Node-type-based load-balancing routing for Parallel Generalized Fat Trees

John Gliksberg (john.gliksberg@uvsq.fr)
Jean-Noël Quintin (jean-noel.quintin@atos.net)
Pedro Javier García (pedro-javier.garcia@uclm.es)

Atos (BuLL) & UVSQ & UCLM

HiPINEB (IEEE HPCA) 2018, Vienna
Saturday February 24th, 2018

Outline

Outline

Improving routing algorithms for fat trees

Outline

Outline

Context

Parallel Generalized Fat Trees (PGFTs)

Formula defines uplinks, downlinks and duplicate links for each level
$PGFT(n; w_0, \ldots, w_{n-1}; u_0, \ldots, u_{n-1}; p_0, \ldots, p_{n-1})$

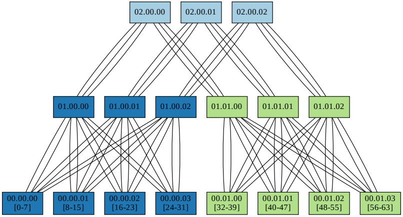


Figure: Example $PGFT(3; 8, 4, 2; 1, 3, 2; 1, 2, 3)$

Context

Parallel Generalized Fat Trees (PGFTs)

Benefits:

Context

Parallel Generalized Fat Trees (PGFTs)

Benefits:

- Deadlock-free routing algorithms

Context

Parallel Generalized Fat Trees (PGFTs)

Benefits:

- Deadlock-free routing algorithms
- Low-radix switches

Context

Parallel Generalized Fat Trees (PGFTs)

Benefits:

- Deadlock-free routing algorithms
- Low-radix switches
- High cross-bisectional bandwidth to number of switches ratio when pruning upper levels

Context

Parallel Generalized Fat Trees (PGFTs)

Benefits:

- Deadlock-free routing algorithms
- Low-radix switches
- High cross-bisectional bandwidth to number of switches ratio when pruning upper levels
- Fault tolerance

Outline

Context

Random routing for PGFTs

- Choice among up-down shortest paths

Context

Random routing for PGFTs

- Choice among up-down shortest paths
- Works, deadlock free

Context

Random routing for PGFTs

- Choice among up-down shortest paths
- Works, deadlock free
- Uses all available resources

Context

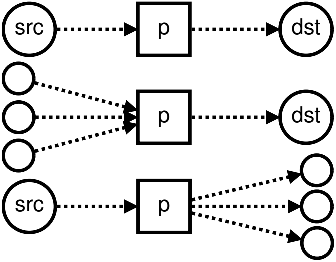
Random routing for PGFTs

- Choice among up-down shortest paths
- Works, deadlock free
- Uses all available resources
- Never perfect, frequent congestion

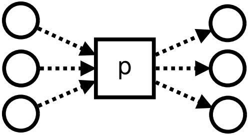
Context

Random routing for PGFTs

Congestion metric: $\min(src, dst)$



(a) Minimal congestion



(b) Non-minimal congestion

Context

Random routing for PGFTs

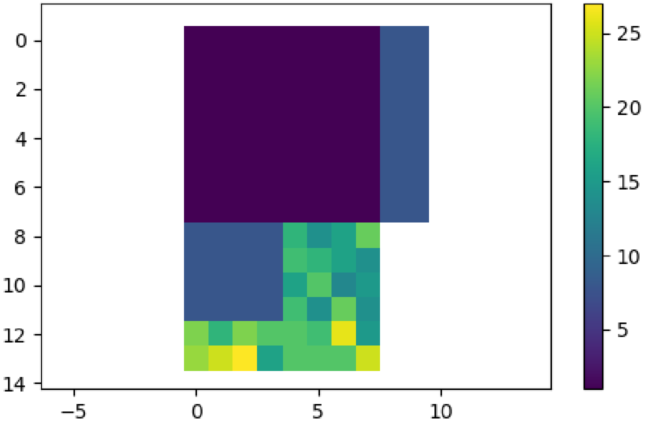


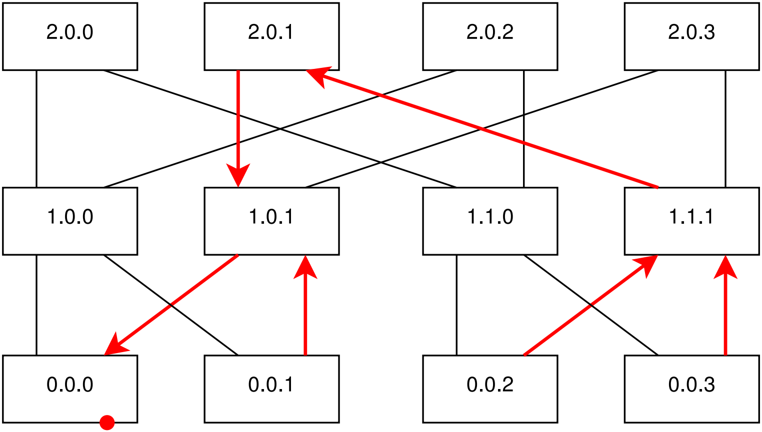
Figure: Congestion metric example for random routing (under all-to-all traffic)

Outline

- Context
  - Parallel Generalized Fat Trees (PGFTs)
  - Random routing for PGFTs
  - Dmodk routing
  - Smodk routing
- Heteregoneous clusters
  - Periodicity
- Reindexing
- Conclusions

Context

Dmodk routing



Context

Dmodk routing

- Deterministic function

Context

Dmodk routing

- Deterministic function
- Coalesce routes to the same destination

Context

Dmodk routing

- Deterministic function
- Coalesce routes to the same destination
- Lowest congestion metric for all-to-all traffic

Context

Dmodk routing

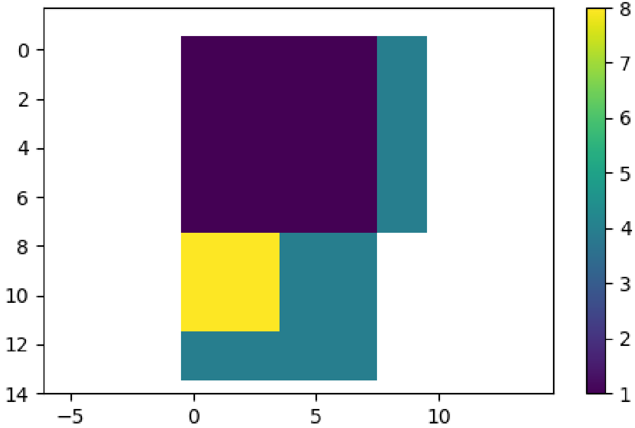


Figure: Congestion metric example for Dmodk (under all-to-all traffic)

Outline

Context

Smodk routing

- Coalesce routes from the same source

Context

Smodk routing

- Coalesce routes from the same source
- Similar to Dmodk for all-to-all

Context

Smodk routing

- Coalesce routes from the same source
- Similar to Dmodk for all-to-all
- Difference of congestion in asymmetric traffic

Context

Smodk routing

Few destinations and many sources: Dmodk will probably fare better

Few sources and many destinations: Smodk will probably fare better

Outline

Heteregoneous clusters

Actual traffic is not all-to-all but instead reflects usage of:

- Compute nodes
- I/O nodes
- Service nodes
- Management nodes
- FPGA, GPGPU nodes

Heteregoneous clusters

Existing algorithms do not take this into account
Oftentimes few ports are consistently congested, while the rest is underused.

Outline

Heteregoneous clusters

Periodicity

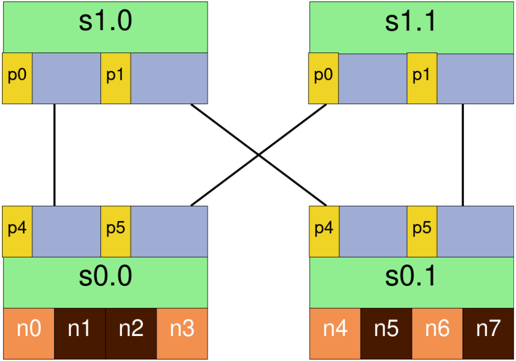


Figure: Example heterogenous topology;
s=switch, p=port, n=node; nodes 1,2,5,7 are I/O nodes

Heteregoneous clusters

Periodicity

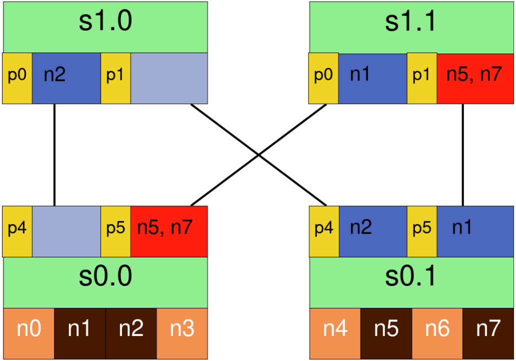


Figure: Dmodk periodicity collision example
s0.0p5 and s1.1p1 have $\min(src, dst) = 2$

Heteregoneous clusters

Periodicity

Two I/O nodes had NIDs which collided after modulo
($5 \bmod 2 = 7 \bmod 2$)

Heteregoneous clusters

Periodicity

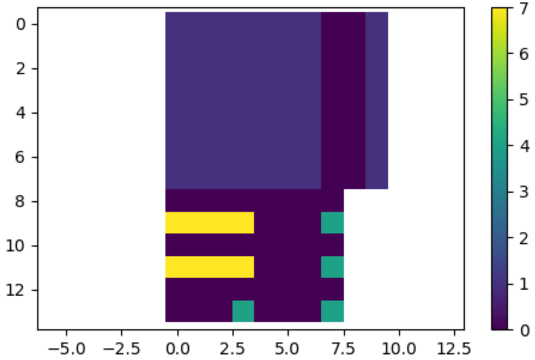


Figure: Dmodk congestion metrics on topology with all I/O nodes on periodic NIDs (under compute to I/O traffic)

Outline

Reindexing

Periodicity

Avoid periodicity issues by routing groups of nodes of the same type as if they were independent

In practice: inject grouped NIDs (gNIDs) instead of NIDs into existing Xmodk algorithms

Reindexing

Periodicity

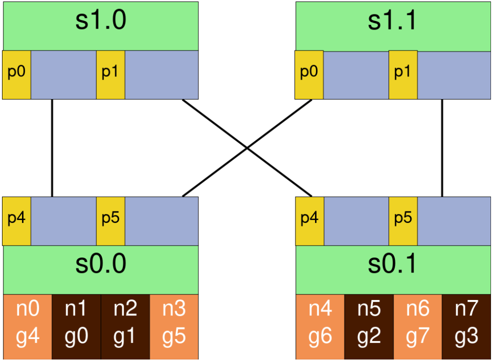


Figure: Example topology with reindexed gNIDs
gNIDS 0–3 are I/O nodes

Reindexing

Periodicity

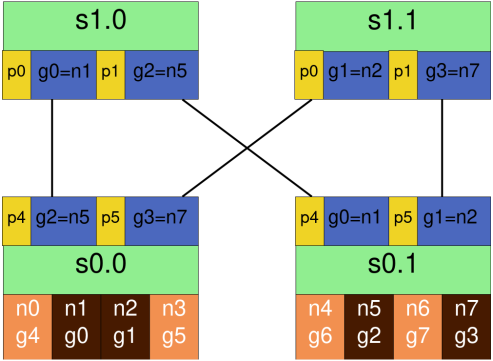


Figure: Example topology under Gdmodk routing with compute to I/O traffic

Reindexing

Periodicity

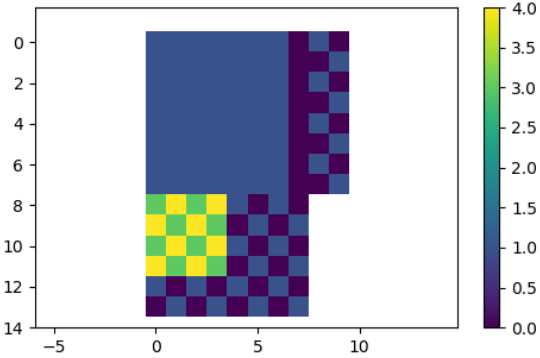


Figure: Gdmodk metrics on topology with all I/O nodes on periodic NIDs (under compute to I/O traffic)

Reindexing

Periodicity

Gsmodk behaves similarly, but symetrically

If the source set is close to all nodes, Gsmodk cannot improve much

Outline

Context
- Parallel Generalized Fat Trees (PGFTs)
- Random routing for PGFTs
- Dmodk routing
- Smodk routing

Heteregoneous clusters
- Periodicity

Reindexing

Conclusions

Conclusions

Cheap improvement (no hardware, little software)
Real-life experimentation?

Conclusions

Applicable to other routing algorithms

- If decisions rely on NIDs
- If resource allocation is mapped on NIDs

Conclusions

- Node type is good

Conclusions

- Node type is good
- Job placement is better!

Thank you
Any questions?