Interconnection Network Architectures for High-Performance Computing

Cyriel Minkenberg
IBM Research — Zurich
HPC interconnection networks

- This lecture relates most closely to Lecture 9 (Apr. 23) “High-Performance Networking I”

- Main objective: Provide you with the basic concepts encountered in HPC networks and how these are applied in practice
Outline

1. Brief introduction to HPC
2. Current HPC landscape (Top 500)
3. Role of the interconnect
4. Topology
5. Routing
6. Flow control & deadlock avoidance
7. Workloads
8. Performance evaluation
9. Outlook
Brief intro to HPC
HPC – a brief history

- Vector machines
- SMPs
- Clusters
- MPPs
HPC use cases

- **Technical and scientific computing, for instance:**
  - Molecular modeling (e.g. CPMD)
  - Quantum mechanics
  - Weather forecasting
  - Climate research
  - Ocean modeling
  - Oil and gas exploration
  - Aerodynamics (airplanes, automobiles), aquadynamics (ships)
  - Nuclear stockpile stewardship
  - Nuclear fusion
Capability vs. capacity systems

A capability system aims to solve a single huge problem as quickly as possible by applying its entire compute power to that one problem
- 1 machine = 1 job
- For Top500 ranking purposes, a machine is typically operated in this mode

A capacity system aims to solve a plurality of problems simultaneously by partitioning its compute power and applying one partition to each job (in parallel)
- 1 machine = several large jobs or many small jobs
- Many large “open” machines at government labs or universities are operated in this mode
- Sophisticated job scheduling engines try to allocate resources to match job requirements in fair way
Strong scaling vs. weak scaling

- **Scaling:** How much faster is a given problem solved by applying more processors?
  - Speedup\( (n) = S(n) = (t_{\text{execution with 1 processor}}) / (t_{\text{execution with n processors}}) \)
  - Strong scaling: Increase the number of processor \( n \) while keeping the problem size constant
  - Weak scaling: Increase the number of processors \( n \) while increasing the problem size (commensurately)
  - Strong scaling is generally (much) harder than weak scaling

- **Amdahl’s Law**
  - Speedup is limited by the serial (non-parallelizable part of the program)
  - \( S(n) \leq 1 / (B + (1-B)/n) \), where \( B \) is the percentage of the program that is serial
  - For most codes, beyond a certain \( n \) performance stops improving
  - …and may get worse, because of increasing communication overheads!
Amdahl’s law

Source: wikipedia
The main difference between HPC and cloud is not primarily in the hardware, it’s in the WORKLOADS!

Different workloads means different requirements means a Cloud-optimized system looks quite different from an HPC-optimized system.
Role of the interconnect
Memory / CPU performance mismatch

Von Neumann's Bottleneck

The interconnect – no longer 2\textsuperscript{nd} class citizen?

- **Conventional wisdom**
  - Computation = expensive
  - Communication = cheap
  \Rightarrow Corollary: Processor is king

- **Then something happened...**
  - Computation
    - Transistors became “free” \Rightarrow more parallelism:
      superscalar, SMT, multi-core, many-core
    - Huge increase in FLOPS/chip
  - Communication
    - Packaging & pins remained expensive
    - Scaling of per-pin bandwidth did not keep pace with CMOS density
  \Rightarrow Consequence
  - Comp/comm cost ratio has changed fundamentally
  - Memory and I/O bandwidth now an even scarcer resource

Source: Intel & ITRS

For the 45nm node, all I/O and power & ground connections of an Pentium1-equivalent chip will have to be served by ONE package pin!
Computation-communication convergence

- **Communication == data movement**
  - It’s all about the data!

- **Convergence of computation and communication in silicon**
  - Integrated L1/L2/L3 caches; memory controller; PCIe controller; multi-chip interconnect (e.g. HT, QPI replacing FSB); network-on-chip
  - Integrated SMP fabric, I/O hub; NIC/CNA/HCA (taking PCIe out of the loop?); special-purpose accelerators

- **Convergence of multiple traffic types on a common physical wire**
  - LAN, SAN, IPC, IO, management
  - L2 encroaching upon L3 and L4 territory
  - Convergence Enhanced Ethernet
Consequences of scarce bandwidth

- **Performance of communication-intensive applications has scaled poorly**
  - because of lower *global* byte/FLOP ratio

- **Yet *mean* utilization is typically very low**
  - because of synchronous nature of many HPC codes; regularly alternating comp/comm phases
  - massive underutilization for computation-intensive applications (e.g. LINPACK)

- **Full-bisection bandwidth networks are no longer cost-effective**

- **Common practice of separate networks for clustering, storage, LAN has become inefficient and expensive**
  - File I/O and IPC can’t share bandwidth
    - I/O-dominated initialization phase could be much faster if it could exploit clustering bandwidth: poor speedup, or even slowdown with more tasks...
**Interconnect becoming a significant cost factor**

- Current interconnect cost percentage increases as cluster size increases
- About one quarter of cost due to interconnect for ~1 PFLOP/s peak system

<table>
<thead>
<tr>
<th></th>
<th>Fat Tree</th>
<th>Torus 1</th>
<th>Torus 2</th>
<th>Torus 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compute</strong></td>
<td>63.3%</td>
<td>72.5%</td>
<td>76.5%</td>
<td>79.7%</td>
</tr>
<tr>
<td><strong>Adapters + cable</strong></td>
<td>10.4%</td>
<td>11.9%</td>
<td>12.6%</td>
<td>13.1%</td>
</tr>
<tr>
<td><strong>Switches + cables</strong></td>
<td>26.3%</td>
<td>15.6%</td>
<td>10.9%</td>
<td>7.2%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Current HPC landscape
Top 500

- **List of 500 highest-performing supercomputers**
  - Published twice a year (June @ ISC, November @ SC)
  - Ranked by maximum achieved GFLOPS for a specific benchmark
  - See [www.top500.org](http://www.top500.org)
  - “Highest-performing” is measured by sustained Rmax for the LINPACK (HPL) benchmark
  - LINPACK solves a large dense system of linear equations
  - Note that not all supercomputers are listed (not every supercomputer owner care to see their system appear on this list...)

If your problem is not represented by a dense system of linear equations, a system’s LINPACK rating is not necessarily a good indicator of the performance you can expect!
Caveat emptor!

1) HPL is embarrassingly parallel
   - Very little load on the network
   - Highly debatable whether it is a good yardstick at all
   - Yet the entire HPC community remains ensnared by its spell

2) FLOPS rating says NOTHING about usefulness or efficiency of computation
   - The underlying algorithm is what matters!
   - Consider two algorithms solving the same problem:
     - Alg A solves it in \( t \) time with \( f \) FLOPS
     - Alg B solves it in \( 10^*t \) time with \( 10^*f \) FLOPS
     - A is 100 times more efficient than B, yet has one-tenth of the FLOPS rating
Top 500 since 1993
Top 500 since 1993

- Gigaflop
- Teraflop
- Petaflop
- Exaflop

~2019

GFLOPS

year


© 2013 IBM Corporation
<table>
<thead>
<tr>
<th>#</th>
<th>Rmax (Pflops)</th>
<th>Name</th>
<th>Computer design</th>
<th>Vendor</th>
<th>Site Country, year</th>
<th>Operating system</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17.590/27.113</td>
<td><em>Titan</em></td>
<td>Cray XK7 + Opteron 6274 + Tesla K20X, Custom</td>
<td>Cray</td>
<td>Oak Ridge National Laboratory (ORNL) in Tennessee, United States, 2012</td>
<td>Linux (CLE, SLES based)</td>
</tr>
<tr>
<td>2</td>
<td>16.325/20.133</td>
<td><em>Sequoia</em></td>
<td>Blue Gene/Q PowerPC A2, Custom</td>
<td>IBM</td>
<td>Lawrence Livermore National Laboratory United States, 2011</td>
<td>Linux (RHEL and CNK)</td>
</tr>
<tr>
<td>3</td>
<td>10.510/11.280</td>
<td><em>K computer</em></td>
<td>RIKEN SPARC64 VIIIfx, Tofu</td>
<td>Fujitsu</td>
<td>RIKEN Japan, 2011</td>
<td>Linux</td>
</tr>
<tr>
<td>4</td>
<td>8.162/10.066</td>
<td><em>Mira</em></td>
<td>Blue Gene/Q PowerPC A2, Custom</td>
<td>IBM</td>
<td>Argonne National Laboratory United States, 2012</td>
<td>Linux (RHEL and CNK)</td>
</tr>
<tr>
<td>5</td>
<td>4.141/5.033</td>
<td><em>JUQUEEN</em></td>
<td>Blue Gene/Q PowerPC A2, Custom</td>
<td>IBM</td>
<td>Forschungszentrum Jülich Germany, 2012</td>
<td>Linux (RHEL and CNK)</td>
</tr>
<tr>
<td>6</td>
<td>2.897/3.185</td>
<td><em>SuperMUC</em></td>
<td>iDataPlex DX360M4 Xeon E5–2680, Infiniband</td>
<td>IBM</td>
<td>Leibniz-Rechenzentrum Germany, 2012</td>
<td>Linux</td>
</tr>
<tr>
<td>7</td>
<td>2.660/3.959</td>
<td><em>Stampede</em></td>
<td>PowerEdge C8220 Xeon E5–2680, Infiniband</td>
<td>Dell</td>
<td>Texas Advanced Computing Center United States, 2012</td>
<td>Linux</td>
</tr>
<tr>
<td>8</td>
<td>2.566/4.701</td>
<td><em>Tianhe-1A</em></td>
<td>NUDT YH Cluster Xeon 5670 + Tesla 2050, Arch6</td>
<td>NUDT</td>
<td>National Supercomputing Center of Tianjin China, 2010</td>
<td>Linux</td>
</tr>
<tr>
<td>9</td>
<td>1.725/2.097</td>
<td><em>Fermi</em></td>
<td>Blue Gene/Q PowerPC A2, Custom</td>
<td>IBM</td>
<td>CINECA Italy, 2012</td>
<td>Linux (RHEL and CNK)</td>
</tr>
<tr>
<td>10</td>
<td>1.515/1.944</td>
<td><em>DARPA Trial Subset</em></td>
<td>Power 775 POWER7, Custom</td>
<td>IBM</td>
<td>IBM Development Engineering United States, 2012</td>
<td>Linux (RHEL)</td>
</tr>
</tbody>
</table>
System/performance share by vendor

Vendors System Share

Vendors Performance Share
System/performance share by processor

Processor Generation System Share

Processor Generation Performance Share
HPC interconnect landscape, Nov. 2012

Source: www.top500.org
Top 500 Interconnects

- List of June 2010

- Dominated by Ethernet & Infiniband
  - Ethernet by volume (49%)
  - Infiniband by PFLOPs (49%)

- Proprietary still plays a significant role
  - 9% system share, 26% performance share
  - High-end HPC vendors

- Myrinet, Quadrics dwindling

- 10 GigE: two installations listed

Source: www.top500.org
Interconnects by system share

Ethernet 37.8%
Infiniband 44.8%
Proprietary 17.4%
Other 17.4%

Nov. 2012
Based on Nov. 2012 Top 500 data

<table>
<thead>
<tr>
<th>Interconnect</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infiniband</td>
<td>224</td>
</tr>
<tr>
<td>Ethernet</td>
<td>189</td>
</tr>
<tr>
<td>Myrinet</td>
<td>13</td>
</tr>
<tr>
<td>NEC SX-9</td>
<td>107</td>
</tr>
<tr>
<td>IBM BlueGene</td>
<td>6.4%</td>
</tr>
<tr>
<td>Fujitsu K</td>
<td>0.6%</td>
</tr>
<tr>
<td>IBM p775</td>
<td>2.6%</td>
</tr>
<tr>
<td>Tianhe-1</td>
<td>0.6%</td>
</tr>
<tr>
<td>Cray</td>
<td>6.2%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ethernet</th>
<th>10GE</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1GE</td>
<td>159</td>
</tr>
</tbody>
</table>

Based on Nov. 2012 Top 500 data
Interconnects by performance share

Based on Nov. 2012 Top 500 data

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>49%</td>
<td>45%</td>
<td>38%</td>
<td>24%</td>
<td>19%</td>
<td>13%</td>
</tr>
<tr>
<td>Infiniband</td>
<td>41%</td>
<td>42%</td>
<td>45%</td>
<td>49%</td>
<td>39%</td>
<td>32%</td>
</tr>
<tr>
<td>Proprietary</td>
<td>10%</td>
<td>13%</td>
<td>17%</td>
<td>27%</td>
<td>42%</td>
<td>55%</td>
</tr>
</tbody>
</table>

Nov. 2012
Interconnects by topology

By system share
- Tree: 84.0%
- 3D Torus: 6.8%
- 5D Torus: 5.8%
- Dragonfly: 3.4%

By performance share
- Tree: 47.6%
- 3D Torus: 17.8%
- 5D Torus: 31.1%
- Dragonfly: 3.6%
Interconnection network basics
Interconnection network basics

- **Networks connect processing nodes, memories, I/O devices, storage devices**
  - System comprises network nodes (aka switches, routers) and compute nodes

- **Networks permeate the system at all levels**
  - Memory bus
  - On-chip network to connect cores to caches and memory controllers
  - PCIe bus to connect I/O devices, storage, accelerators
  - Storage area network to connect to shared disk arrays
  - Clustering network for inter-process communication
  - Local area network for management and “outside” connectivity

- **We’ll focus mostly on clustering networks here**
  - Don’t equate “clustering network” with “cluster” here
  - These networks tie many nodes together to create one large computer
  - Nodes are usually identical and include their own cores, caches, memory, I/O
Interconnection network basics

- By now, buses have mostly been replaced by networks, off-chip as well as on-chip

- System performance is increasingly limited by communication instead of computation

- Hence, network has become a key factor determining system performance

- Therefore, we should choose its design wisely

- We review several options here (but not all by far)
Main design aspects

- **Topology**
  - Rules determining how compute nodes and network nodes are connected
  - Unlike LAN or data center networks, HPC topologies are highly *regular*
  - Interconnection patterns can be described by simple algebraic expressions
  - All connections are full duplex

- **Routing**
  - Rules determining how to get from node A to node B
  - Because topologies are regular & known, routing algorithms can be designed a priori
  - Source- vs. table-based; direct vs. indirect; static vs. dynamic; oblivious vs. adaptive

- **Flow control**
  - Rules governing *link traversal*
  - Deadlock avoidance
Interconnect classification

- **Direct vs. indirect**
  - Direct network: Each network node attaches to at least one compute node.
  - Indirect network: Compute nodes are attached at the edge of the network only; clear boundary between compute nodes & network nodes; many routers only connect to other routers.

- **Discrete vs. integrated**
  - Discrete: network nodes are physically separate from compute nodes
  - Integrated: network nodes are integrated with compute nodes

- **Standard vs. proprietary**
  - Standard (“open”): Network technology is compliant with a specification ratified by standards body (e.g., IEEE)
  - Proprietary (“closed”): Network technology is owned & manufactured by one specific vendor
Topologies

- **Trees [indirect]**
  - Fat tree
  - $k$-ary $n$-tree
  - Extended generalized fat tree (XGFT)

- **Mesh & torus [direct]**
  - $k$-ary $n$-mesh
  - $k$-ary $n$-cube

- **Dragonfly [direct]**

- **HyperX/Hamming Graph [direct]**
The origin of the term “fat tree”

- First described by C. Leiserson, 1985.
- A fat tree of height $m$ and arity $k$ has $k^m$ end nodes and $k^{m-j}$ switches in level $j$ ($1 \leq j \leq m$).
- Each switch has at level $j$ has $k$ “down” ports with capacity $C_{j-1} = k^{j-1}C$ and 1 “up” port with capacity $C_j = k^jC$.

- A fat tree’s main characteristic: Constant bisectional bandwidth
- Achieved by increasingly “fatter” links in subsequent levels
- Top level can be eliminated if further expansion not desired (“double sized”)
Transmogrification into $k$-ary $m$-tree

- Switch capacity doubles with each level
  - Constant radix, but increasing link speed
  - Not feasible in practice for large networks
  - Routing is trivial (single path)

- Construct fat tree from fixed-capacity switches
  - Constant radix, constant link speed
  - Requires specific interconnection pattern and routing rules (multi-path)
  - Top-level switches radix is $k/2$ (without expansion)
$k$-ary $n$-tree network

- $N = k^n$ end nodes
- $nk^{(n-1)}$ switches arranged in $n$ stages
- $(nk^n)/2$ inter-switch links

- Diameter = $2n-1$ switch hops
- Bisection = $B_{\text{bis}} = \frac{1}{2}k^n$ bidir links
- Relative bisection = $B_{\text{bis}}/(N/2) = 1
Extended Generalized Fat Tree

- **XGFT** \(( h ; m_1, \ldots, m_h ; w_1, \ldots, w_h )\)
- \( h = \text{height} \)
  - number of levels - 1
  - levels are numbered 0 through \( h \)
  - level 0: compute nodes
  - levels 1 \( \ldots h \): switch nodes
- \( m_i = \text{number of children per node at level } i, \ 0 < i \leq h \)
- \( w_i = \text{number of parents per node at level } i-1, \ 0 < i \leq h \)

- Number of level 0 nodes = \( \prod_i m_i \)
- Number of level \( h \) nodes = \( \prod_i w_i \)

\[ \text{XGFT ( 3 ; 3, 2, 2 ; 2, 2 ,3 )} \]

Number of children per level

Number of parents per level
Fat tree example: Roadrunner

- **First system to break the petaflop barrier**
  - Operational in 2008, completed in 2009
  - National Nuclear Security Administration, Los Alamos National Laboratory
  - Used to modeling the decay of the U.S. nuclear arsenal

- **First hybrid supercomputer**
  - 12,960 IBM PowerXCell 8i CPUs
  - 6,480 AMD Opteron dual-core processors
  - InfiniBand interconnect

- **Numbers**
  - Power: 2.35 MW
  - Space: 296 racks, 560 m²
  - Memory: 103.6 TiB
  - Storage: 1,000,000 TiB
  - Speed: 1.042 petaflops
  - Cost: US$100 million
Cell BE commercial application

Source: Sony
Roadrunner fat tree network

- **“TriBlade” hybrid compute node**
  - One IBM LS21 AMD Opteron blade: 2 dual-core AMD Opteron, 32 GB RAM
  - Two IBM QS22 Cell BE blades: 2x2 3.2 GHz IBM PowerXCell 8i processors, 32 GB RAM, 460 GFLOPS (SP)

- **One Connected Unit (CU) = 180 TriBlades**
- **Full system = 18 CUs = 3,240 compute nodes = 6,480 Opterons + 12,960 Cells**

- **InfiniBand 4x DDR network (2 GB/s/port)**
  - two-stage fat tree
    - Stage 1: 18 switches: 180 ports down, 8x12 ports up
    - Stage 2: 8 switches, 18x12 ports down

Diagrams and equations are used to illustrate the network topology and specifications.
Ashes to ashes…

Source: arstechnica.com
Mesh and torus

2D mesh: 4-ary 2-mesh

Ring: 8-ary 1-cube

2D torus: 4-ary 2-cube

Hypercube: 2-ary 4-cube
Blue Gene/Q packaging

1. Chip
   Single chip
   16+2 cores

2. Module
   One single chip module,
   16 GB DDR3 memory

3. Compute Card
   32 Compute Cards,
   Optical Modules, Link Chips,
   2x2x2x2x2 5D Torus

4. Node Card
   16 Node Cards

5a. Midplane
    16 Node Cards
    4x4x4x4x2

5b. I/O Drawer
    8 I/O Cards
    8 PCIe Gen2 slots

6. Rack
    2 Midplanes
    1, 2 or 4 I/O Drawers
    4x4x4x8x2

7. System
    96 racks, 20 PF/s

Sustained single node perf: 10x BG/P, 20x BG/L
MF/Watt: > 2 GFLOPS per Watt (6x BG/P, 10x BG/L)

Full system: 5D Torus 12x16x16x16x2
BlueGene/Q

9 link modules, each having 1 link chip + 4 optical modules; total 384 fibers (256 torus + 64 ECC + 64 spare)

32 compute cards (nodes) forming a 2x2x2x2x2 torus

1 optical module = 12 TX + 12 RX fibers @ 10 Gb/s with 8b/10b coding; eff. 1 GB/s per fiber
8 fibers for 4 external torus dimensions (2 GB/s/port)
2 fibers for ECC (1 per group of 4 fibers)
2 spares
1 link module = 4 optical modules
BG/Q Sequoia, Top500 #2

- 1,572,864 cores
- 17 PF sustained performance (20 PF peak)
- 8 MW power consumption

Source: LLNL
Full mesh aka “complete graph”

- Fully connected or Complete Graph $K_n$: all nodes directly connected to all other nodes by bidirectional dedicated links
  - $R$ routers and $R(R-1)/2$ bidirectional links
  - Bisection bandwidth = $R^2/4$ if $R$ even or $(R^2-1)/4$ if $R$ odd
  - Assuming $n = R$ and random traffic, $R^2/4$ packets in each direction for a total of $R^2/2$ bisection traffic
Dragonfly topology

- **Hierarchical network**
  - Full mesh of routers at 1\textsuperscript{st} level
  - Full mesh of groups at 2\textsuperscript{nd} level

- **DF(p, a, h)**
  - $p$: \#nodes-router
  - $a$: \#routers/group
  - $h$: \#remote links/router
  - \#groups $ah+1$
  - \#routers $a(ah+1)$
  - \#nodes $pa(ah+1)$
  - \#links $a(ah+1)(a-1+h)/2$
  - Diameter 3
  - Bisection $\sim((ah+1)^2-1)/4$

\begin{align*}
  \text{DF}(1, 4, 1)
\end{align*}
HPCS PERCS topology (IBM p775)

- 8 cores/processor
- $p = 4$ processors/router
- $a = 32$ routers/group (group = “supernode”)
- $h = 16$ remote links/router
- Total #routers = $32 \times (32 \times 16 + 1) = 16,416 \ [16,384]$
- Total #processors = $4 \times 32 \times (32 \times 16 + 1) = 65,664 \ [65,536]$
- Total #cores = 524,288
HPCS PERCS interconnect (IBM p775)

Source: W. Denzel
IBM Power 775 (aka P7-IH, PERCS, TSFKABW)

- **Packaging**
  - 1 P7 = 8 cores @ 3.84 GHz
  - 1 quad = 4 P7 = 32 cores
  - 1 2U drawer = 8 quads = 32 P7 = 256 cores
  - 1 supernode = 4 drawers = 32 quads = 128 P7 = 1’024 cores
  - 1 rack = 3 supernodes = 3’072 cores
  - 1 full-scale PERCS system = ~171 racks = 512 supernodes = 524’288 cores

- **Drawer is smallest unit**
  - 8 MCMs with 4 P7 chips each
  - 8 hub modules with optical modules & fiber connectors
  - Memory DIMMs (up to 2 TB)
  - Power, cooling

- **Integrated network**
  - MCM including 4 P7 chips plus hub chip
  - No external adapters or switches

- **96 TFLOPs per rack**
  - 24 TB memory
  - 230 TB storage
  - Watercooled
p775 hub chip & module

- **Hub chip for POWER7-based HPC machines**
  - Coherency bus for 4 POWER7 chips (“quad”)
  - Integrated adapter: Host Fabric Interface (HFI)
  - Integrated memory controllers
  - Integrated Collective Acceleration Unit (CAU)
  - Integrated PCIe controllers
  - Integrated switch/router (ISR)
  - Hardware acceleration (CAU, global shared memory, RDMA)

- **56-port integrated switch**
  - 7 electrical intra-drawer links (LL) to all other hubs within same drawer @ (24+24) GB/s per link
  - 24 optical inter-drawer links (LR) to all other hubs within other drawers of same supernode @ (5+5) GB/s per link
  - 16 optical inter-supernode links to hubs in other supernodes @ (10+10) GB/s per link

- **336 × 10 Gb/s optical links in 56 modules**
  - 28 Tx/Rx pairs × 12 fibers/module = 336 bidir links
  - Avago MicroPOD™ 12× 10 Gb/s with PRIZM™ Light-Turn® optical connector
  - 8b/10b coding ➔ (2 × 336 × 10 × 8/10)/8 = 672 GB/s of aggregate optical link bandwidth
Hamming graph aka HyperX

- Cartesian product of complete graphs
- HG(p, g, h)
  - Hierarchical topology with h levels
  - Each level comprises g groups of switches
  - p end nodes are connected to each switch
  - Recursive construction
    - At level 0, each group comprises one switch
    - At level i+1, replicate the network of level i g times and connect each switch of each group directly to each switch at the same relative position in the other g-1 groups

- Each group at a given level is directly connected to all other groups at that level
  - Complete graph ("full mesh") among groups at each level
  - With \( g^{i-1} \) links in between each pair of groups at level \( i \), \( 1 \leq i \leq h \)

- Total number of switches = \( g^h \)
- Total number of end nodes = \( pg^h \)
- Each switch has radix \( r = n + h(g - 1) \)
- Total number of internal links = \( g^h h (g - 1) / 2 \)
- Diameter = \( h+1 \)
Hamming graphs

HG(3, 4, 1) [K₄]

HG(p, 4, 3) [K₄ x K₄ x K₄]

4 links

16 links
Cray XC30

- Aries interconnect: combination of Hamming graph & dragonfly
- **Router radix = 48 ports**
  - node: 8 (2 ports per node)
  - rank-1: 15
  - rank 2: 15 (3 ports per connection)
  - rank 3: 10 (global)
- **Rank 1: complete graph of 16 routers**
  - 16 Aries, 64 nodes
- **Rank 2: group of 6 rank-1 graphs; Hamming graph $K_{16} \times K_6$**
  - 96 Aries, 384 nodes
- **Up to 96x10+1 = 961 groups; in practice limited to 241 groups**
  - 23,136 Aries, 92,544 nodes

Source: Cray
Routing
Routing

- How do we get from source node A to destination node B?

- HPC topologies are regular and static
  - Optimal routing algorithms can be designed a priori
  - No need for flooding-based learning methods
  - Many of such algorithms could be implemented by algebraic manipulations on current & destination address

- Examples
  - Fat tree: destination[source]-modulo-radix
  - Mesh + torus: dimension order routing
  - Dragonfly: direct (local-global-local)
  - Hamming graph: level order routing

- Variations/enhancements
  - Source- (end node determines route) vs. hop-by-hop (routing decision at each router)
  - Direct (shortest path) vs. indirect (detour)
  - Static (path for a given src/dst pair is fixed) vs. dynamic (multiple paths may be used)
  - Oblivious (path alternative chosen obliviously) vs. adaptive (path alternative chosen based on network state)

- This topic alone would be good for several hours
Flow control
Message, packets & flits

- **A message is divided into multiple packets**
  - Each packet has header information that allows the receiver to re-construct the original message
  - Packets of same message may follow different paths

- **A packet may further be divided into flow control units (flits)**
  - Only head flits contains routing information
  - Flits of same packet must follow same path
  - Flit = finest granularity at which data flow is controlled (atomic unit of flow control)
  - Flits from different packets may (usually) not be interleaved ➔ **wormhole routing**
  - If flit == packet size: **virtual cut-through** (VCT)

- Segmentation into packets & flits allows the use of a large packets (low header overhead) and fine-grained resource allocation on a per-flit basis
Flow control

- **HPC networks employ link-level flow control**
  - Avoid packet drops due to buffer overflows; retransmission latency penalty is considered too expensive
  - Enable buffer resource partitioning for deadlock prevention & service differentiation
  - This is a crucial difference from TCP/IP networks

- **Routing of a message requires allocation of various resources**
  - Channel (link bandwidth)
  - Buffer
  - Control state (virtual channel)

- **Non-packet-switched flow control**
  - Bufferless switching: drop & retransmit
  - Circuit switching: set up end-to-end circuit before sending data
Buffered flow control

- Buffers enable contention resolution in packet-switched networks
  - Temporarily store packets contending for same channel
  - Decouple resource allocation for subsequent channels – buffers are cheaper bandwidth

- Packet-buffer flow control: channels and buffers are allocated per packet
Flit-buffer flow control (Wormhole)

- Wormhole flow control is virtual cut-through at the flit instead of packet level

- A head flit must acquire three resources at the next switch before being forwarded:
  - Channel control state (virtual channel)
  - Buffer for one flit
  - Channel bandwidth for one flit
  - Body and tail flits use the same VC as the head, but must still acquire a buffer and physical channel individually

- Can operate with much smaller buffers than VCT
- May suffer from channel blocking: channel goes idle when packet is blocked in the middle of traversing the channel
Virtual channel flow control

- **Multiple virtual channels per physical channel**
  - Partition each physical buffer into multiple logical buffer spaces, one per VC
  - Flow control each logical buffer independently
  - Flits are queued per VC
  - VCs may be serviced independently
  - Incoming VC may be different from outgoing VC
  - Each VC keeps track of the output port + output VC of head flit
  - Head flit must allocate the same three resources at each switch before being forwarded

- **Advantages**
  - Alleviate channel blocking
  - Enable deadlock avoidance
  - Enable service differentiation
Example

Message A is blocked in the middle switch
Message B needs to traverse the middle switch
Deadlock avoidance
Routing deadlocks

- Most topologies (other than trees) have loops
  - Loops can be dangerous!
  - Packets routed along a loop would never get to their destination
  - Broadcast packets in a loop can cause devastating broadcast storms
  - For this reason, Ethernet network use the Spanning Tree Protocol: Overlay a loop-free logical topology on a loopy physical one
  - In regular topologies, routing loops can be avoided by a sound routing algorithm

- However, because HPC networks also use link-level flow control, loops can induce routing deadlocks
  - In deadlock, no packet can make forward progress anymore
  - Formally, a routing deadlock can occur when there is a cyclic dependency in the channel dependency graph
  - Such dependencies can be broken
    - By restricting routing
    - By restricting injection (e.g., bubble routing)
    - By means of virtual channels
Deadlock-free routing in tori

- Does dimension-ordered routing avoid deadlocks in a torus?

- No, because there is a cycle within each dimension

- Introduce a second VC to break the cyclic dependency: “Dateline” routing
  - Each ring has a dateline link
  - Each packet starts on VC1
  - When a packet crosses the dateline, it moves to the VC2
  - Visit dimension in fixed order
  - When moving to another dimension, go back to VC1
Recall the \( k \)-ary \( n \)-tree network

Are loops a problem in \( k \)-ary \( n \)-trees?

- Not really: Shortest-path routing never performs a down-up turn
  - Always \( n \) hops up followed by \( n \) hops down
  - No cyclic dependencies in channel dependency graph
A $k$-ary $n$-tree network with more nodes

Why aren’t there any nodes connected to these switches?
**b-way k-ary n-direct-tree**

Level 4

- 2-way binary 4-tree

Level 3

- 2-way binary 4-tree

Level 2

- 2-way binary 4-tree

Level 1

- 2-way binary 4-tree

- **bnk^{n-1}** end nodes
- **nk^{n-1}** switches arranged in n stages
- **nk^{n}/2** inter-switch links
Deadlock prevention in direct trees

- \[ \text{NCA}(S(3;000), S(3,101)) = S(4; x00) \]
- \[ \text{NCD}(S(3;000), S(3,101)) = S(1; 1yz) \]
- 8 alternative shortest paths
- Route on VC0 (red) until NCD, then switch to VC1 (blue)
Workloads
Workload communication patterns

Understanding workloads characteristics is crucial to designing a cost-performance-optimal system.

For the interconnect designer, this implies understanding temporal, spatial, and causal patterns of inter-node communications.
Let’s look at bytes per FLOP

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>Country</th>
<th>System</th>
<th>Cores</th>
<th>Rmax (TF/s)</th>
<th>Rpeak (TF/s)</th>
<th>Eff. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DOE/NNSA/LLNL</td>
<td>United States</td>
<td>BlueGene/Q</td>
<td>1572864</td>
<td>16325</td>
<td>20133</td>
<td>81</td>
</tr>
<tr>
<td>3</td>
<td>DOE/SC/ANL</td>
<td>United States</td>
<td>BlueGene/Q</td>
<td>786432</td>
<td>8162.4</td>
<td>10066</td>
<td>81</td>
</tr>
<tr>
<td>7</td>
<td>CINECA</td>
<td>Italy</td>
<td>BlueGene/Q</td>
<td>163840</td>
<td>1725.5</td>
<td>2097.2</td>
<td>82</td>
</tr>
<tr>
<td>8</td>
<td>Forschungszentrum Juelich</td>
<td>Germany</td>
<td>BlueGene/Q</td>
<td>131072</td>
<td>1380.4</td>
<td>1677.7</td>
<td>82</td>
</tr>
<tr>
<td>13</td>
<td>Daresbury Laboratory</td>
<td>United Kingdom</td>
<td>BlueGene/Q</td>
<td>114688</td>
<td>1207.8</td>
<td>1468</td>
<td>82</td>
</tr>
<tr>
<td>20</td>
<td>University of Edinburgh</td>
<td>United Kingdom</td>
<td>BlueGene/Q</td>
<td>98304</td>
<td>1035.3</td>
<td>1258.3</td>
<td>82</td>
</tr>
<tr>
<td>29</td>
<td>EDF R&amp;D</td>
<td>France</td>
<td>BlueGene/Q</td>
<td>65536</td>
<td>690.2</td>
<td>838.9</td>
<td>82</td>
</tr>
<tr>
<td>36</td>
<td>High Energy Accelerator Research Organization</td>
<td>Japan</td>
<td>BlueGene/Q</td>
<td>49152</td>
<td>517.6</td>
<td>629.1</td>
<td>82</td>
</tr>
<tr>
<td>48</td>
<td>DOE/NNSA/LLNL</td>
<td>United States</td>
<td>BlueGene/Q</td>
<td>32768</td>
<td>345.1</td>
<td>419.4</td>
<td>82</td>
</tr>
<tr>
<td>100</td>
<td>DOE/SC/ANL</td>
<td>United States</td>
<td>BlueGene/Q</td>
<td>16384</td>
<td>172.7</td>
<td>209.7</td>
<td>82</td>
</tr>
<tr>
<td>252</td>
<td>DOE/NNSA/LLNL</td>
<td>United States</td>
<td>BlueGene/Q</td>
<td>8192</td>
<td>86.3</td>
<td>104.9</td>
<td>82</td>
</tr>
</tbody>
</table>

BG/Q achieves higher parallel efficiency at scale despite much lower B/F ratio! (and much less memory per core, and higher-diameter network...) Therefore, for the LINPACK workload:

- If problem size is large enough, then network bandwidth does not matter
- Corollary: If bandwidth does matter, then the problem size is too small!

Source: top500.org

© 2013 IBM Corporation
LINPACK performance: IB vs GigE

- TOP500 systems listed according to their efficiency
- InfiniBand is the key element responsible for the highest system efficiency
- Up to 96% efficiency
High-Performance LINPACK MareNostrum, 64 MPI tasks
N = 8192, BS = 128, P=Q=8

Problem too small:
60% computation
40% communication
The communication matrix

- Each task communicates with a limited subset of other tasks
- Very regular structure
- This is typical for many scientific codes
- If one were to design an interconnect for a specific code, one could exploit this to great effect!
- However
  - Depends on task placement
  - Patterns vary across codes
  - For some codes the matrix is even
  - Says nothing about comm/comp ratio!

Each graph shows data volume per task pair (x,y)
Performance of communication-intensive codes depends strongly on interconnect design: topology, routing, etc.
All-to-all exchange pattern

- Each color marks the physical reception of the payload of a message
- Gaps between messages are exchanges of rendez-vous protocol messages
- Dependencies cause delays to accumulate and propagate
- From 1 microsecond imbalance to 1 millisecond loss!
Workload predictability

- **HPC systems are usually “closed”**
  - Not directly exposed to the outside world, only local research community
  - Jobs controlled by a scheduler

- **Scientific codes**
  - Comm patterns of many scientific code are quite static and regular
  - With some codes, comm depends on the input/problem (e.g., adaptive mesh refinement)

- **Capacity vs capability**
  - Need to control/be aware of task placement

- **Certain classes of data centers are “open”**
  - North-south vs east-west traffic
  - Much less predictable workloads
  - Virtualization will affect this negatively – but is also an opportunity
Performance evaluation
Modeling of large-scale HPC systems

- **Dichotomy in performance evaluation in computer architecture**
  - Node architecture
    - Execution-driven, cycle-accurate CPU (ISA), cache and memory models
    - Highly accurate
    - Too much detail to scale to large node counts
  - Network architecture
    - Highly accurate (flit level) at network level
    - Several orders of magnitude fewer network nodes than CPU cores
    - Network node much simpler than CPU core
    - But usually driven by either purely random or purely deterministic and non-reactive traffic patterns

- **Need to adopt a holistic approach, taking into account application, node architecture, and network architecture**
  - Given the scale of current and future systems, parallel simulation is the only way to manage simulation run time and memory footprint
Application-level performance prediction

1. **Instrument applications to collect computation, communication and their inter-dependencies**
   - For apps or benchmarks of interest

2. **Collect traces on a production system**
   - e.g., BG/P, MareNostrum

3. **Perform full-system trace-driven simulations with Dimemas+Venus**
   - Tune model parameters to match reality
   - Perform parameter sensitivity studies
     - Network technology
     - Network topology
     - Routing, etc...

4. **Optimize**
   - Interconnect: e.g. performance/$
   - Application: e.g. communication scheduling
Simulation tool chain

Goal: Understand application usage of physical communication resources and facilitate optimal communication subsystem design
Outlook:
Exascale interconnects
Exascale interconnect challenges

- Proprietary network technologies are not commercially viable
- Indirect networks are unaffordable for extreme-scale HPC
- Integrated networks (NIC/switch on CPU) are becoming a reality
Towards exascale computing

- Current #1 on Top 500: 17.59 PetaFLOPS @ 8.2 MW
- 1 Exaflop = $10^{18}$ floating point operations per second @ 20 MW
- Timeframe: 2018/19/20

How do we increase FLOPs by two orders of magnitude while keeping power and space within the current order of magnitude?

1,000,000,000,000,000,000

- ~ 1000 racks per machine
- ~ 1000 chips per rack
- ~ 100 cores per chip
- ~ 1 TFLOPS per chip
- ~ 1 PFLOPS per rack
- ~ 10 GFLOPS per core

1 EXAFLOPS
Exascale network implications

- **Assume improve FLOPS/node improve 10x**
  - General purpose CPUs
  - 2 TFLOP/node
  - i.e. about 500’000 nodes

- **Byte/FLOP ratio**
  - Typically between 0.1 and 1
  - Let’s assume “cheap” 0.1 scenario
  - $10^{18}$ FLOPS $\Rightarrow 10^{17}$ B/s aggregate network injection bandwidth
  - i.e. about 200 GB/s/node

- **Network speed**
  - Speed will be driven by link technology: transceivers, connectors, cables/fibers; physical link probably ~50 Gb/s
  - By 2018/2019 100 Gb/s will be ramping up, 400 Gb/s emerging (standard ratification expected 2017)
  - $(10^{17}) / (12.5\times10^9) = 8$ million 100G ports!!

- **Network power**
  - Let’s be generous and allocate 25% of power budget to the network, i.e. 5 MW
  - Assume mean #hops per communication is 5 (which is VERY low) and links consume ZERO power when idle
  - Link efficiency: $(5\times10^6$ W) / $(5\times8\times10^{17}$ b/s) = $1.25\times10^{-12}$ = 1.25 pJ/bit
Data Center Port Shipments – Ethernet Switching

Source: Dell’Oro, Ethernet Alliance panel @ OFC’13

Source: networkworld.com
Silly example

- How much would a 128K-node fat tree network with 200 GB/s bandwidth per node cost if we tried to build it using commercial Ethernet switches today?

- 64 port switches
- 4 layers: 32*32*32*4 = 128K
- #switches = (128K/4) * 4 = 16K
- To achieve 200 GB/s we need 160 identical parallel networks
- Total number of switches = 160x16K = 2560K
- Such a switch costs at least $10K today

US$ 26B
(and then you haven’t bought a single cable yet)
Topology options

- **Fat tree**
  - Switch radix 64: tree arity = 32
  - Requires at least 4 layers ($32^3 = 32K$) to scale to 100-500K nodes
  - Diameter = 7
  - Bisection: full
  - Total of 64K switches for a 512K node full bisection bandwidth network!
  - Indirect topology with high-radix switches and many long links
    - For small radix ($k<n$), the number of switches may exceed number of nodes
  - Switch radix too large for integration on CPU: Discrete switch chips & boxes (e.g. Ethernet or IB)
  - Best performance, but completely unaffordable

- **Torus**
  - Switch radix 10: 5 dimensions
  - Requires 10-14 nodes along each dimension to scale to 100-500K nodes
  - Diameter = 50-70
  - Bisection: low
  - Direct topology with low-radix switches and mostly short links
  - Switches can be integrated: vastly reduced cost, but low bisection bandwidth and huge diameter

- **Dragonfly**
  - Number of nodes = $a(ah+1)$
  - Requires switch radix 90-150: $a = 60-100$, $h = 30-50$ to scale to 100-500K nodes
  - Diameter = 3
  - Bisection: high (but requires non-minimal routing)
  - Direct topology with very-high-radix switches and many long links
  - Switch radix too large for integration on CPU: Discrete switch chips (e.g. p775 hub chip); 1 switch per node

- **What we need is a direct topology with low-radix switches, small diameter, high bisection bandwidth and not too many long links**
  - Is it too much to ask for?

“Die eierlegende Wollmilchsau”
Outlook:
Trends in commercial servers
Three key trends in commercial servers

- Move towards “density optimized” servers
- ARM-based SoCs moving into server domain
- Off-chip network moving on-chip
HPC vs overall server market

- **2012 server shipments:** US$ 51.3B (IDC)
- **HPC market ~ US$ 10B**
  - Supercomputer (> 500 K$): ~45%
  - Divisional (250-500 K$): ~13%
  - Departmental (100-250 K$): ~31%
  - Workgroup (< 100 K$): ~11%
- **Modular form factors**
  - Blade servers revenue grew 3.3% year over year to $2.4 billion (16.3% of total server revenue)
  - Density Optimized servers revenue grew 66.4% year over year in 4Q12 to $705 million (4.8% of all server revenue and 9.2% of all server shipments).

*Source: IDC*
High-density server platforms based on embedded CPUs

- **AMD SeaMicro™**
  - AMD Opteron™ + SeaMicro™ “Freedom” 3D torus network
- **HP Moonshot**
  - Intel® Atom™ /Calxeda® + Ethernet & backplane Torus
- **IBM DOME/SKA**
  - Freescale™ + Ethernet
- **Intel®’s acquisition of Fulcrum/Ethernet, QLogic®/InfiniBand, Cray®/Aries**
  - Intel® Atom™ “Avoton” to have integrated Ethernet controller

- **How long before a small Ethernet or InfiniBand switch moves onto CPU?**

- **Mid-range commodity server business getting squeezed between high-end and embedded?**
PC market collapse erodes x86 volume base

<table>
<thead>
<tr>
<th>Vendor</th>
<th>1Q13 Shipments</th>
<th>1Q13 Market Share</th>
<th>1Q12 Shipments</th>
<th>1Q12 Market Share</th>
<th>1Q13/1Q12 Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. HP</td>
<td>11,997</td>
<td>15.7%</td>
<td>15,726</td>
<td>17.7%</td>
<td>-23.7%</td>
</tr>
<tr>
<td>2. Lenovo</td>
<td>11,700</td>
<td>15.3%</td>
<td>11,705</td>
<td>13.2%</td>
<td>0.0%</td>
</tr>
<tr>
<td>3. Dell</td>
<td>9,010</td>
<td>11.8%</td>
<td>10,110</td>
<td>11.4%</td>
<td>-10.9%</td>
</tr>
<tr>
<td>4. Acer Group</td>
<td>6,150</td>
<td>8.1%</td>
<td>8,952</td>
<td>10.1%</td>
<td>-31.3%</td>
</tr>
<tr>
<td>5. ASUS</td>
<td>4,363</td>
<td>5.7%</td>
<td>5,401</td>
<td>6.1%</td>
<td>-19.2%</td>
</tr>
<tr>
<td>Others</td>
<td>33,075</td>
<td>43.4%</td>
<td>36,739</td>
<td>41.5%</td>
<td>-10.0%</td>
</tr>
<tr>
<td>Total</td>
<td>76,294</td>
<td>100.0%</td>
<td>88,635</td>
<td>100.0%</td>
<td>-13.9%</td>
</tr>
</tbody>
</table>

Source: IDC Worldwide Quarterly PC Tracker, April 10, 2013
Network moving onto processor

- Driven by considerations of cost and power

- Convergence of computation and communication in silicon
  - Integrated L1/L2/L3 caches; memory controller; PCIe controller; multi-chip interconnect (e.g. HT, QPI replacing FSB); network-on-chip
  - Integrated SMP fabric, I/O hub; NIC/CNA/HCA; special-purpose accelerators

- High-end and “low”-end are at the vanguard of this trend
  - Extreme scale HPC: BlueGene/Q
  - Low-end: embedded SoCs for mobile & server applications
  - In both cases, cost and power are critical

- Commodity desktop & server processors are lagging somewhat
  - But with Intel®’s recent networking acquisitions in Ethernet, IB & HPC we can expect this to change soon
  - HP Moonshot: example for HPC topology in commercial server platform
Additional reading

  - Online lectures, EE382C: Interconnection Networks (Bill Dally, Stanford): http://classx.stanford.edu/ClassX/system/users/web/pg/view_subject.php?subject=EE382C_SPRING_2010_2011

Thank you!