Keynote

High-Performance Interconnection Networks on the Road to Exascale HPC: Challenges and Solutions

Pedro Javier García García
Jesús Escudero-Sahuquillo
Francisco J. Quiles
Universidad de Castilla-La Mancha (UCLM)
SPAIN

José Duato
Universtitat Politècnica de València (UPV)
SPAIN

September 12th, Barcelona, SPAIN
Outline

• Introduction
• Topologies: Scalability, Routing and Fault-Tolerance
• Power Efficiency
• Congestion Awareness
• Conclusions
## Introduction

What does the Exascale challenge consist in?

<table>
<thead>
<tr>
<th></th>
<th>2010</th>
<th>2018</th>
<th>Factor Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>System peak</td>
<td>2 Pf/s</td>
<td>1 Ef/s</td>
<td>500</td>
</tr>
<tr>
<td>Power</td>
<td>6 MW</td>
<td>20 MW</td>
<td>3</td>
</tr>
<tr>
<td>System Memory</td>
<td>0.3 PB</td>
<td>10 PB</td>
<td>33</td>
</tr>
<tr>
<td>Node Performance</td>
<td>0.125 Gf/s</td>
<td>10 Tf/s</td>
<td>80</td>
</tr>
<tr>
<td>Node Memory BW</td>
<td>25 GB/s</td>
<td>400 GB/s</td>
<td>16</td>
</tr>
<tr>
<td>Node Concurrency</td>
<td>12 cpus</td>
<td>1,000 cpus</td>
<td>83</td>
</tr>
<tr>
<td>Interconnect BW</td>
<td>1.5 GB/s</td>
<td>50 GB/s</td>
<td>33</td>
</tr>
<tr>
<td>System Size (nodes)</td>
<td>20 K nodes</td>
<td>1 M nodes</td>
<td>50</td>
</tr>
<tr>
<td>Total Concurrency</td>
<td>225 K</td>
<td>1 B</td>
<td>4,444</td>
</tr>
<tr>
<td>Storage</td>
<td>15 PB</td>
<td>300 PB</td>
<td>20</td>
</tr>
<tr>
<td>Input/Output bandwidth</td>
<td>0.2 TB/s</td>
<td>20 TB/s</td>
<td>100</td>
</tr>
</tbody>
</table>

Introduction

Current situation

• **Breakdown of Moore’s Law** and **Dennard Scaling**: Transistors may become smaller but power density is no longer constant but increases, so no way for “ever faster chips”

• **Current multicore processors** on the way to achieve **more computing power** and **less power consumption**
  – Current ARM products offer a good performance/watt ratio
  – Expected Intel, AMD or NVIDIA power-efficient solutions

• **Accelerators** can help to increase performance in heterogeneous systems while keeping power consumption
## Introduction

### Current Green500 list

![Table showing Green500 list]

1 ExaFLOP = 311 MW

*Performance data obtained from publicly available sources including TOP500*
### Introduction

Current TOP500 list

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>System</th>
<th>Cores</th>
<th>Rmax (TFlop/s)</th>
<th>Rpeak (TFlop/s)</th>
<th>Power (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National University of Defense Technology</td>
<td>Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.20GHz, TH Express-2, Intel Xeon</td>
<td>3,120,000</td>
<td>33,862.7</td>
<td>54,902.4</td>
<td>17,808</td>
</tr>
<tr>
<td>2</td>
<td>DOE/SC/Oak Ridge National Laboratory</td>
<td>Linux, Cray XK7, Opteron 6274 16C 2.20GHz, Cray Gemini Connect, NVIDIA K20x</td>
<td>560,640</td>
<td>17,590.0</td>
<td>27,112.5</td>
<td>8,209</td>
</tr>
<tr>
<td>3</td>
<td>DOE/NNSA/LLNL</td>
<td>Sequoia - BlueGene/Q, BOC 16C 1.60 GHz, Custom</td>
<td>1,572,864</td>
<td>17,173.2</td>
<td>20,132.7</td>
<td>7,890</td>
</tr>
<tr>
<td>4</td>
<td>RIKEN Advanced Institute for Computational Science (AICS)</td>
<td>K computer - Custom</td>
<td>1,280.4</td>
<td>12,660</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/Argonne National Laboratory</td>
<td>Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom</td>
<td>768,432</td>
<td>8,560.8</td>
<td>10,066.3</td>
<td>3,945</td>
</tr>
<tr>
<td>6</td>
<td>Texas Advanced Computing Center/Univ. of Texas</td>
<td>Stampede - PowerEdge C8220, Xeon E5-2680 8C 2.70GHz, Infiniband FDR, Intel Xeon Phi SE10P</td>
<td>462,462</td>
<td>5,168.1</td>
<td>8,520.1</td>
<td>4,510</td>
</tr>
</tbody>
</table>

**Tianhe – 1\textsuperscript{st} TOP500**

55 PFLOPS (peak) / 17,8 MW
Introduction

How to achieve Exascale goals?

• It is still clearly necessary to increase drastically the performance/watt ratio to achieve Exascale goals, but HOW?

• Most likely approach: Exascale processors are likely to reduce their peak performance to save power, while Exascale systems are likely to require many more processors
## Introduction

Massive parallelism in Exascale systems

<table>
<thead>
<tr>
<th></th>
<th>2010</th>
<th>2018</th>
<th>Factor Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>System peak</td>
<td>2 Pf/s</td>
<td>1 Ef/s</td>
<td>500</td>
</tr>
<tr>
<td>Power</td>
<td>6 MW</td>
<td>20 MW</td>
<td>3</td>
</tr>
<tr>
<td>System Memory</td>
<td>0.3 PB</td>
<td>10 PB</td>
<td>33</td>
</tr>
<tr>
<td>Node Performance</td>
<td>0.125 Gf/s</td>
<td>10 Tf/s</td>
<td>80</td>
</tr>
<tr>
<td>Node Memory BW</td>
<td>25 GB/s</td>
<td>400 GB/s</td>
<td>16</td>
</tr>
<tr>
<td><strong>Node Concurrency</strong></td>
<td>12 cpus</td>
<td>1,000 cpus</td>
<td>83</td>
</tr>
<tr>
<td>Interconnect BW</td>
<td>1.5 GB/s</td>
<td>50 GB/s</td>
<td>33</td>
</tr>
<tr>
<td><strong>System Size (nodes)</strong></td>
<td>20 K nodes</td>
<td>1 M nodes</td>
<td>50</td>
</tr>
<tr>
<td>Total Concurrency</td>
<td>225 K</td>
<td>1 B</td>
<td>4,444</td>
</tr>
<tr>
<td>Storage</td>
<td>15 PB</td>
<td>300 PB</td>
<td>20</td>
</tr>
<tr>
<td>Input/Output bandwidth</td>
<td>0.2 TB/s</td>
<td>20 TB/s</td>
<td>100</td>
</tr>
</tbody>
</table>


*U.S. Department of Energy, Fall 2010*
Introduction

How to achieve Exascale goals?

- It is still clearly necessary to increase drastically the performance/watt ratio to achieve Exascale goals, but HOW?

- Most likely approach: Exascale processors are likely to reduce their peak performance to save power, while Exascale systems are likely to require many more processors

- Consequently, interconnection networks able to connect a huge number of nodes and processors are likely to be required in future Exascale systems

- However, designing interconnection networks suitable to Exascale systems is not obvious
## Introduction

### Interconnection Networks in the Exascale challenge

<table>
<thead>
<tr>
<th></th>
<th>2010</th>
<th>2018</th>
<th>Factor Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>System peak</td>
<td>2 Pf/s</td>
<td>1 Ef/s</td>
<td>500</td>
</tr>
<tr>
<td>Power</td>
<td>6 MW</td>
<td>20 MW</td>
<td>3</td>
</tr>
<tr>
<td>System Memory</td>
<td>0.3 PB</td>
<td>10 PB</td>
<td>33</td>
</tr>
<tr>
<td>Node Performance</td>
<td>0.125 Gf/s</td>
<td>10 Tf/s</td>
<td>80</td>
</tr>
<tr>
<td>Node Memory BW</td>
<td>25 GB/s</td>
<td>400 GB/s</td>
<td>16</td>
</tr>
<tr>
<td>Node Concurrency</td>
<td>12 cpus</td>
<td>1,000 cpus</td>
<td>83</td>
</tr>
<tr>
<td>Interconnect BW</td>
<td>1.5 GB/s</td>
<td>50 GB/s</td>
<td>33</td>
</tr>
<tr>
<td>System Size (nodes)</td>
<td>20 K nodes</td>
<td>1 M nodes</td>
<td>50</td>
</tr>
<tr>
<td>Total Concurrency</td>
<td>225 K</td>
<td>1 B</td>
<td>4,444</td>
</tr>
<tr>
<td>Storage</td>
<td>15 PB</td>
<td>300 PB</td>
<td>20</td>
</tr>
<tr>
<td>Input/Output bandwidth</td>
<td>0.2 TB/s</td>
<td>20 TB/s</td>
<td>100</td>
</tr>
</tbody>
</table>

Introduction
Power Consumption in Interconnection Networks

- Power consumption fraction of the interconnection network near 35% of total
- Most of the network power consumption is devoted to the links
- Depending on the application, the power consumption can be significantly affected

Introduction
Challenges in Exascale Interconnection Networks

- Performance Requirements
- Scalability
- Simplicity
- Reliability
- Fault Tolerance
- Cost and Power Consumption
- Congestion Management

They must not be considered separately, since they are closely related.
Outline

• Introduction

• Topologies: Scalability, Routing and Fault-Tolerance

• Power Efficiency

• Congestion Awareness

• Conclusions
Topologies
Scaling to 1M endnodes

• **Main objectives:**
  – High connectivity
  – Low latency and high throughput
  – Reducing *cost and power consumption*

• **Design trends:**
  – Reducing *network diameter* (reaching more nodes in fewer hops)
  – Optimizing the *number of components* (no overdimension)
  – Cost-efficient *routing algorithms*
  – Increasing *path diversity*
Topologies
Direct Networks

- Network Latency is related to network diameter
- **Routing algorithms**: DOR, Oblivious, Adaptive, etc. Most of them impose routing restrictions to avoid **deadlocks**
- **High number of dimensions** increase the switch/routing complexity

Mesh

Iorus

Hypercube
Topologies
Indirect Networks

- **Fat-Trees** are widely used in real systems
- **High effective bandwidth**
- **Cost-efficient routing algorithms** (e.g. DESTRO / D-mod-K)
- **Tradeoff**: high-radix switches (fewer switches but more complex) versus low-radix switches (more switches, simplicity, high cost)
- **Network diameter** depends on the number of stages

![k-ary n-tree and n-stage k-shuffle-exchange](image)
Routing
Efficient Deterministic Routing Algorithms for Indirect Networks

• Tailored to specific network topologies
• Balance the destinations among the different paths
• Offer the same performance as do adaptive routing while they require fewer resources to be implemented
• They solve packet out-of-order delivery problems
• Can be recalculated if some faults appear in the network


Routing

Example of Efficient Routing: DESTRO in a k-ary n-tree

Balances the use of links by different paths
Limitations of the classic topologies in large networks

- **Direct networks:**
  - Cheap: fewer switches and links
  - Lower performance
  - Higher average length of paths

- **Indirect networks:**
  - Expensive: many switches and links
  - Higher performance
  - Lower average length of paths
Topologies

Hierarchical Networks

- Most prominent example are **Dragonflies**
- **Hierarchical network** (3-levels): switch, group and system
- Global links are significantly long
- Network diameter reduction
- High number of links makes them **expensive**
Topologies
Hybrid Networks (KNS)

- Designed for **large networks**
- Based on **direct** and **indirect topologies**
- Reduces the **diameter**, **number of switches** and links
- **High path diversity**, which allows a high level of fault-tolerance
- **Low latency**, **high-throughput** and **lower cost** than indirect networks
- **Hybrid-DOR routing**
Topologies
KNS hybrid topology

• Nodes are organized orthogonally, in several dimensions, like in direct networks:
  – Routers

• Dimensions are connected by means of indirect networks:
  – Crossbar, Fat-tree, ...

• Defined using three parameters: K, N and S
Routing
Example of Hybrid-DOR in a KNS hybrid topology

Roberto Peñaranda, Crispín Gómez Requena, María Engracia Gómez, Pedro López, José Duato: A New Family of Hybrid Topologies for Large-Scale Interconnection Networks. NCA 2012: 220-227
Topologies

KNS hybrid topology

• KNS is **superior** to existing topologies because:
  
  – It provides **switching capabilities at both switches and network interfaces**, and not only at switches (like indirect networks) or at network interfaces (like direct networks).
  
  – It **provides a large number of alternative paths**, all of them having the same length, unlike other topologies with high connectivity (e.g. the flattened butterfly provides many alternative paths longer than the minimal one).
  
  – It **directly benefits from the best routing techniques** for orthogonal direct networks and for fat trees, **requiring neither hierarchical nor non-minimal routing algorithms** for achieving a high path diversity.
Topologies
KNS hybrid topology

• **KNS summary:**
  - A huge number of nodes may be connected efficiently
  - Higher performance and lower cost than other topologies (e.g. Flattened Butterflies)
  - Small network diameter
  - High scalability

• **Open issues to be solved** by current infrastructure:
  - Is current technology able to implement the router features? (even for 3D, 4D KNS networks)
  - Fault tolerance and power efficiency
  - Congestion management
Topologies and Scalability
Fault Tolerance

- Hybrid topologies offer a high number of *alternative paths*, thus easing fault tolerance.

- **Current techniques** (DFSSSP, LASH) could be applied to hybrid topologies with minimal cost.

- Considering the huge number of nodes and cores in Exascale systems, fault tolerance may become a mandatory issue.
Outline

• Introduction
• Topologies: Scalability, Routing and Fault-Tolerance
• **Power Efficiency**
• Congestion Awareness
• Conclusions
Power Efficiency

Motivation

- High **cost of the power consumption bill** for large HPC systems: power and cooling

- The **interconnection network power consumption fraction** is about 20% of the total idle power, increasing an additional 20% when simple benchmarks are used [1]

- Some **advances in power consumption for CPUs** and/or memories, but there is a gap to cover in interconnects

- Power Efficiency in HPC interconnect is still a **challenge**:  
  - **Idle networks** have a high power consumption  
  - **Hw/Sw infrastructure** must offer power efficiency

Power Efficiency
Energy consumption

• Most of the interconnects energy spent by the links

• **Number and length** of the links is important

• **Contention** increases the power consumption

• Current solutions:
  – Hardware
  – Software
Power Efficiency

Software solutions

• **Proactive solutions:**
  – Schedule the traffic so that hot-spots are minimized
  – Maintain the network with low utilization

• **Problems of software solutions:**
  – Medium term technologies increase the link speed
  – Exascale topologies make the traffic scheduling very complex
  – Even at low network utilization, the idle power consumed by the links is significant
Power Efficiency

Hardware solutions

• Dynamic Voltage Scaling (DVS)
  – Adds complexity
  – Introduces delay overhead

• Turn off the links completely:
  – Requires a fault-tolerant routing algorithm
  – Path diversity is also required
  – Adds complexity
  – Slow reaction to traffic bursts
Power Efficiency

Hardware solutions

• If ports are connected to **aggregated parallel links** (i.e. 4x, 8x...): Turning **on and off dynamically individual links** of the same port (w/o disabling it completely):
  
  – Connectivity is not affected

  – The routing algorithm is preserved

• **Common problems of hardware solutions**:
  
  – **Slow reaction** when traffic bursts appear

  – Traffic bursts may **lead the system to congestion**

---

Outline

- Introduction
- Topologies: Scalability, Routing and Fault-Tolerance
- Power Efficiency
- Congestion Awareness
- Conclusions
Congestion Awareness

Why is congestion management necessary?

• Exascale networks: around one million of endnodes

• Cost and power consumption constraints lead to use the minimum number of components, thus working close to the saturation zone and increasing congestion probability

• Power efficiency policies react slowly to traffic bursts
Congestion-Derived Problems
Low-Order Head-of-Line (HoL) Blocking

Flow Control backpressure

Low-order HoL-blocking

- 33 % Sending
- 33 % Stopped
- 33 % Sending
Congestion-Derived Problems
High-Order Head-of-Line (HoL) Blocking

High-order HoL-blocking
33% Sending
33% Stopped
33% Sending

Hot packets (Dst 1)
Cold packets (Dst 2)
Cold packets (Dst 3)
Congestion-Derived Problems
High-Order Head-of-Line (HoL) Blocking

High-order HoL-blocking
33 % Sending
33 % Stopped
33 % Sending

Hot packets (Dst 1)
Cold packets (Dst 2)
Cold packets (Dst 3)
Congestion-Derived Problems
Buffer Hogging / Intra-VL hogging

Kenji Yoshigoe: Threshold-based Exhaustive Round-Robin for the CICQ Switch with Virtual Crosspoint Queues. ICC 2007: 6325-6329

High-Performance Interconnection Networks on the Road to Exascale HPC: Challenges and Solutions
Pedro Javier García , UCLM (Spain)
Congestion-Derived Problems
Buffer Hogging / Intra-VL hogging

Kenji Yoshigoe: Threshold-based Exhaustive Round-Robin for the CICQ Switch with Virtual Crosspoint Queues. ICC 2007: 6325-6329
Congestion Awareness
How can congestion be managed?

- Different approaches to congestion management:
  - Packet dropping
  - Proactive techniques
  - Reactive techniques
  - HoL-blocking prevention techniques
  - Hybrid techniques
  - Related techniques
Congestion Awareness

Reactive congestion management

- A.K.A. congestion recovery

- Injection limitation techniques (injection throttling) using closed-loop feedback

- Does not scale with network size and link bandwidth
  - Notification delay (proportional to distance / number of hops)
  - Link and buffer capacity (proportional to clock frequency)
  - May produce traffic oscillations (closed loop system with pure delay)
Congestion Awareness
Reactive congestion management

- Example: Infiniband FECN/BECN mechanism:
  - Two bits in the packet header are reserved for congestion notification.
  - If a switch port is considered as congested, the Forward Explicit Congestion Notification (FECN) bit in the header of packets crossing that port is set.
  - Upon reception of such a “FECN-marked” packet, a destination will return a packet (Congestion Notification Packet, CNP) whose header will have the Backward Explicit Congestion Notification (BECN) bit set back to the source.
  - Any source receiving a “BECN-marked” packet will then reduce its packet injection rate for this traffic flow.

Congestion Awareness
HoL-blocking prevention techniques

- In general, these techniques rely on having several queues (or VLs) and/or several read ports, at the buffer of each port to separate different packet flows.

- Queuing schemes differ mainly in the criteria to map packets to queues and in the number of required queues per port.

![Diagram of storage queues](image)
### Congestion Awareness

Classical Generic “Static-Mapping” Queuing Schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Low-order prevention</th>
<th>High-order prevention</th>
<th>Scalable (network size)</th>
<th>Scalable (#switch ports)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOQnet</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>VOQsw</td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DAMQs</td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DBBM</td>
<td>Partial</td>
<td>Partial</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

In general, some queues are wasted at some ports as they are “topology agnostic” schemes.
### Congestion Awareness

#### Topology- & Routing –Aware “Static-Mapping” Schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Topology</th>
<th>Low-order prevention</th>
<th>High-order prevention</th>
<th>Scalable (network size)</th>
<th>Scalable (#switch ports)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBQA</td>
<td>Fat-Tree</td>
<td>Partial</td>
<td>Partial</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>vFtree</td>
<td>Fat-Tree</td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Flow2SL</td>
<td>Fat-Tree</td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>BBQ</td>
<td>KNS</td>
<td>Partial</td>
<td>Partial</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

In general, they achieve similar or better performance than topology-agnostic schemes while requiring fewer queues per port, so improving cost- and power- efficiency.
The KNS network is divided into logic horizontal “bands”, every port having as many queues as bands.

The packets addressed to different bands never share queues.

Band-Based Queuing (BBQ)
Congestion Awareness
Example of Topology-Aware Queuing Scheme: BBQ

• At each port, BBQ maps packets to queues according to the following formula:

\[
\text{SelectedQueue} = \frac{\text{Packet}\_\text{Destination} \times \text{Number}\_\text{Queues}}{\text{Number}\_\text{EndNodes}}
\]

• Easy implementation in InfiniBand technology:
  – Assigning each packet an SL equal to the queue given by the formula
  – Filling the SL-to-VL tables so that VL=SL

Congestion Awareness

Example of Topology-Aware Queuing Scheme: BBQ

- Packet Latency vs. Normalized Efficiency, Uniform Traffic Pattern (100% traffic addressed to random destinations),

16ary-2direct-1indirect
256 nodes

32ary-2direct-1indirect
1024 nodes
Congestion Awareness
Example of Topology-Aware Queuing Scheme: BBQ

- Normalized efficiency vs. Generated traffic, Hot-Spot Traffic pattern (75% of endnodes generating traffic to random destinations and 25% of endnodes generating traffic to a single destination)
Congestion Awareness
Tailoring Queuing Schemes to Exascale Topologies

• The **queue assignment** criterion (i.e. the mapping policy) **should exploit the properties** of both network topology and routing scheme

• Metrics to analytically evaluate a **specific mapping of traffic flows** (SLID, DLID) to **SLs** (i.e. to VLs):
  
  – **VL Load**: Number of flows mapped to a VL in a specific port (strongly depends on the routing algorithm)
  
  – **Balancing Degree**: Variation between the maximum and minimum values of VL loads (ideally identical values)
  
  – **Overlapping Degree**: Meassures the number of flows simultaneously mapped to several VLs at the same port (must be low to reduce intra-VL hogging probability, ideally zero)
Congestion Awareness

“Dynamic-Mapping” Queuing Schemes

• “Static-mapping” schemes prevent HoL-blocking and buffer-hogging as much as possible with the available queues, but not completely.

• A complete effectiveness in solving these problems would require to pay an “extra-price” in terms of complexity and additional resources, if **Dynamic-Mapping Queuing Schemes** (i.e. “RECN-like” schemes) were implemented:
  – **RECN** (deterministic source-based routing)
  – **FBICM** (deterministic distributed-based routing)
  – **DRBCM** (fat-trees with deterministic distributed-based routing, DESTRO-like routing)
  – ....
Congestion Awareness
“Dynamic-Mapping” Queuing Schemes Basics

- **Congested points are detected** at any port of the network by measuring *queue occupancy*.
- The **location** of any detected congested point is stored in a **control memory** (a CAM or T-CAM line) at any port forwarding packets towards the congested point.
- A **special queue** associated to the CAM line is also allocated to exclusively store packets addressed to that congested point.
- Congestion information is progressively notified to every port at upstream switches crossed by congested flows, where new CAM (or T-CAM) lines and special queues are allocated.
- A packet arriving at a port is stored in the **standard queue** only if its **routing information does not match any CAM line**.
Congestion Awareness
Example of Dynamic-Mapping Scheme: DRBCM

- Cold Packets Queue (CPQ)
- Hot Packets Queues (HPQ)

To Isolate hot flows from cold ones

Ternary Content Addressable Memory (T-CAM)
Congestion Awareness
Example of Dynamic-Mapping Scheme: DRBCM

- The mask field (using values 0, 1 and X) identifies all the destinations crossing a congestion root
- The mask is updated as congestion information is propagated
- The rest of the fields are required to manage the T-CAM line operations (flow-control, deallocation timer, etc.)
Congestion Awareness

Example of Dynamic-Mapping Scheme: DRBCM

<table>
<thead>
<tr>
<th>#</th>
<th>Root</th>
<th>Mask</th>
<th>Hops</th>
<th>oPort</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rz</td>
<td>010000</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Ry</td>
<td>xxx0000</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Rx</td>
<td>xxx0000</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Root</th>
<th>Mask</th>
<th>Hops</th>
<th>oPort</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rz</td>
<td>010000</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Ry</td>
<td>xxx0000</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
Congestion Awareness
Example of Dynamic-Mapping Scheme: DRBCM

- Execution Time of Real-Traffic Traces

![Bar chart showing execution times for different benchmarks]

4-ary 4-tree
256 nodes

Jesus Escudero-Sahuquillo, Pedro J. García, Francisco J. Quiles, Jose Flich, Jose Duato, *An Effective and Feasible Congestion Management Technique for High-Performance MINs with Tag-Based Distributed Routing*, IEEE Transactions on Parallel and Distributed Systems, October. 2013.
Congestion Awareness

Drawbacks of “RECN-like” Schemes

- In scenarios with several different congested points, it is possible to run out of special queues at some ports.
- The need for CAMs at switch ports increases switch complexity, implementation cost and required silicon area per port.
- Unfairness in the scheduling of hot flows may appear.
Congestion Awareness
Hybrid Congestion Management Strategies

• Combining Injection Throttling and Dynamic Mapping:
  – Using **Dynamic Mapping** to quickly and locally eliminate **HoL-blocking**, propagating congestion information and allocating queues as necessary
  – Using **Injection Throttling** to slowly eliminate congestion, deallocating special queues whenever possible
  – Use of **Dynamic Mapping** provides immediate response and allows reactive congestion management to be tuned for slow reaction, thus avoiding oscillations
  – **Injection Throttling** drastically reduces **Dynamic Mapping buffer requirements** (just one or two queues per port)
Congestion Awareness
Example of Hybrid Congestion Management: CCFIT

- Input ports like RECN (CAMs at input/output ports)
- HPQs assigned when the CPQ exceeds a threshold
- Output ports in congestion state, when HPQ reaches a High Threshold
- Packets are marked (FECN) at output ports in congestion state
- Output ports congestion state are deactivated when all the HPQs of the switch are below the Low Threshold
Congestion Awareness

Example of Hybrid Congestion Management: CCFIT

- HCAs must support both RECN-like queues + CAMs and typical InfiniBand Injection-Throttling structures (CCT, Timers, etc.)

- HCAs arbiter must take into account information from different structures
Congestion Awareness
Example of Hybrid Congestion Management: CCFIT

- Normalized Throughput vs. Time, 4 Hot-Spots

4-ary 4-tree
256 nodes

Jesús Escudero-Sahuquillo, Ernst Gunnar Gran, Pedro Javier García, Jose Flich, Tor Skeie, Olav Lysne, Francisco J. Quiles, José Duato: *Combining Congested-Flow Isolation and Injection Throttling in HPC Interconnection Networks*. Proceedings of ICPP 2011:
Congestion Awareness

Summary

Static Mapping of Hot-Flows to queues (or VLs)

Dynamic Mapping of Hot-Flows to queues (or VLs)

Injection Throttling

Combining Dynamic Hot-Flow Isolation and Injection Throttling

Simplicity
(Optimal efficiency with minimum complexity)

Use of Additional Resources

Effectiveness
(Complete HoL-blocking prevention)
Outline

• Introduction

• Topologies: Scalability, Routing and Fault-Tolerance

• Power Consumption

• Congestion Awareness

• Conclusions
Conclusions

• The performance/watt ratio of HPC systems must be significantly improved to reach **Exascale goals**

• Processor cores are likely to reduce their peak performance to reduce power consumption (unless new materials could improve the level of integration and power density)

• Thus, **many more processor nodes and much larger and improved networks** are likely to be required:
  – Endnodes are likely to contain one thousand interconnected cores
  – Network interfaces will increase their link speed
  – Networks of Exascale HPC Systems are likely to interconnect around 1 million endnodes
Conclusions

• Interconnects trends to meet Exascale requirements:
  – High network connectivity by means of topologies with reduced diameter to achieve low latency while keeping high-throughput
  – Efficient routing algorithms to evenly balance traffic
  – Increasing importance of fault tolerance and path diversity
  – Reducing the network power consumption fraction:
    – Power-efficiency solutions
    – Non-overdimensioned topologies
  – Congestion Management to prevent performance degradation:
    – Optimizing the use of available resources
    – Improving efficiency with additional resources
Questions???

Pedro Javier García García
Jesús Escudero-Sahuquillo
Francisco J. Quiles
Universidad de Castilla-La Mancha (UCLM)
SPAIN

José Duato
Universtitat Politècnica de València (UPV)
SPAIN

September 12th, Barcelona, SPAIN
Keynote

High-Performance Interconnection Networks on the Road to Exascale HPC: Challenges and Solutions

Pedro Javier García García
Jesús Escudero-Sahuquillo
Francisco J. Quiles
Universidad de Castilla-La Mancha (UCLM)
SPAIN

José Duato
Universtitat Politècnica de València (UPV)
SPAIN

September 12th, Barcelona, SPAIN