



Bridging the gap between Fabric Management and application performance

Ghislain de Jacquilot

March 22, 2010

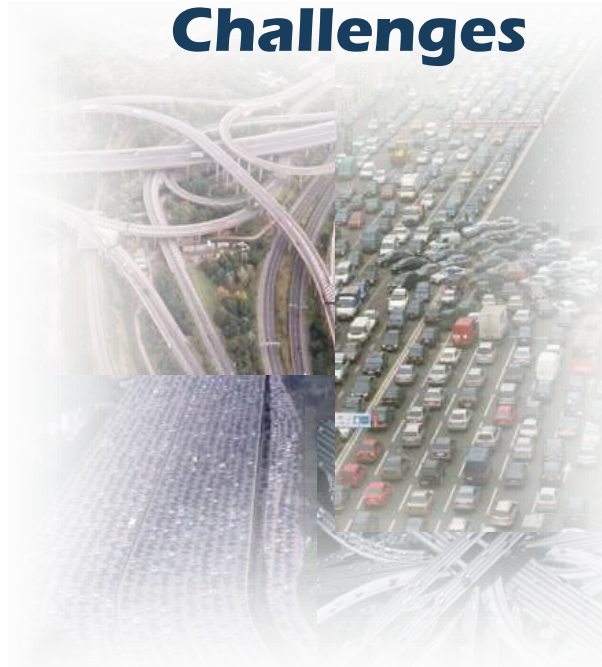
Data Center Challenges

Complexity



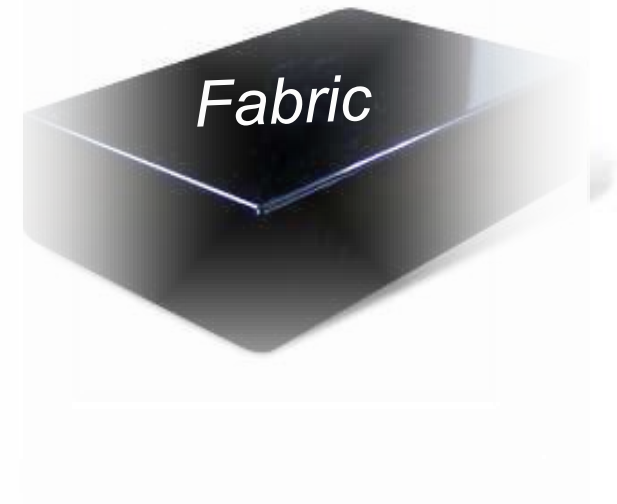
- Traditional device management is not enough

Traffic Challenges



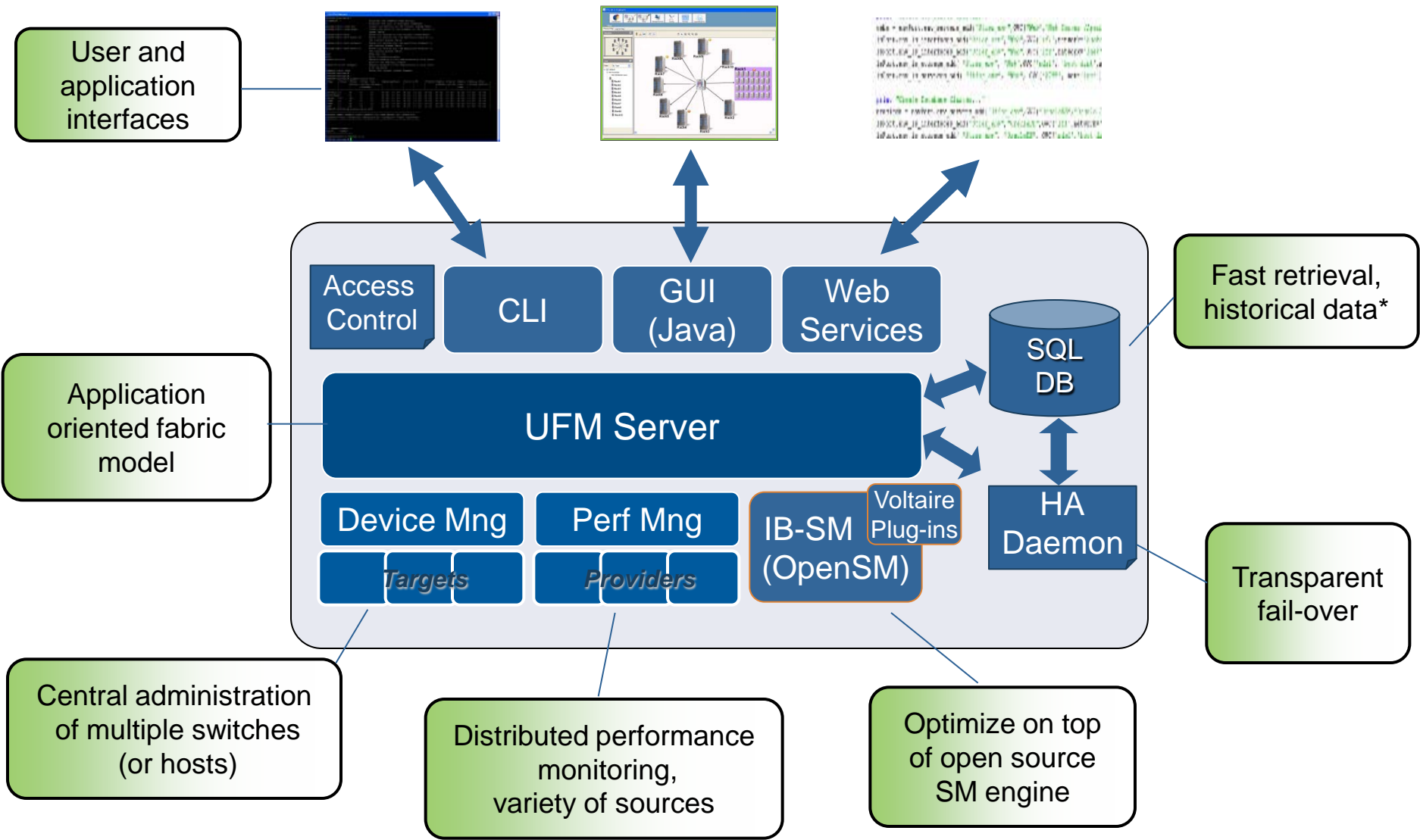
- Benchmarks differ from real-life traffic patterns

Lack of Visibility

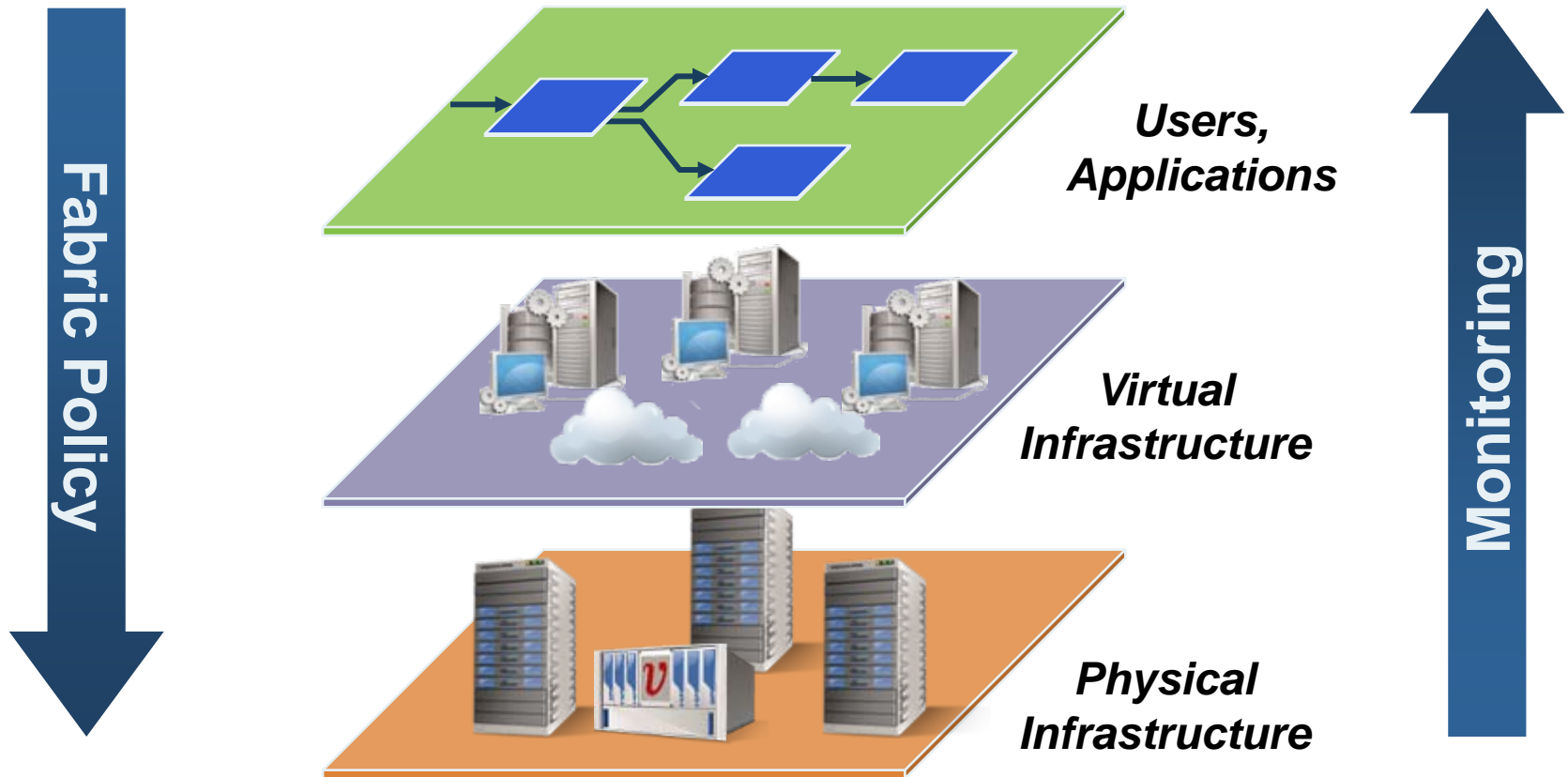


- Undetected issues, unutilized fabric, OPEX burden

UFM2.0 Architecture



Application correlated management



Map application requirements to fabric policies and

Map element status to application status

Advanced Monitoring and Analysis

▶ Monitor & analyze fabric performance

- Bandwidth utilization
- Unique congestion monitoring
- Dashboard for aggregated fabric view

▶ Real-time fabric-wide health monitoring

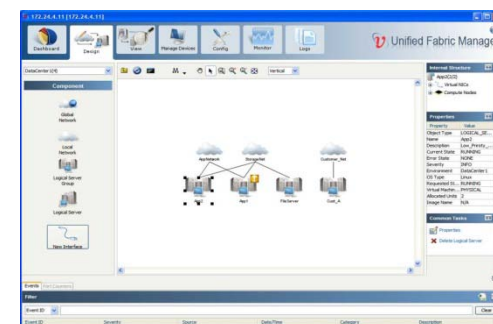
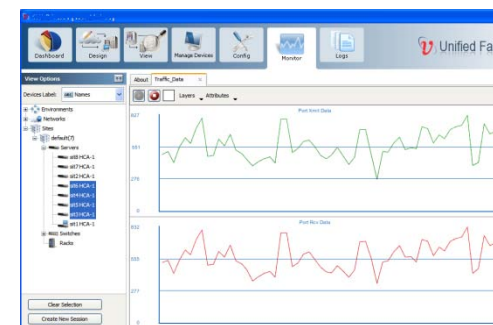
- Monitor events and errors through-out the fabric
- Threshold based alarms
- Granular monitoring of host and switch parameters

▶ Innovative congestion mapping

- One view for fabric-wide congestion and traffic patterns
- Enables root cause analysis for routing, job placement or resource allocation inefficiencies

▶ All is managed at the application/aggregation level

- Event effects are clearly visible
- Pro-active measures can be taken



Central Dashboard

*Top 10's
B/W, Congestion*

*Resource Utilization
& Status*

Congestion Map



Event Pane

B/W Consumers

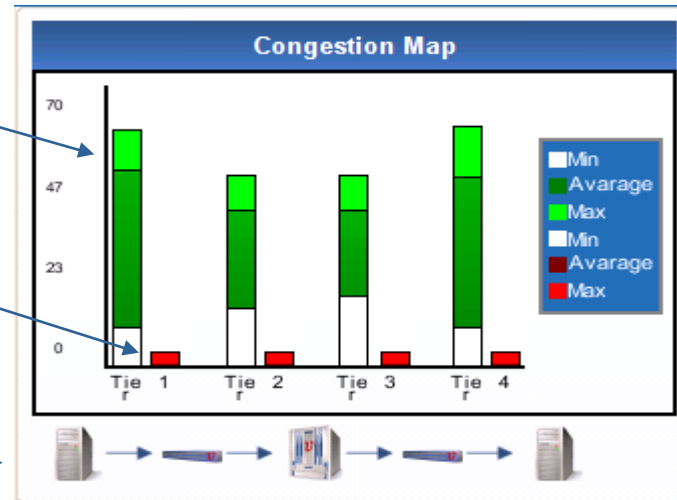
Top 10 alerted nodes

Unique Congestion Map

*Transmitted bandwidth (green)
Aggregation of all ports on tier*

*Congestion, lost bandwidth (red)
Aggregation of all ports on tier*

*Unfolded fabric topology
Source to destination tier analysis
(left to right)*

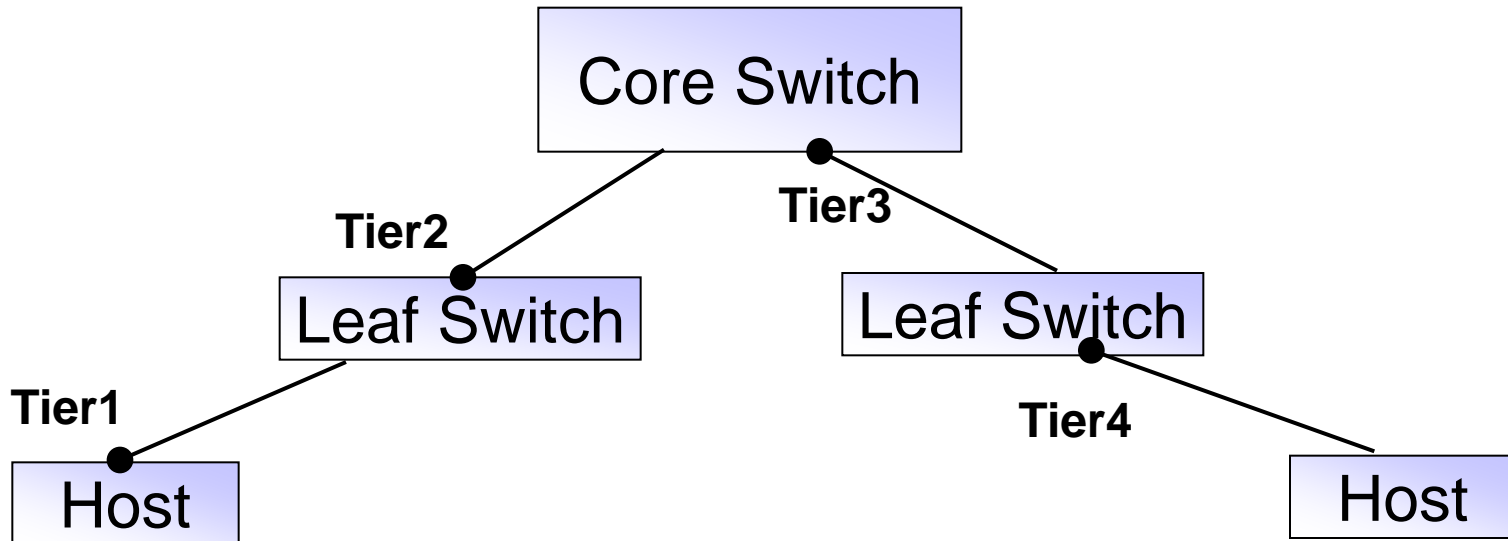


*Min ,
Max
and
AVG*

Data is normalized to % of total tier bandwidth (e.g. Avg of 50% means the average bandwidth across all nodes is 50%)

UFM2.0 Training – Fabric Monitoring – Congestion Map

- ▶ Tier1 – Tx Traffic from HCA to Leaf Switch
- ▶ Tier2 – Tx Traffic from Leaf Switch to Switch
- ▶ Tier3 – Tx Traffic from Non Leaf Switch to Switch
- ▶ Tier4 – Tx Traffic from Leaf Switch to HCA (opposite to T1)



Fabric Monitoring – Congestion Map

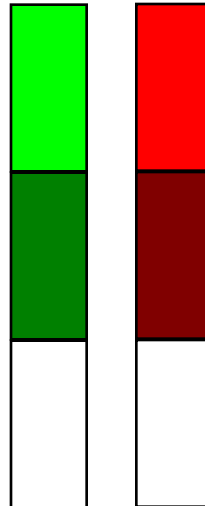
▶ **Normalized Congestion Bandwidth (NCB) [%]**

$$\Delta X_{mitWait} / Time\ Interval$$

▶ **Bandwidth Utilization [%]**

$$X_{mitRate} / Port\ Bandwidth$$

Max. BW Utilization within Tier



Max. % of Congested Bw within Tier

Avg. BW Utilization within Tier

Avg. % of Congested Bw within Tier

Min. BW Utilization within Tier

Min. % of Congested Bw within Tier

Understanding The Congestion Bars (Red)

- ▶ **The red bars indicate the level of congestion in the fabric**
- ▶ **Must analyze where the congestion starts**
- ▶ **Tier 1/2 congestion: Over Subscription**
 - Routing problem or too much blocking
 - If difference high between max and average: unbalanced
 - Solutions: TOR, add switches, implement QoS (to reduce congestion), different placement
- ▶ **Tier 2/3 congestion: Many to One**
 - Solution: QoS, Future: CCA (slow the sender)
- ▶ **Tier 4 congestion: Slow end-node**
 - Typically application/host related



Congestion Map Illustration



Advanced Monitoring Engine

*Multiple sessions
On demand*

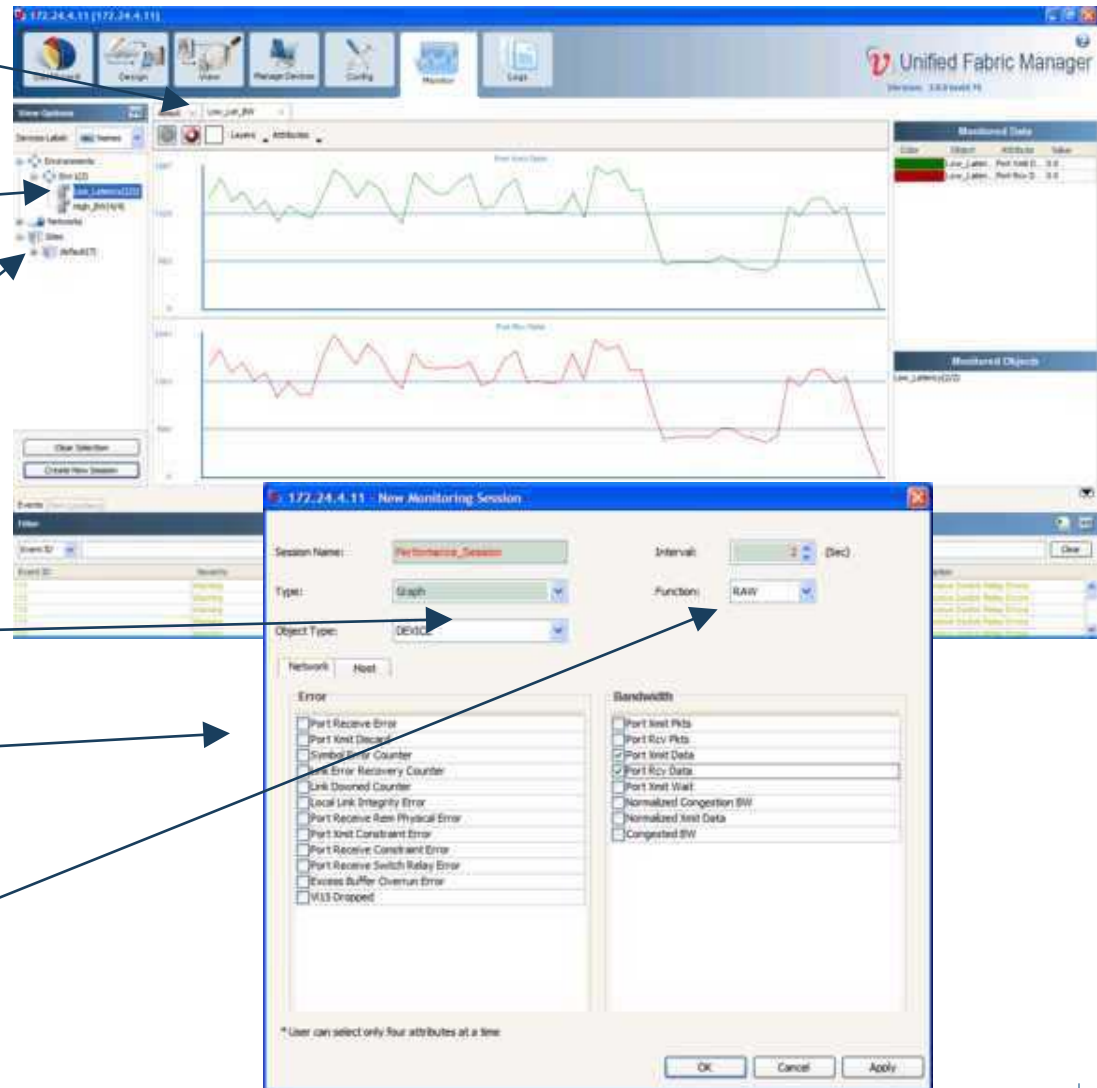
*Sessions per Logical
Groups – no need to
know physical nodes*

*Aggregation per
Multiple devices*

*Various graphs (linear,
bar, histogram, pie...)*

*Correlate switch and
host information*

*Formulas (AVG, Max,
Min, Sum)*

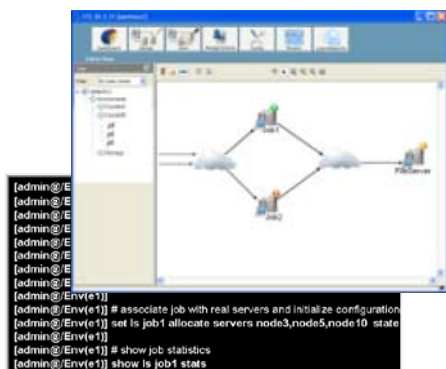


Performance Optimization Cycle with UFM

Feedback and Analysis



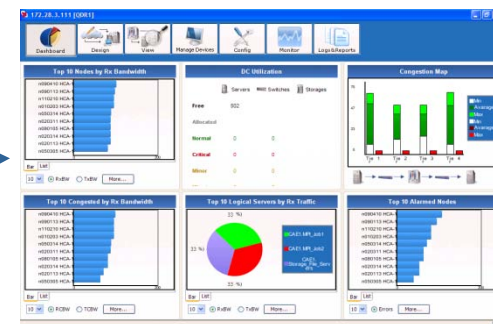
Application Requirements



UFM Optimization



UFM Monitoring



Orchestrators
& Schedulers

Optional

Characterize traffic pattern and priorities
Unique logical fabric model

QoS to prioritize critical apps.
Optimize routing with Voltaire's Traffic Optimized Routing (TOR)

Show traffic and congestion information
Unique Congestion Map

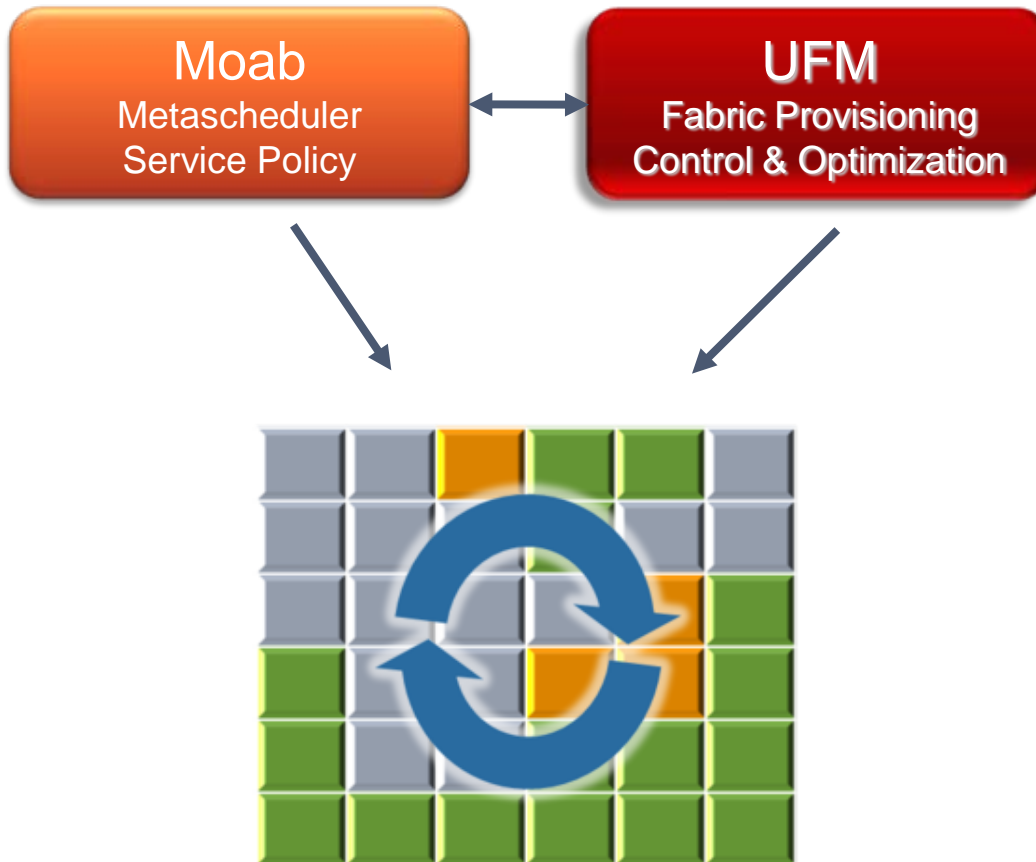
CLI Example: Configuring and Operating

Defining couple of networks, creating a new job/application, and monitoring it

```
[admin@/Env(e1)] # Create Networks for MPI and Lustre (once per cluster)
[admin@/Env(e1)] set network mpinet p_key 0xffff cos 0 typical_dst internal
[admin@/Env(e1)] set network lustre cos 2 typical_dst ost load 500
[admin@/Env(e1)]
[admin@/Env(e1)] # Create new job with 2 interfaces (to MPI and Storage)
[admin@/Env(e1)] set ls job1 description "New Fluent Job"
[admin@/Env(e1)] set ifc job1/ifc1 network mpinet load 300
[admin@/Env(e1)] set ifc job1/ifc1 network lustre
[admin@/Env(e1)] # associate job with real servers and initialize configuration
[admin@/Env(e1)] set ls job1 allocate servers node3,node5,node10 state running
[admin@/Env(e1)]
[admin@/Env(e1)] # show job statistics
[admin@/Env(e1)] show ls job1 stats
```

UFM Adaptive Suite

- Separate UFM offering integrated with Moab



- Intelligent & automatic resource allocation
- Optimize fabric performance
- Maintain connectivity upon changes
- Central monitoring

This is the first integrated solution that correlates network fabric management and workload management for dynamic data centers

Advanced Performance Optimization Mechanisms

▶ Fabric virtualization and Quality of Service (QoS)

- Run multiple clusters or multiple jobs on the same infrastructure
- Assure critical applications get priority through QoS policy
- Provide the required isolation for different departments or jobs

▶ Traffic Aware Routing Algorithm (TARA)

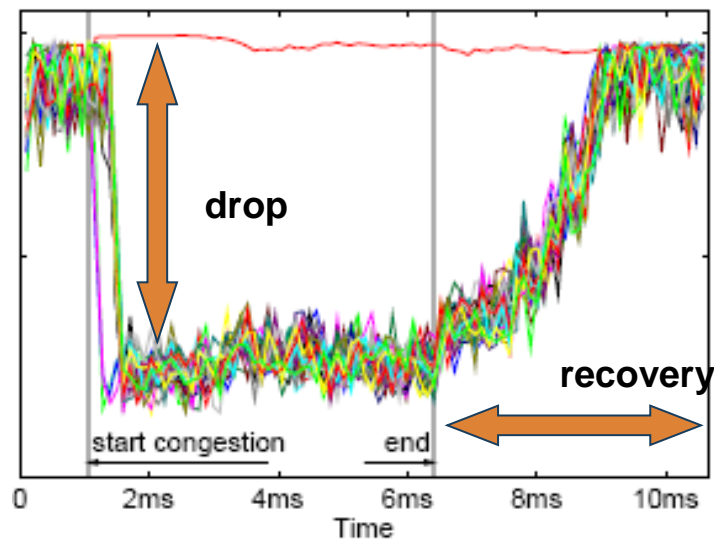
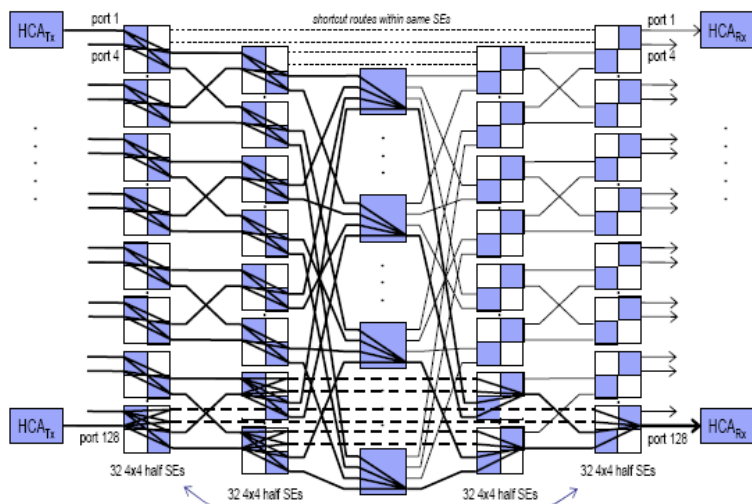
- Voltaire's major shift from static to traffic aware routing
- Routing enhancements are built on top of OpenSM in a modular plug-in architecture
- Takes into consideration traffic patterns and loads
- Traffic model can be derived automatically from fabric model or via API with 3rd party schedulers

Applicable to both DDR and QDR Environments

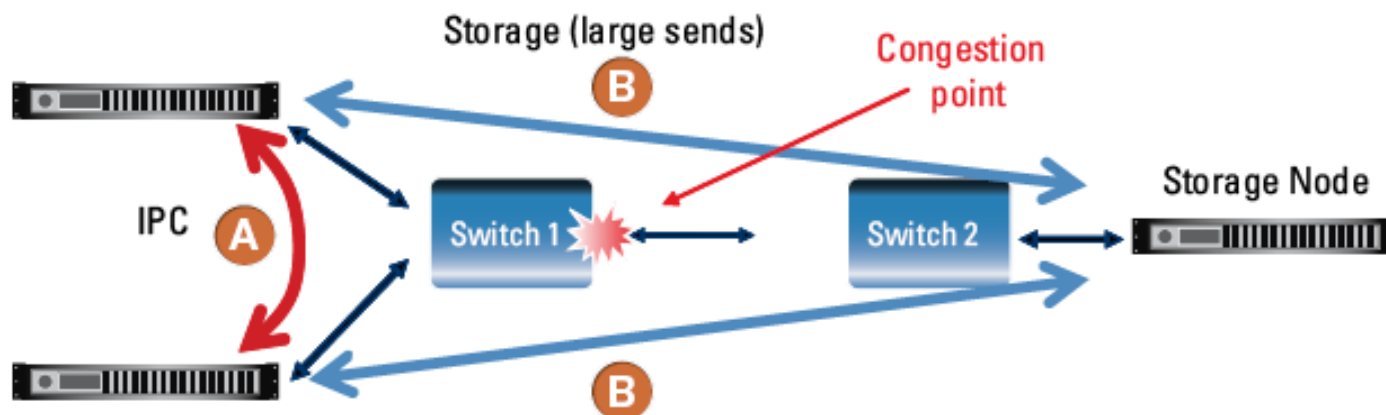
Congestion Example

► Degradation due to node oversubscription

- Destination port in saturation (multiple sources)
- Congestion spread across the fabric
- ALL other flows drop to **20%** of capacity
- Take time to recover
- Common with storage traffic



Quality of Service Optimization



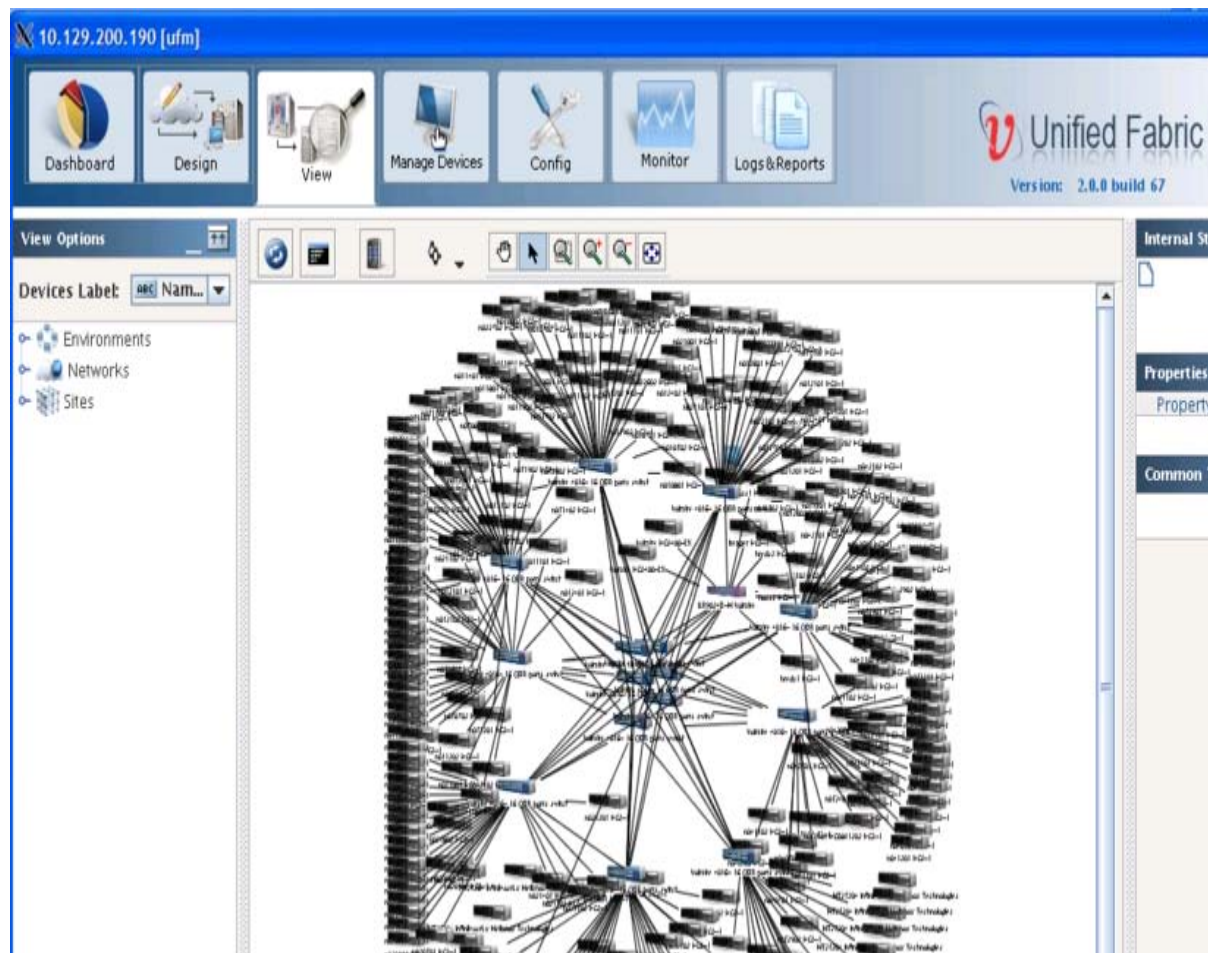
Case	A CoS	B CoS	IPC (A) Latency	
			Typ	Max
Just IPC traffic	0	None	1.2us	1.7us
IPC + Storage use the same CoS	0	0	18us	20us
IPC + Storage separate CoS	0	1	1.5us	2us

10x better latency!

UFM Enables QoS Optimization

Test Environment

- ▶ 2 nodes running a latency critical job
- ▶ 12 nodes running a bandwidth consuming job
- ▶ Goal: achieve best performance with Latency critical tasks



W/O Partitioning Latency degradation of ~215%

Latency job running alone
(Latency = ~2.1 us)

Bandwidth job added on
same partition
(Latency = ~4.5 us)

```
hwwnec5 @ cl3fr0 /nfs/home6/nec/hwwnec5
avg/min/max: 0.000207 0.000177 0.000330
avg/min/max: 0.000207 0.000177 0.000328
avg/min/max: 0.000208 0.000178 0.000337
avg/min/max: 0.000208 0.000177 0.000389
avg/min/max: 0.000208 0.000177 0.000343
avg/min/max: 0.000208 0.000178 0.000362
avg/min/max: 0.000205 0.000177 0.000342
avg/min/max: 0.000207 0.000177 0.000353
avg/min/max: 0.000206 0.000178 0.000347
avg/min/max: 0.000207 0.000177 0.000357
avg/min/max: 0.000209 0.000177 0.000358
avg/min/max: 0.000206 0.000177 0.000350
avg/min/max: 0.000206 0.000177 0.000356
avg/min/max: 0.000206 0.000177 0.000353
avg/min/max: 0.000206 0.000177 0.000359
avg/min/max: 0.000209 0.000177 0.000361
avg/min/max: 0.000209 0.000177 0.000364
avg/min/max: 0.000212 0.000178 0.000364
avg/min/max: 0.000210 0.000177 0.000364
avg/min/max: 0.000212 0.000177 0.000366
avg/min/max: 0.000213 0.000177 0.000361
avg/min/max: 0.000214 0.000177 0.000375
avg/min/max: 0.000214 0.000177 0.000369
```

```
hwwnec5 @ cl3fr0 /nfs/home6/nec/hwwnec5
avg/min/max: 0.000222 0.000177 0.000408
avg/min/max: 0.000221 0.000177 0.000405
avg/min/max: 0.000217 0.000177 0.000414
avg/min/max: 0.000221 0.000177 0.000421
avg/min/max: 0.000221 0.000177 0.000397
avg/min/max: 0.000219 0.000177 0.000415
avg/min/max: 0.000221 0.000177 0.000407
avg/min/max: 0.000319 0.000177 0.000593
avg/min/max: 0.000447 0.000178 0.000624
avg/min/max: 0.000480 0.000272 0.000628
avg/min/max: 0.000456 0.000178 0.000633
avg/min/max: 0.000479 0.000306 0.000655
avg/min/max: 0.000476 0.000303 0.000657
avg/min/max: 0.000479 0.000290 0.000644
avg/min/max: 0.000444 0.000178 0.000643
avg/min/max: 0.000488 0.000301 0.000606
avg/min/max: 0.000487 0.000310 0.000631
avg/min/max: 0.000487 0.000324 0.000665
avg/min/max: 0.000484 0.000317 0.000631
avg/min/max: 0.000483 0.000332 0.000688
avg/min/max: 0.000481 0.000327 0.000636
avg/min/max: 0.000480 0.000323 0.000694
avg/min/max: 0.000424 0.000177 0.000626
```

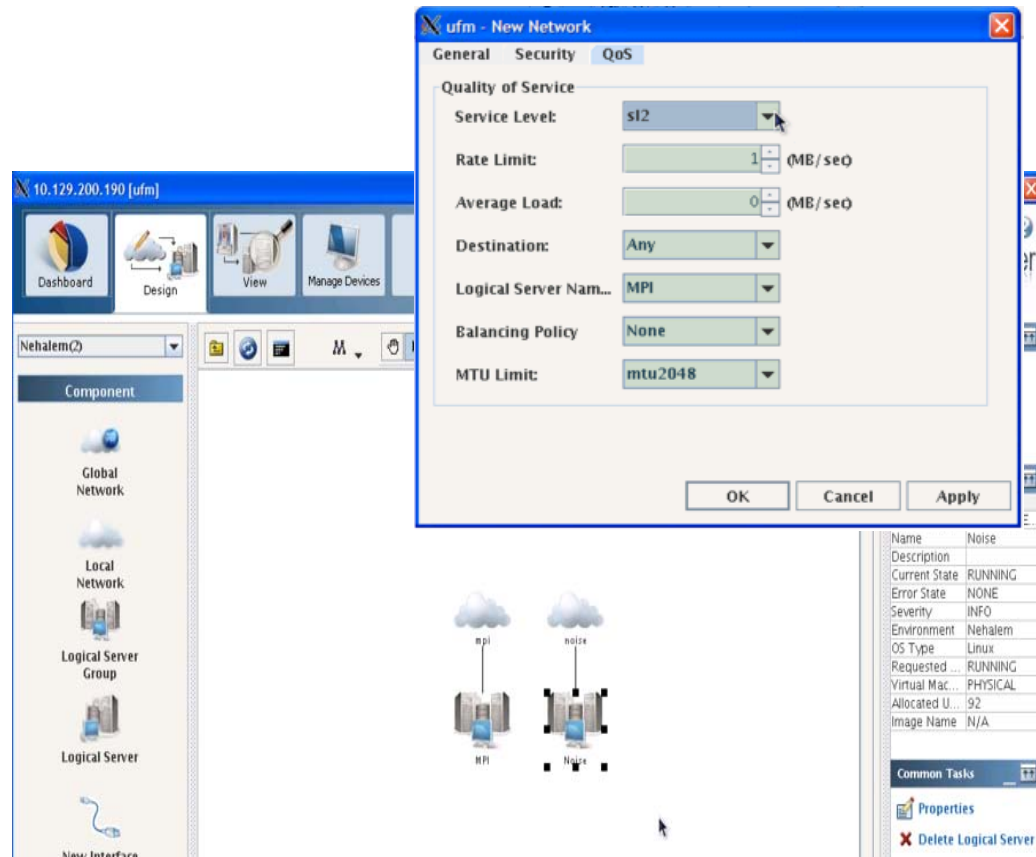
UFM Logical Model Creates Partition and Sets QoS

▶ 2 Logical Groups

- Latency job
- B/W oriented job

▶ QoS settings

▶ UFM creates virtual NICs, partitions and assigns Service Levels on the fabric



The screenshot displays the UFM (Unified Fabric Manager) interface. The main window shows a navigation pane on the left with components like Global Network, Local Network, Logical Server Group, and Logical Server. The main area shows a network diagram with two logical servers, 'mpi' and 'noise', connected to a central fabric. A 'ufm - New Network' dialog box is open, showing the 'QoS' tab. The 'Quality of Service' section includes the following settings:

- Service Level: sl2
- Rate Limit: 1 (MB/seo)
- Average Load: 0 (MB/seo)
- Destination: Any
- Logical Server Name: MPI
- Balancing Policy: None
- MTU Limit: mtu2048

At the bottom right, a properties table is visible:

Name	Noise
Description	
Current State	RUNNING
Error State	NONE
Severity	INFO
Environment	Nehalem
OS Type	Linux
Requested	RUNNING
Virtual Mac...	PHYSICAL
Allocated U...	92
Image Name	N/A

Below the table are 'Common Tasks' including 'Properties' and 'Delete Logical Server'.

With UFM QoS

Cross Application Interference fixed

Single job in cluster
(Latency = 2.1us)

```
hwwnec5 @ cl3fr0 /nfs/home6/nec/hwwnec5
avg/min/max: 0.000207 0.000177 0.000330
avg/min/max: 0.000207 0.000177 0.000328
avg/min/max: 0.000208 0.000178 0.000337
avg/min/max: 0.000208 0.000177 0.000389
avg/min/max: 0.000208 0.000177 0.000343
avg/min/max: 0.000208 0.000178 0.000362
avg/min/max: 0.000205 0.000177 0.000342
avg/min/max: 0.000207 0.000177 0.000353
avg/min/max: 0.000206 0.000178 0.000347
avg/min/max: 0.000207 0.000177 0.000357
avg/min/max: 0.000209 0.000177 0.000358
avg/min/max: 0.000206 0.000177 0.000350
avg/min/max: 0.000206 0.000177 0.000356
avg/min/max: 0.000206 0.000177 0.000353
avg/min/max: 0.000206 0.000177 0.000359
avg/min/max: 0.000209 0.000177 0.000361
avg/min/max: 0.000209 0.000177 0.000364
avg/min/max: 0.000212 0.000178 0.000364
avg/min/max: 0.000210 0.000177 0.000364
avg/min/max: 0.000212 0.000177 0.000366
avg/min/max: 0.000213 0.000177 0.000361
avg/min/max: 0.000214 0.000177 0.000375
avg/min/max: 0.000214 0.000177 0.000369
```

2nd job added
(Latency = 4.5us)

```
hwwnec5 @ cl3fr0 /nfs/home6/nec/hwwnec5
avg/min/max: 0.000222 0.000177 0.000408
avg/min/max: 0.000221 0.000177 0.000405
avg/min/max: 0.000217 0.000177 0.000414
avg/min/max: 0.000221 0.000177 0.000421
avg/min/max: 0.000221 0.000177 0.000397
avg/min/max: 0.000219 0.000177 0.000415
avg/min/max: 0.000221 0.000177 0.000407
avg/min/max: 0.000319 0.000177 0.000593
avg/min/max: 0.000447 0.000178 0.000624
avg/min/max: 0.000480 0.000272 0.000628
avg/min/max: 0.000456 0.000178 0.000633
avg/min/max: 0.000479 0.000306 0.000655
avg/min/max: 0.000476 0.000303 0.000657
avg/min/max: 0.000479 0.000290 0.000644
avg/min/max: 0.000444 0.000178 0.000643
avg/min/max: 0.000488 0.000301 0.000606
avg/min/max: 0.000487 0.000310 0.000631
avg/min/max: 0.000487 0.000324 0.000665
avg/min/max: 0.000484 0.000317 0.000631
avg/min/max: 0.000483 0.000332 0.000688
avg/min/max: 0.000481 0.000327 0.000636
avg/min/max: 0.000480 0.000323 0.000694
avg/min/max: 0.000424 0.000177 0.000626
```

2 jobs, UFM optimization
(Latency = 2.2us)

```
hwwnec5 @ cl3fr0 /nfs/home6/nec/hwwnec5
avg/min/max: 0.000227 0.000180 0.000403
avg/min/max: 0.000230 0.000180 0.000403
avg/min/max: 0.000227 0.000179 0.000407
avg/min/max: 0.000228 0.000180 0.000402
avg/min/max: 0.000230 0.000179 0.000412
avg/min/max: 0.000230 0.000180 0.000437
avg/min/max: 0.000222 0.000179 0.000408
avg/min/max: 0.000229 0.000180 0.000413
avg/min/max: 0.000226 0.000180 0.000455
avg/min/max: 0.000223 0.000180 0.000411
avg/min/max: 0.000217 0.000180 0.000411
avg/min/max: 0.000214 0.000180 0.000392
avg/min/max: 0.000214 0.000179 0.000382
avg/min/max: 0.000216 0.000178 0.000378
avg/min/max: 0.000217 0.000179 0.000347
avg/min/max: 0.000220 0.000179 0.000304
avg/min/max: 0.000220 0.000180 0.000305
avg/min/max: 0.000219 0.000178 0.000303
avg/min/max: 0.000219 0.000180 0.000305
avg/min/max: 0.000225 0.000180 0.000424
avg/min/max: 0.000223 0.000178 0.000306
avg/min/max: 0.000222 0.000180 0.000306
avg/min/max: 0.000227 0.000179 0.000308
```

100% Better Performance Through QoS Implementation

Optimize performance #2: routing

▶ Existing routing algorithms

- Are not aware of application communication flow
- They distribute paths evenly across the fabric links

▶ In real life, fabrics have non uniform usage

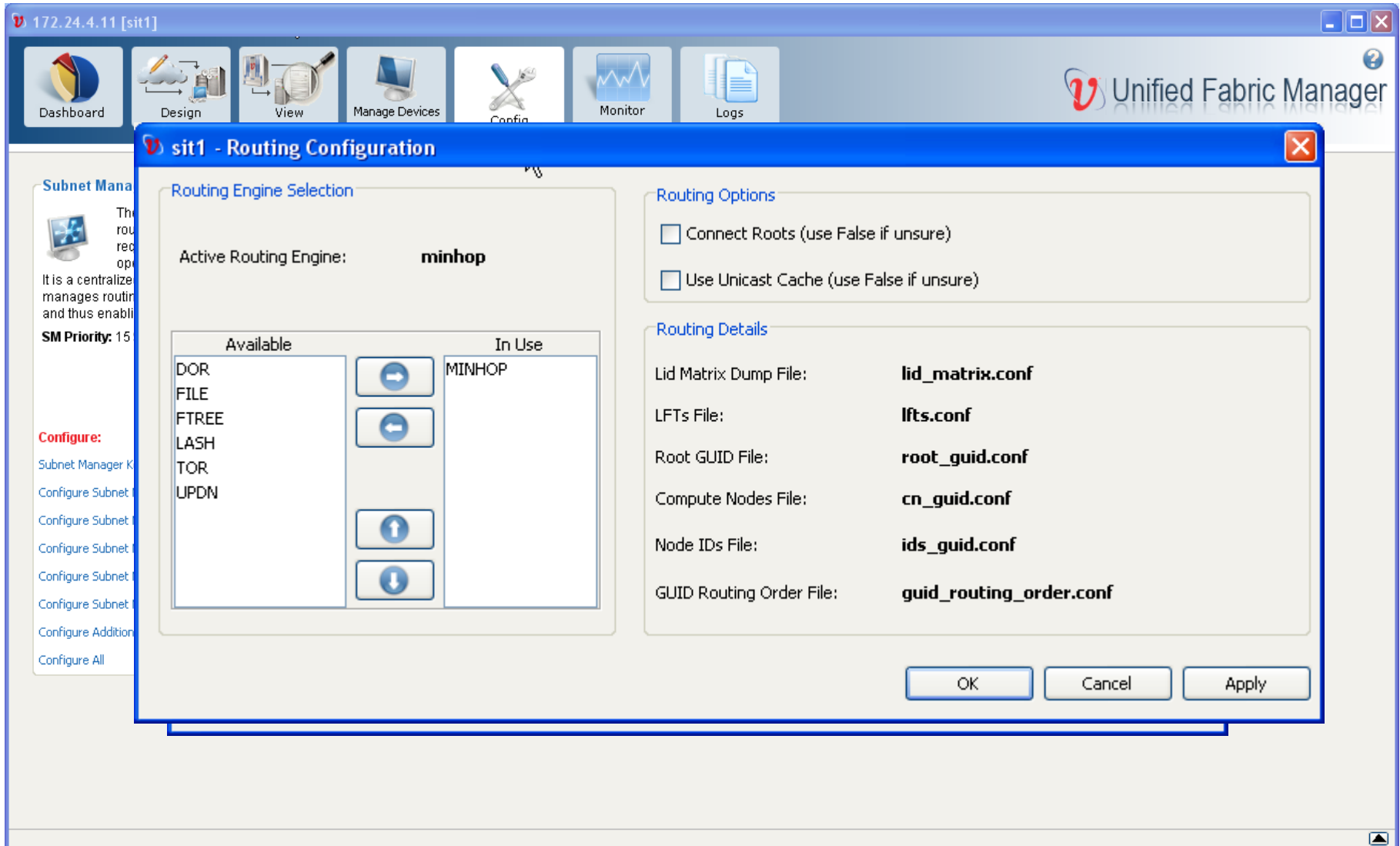
- Some endpoints “talk” a lot, some don’t “talk” at all
- Many-to-many (cluster) and any-to-many (storage) topologies

▶ Result

- Unbalanced fabric
- Congestion is created leading to slower performance and high latency

Congestion = Latency

UFM2.0 - Fabric Management



The screenshot shows the Unified Fabric Manager (UFM2.0) interface. The main window title is '172.24.4.11 [sit1]'. The top navigation bar includes icons for Dashboard, Design, View, Manage Devices, Config, Monitor, and Logs. The 'Unified Fabric Manager' logo is visible in the top right corner.

The 'sit1 - Routing Configuration' dialog box is open, displaying the following configuration details:

Routing Engine Selection

Active Routing Engine: **minhop**

Available		In Use
DOR	➔	MINHOP
FILE	➔	
FTREE	➔	
LASH	⬆	
TOR	⬆	
UPDN	⬆	

Routing Options

- Connect Roots (use False if unsure)
- Use Unicast Cache (use False if unsure)

Routing Details

- Lid Matrix Dump File: **lid_matrix.conf**
- LFTs File: **lfts.conf**
- Root GUID File: **root_guid.conf**
- Compute Nodes File: **cn_guid.conf**
- Node IDs File: **ids_guid.conf**
- GUID Routing Order File: **guid_routing_order.conf**

Buttons: OK, Cancel, Apply

TARA (TOR) Optimization

▶ TARA provides the following benefits:

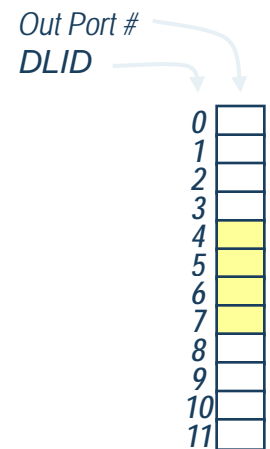
- Reduces competition between fabric resources, thus decreasing congestion
- Increases available bandwidth, resulting in improved fabric utilization
- Delivers lower latency and shorter application runtime

▶ How ?

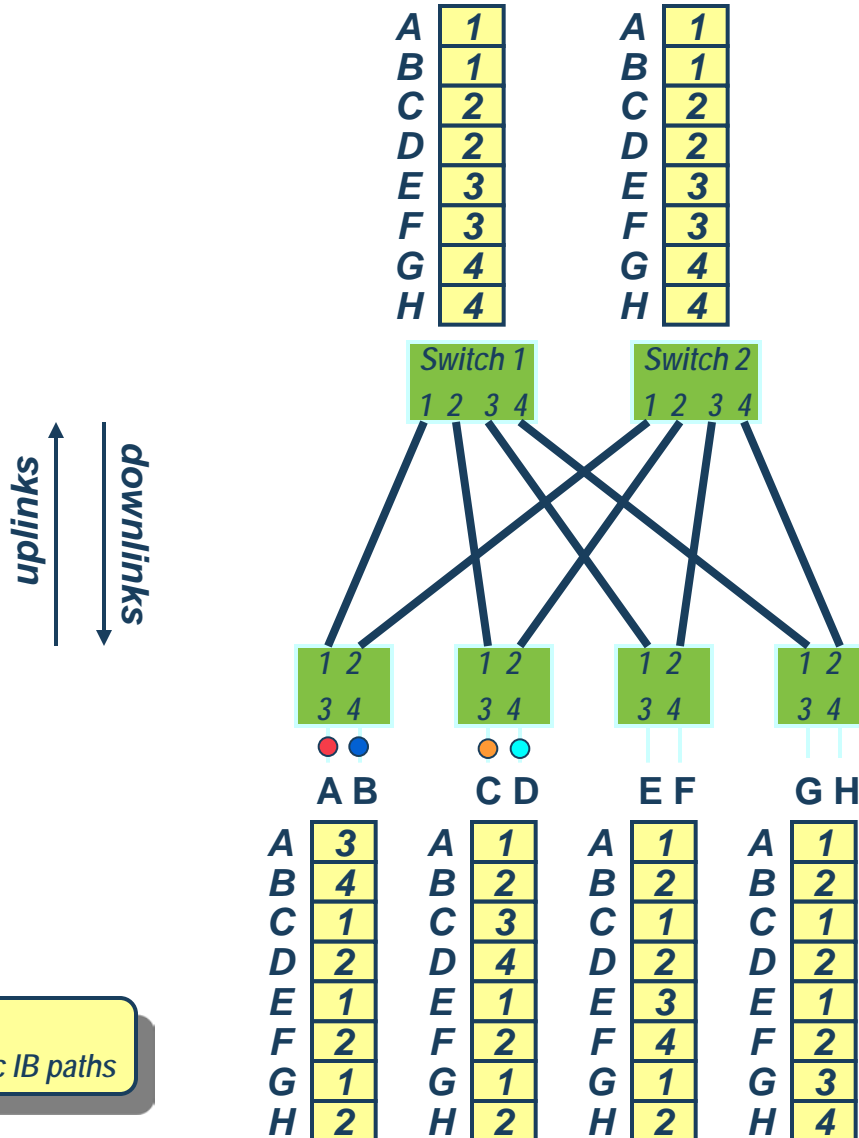
- Uses knowledge of cluster usage: logical servers, networks.
- Balances routes depending on usage
- Not based on real-time analysis of bandwidth / congestion

Routing ?

- ▶ **InfiniBand packets are ‘destination routed’ based on the Destination Logical ID (DLID) field in the header**
- ▶ **In IB: DLID=route (not only remote address)**
- ▶ **DLIDs are 16 bits**
 - 48K values are used for unicast
 - 16K values are used for multicast
- ▶ **At each switch ASIC, the incoming unicast DLID is used as an index into a Linear Forwarding Table (LFT) that returns the outgoing switch port number**
 - E.g. the InfiniScale III ASIC supports all 48K possible LFT entries

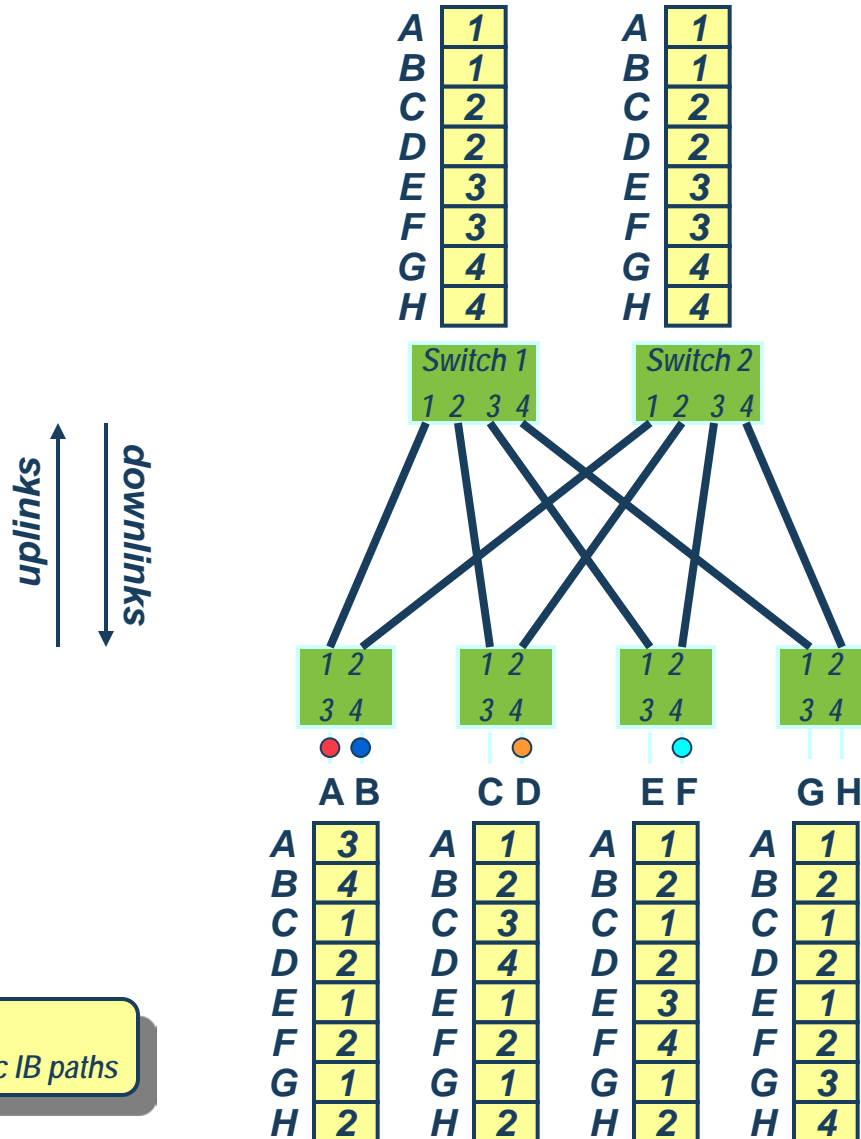


Communication Patterns (balanced)



- Communication pattern:
 - A-E
 - B-H
 - C-G
 - D-F
- No link contention

Communication Patterns (un-balanced)



- Communication pattern:
 - A-C ●
 - B-E ●
 - D-G ●
 - F-H ●
- 2:1 link contention:
 - A->C and B->E share uplink to Switch 1 port 1
 - G->D and H->F share uplink to Switch 2 port 4

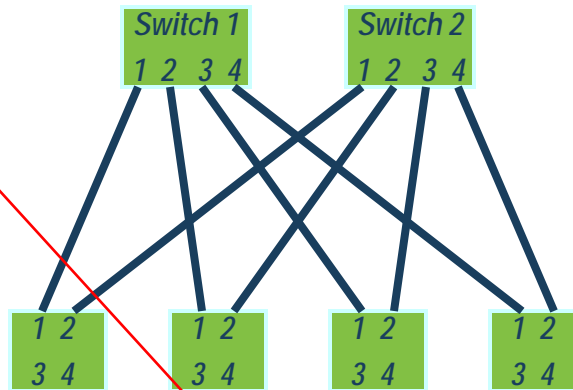
Communication Patterns with TARA

TARA makes local changes to balance routes per job

uplinks
downlinks

A	1
B	1
C	2
D	2
E	3
F	3
G	4
H	4

A	1
B	1
C	2
D	2
E	3
F	3
G	4
H	4



●	●
●	●
●	●
●	●
●	●
●	●
●	●
●	●

A	3
B	4
C	1
D	2
E	2
F	2
G	1
H	2

A	1
B	2
C	3
D	4
E	1
F	2
G	1
H	2

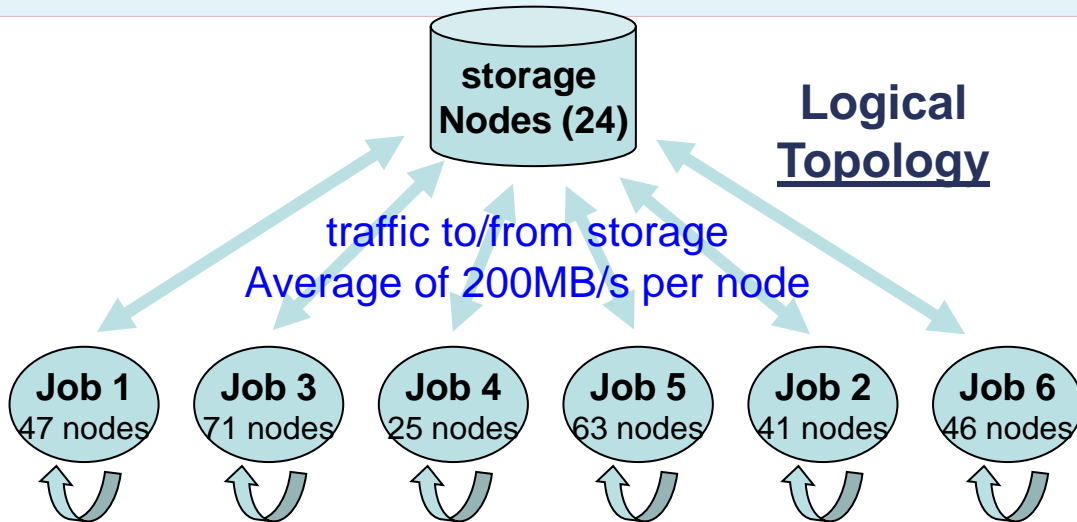
A	1
B	2
C	1
D	2
E	3
F	4
G	1
H	2

A	1
B	2
C	1
D	2
E	1
F	1
G	3
H	4

.....> IB path
↔ 2 symmetric IB paths

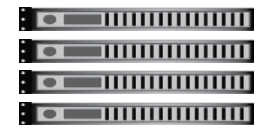
- Communication pattern:
 - A-C-A ●
 - B-E-B ●
 - D-G-D ●
 - F-H-F ●
- 2:1 link contention:
 - A->C and B->E share uplink to Switch 1 port 1
 - G->D and H->F share uplink to Switch 2 port 4

Example: TARA with 324 nodes cluster



Physical Topology

300 servers

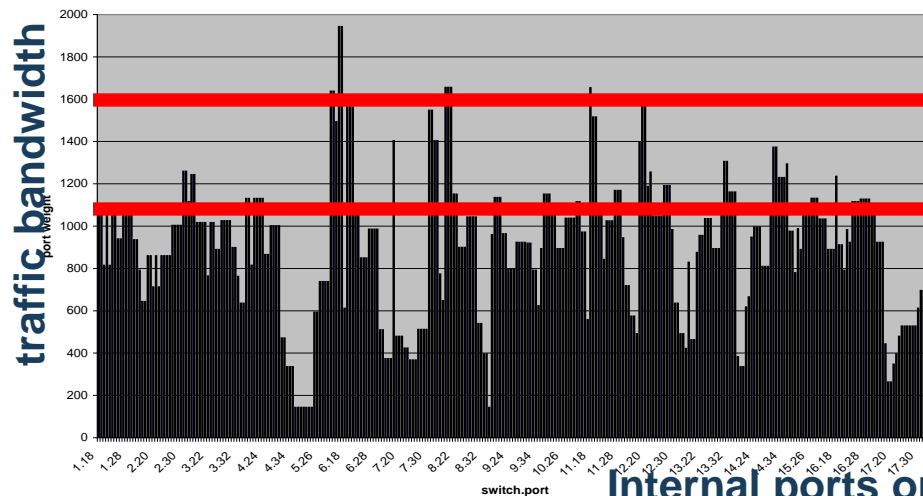


24 storage nodes

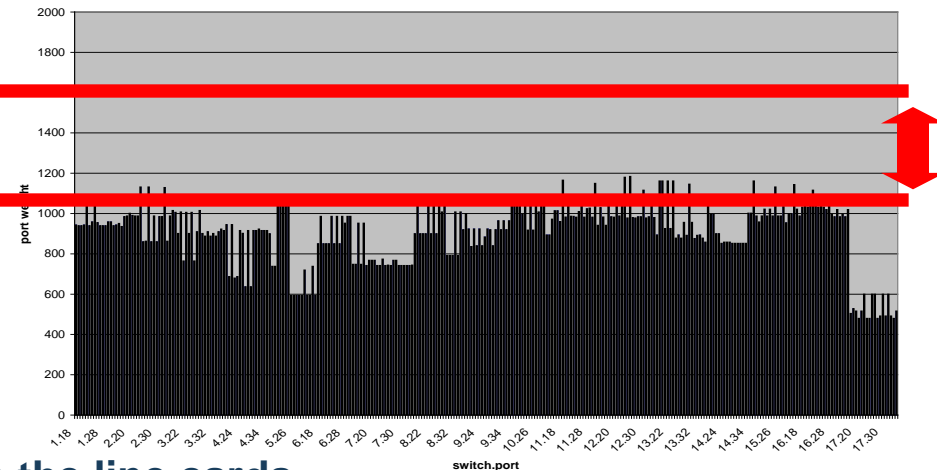


Internal traffic inside each job, 1000 MB/s from each node

OpenSM



Traffic Optimized Routing

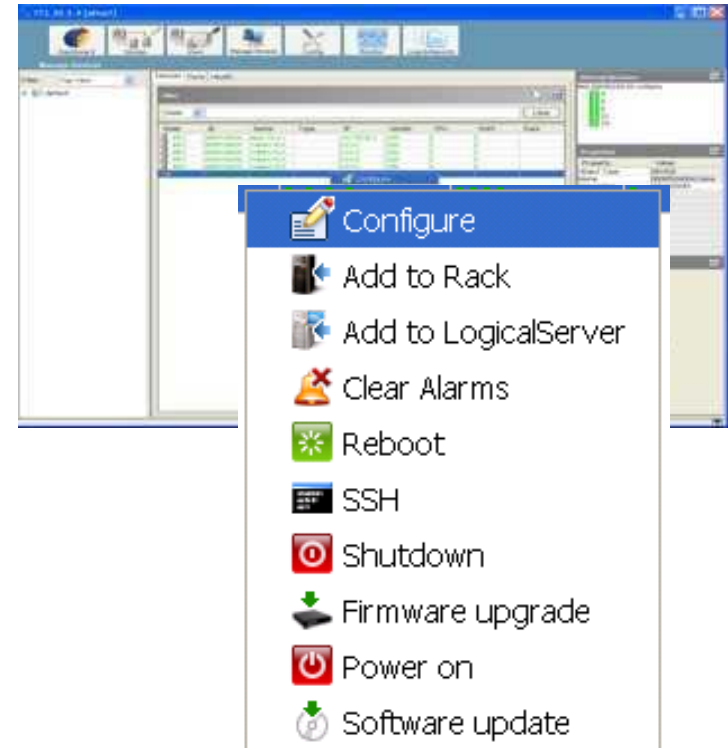


Internal ports on the line cards

Confidential - Internal

Scale-out and Maintain Control on Fabric

- ▶ **Dozens of switches and 1000s of nodes become a massive operational burden**
- ▶ **UFM automates I/O and switch configuration enabling isolation and QoS**
- ▶ **Central Device Management for switches and hosts**
- ▶ **High-availability and seamless failover of SM and UFM**
- ▶ **Advanced API for seamless integration in existing environments**

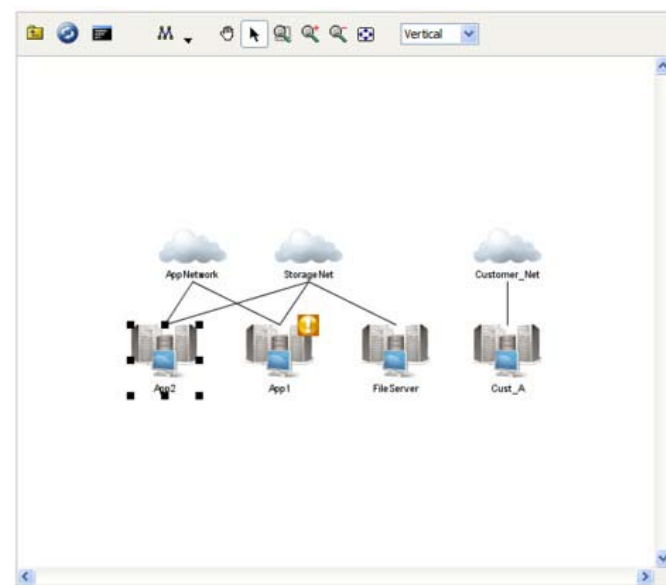


Automatic, seamless operations save hours of configuration and set-up work

Efficient Troubleshooting

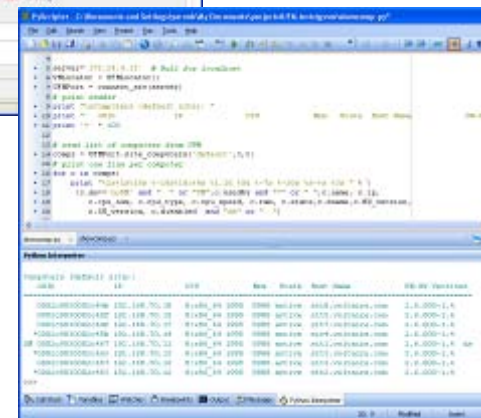
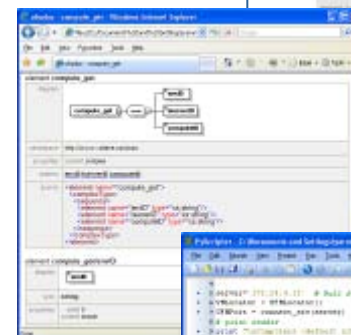
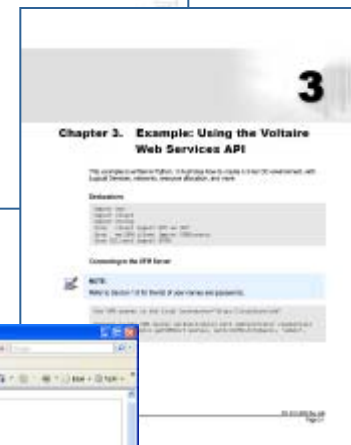
- ▶ **Dozens of traffic and health events**
 - Easy central drill-down to counters, alerts and events to the port level
- ▶ **Configurable thresholds and criticality levels**
- ▶ **GUI and log level alarms**
- ▶ **Alerts correlated to the application level**
- ▶ **Alerts correlated to the DC rack level**

Alarm	Log File	Threshold	TTL(Sec)	Severity
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	200	300	Minor
<input type="checkbox"/>	<input checked="" type="checkbox"/>	1	300	Info
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	100	300	Minor
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	50	300	Warning
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	200	300	Minor

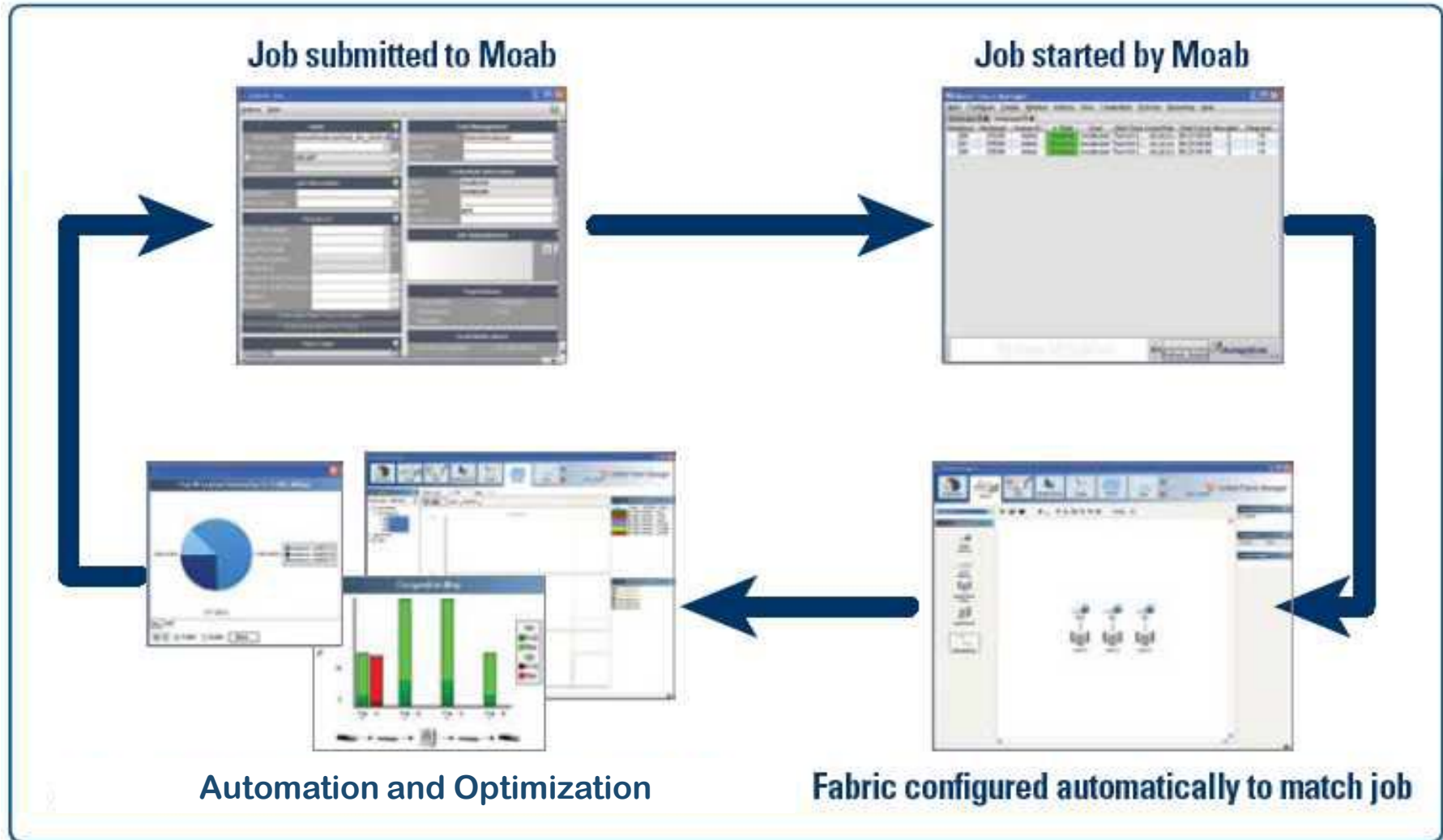


Open system

- ▶ Extensible architecture based on Web-services
- ▶ Open API for users or 3rd party extensions
- ▶ Expose entire fabric and datacenter object model
- ▶ Allow simple reporting, provisioning, monitoring, and task automation
- ▶ Tools already benefiting from UFM API
 - Scheduler integration (e.g. Moab)
 - UFM Support tool kit
 - Various command line tools/extensions to UFM
 - Web fabric portal
 - * Provided in UFM Advanced packages



Integration with Moab – how does it work ?



UFM and Moab Adaptive HPC Suite Integrated Solution

UFM Benefits

Fabrics w/o UFM

Little Fabric Visibility

*Unnoticed performance degradation
Difficult to assess impact*

Ineffective Troubleshooting

*Long troubleshooting time
Performance issues take days to analyze*

Complex and Manual Processes

*Needs admin skills
Many options left unused at all*

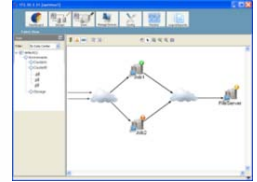
Low Performing Unutilized Fabrics

*Arbitrary routing algorithms, QoS seldom implemented
Congested fabrics, latency affected*

UFM Customers

In-Depth Visibility and Control

*Clear health and performance visualization
Business oriented impact and root analysis*



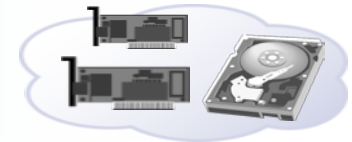
Quick Issue Resolution

*Dashboard, Alarms, Congestion Map
Reduces downtime, high fabric utilization*



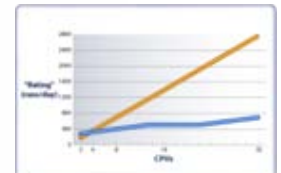
Simple and Automated

*Lowers administration tasks
time from days to minutes*



Increased Performance

*Reduce congestion, lower latency
Quicker application runtime*



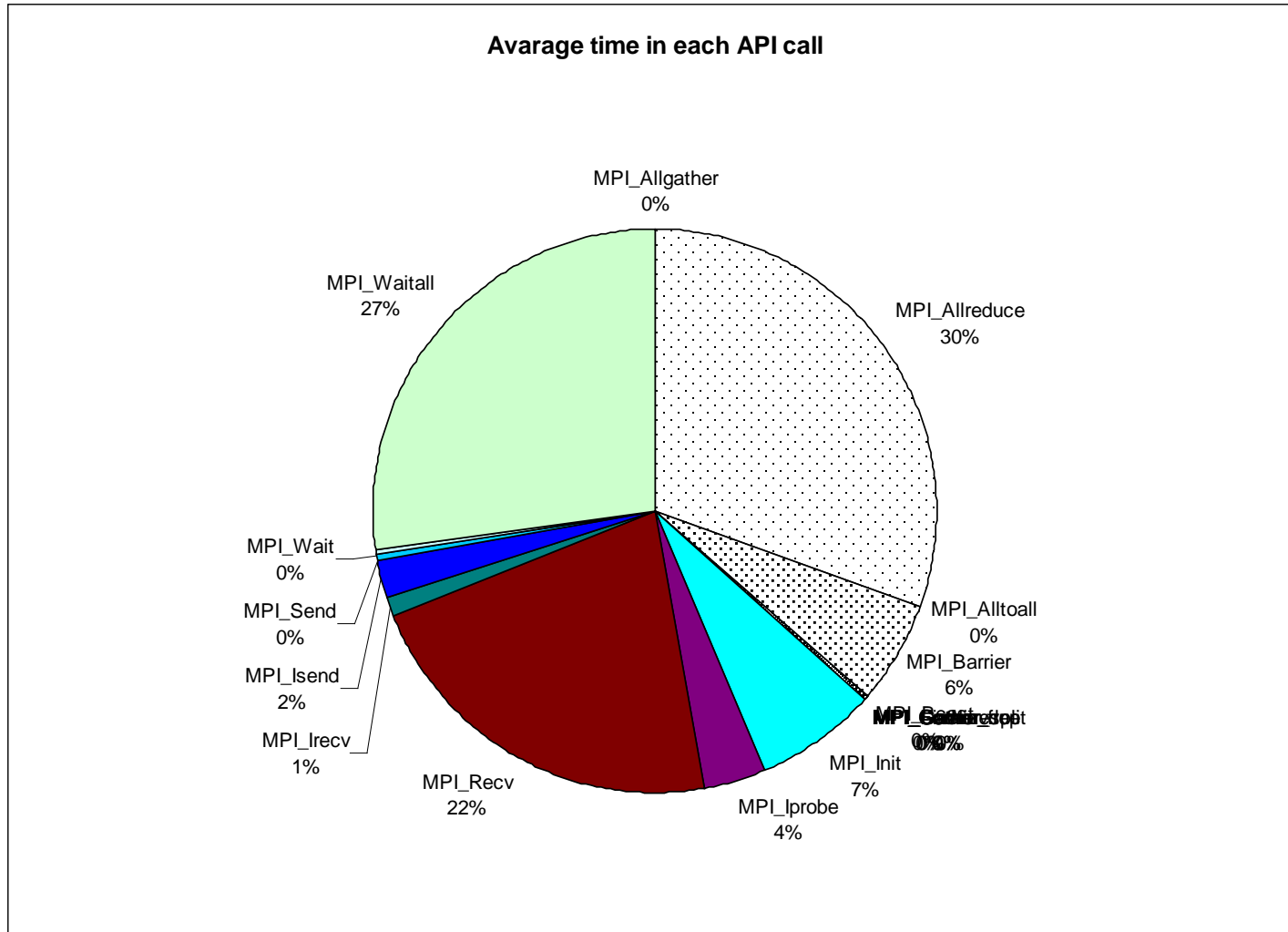


Fabric Collective Accelerator

Ghislain de Jacquilot

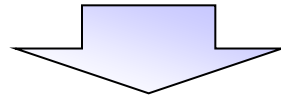
March 22, 2010

Fluent (eddy_417k): 32 Ranks MPI API calls, % of time ranks spent in each call



The challenge: Collective Operations performance

- ▶ **Collective operations takes large chunk of the application run time and don't scale**
 - Examples: OpenFoam spends 33% of its MPI time in Allreduce, large scale SAGE spends 55% of its MPI time in Allreduce
- ▶ **System/OS “noise” affects on scalability**
- ▶ **Simple offload solutions DON'T address the key problems:**
 - Huge network congestion due to “All-to-All” communication
 - Computation & messaging performance
 - Difficult to manage and orchestrate

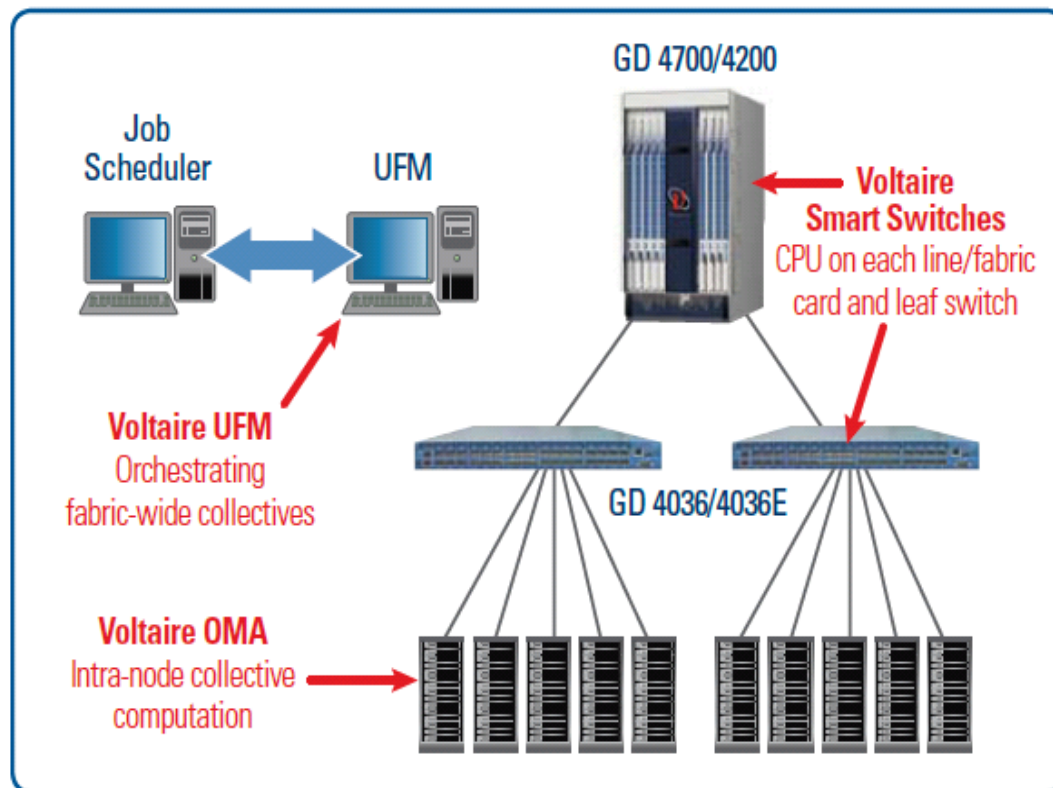


Poor application scalability and low cluster efficiency

What is Voltaire FCA ?

First Fully Integrated solution to offload collectives combine intelligence on server, switches, and management

- UFM™ Automate fabric collective offload/monitoring and integrate with schedulers
- Voltaire “smart” switch based CPUs performing reduction and messaging operation
- Voltaire OMA (Open MPI plug-in) address server side

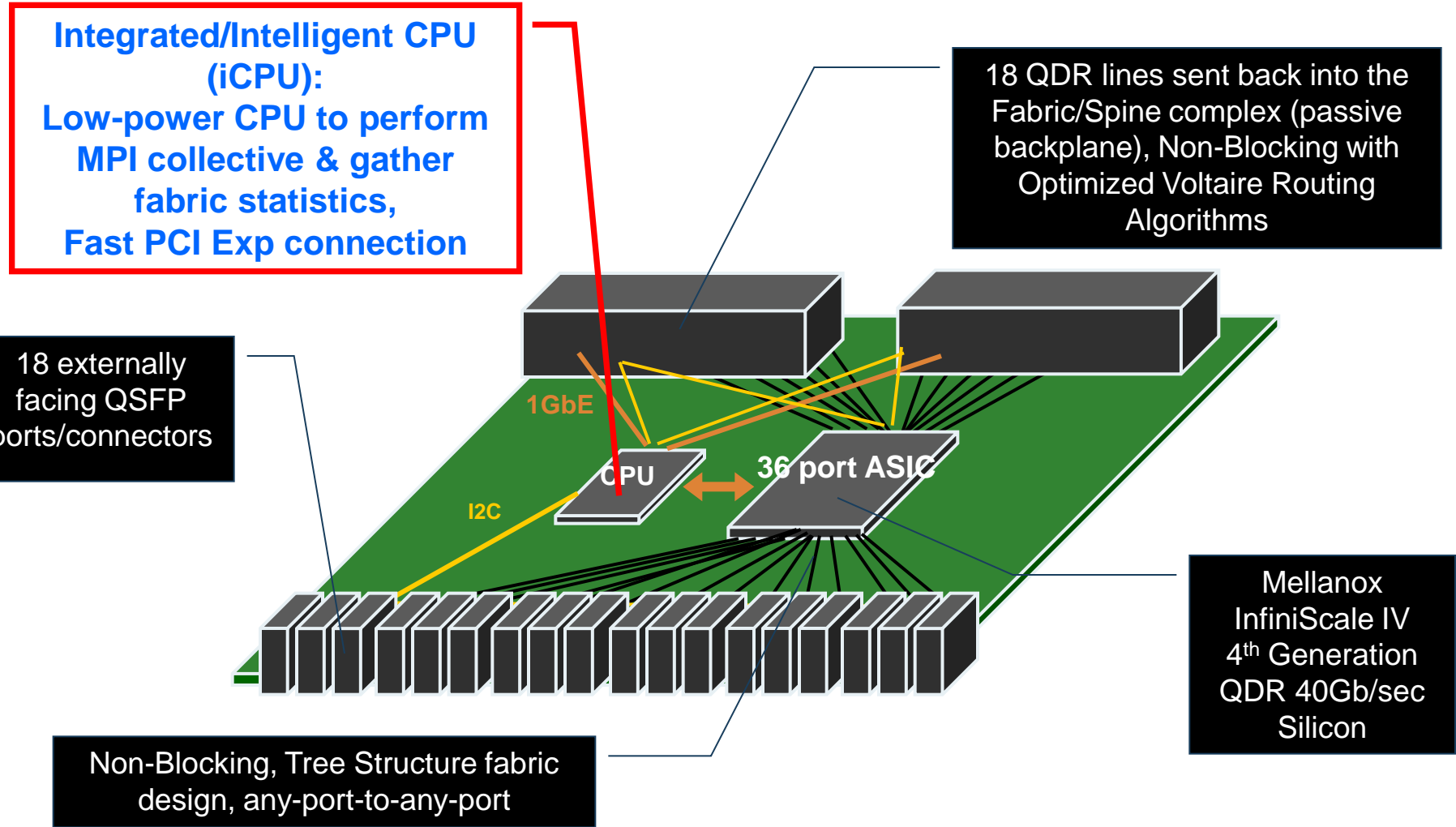


Voltaire FCA Building Blocks

- ▶ **UFM™ - Smart communicator map creation:**
 - Full job scheduler integration
 - Topology aware
 - Balance loads between fabric elements
 - Consider compute & messaging capabilities
 - Ensures single message per physical wire for any collective operation
 - Separate VL buffering and QoS
- ▶ **Switch hardware integrated CPU (iCPU) performing computation & messaging**
- ▶ **Voltaire OMA (Open MPI Plug-in) - node collective acceleration and offloading to iCPU**
- ▶ **Broadcast communication is managed with Multicast**
 - Coupled with reliable protocol to reduce and overcome loss

Voltaire Grid Director 4700

Modular 18-Port QDR Line Card Architecture



Voltaire FCA and “Smart” products

Addressing The Problem End To End

► Increase performance, reduce congestion

- Reduce fabric traffic to single message per wire, dramatically reduce congestion
- FCA offload “shield” collective operation from node “noise”
- Enable non-blocking collective (overlap communication and calculation)
- Linear scalability to many thousands of nodes with predictable Hardware performance

► Simple, Fully integrated

- No change in application – OMA drop-in Open MPI plug-in
- Switches come equipped with FCA offload out of the box
- UFM automate the process and integrate with scheduler, saving huge setup burden
- Fully integrated monitoring capabilities

► FCA Reduced collective operations runtime by up to 100X

- **11K nodes MPI collective operations within 25 usec**

FCA – Collective Operation

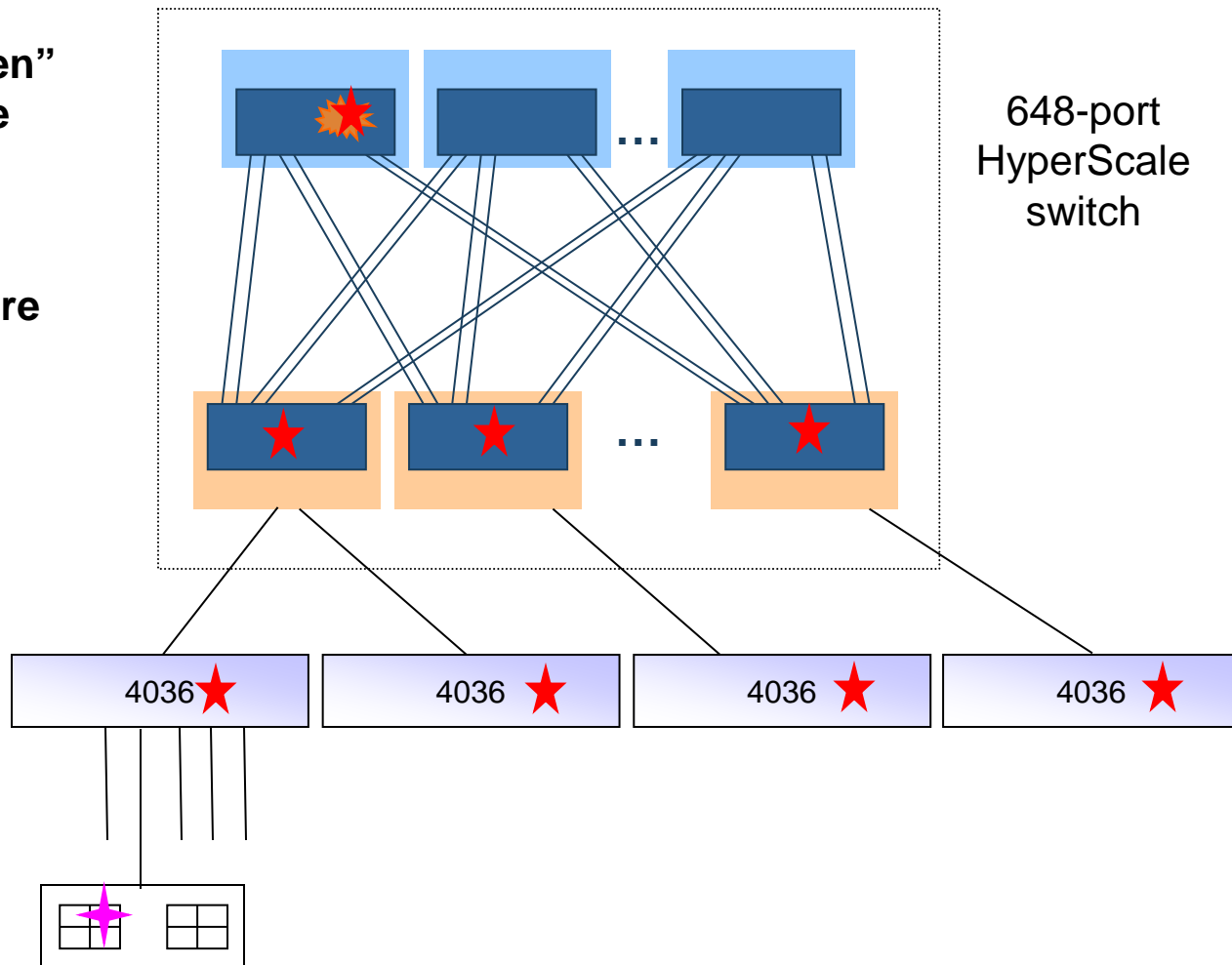
FCA Initialization & Configuration - FMM

1. Select collective root & "tree"

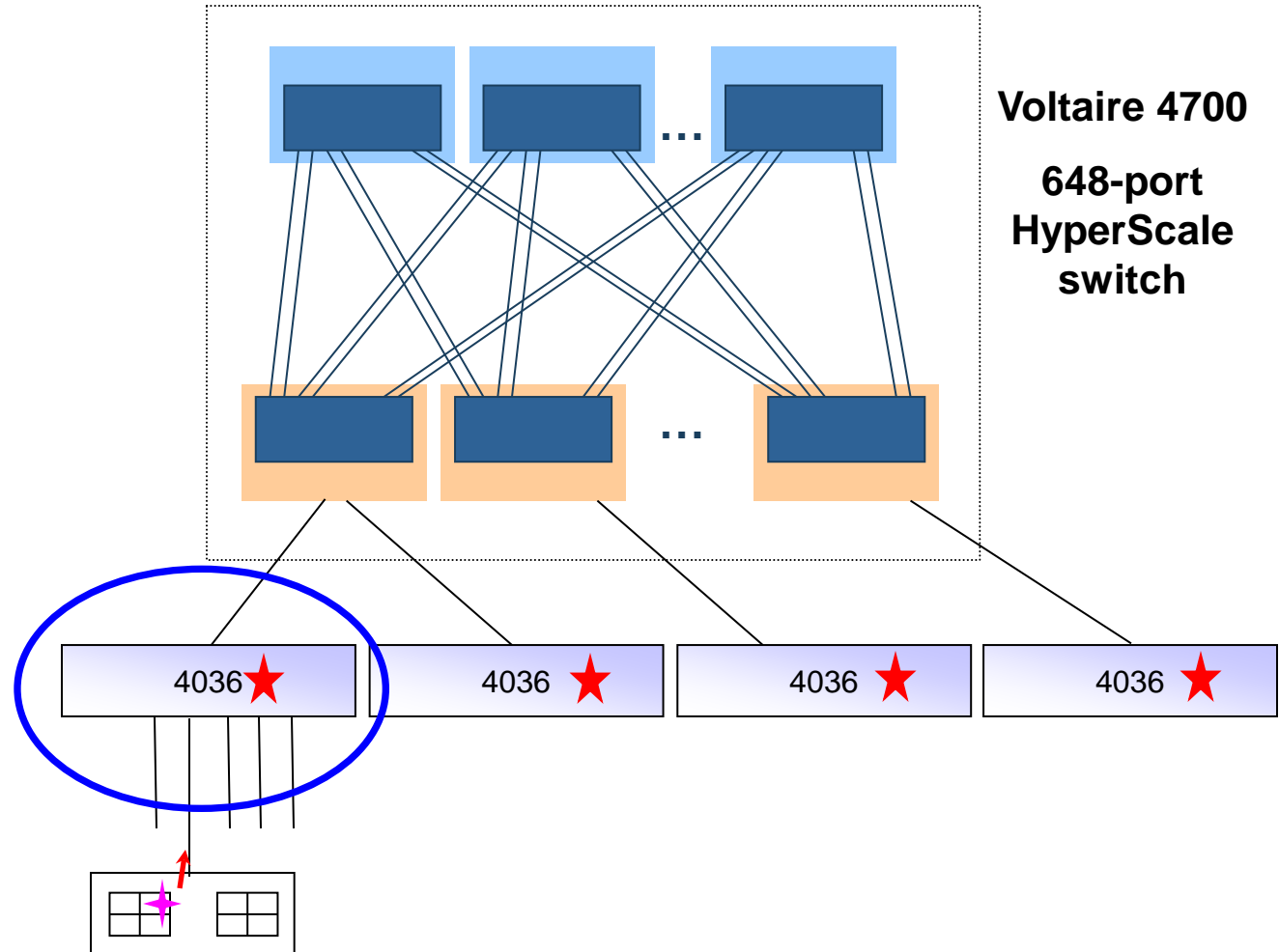
2. Create "parent & children" logical "tree" on top of the physical connections

3. Instruct UFM to configure multicast group for all elements

4. Configure all element information: "parent & children"



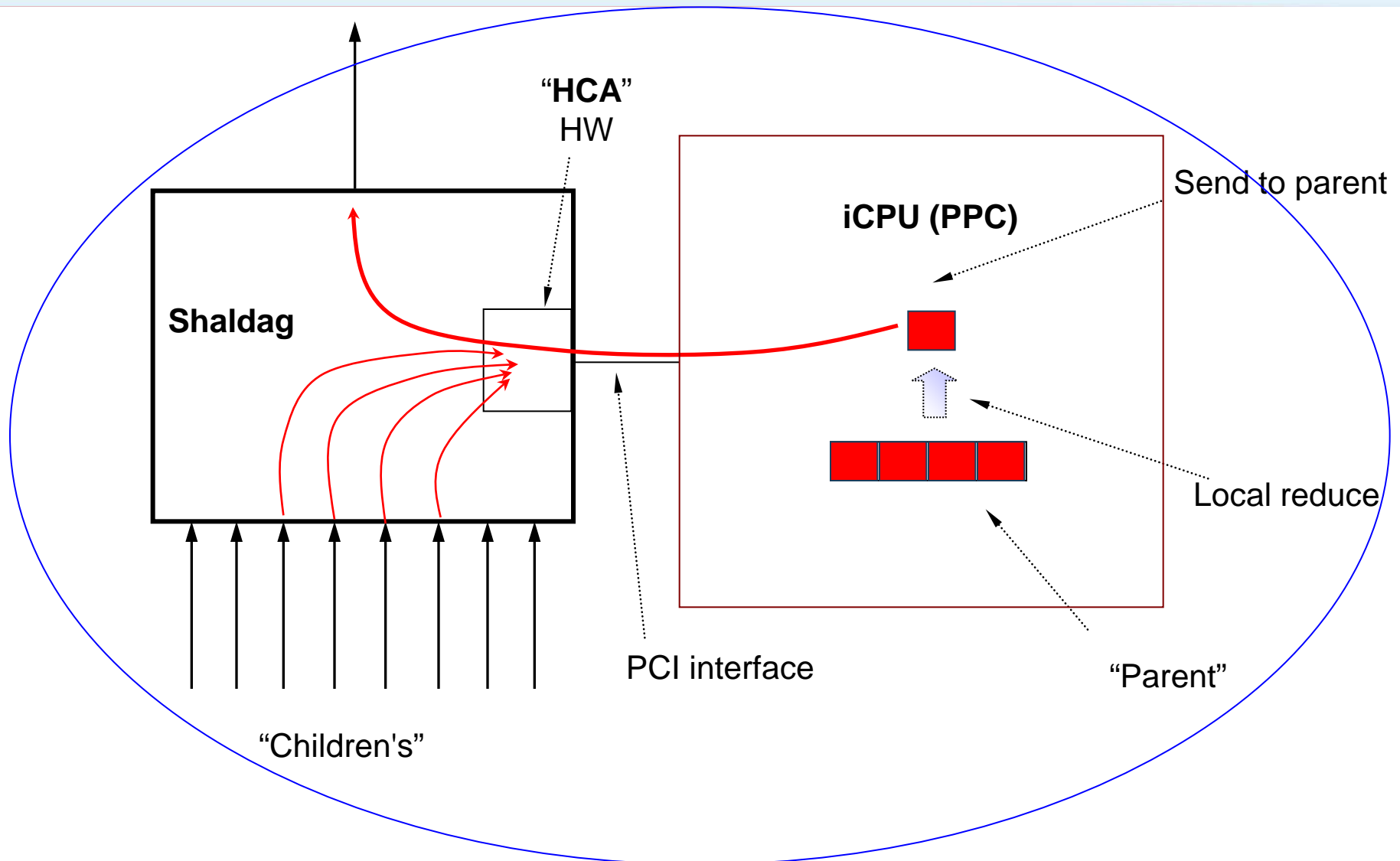
FCA Allreduce flow



2. iCPU collective offload

1. Node reduce & Send reduce message to parent

FCA – Switch iCPU operations



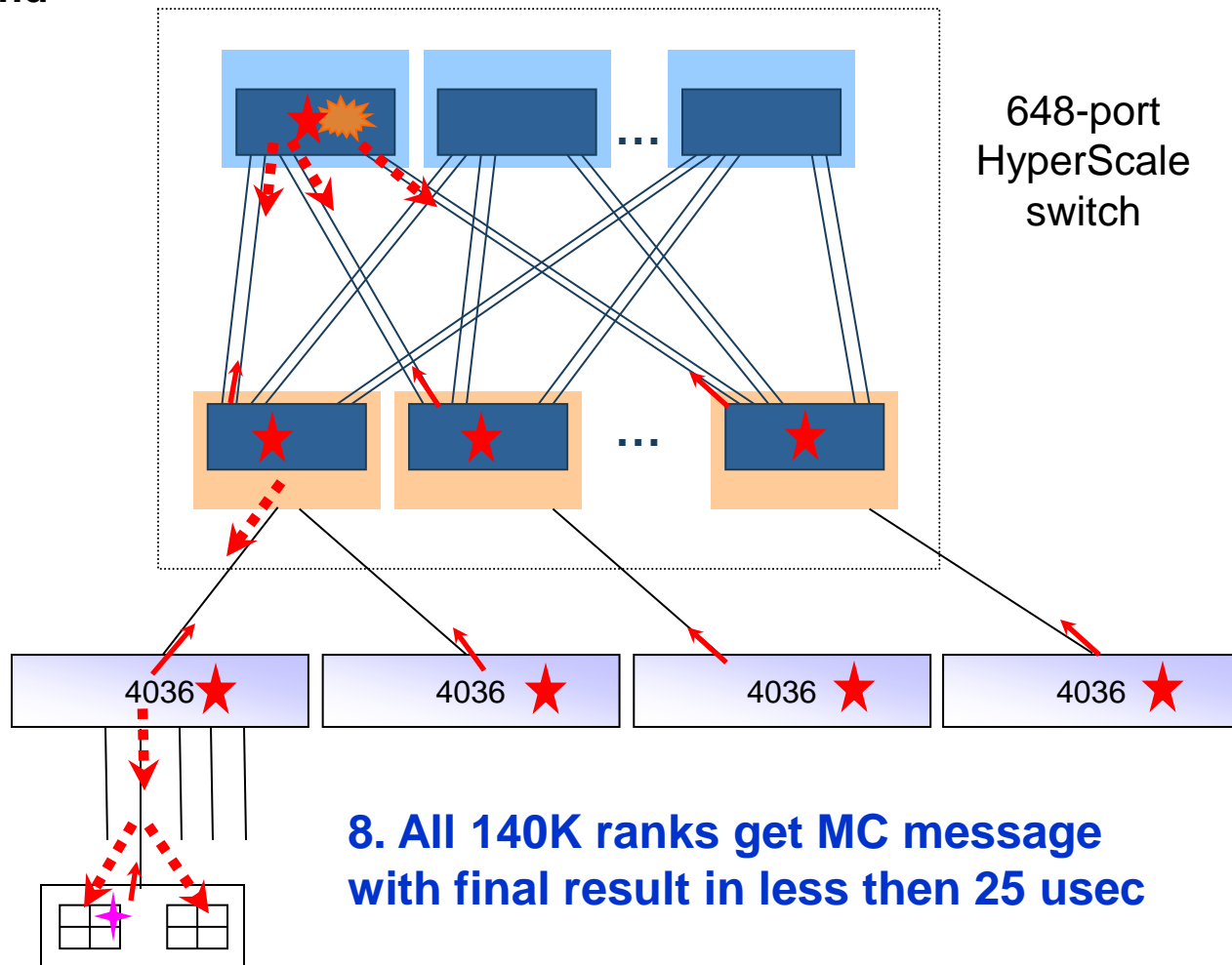
FCA Allreduce flow

1. Select collective root (part of setup)

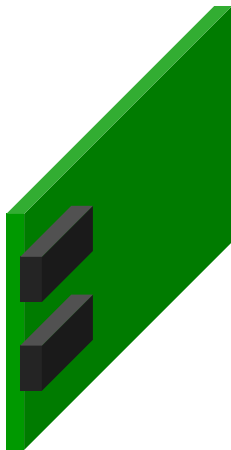
7. Have final result - iCPU send multicast

6. iCPU (master root) reduce all messages

5. iCPU reduce 18 nodes & forward reduce message

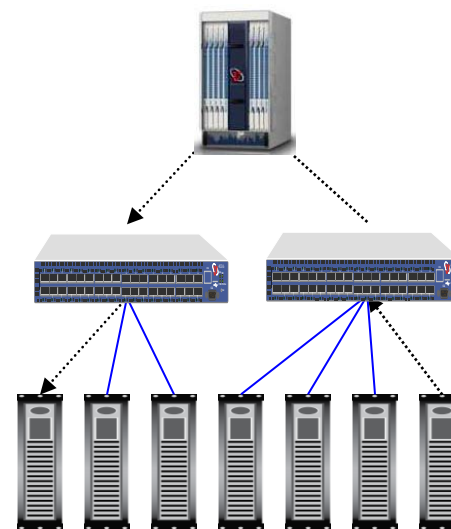


Comparison means of Collective Offload



▶ NIC based offload

- Address OS noise only
- Modest improvement of performance
- Network congestion due to “All-to-All” communication

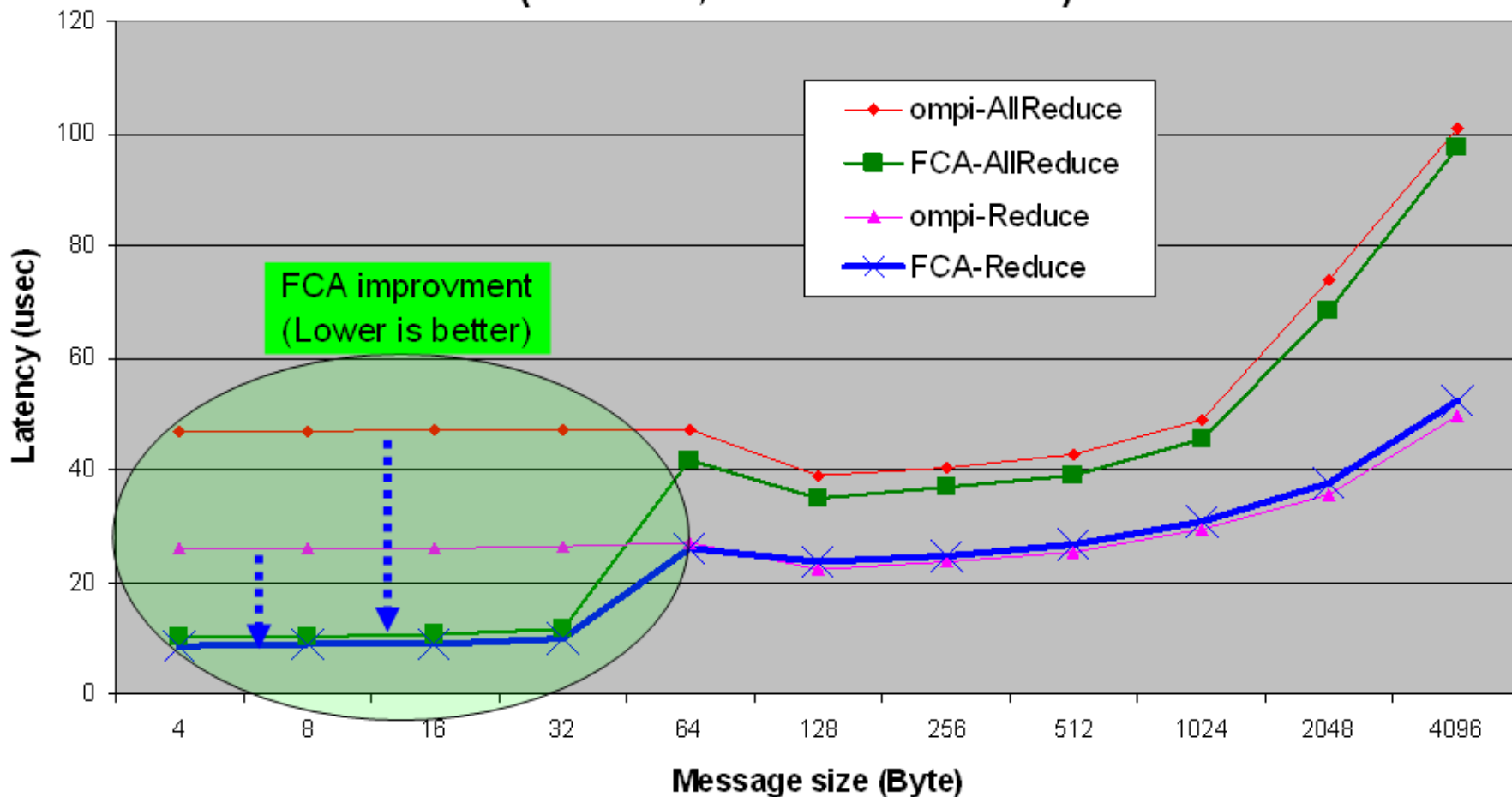


▶ Fabric based offload

- Drastic reduced run time
- Adds on top of reduced OS noise
- Reduced fabric congestion
- Central management – topology and settings

Measured performance benefits

IMB Allreduce and Reduce results for ompi and FCA
(40 ranks, 5 x Nehalem 5520)



FCA results on 512 ranks

	OpenMPI	Voltaire FCA	% improve
Allreduce	145	22	85%
Reduce	57	21	64%
Barrier	139	22	84%

