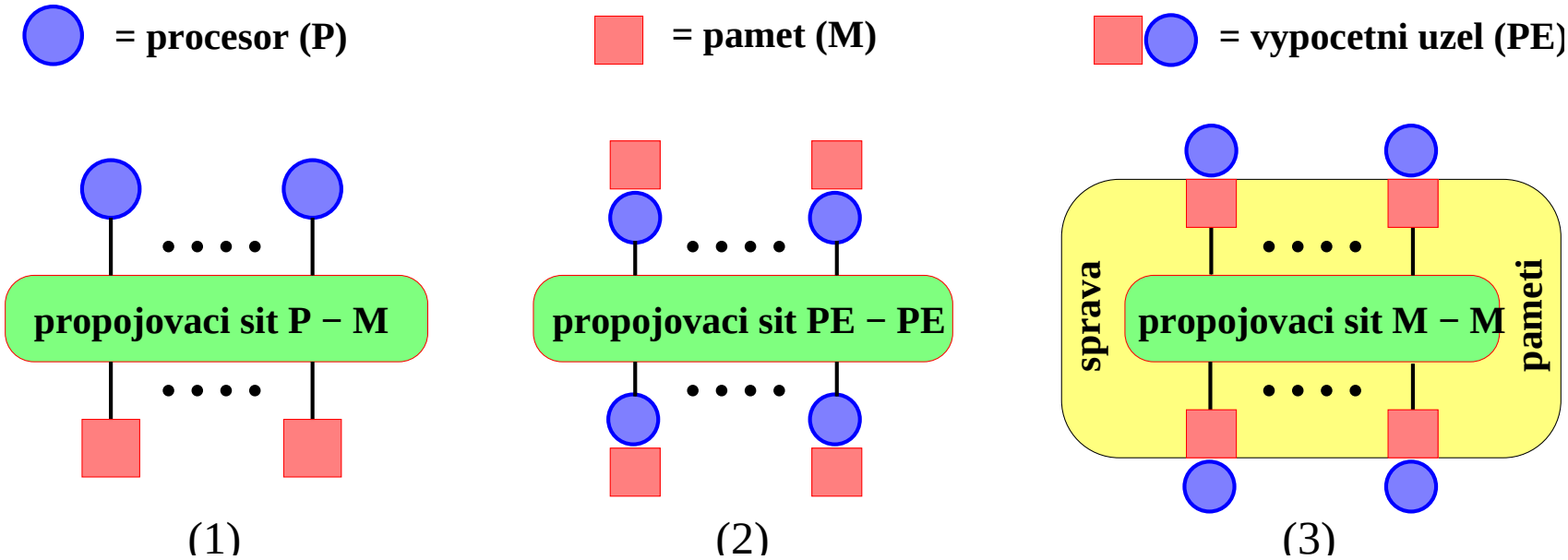
- Předpověď počasí v **oblasti** o velikosti $3000 \times 3000 \times 11\ km^3$ na dobu **2 dnů**.
- Oblast rozdělena na **segmenty** (např. metodou konečných prvků) o velikosti $0.1 \times 0.1 \times 0.1\ km^3 \quad \Rightarrow \quad \#$ segmentů $\sim 10^{11}$.
- Parametry modelu (teplota, rychlost větru,...) jsou počítány s časovým krokem 30 minut.
- **Nové** hodnoty parametrů segmentu jsou počítány z **předchozích** hodnot parametrů sousedních segmentů.
- Nechť výpočet parametrů 1 segmentu spotřebuje 100 instrukcí.
  - Pak 1 **iterace** = **aktualizace hodnot** všech parametrů v celé oblasti vyžaduje cca $10^{11} \times 100 \times 96 \doteq 10^{15}$ operací.
  - Sekvenční počítač s 1GFlops spotřebuje 10^6 sec $\doteq 11$ dní.
  - To je ale pozdě, protože modelujeme počasí pro příští 2 dny.
- Paměťový problém (data se nevejdou do hlavní paměti sekv. počítače a musí být odkládána na disk) řešení ještě mnohonásobně zpomalí!
- Pro výpočet spolehlivého modelu počasí je třeba mnoho iterací!!!!!

  $\Rightarrow$ paralelní rozdělení dat a výpočet = **jediné schůdné řešení**.

Taxonomy of parallel computers

- Instruction and data flow architecture view: SIMD, MIMD.
- Memory organization view.
- Interconnection network view.

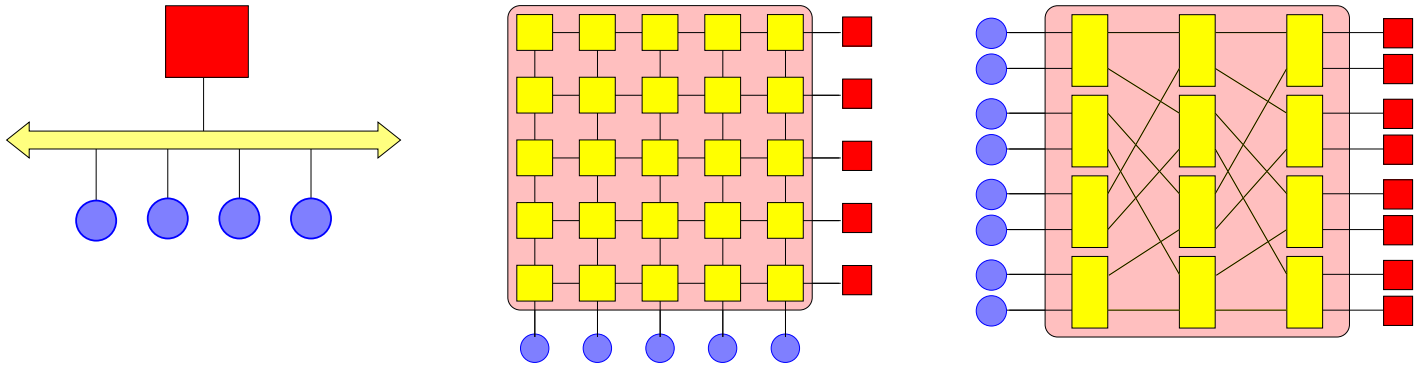
Memory organization view



1. multiprocessor systems with **shared memory**, symmetric multiprocessors (SMP, UMA)
   - HW/SW communication = Read/Write
2. multiprocessor systems with **distributed memory** (NUMA)
   - HW/SW communication = Send/Receive
3. multiprocessor systems with **virtually shared** (distributed shared) memory (CC-NUMA (Cache-Coherent))
   - HW communication = Send/Receive, SW comm. = Read/Write

Interconnection network view

Multiprocessor systems with shared memory



(a) (b) (c)

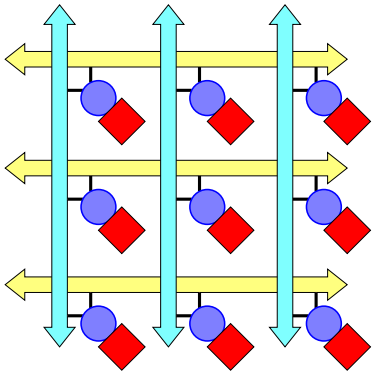
(a) multiprocessor bus

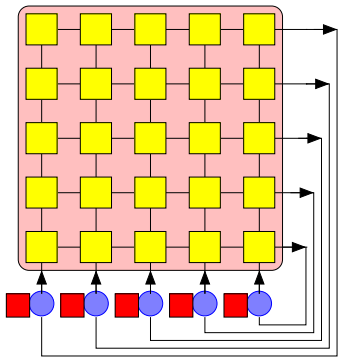
(b) crossbar switch

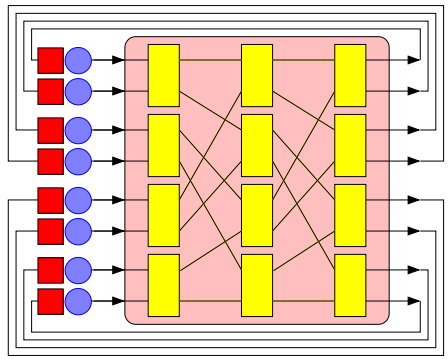
(c) Multistage indirect network

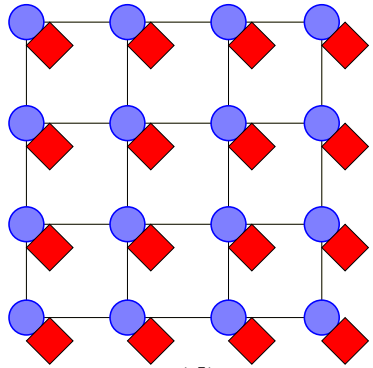
Interconnection network view

Multiprocessor systems with distributed memory (clusters included)


(a)


(b)


(c)


(d)

(a) 2-D mesh of multiprocessor busses

(b) crossbar switch

(c) multistage indirect network

(d) direct interconnect

Parallel computers on the Internet

`www.top500.org`

- As a part of the prestigious conference **Supercomputing**, a group of scientists and vendors in HPC has been publishing for the **last 20 years** twice a year a list of the **500 most powerful computer systems**.
- The performance is measuder using the **LINPACK** benchmark, a SW package to solve a dense system of linear equations. The benchmark allows the user to scale the size of the problem and to optimize the SW in order to achieve the best performance for a given machine.
- Even though `www.top500.org` are not all parallel computers in the world, the list represents very expressively the state-of-the-art in the area of both research and commercial high performance computing massively parallel systems and their development trends.

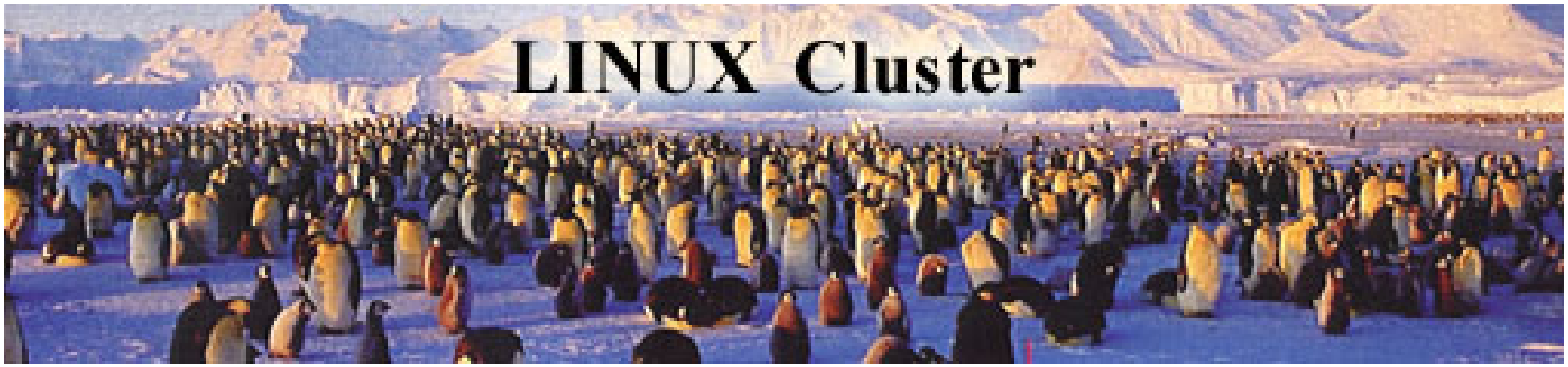
An example of MPP: IBM Blue Gene

Top500.org 10year statistics: Decade of Linux clusters

Attribute	Fall 2000	Fall 2009	Fall 2010
Cluster architecture	28 (5.6%)	417 (83.4%)	414 (82.8%)
MPP architecture	346 (69.2%)	81 (16.2%)	84 (16.8%)
Unix OS	427 (85.4%)	25 (5%)	19 (3.8%)
Linux OS	54 (10.8%)	444 (88.8%)	459 (91.8%)
X86/AMD CPU	6 (1.2%)	438 (87.6%)	455 (91%)

Top500.org 10year statistics: Decade of Linux clusters

Attribute	Fall 2000		Fall 2009		Fall 2010	
Cluster architecture	28	(5.6%)	417	(83.4%)	414	(82.8%)
MPP architecture	346	(69.2%)	81	(16.2%)	84	(16.8%)
Unix OS	427	(85.4%)	25	(5%)	19	(3.8%)
Linux OS	54	(10.8%)	444	(88.8%)	459	(91.8%)
X86/AMD CPU	6	(1.2%)	438	(87.6%)	455	(91%)



Driving forces

- Driving forces behind the growth of Linux clusters:
  - Commodity low-cost hardware (excellent price/performance ratio, 10 times better).
  - Open source software and developer community.
  - Low-cost Ethernet cards, Linux drivers, and networking fabrics.
  - Easy-to-build experimental testbeds, easy proofs of concept, extensive know-how sharing.
- Pioneering projects:
  - **Beowulf** Project at NASA in 1994 made Linux clusters attractive and popular.
  - Dozens of similar do-it-yourself PC clustering projects in the 90s that allowed to build inexpensive virtual parallel computers using commodity PCs.

Why Linux on clusters I

- Dealing with:
  - multiple users,
  - large files,
  - remote access to another machine

  in UNIX-like systems for commodity HW was **NORMAL** and **STANDARD** since the very beginning, in contrast to MS or Apple.
- **Open collaboration**, **sharing**, and **free information circulation** in the developer community: News, packages, distributions, fixes, updates, patches, HowTo's, mailing lists, and even grids.
- Open software packages do not require the payment of **license fees**:
  - Linux OS and the supporting GNU software (compilers, libraries, tools, etc.) are licensed under the GPL license: no additional costs for HPC systems.
  - Ability to scale HW without additional SW costs!!!!!
- BSD legal disputes in 90s.
- Commodity HW: no vendor lock-in.

Why Linux on clusters? II

- Open software could easily be modified to work optimally in an HPC environment.
  - **Drivers** for new hardware could be easily added to existing installations because the source code was available:
    - ★ e.g., early development of Ethernet drivers for Linux kernels.
  - Node **kernel customizations** to get more memory for HPC apps:
    - ★ e.g., computing nodes do not need sound support, laptop power management, enhanced video.
  - Kernel **bypass** optimization for communication ops: communication takes place outside the kernel networking stack and messages are copied between two independent process spaces on different nodes:
    - ★ e.g., drivers provided by Myricom for their Myrinet low-latency interconnect technology.
  - **Process migration** add-ons: OpenMossix.

Why Linux on clusters? III

- Plug-and-play **compatibility** with UNIX.
  - Most large supercomputers used UNIX as their OS.
  - SW packages could easily be ported to Linux clusters.
  - Several open source HPC middleware packages, e.g.,
    - ★ PVM - Parallel Virtual Machine,
    - ★ MPI - Message Passing Interface,

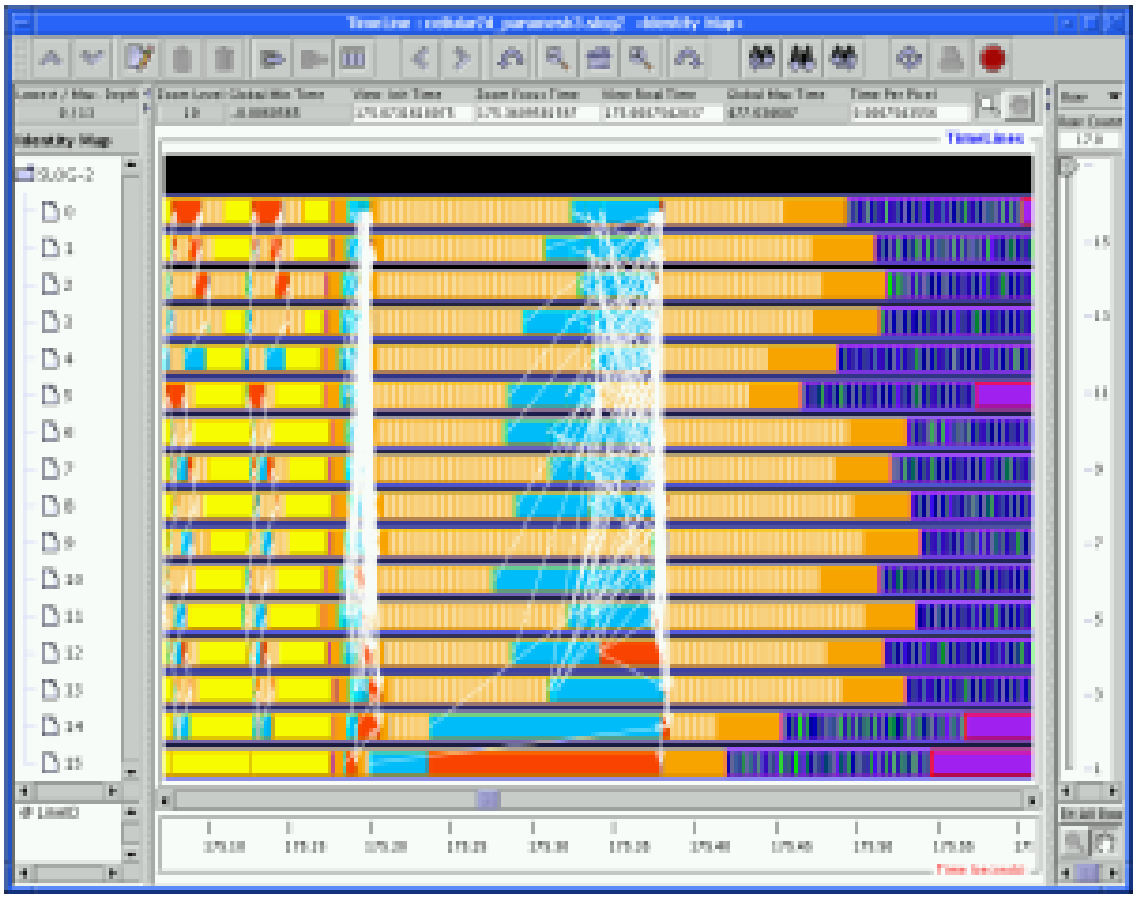
    allowed a standard method of communication among cluster computers.
  - These packages were easily **compiled** under Linux using the same open GNU compilers that were also available on many of the large supercomputers.
  - Typically, moving a message passing application from a large Unix supercomputer to a Linux cluster was often a matter of re-compiling under Linux and took less than a day (plug-recompile-and-play).

A typical Linux cluster software architecture

- **Master-slave** kernel architecture.
- Cluster-**customized** node kernels: no CD, no SCSI, no enhanced video, no sound,...
- **Distributed/parallel shell**, e.g.,
  - `dsh -a reboot`,
  - `dsh -c -g slaves "shutdown -h now"`,
  - `dsh -a df`,
- **Parallel programming environment** (like on MPPs):
  - **Parallel programming** language compilers: `HPF`, `Charm++`, `UPC`.
  - **Communication** middleware libraries: `MPI`, `PVM`.
  - Parallel **job scheduling** (load balancing) tools: `Sun Grid Engine`, `LoadLeveler`, `PBS`, `TORQUE Resource Manager`.
  - Parallel **debugging and node inspection** tools: `padb`.

A typical Linux cluster software architecture

- Parallel **visualization** tools: `JumpShot`



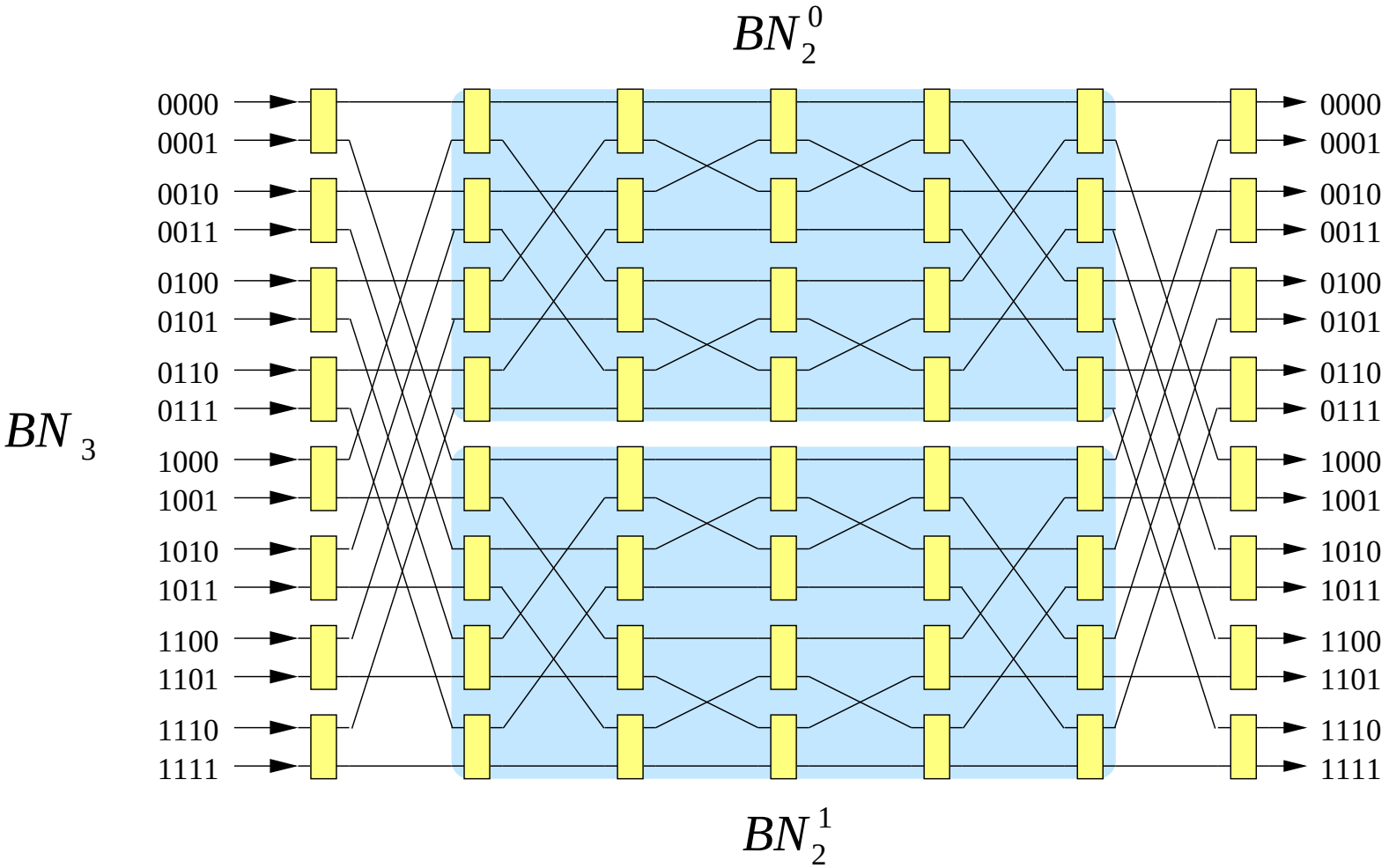
- Cluster **installation** tools: `OSCAR`, `ROCKS`.

Cluster interconnects

- Ethernet (1Gb, 10Gb).
- Myrinet (Myricom).
- Infiniband: switched fabric communications link technology:
  - Point-to-point bidirectional serial links (similar to FC, SATA) connecting
    - ★ processors with processors,
    - ★ processors with high-speed peripherals, such as disks.
  - Multicast operations.
  - Links can be bonded together for additional throughput (up to 200Gb/s).

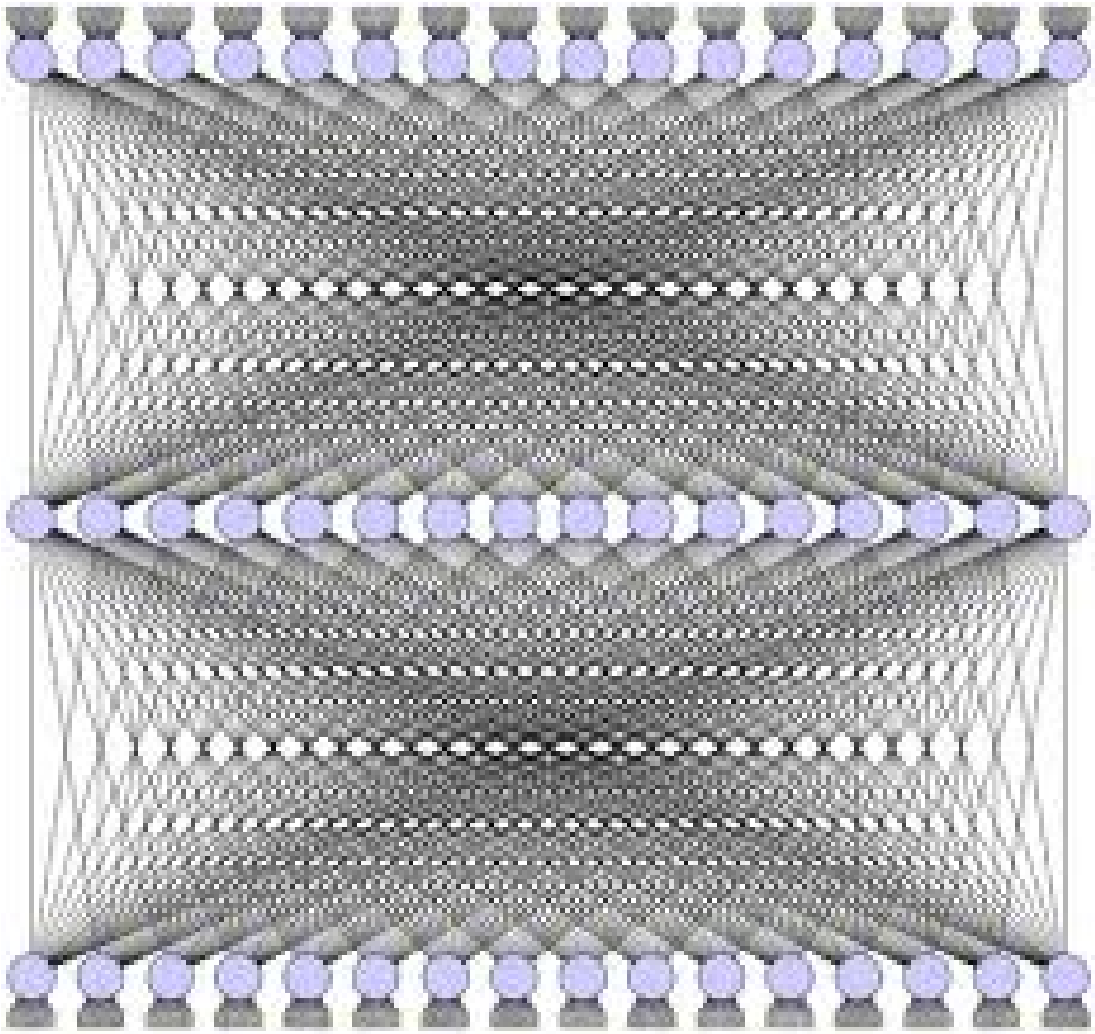
Infiniband topologies

Benes network



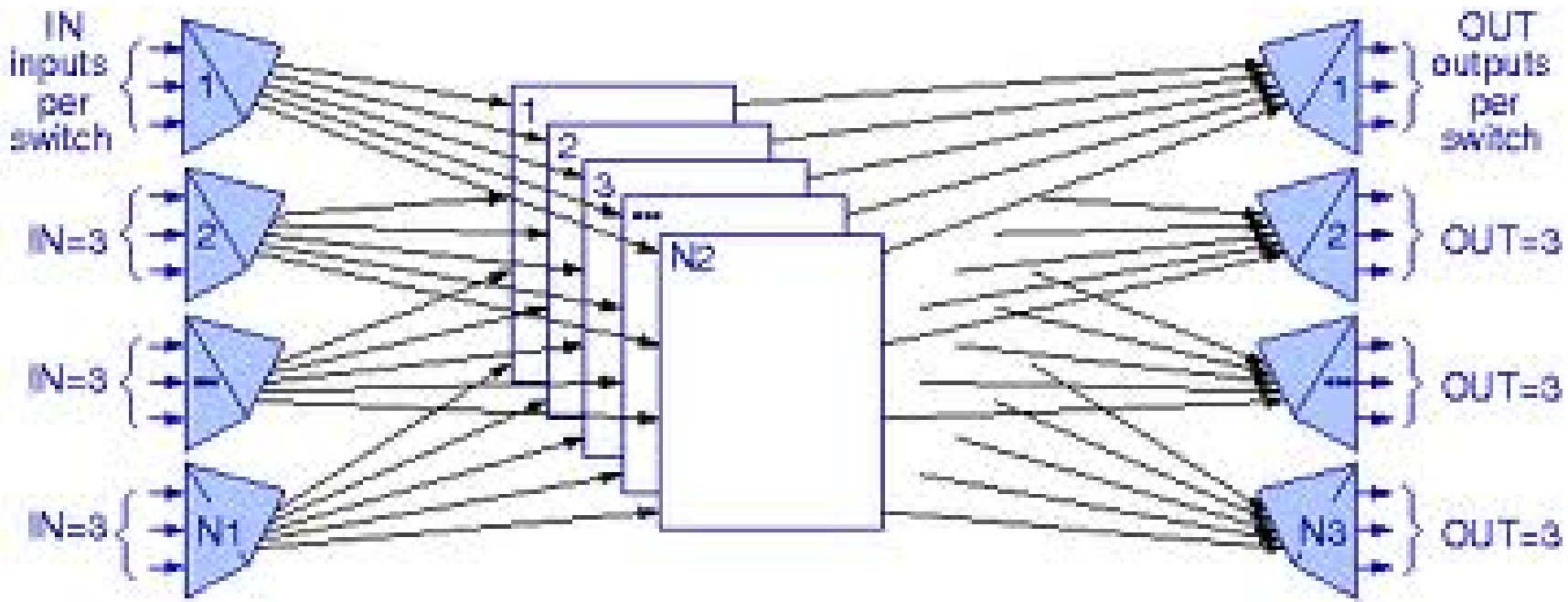
Infiniband topologies

Clos network



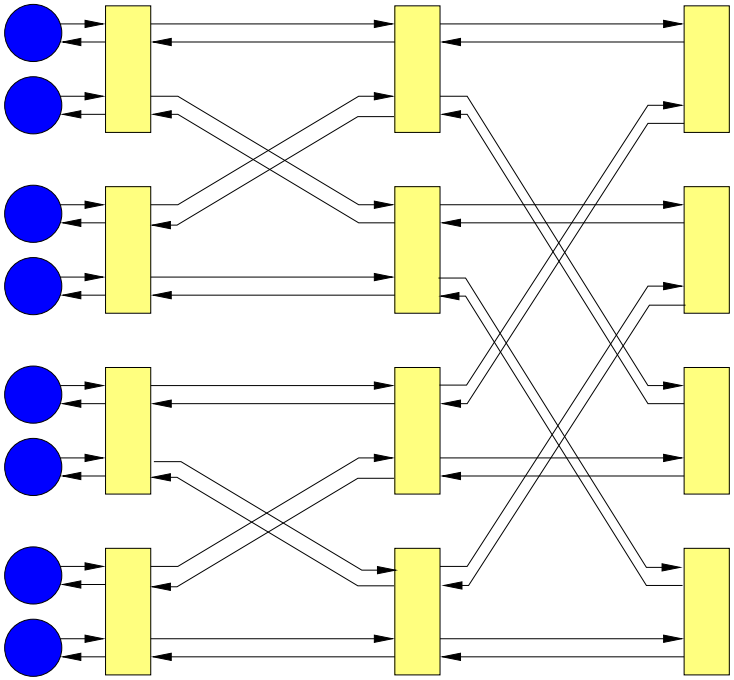
Infiniband topologies

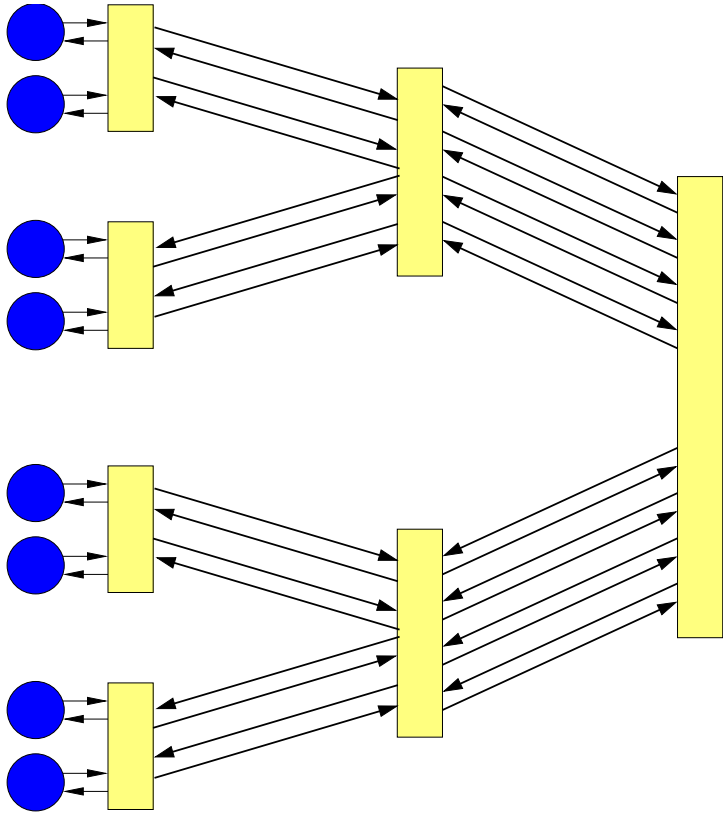
Clos network



Infiniband topologies

Bidirectional butterfly and fat tree





Challenges of large Linux clusters

- Scalability.
- Cluster management.
- Parallel file system.

Biggest computers in the world

- Blue Waters at University of Illinois at Urbana-Champaign: (převzato z `http://www.ncsa.illinois.edu/BlueWaters`)
  - 300 000 cores Power7
  - 10 PFlops (Tera $= 10^{12}$, Peta $= 10^{15}$, Exa $= 10^{18}$)
  - 1 PetaByte of main memory.
  - 5 PetaBytes/s memory bandwidth.
  - 18 PetaBytes of disk storage.
  - 500 PetaBytes of archival storage.
  - A new interconnect.
  - A new parallel file system.
  - A new operating system.
  - A new programming environment.
  - New system administration tools.
  - An integrated toolset to use, analyze, monitor, and control the behavior of Blue Waters.

Biggest computers in the world

- Mira: IBM's BlueGene/Q.
  - 10 PFlops.
  - 750 000 cores.
  - DOE Argonne National Lab.
- Future:
  - ExaFlops systems in 2020.
  - 100s of millions of cores.

Challenges of Blue Waters system I

- Computational and mathematical libraries.
- Cactus: an open-source component-based framework for HPC parallel application development that supports large-scale science and engineering applications and collaborative development teams.
- **Virtualization**.
  - reliability issues,
  - transparent process migration to other processors or drives should the initial components fail or become unavailable,
  - improved performance by hiding some of the latency as work is moved among components and by optimally balancing the workload,
  - overlapping of communication and computation in novel ways.
- New **debugging and performance** tuning tools.

Challenges of Blue Waters system II

- **Integrated Application Development Environment.**
  - Integrated Systems Console: system administration tool providing a single, unified view of the system.
    - ★ aggregation of event reports and performance metrics from across the system.
    - ★ mining performance information using global filters, fault detection, and fault prediction modules.
    - ★ system monitoring,
    - ★ checkpointing of computational runs,
    - ★ file system management including RAID-recovery,
    - ★ interconnect-reconfiguration activities.

Challenges of Blue Waters system III

- **Performance Tools.** Low-overhead agents to collect local data and perform local pre-analyses, including data compression and reduction, to limit system-level impact:
  - automated performance optimization tool,
  - performance data visualization tool (`JumpShot`),
  - instrumentation, event-trace generation, and postprocessing of event traces (`KOJAK`),
  - controlling and tracing performance-monitoring instrumentation (`Tau`).
- **System Simulator**: instruction-level processor simulators and full system network simulators.

Challenges of Blue Waters system IV

- **POWER7** processors:
  - 8 high-performance cores (3.5-4 GHz). Each core has 12 execution units and can perform up to four fused multiply-adds per cycle.
  - Simultaneous multithreading that delivers up to four virtual threads per core.
  - Private L1 (32 KB) instruction and data SRAM cache, private L2 (256 KB) SRAM cache and on-chip L3 (32 MB) eDRAM cache that can be used either as shared cache or separated into dedicated caches for each core.
  - Two dual-channel DDR3 memory controllers, delivering up to 128 GB/sec peak bandwidth (0.5 Byte/flop).
  - Up to 128 GB DDR3 DRAM memory per processor.

Challenges of Blue Waters system V

- Multi-chip modules (MCM):
  - 4 tightly coupled (complete graph) shared-memory POWER7 nodes.
  - 512 GB/sec of aggregate memory bandwidth and 192 GB/sec of bandwidth (0.2 Byte/Flops)
- **Interconnect**: hierarchical combination of switches and InfiniBand links based on a hub/switch chip.
  - 8 MCM are connected to 1 **supernode** with 32 hub/switch chips.
  - topology: fully connected two-tier network
  - hub switch:
    - ★ **Cache-coherence** within the MCM.
    - ★ **Collectives Acceleration Unit** performing multiple independent collective operations occurring simultaneously between various groups of processors
    - ★ hardware support for global address space operations, active messaging, and atomic updates to remote memory locations (support for one-sided communication protocols and ops in X10, UPC, and Co-Array Fortran)

Linux clusters on FIT

- `star.fit.cvut.cz`
  - 3rd generation of Linux cluster at CTU.
  - IBM blade Linux server.
  - Front-end dual-core Xeon processor (`ssh`).
  - $4 \times 8 + 6 \times 12 = 104$ AMD cores.
  - Infiniband + 1Gbit Ethernet.
  - Gentoo 4.4.3-r2 p1.2, gcc, g++, OpenMPI, Sun Grid Engine.
- Semestral projects in Master course **Parallel Algorithms and Systems**.
- Research Master and PhD projects in parallel algorithms.
- Research project **Clondike**: Cluster of Non-dedicated Linux Kernels.