OPTIMIZED ROUTING AND PROCESS MAPPING FOR ARBITRARY NETWORK TOPOLOGIES

Torsten Hoefler

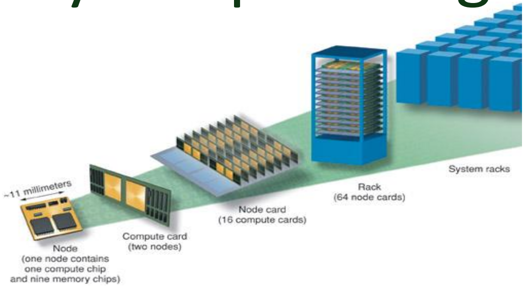
University of Illinois at Urbana-Champaign and ETH Zürich

Talk at Tokyo Institute of Technology, Tokyo, Japan

With Inputs from Jens Domke

MOTIVATION

- Scientific problems **require** more performance
- … which **requires** increased parallelism
- … which **requires** increased number of processing elements (PEs)
- … which **requires** a tightly-coupled larger network
  - On-chip
  - On-node
  - Off-chip



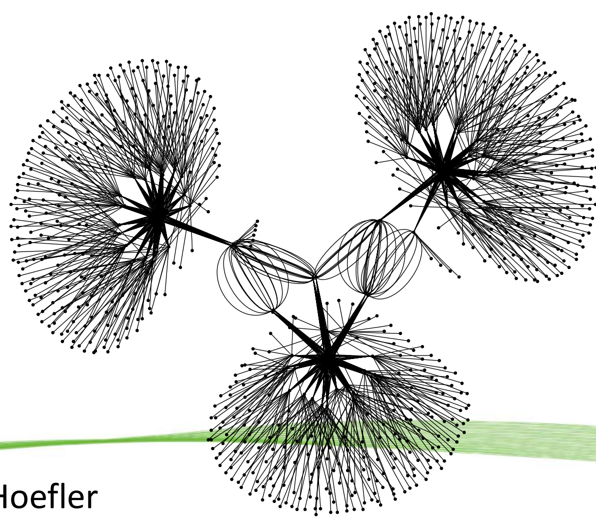
- → Supercomputing is at the fore-front of scalable networking (aka. "Formula 1 of Networking")

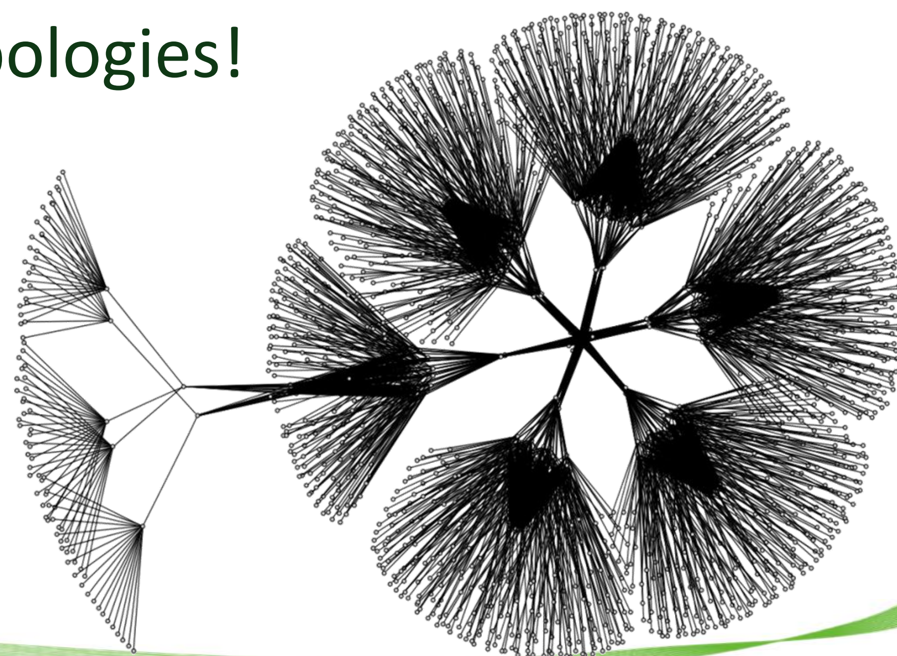
High Performance Networking?

- Important parameters:
  - Endpoint type (InfiniBand (IB), Ethernet, TOFU, …)
  - Topology (Fat Tree, Hypercube, Butterfly variants, …)
  - Routing Mode (static, dynamic, adaptive, …)
- We focus on (for now):
  - InfiniBand (easily available, tools are open source)
  - Routing (the most important variable at scale)
    - IB spec mandates static routing ☹
  - Arbitrary Topologies (next slide)

Why Arbitrary Topologies?

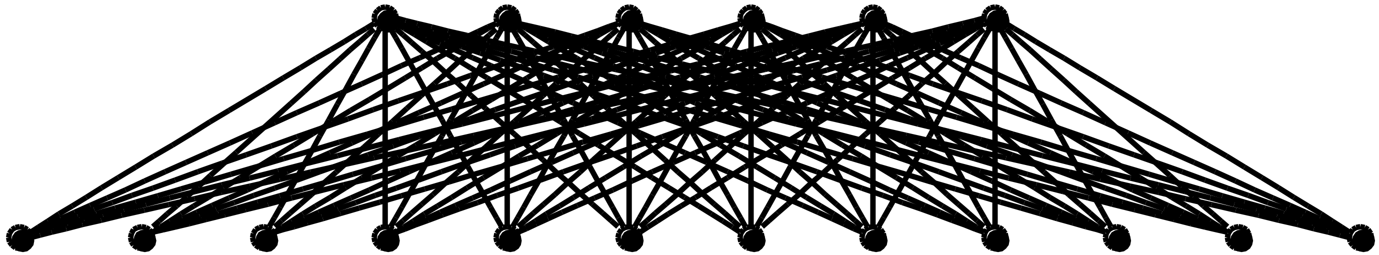
- Many networks grow over time or fulfill more than one purpose
  - Fat Trees and Butterflies are hard to grow
  - Tori networks may have undesirable properties
  - IB supports arbitrary topologies!
- Hybrid networks exist:





FORGET FULL BISECTION BANDWIDTH ☹

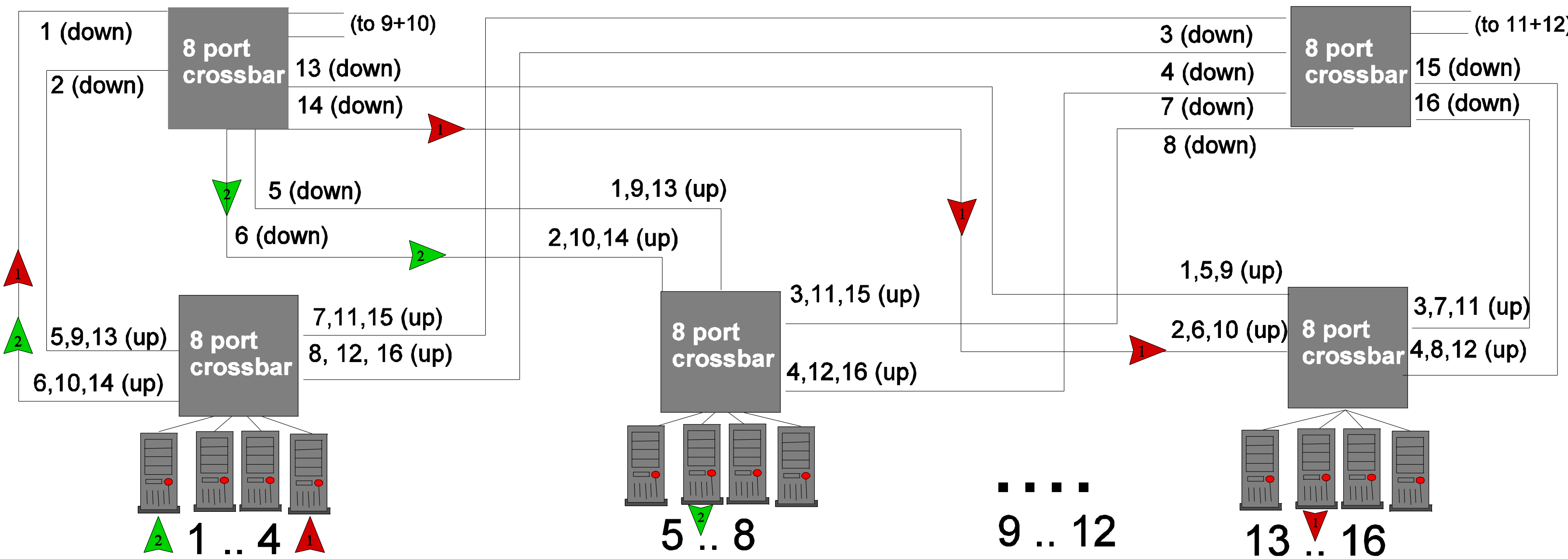
- expensive topologies do not guarantee high bandwidth
- deterministic oblivious routing cannot reach full bandwidth!
  - see Valiant's lower bound
  - random routing is asymptotically optimal but looses locality (see later)



- InfiniBand routing:
  - deterministic oblivious, destination-based, simple
  - linear forwarding table (LFT) at each switch
  - lid mask control (LMC) enables multiple addresses per port

Hoefler et al.: Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks

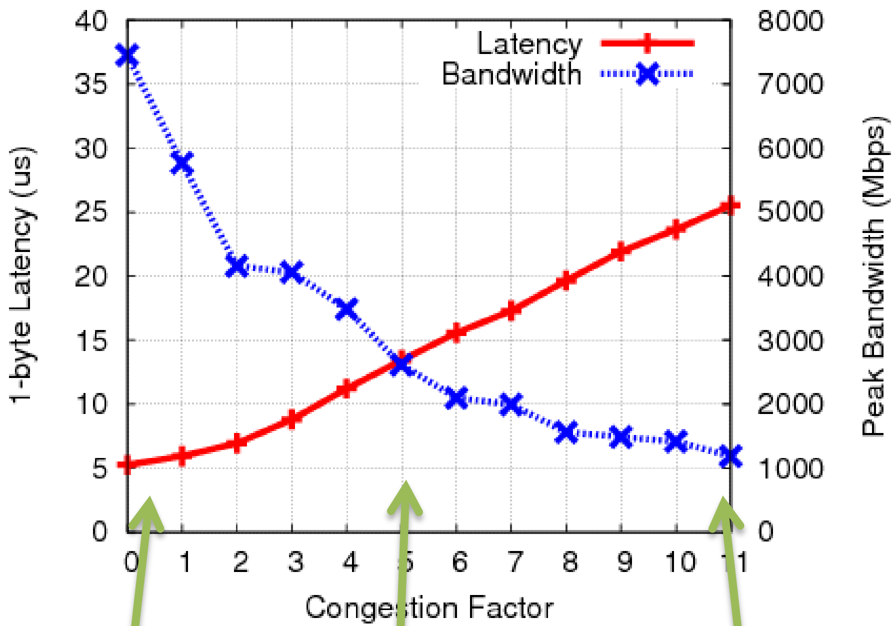
But My Vendor Said "Non-Blocking"



- Two communications 1→6, 4→14
- Full bisection bandwidth network
- No full bandwidth observed!

Hoefler et al.: Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks

So How Bad is Congestion?



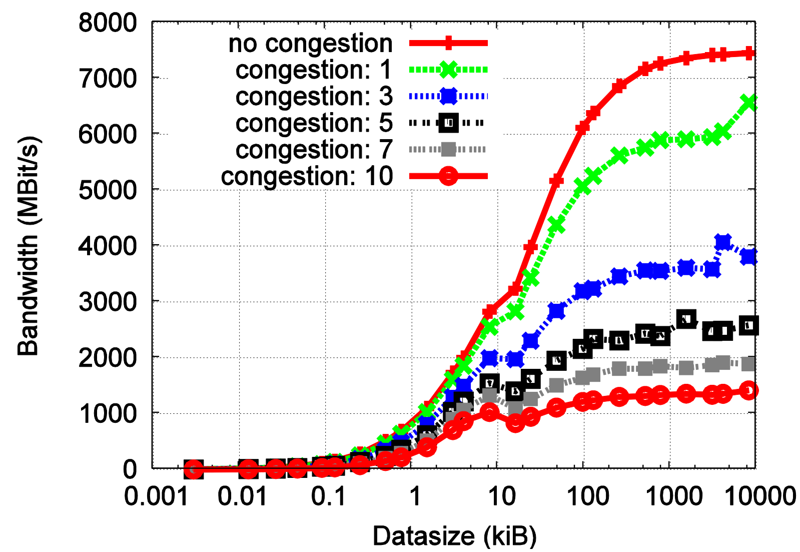
Microbenchmarks (yet to be seen in practice)

Reality?

Lower Bound! (~ GigE Speed)

CHiC Supercomputer:

- slightly aged but reflects routing
- 566 nodes, full bisection IB fat-tree
- **no endpoint congestion!**
- effective Bisection Bandwidth: 0.699



Hoefler et al.: Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks

But I have a Clever Subnet Manager!

- OpenSM (IB) routing algorithms:
  - MINHOP (finds minimal paths, balances number of routes local at each switch)
  - UPDN (uses Up*/Down* turn-control, limits choice but routes contain no credit loops)
  - FTREE (fat-tree optimized routing, no credit loops)
  - DOR (dimension order routing for k-ary n-cubes, might generate credit loops)
  - LASH (uses DOR and breaks credit-loops with virtual lanes)
- It's clever if you have a Fat Tree or a Torus
  - But beware if you add or remove one link!

T. Hoefler, T. Schneider and A. Lumsdaine: Optimized Routing for Large-Scale InfiniBand Networks

Effective Bisection Bandwidth

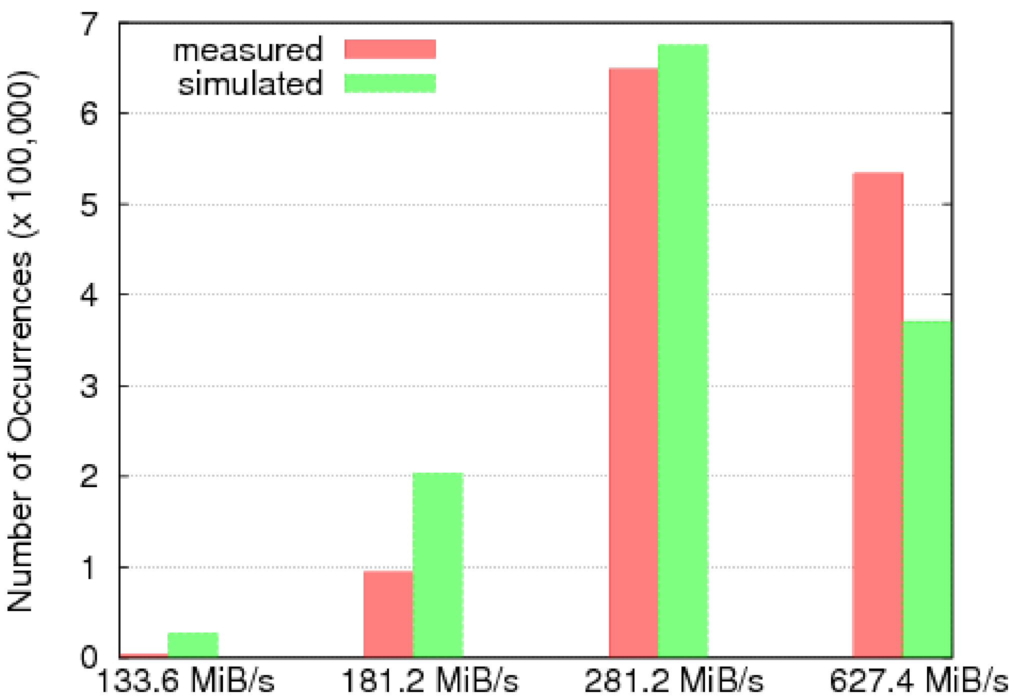
- A measure for global network performance
  - Considers routing! Can be measured with a benchmark!
  - More realistic then bisection bandwidth!
- Effective Bisection Bandwidth (eBB) Benchmark
  - Divide network into equal partitions A and B
    - $\binom{P}{\frac{P}{2}}$ combinations
  - Find one peer in B for each node in A
    - $\frac{P}{2}!$ pairings
  - Huge number of patterns
    - Statistics converge fast (~1000 measurements)
  - Implemented in Netgauge/eBB (download and try!)

Hoefler et al.: Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks

ORCS – A Routing eBB Simulator

- Routes large number of random eBB patterns
  - Count maximum congestion of each
  - Statistical analysis
  - Verified on Chic:
- Other systems

Computer	Nnodes	FBB	eBB
Ranger	3908	Full	57.5%
Atlas	1142	Full	55.6%
Thunderbird	4390	½	40.6%



Schneider, Hoefler, Lumsdaine : ORCS: An Oblivious Routing Congestion Simulator

Achieving High Bandwidth

- Model network as G=($V_P \cup V_C$, E)
- Path r(u,v) is a path between u,$v \in V_P$
- Routing R consists of P(P-1) paths
- Edge load l(e) = number of paths on $e \in E$
- Edge forwarding index π(G,R)=$max_{e \in E}$ l(e)
  - π(G,R) is an **upper bound** to congestion!
- ➢ Goal is to find R that minimizes π(G,R)
  - shown to be NP-hard in the general case

T. Hoefler, T. Schneider and A. Lumsdaine: Optimized Routing for Large-Scale InfiniBand Networks

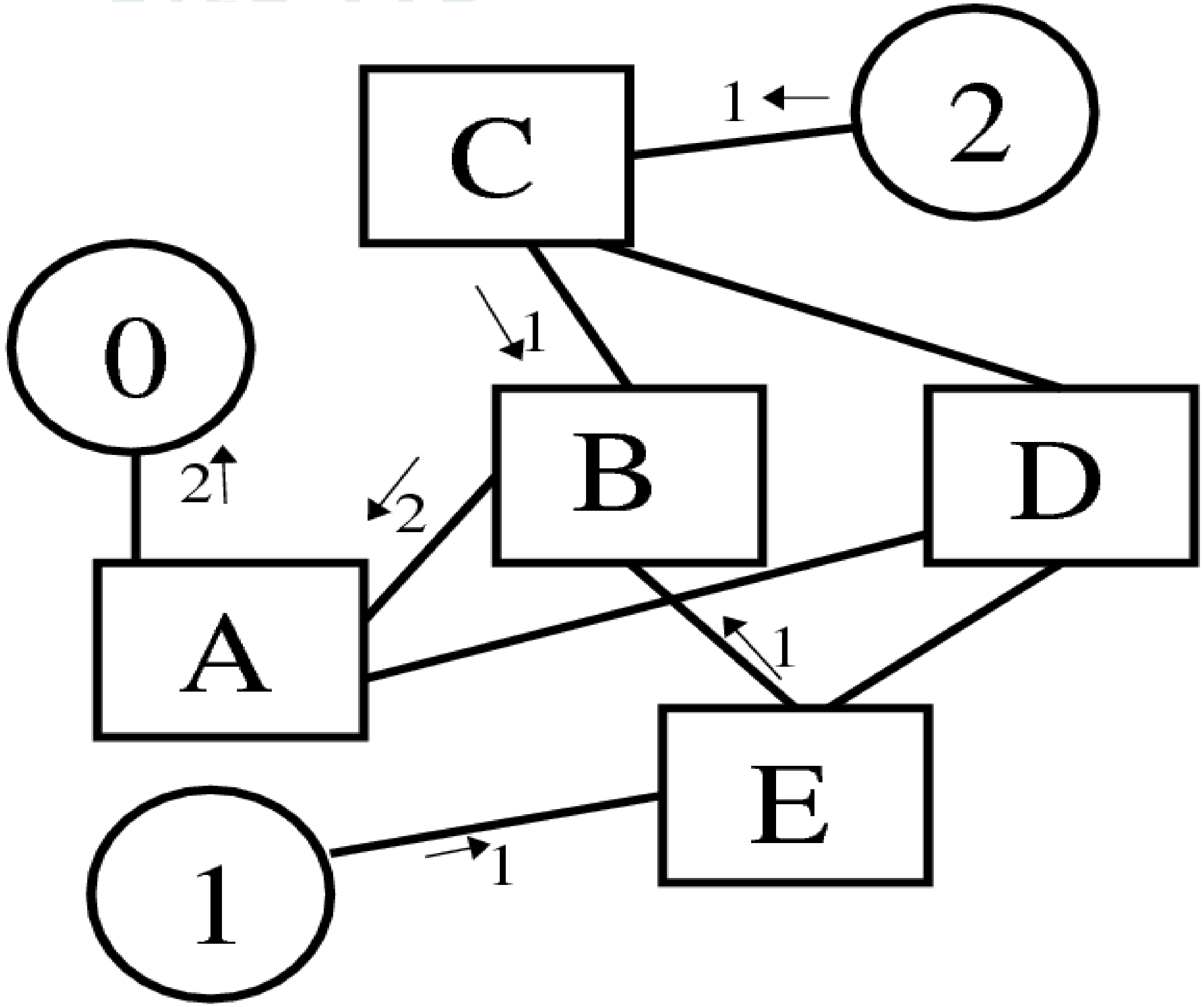
A Simple Heuristic

- Keep it simple, greedily minimize $\pi(G,R)$
- SSSP routing starts a SSSP run at each node
  - Finds paths with minimal edge-load $l(e)$
  - Updates routing tables in reverse
    - essentially SDSP
  - Updates $l(e)$ between runs
  - Strives for global balancing
- An example …

T. Hoefler, T. Schneider and A. Lumsdaine: Optimized Routing for Large-Scale InfiniBand Networks

STEP 1/3

Step 1:
Source-node 0:

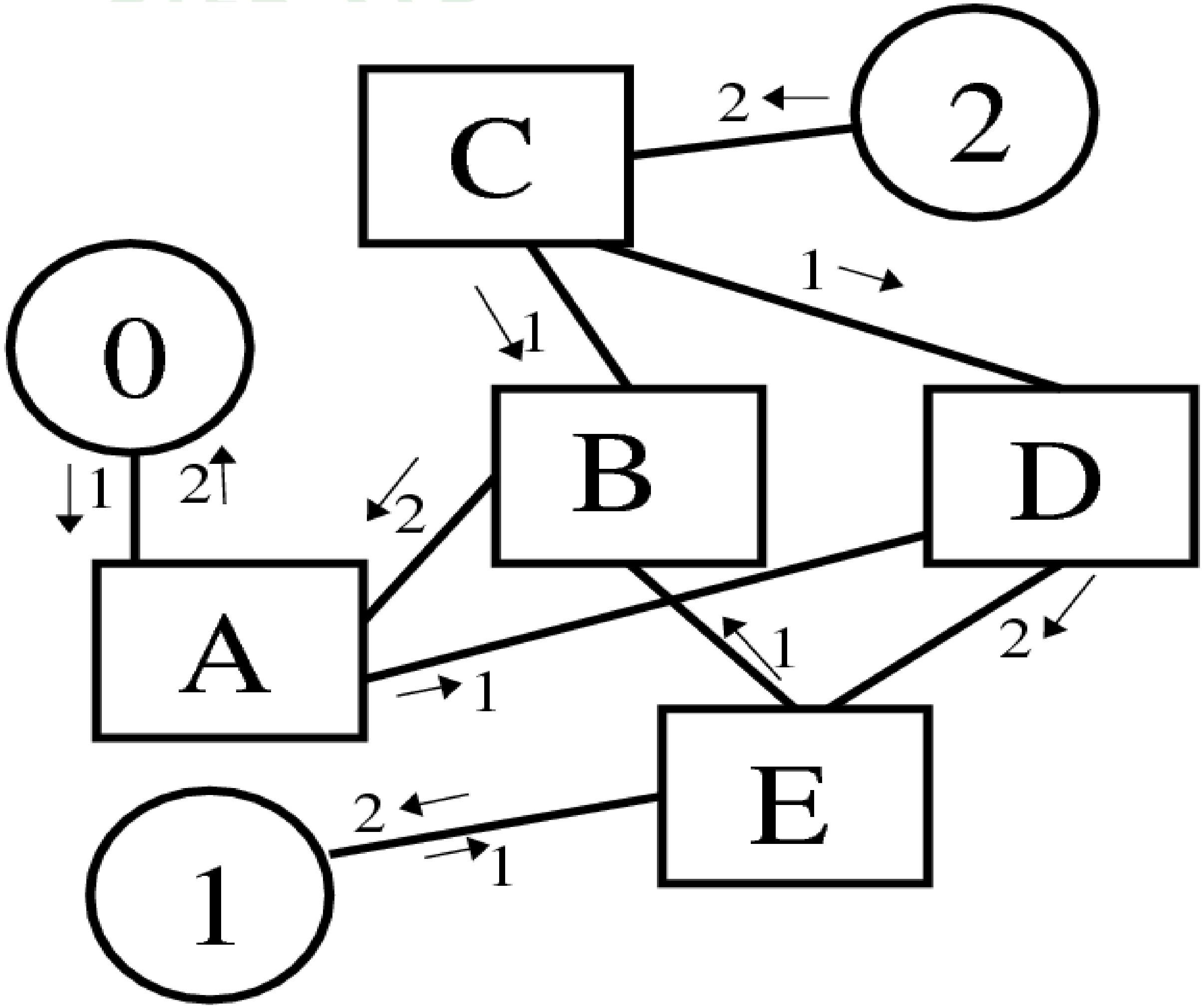


T. Hoefler, T. Schneider and A. Lumsdaine: Optimized Routing for Large-Scale InfiniBand Networks

Step 2/3

Step 2:

Source-node 1:

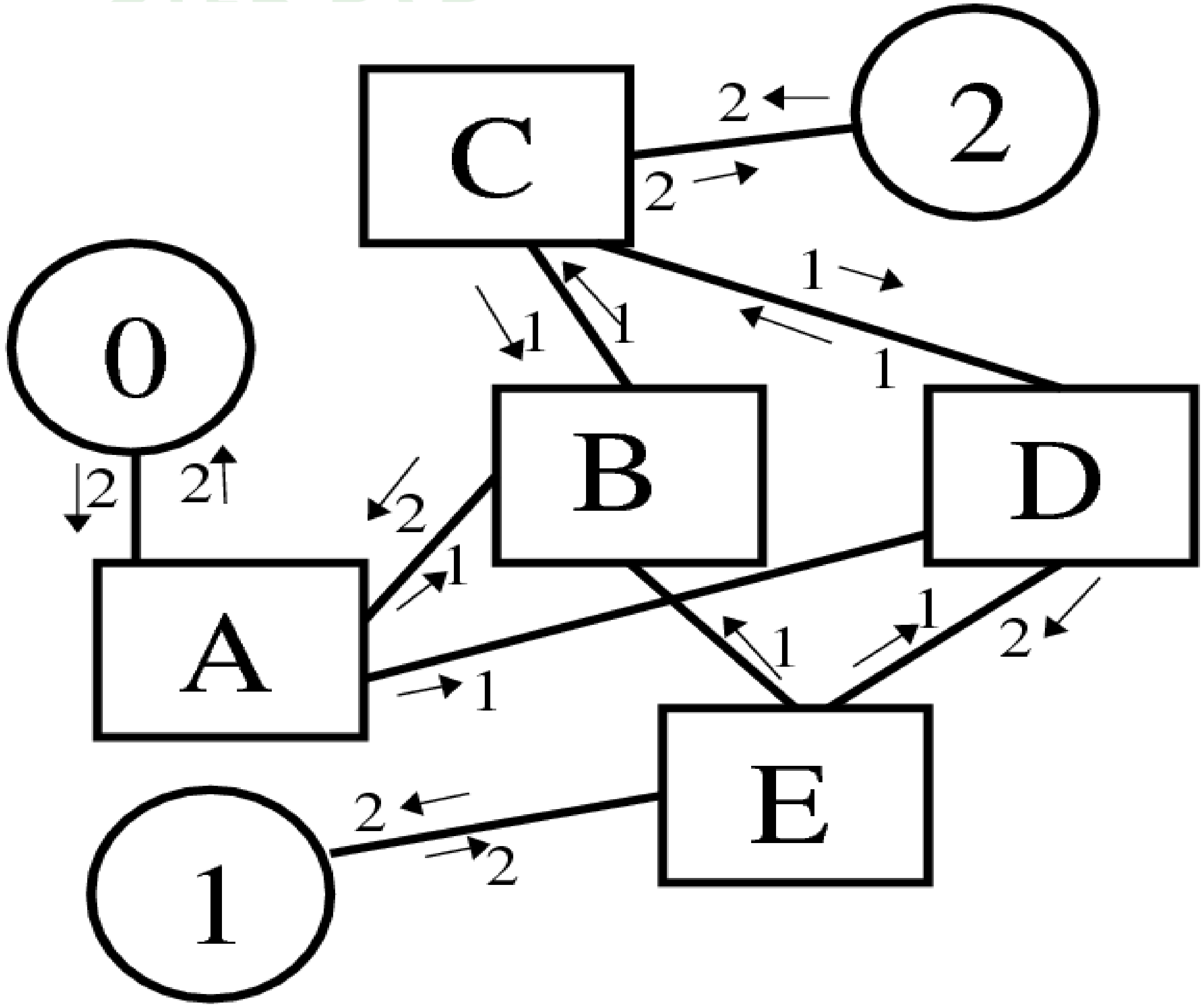


T. Hoefler, T. Schneider and A. Lumsdaine: Optimized Routing for Large-Scale InfiniBand Networks

STEP 3/3

Step 3:

Source-node 2:

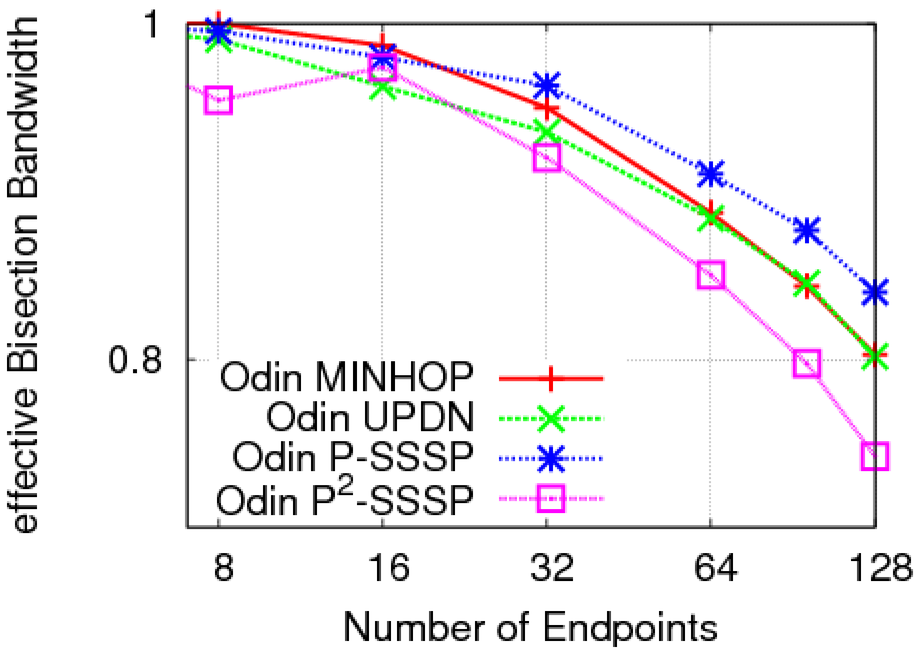


$\pi(G,R)$=2

T. Hoefler, T. Schneider and A. Lumsdaine: Optimized Routing for Large-Scale InfiniBand Networks

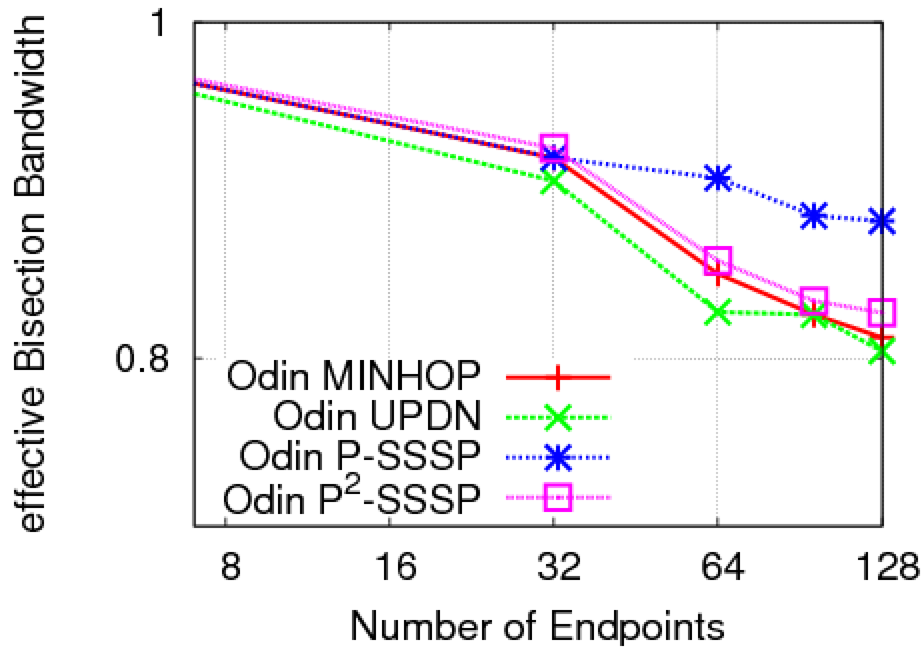
Evaluation - Odin

Simulation



Simulation predicts 5% improvement

Benchmark
(Netgauge Pattern eBB)

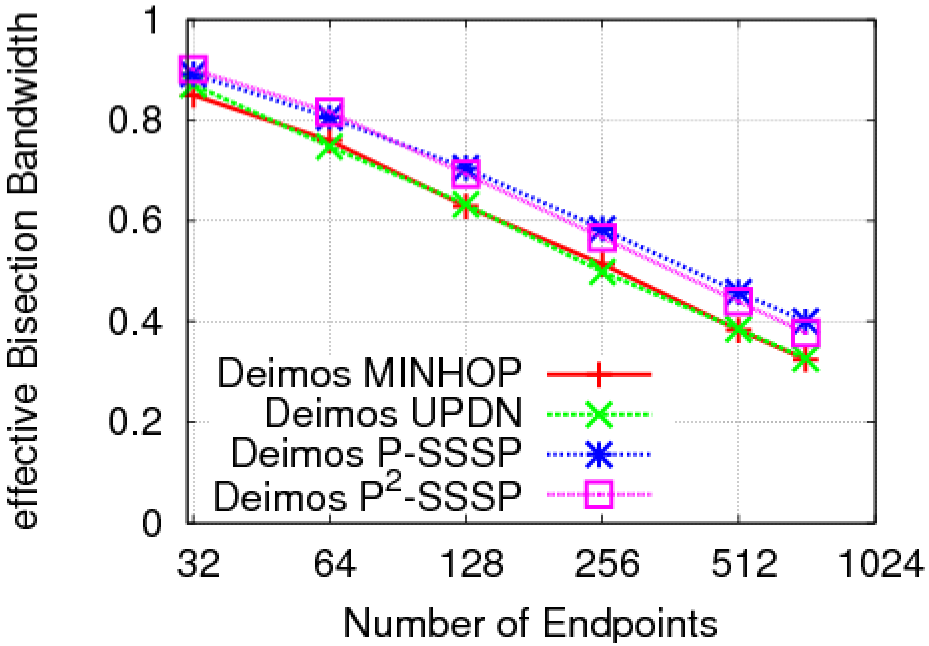


Benchmark shows 18% improvement!

T. Hoefler, T. Schneider and A. Lumsdaine: Optimized Routing for Large-Scale InfiniBand Networks

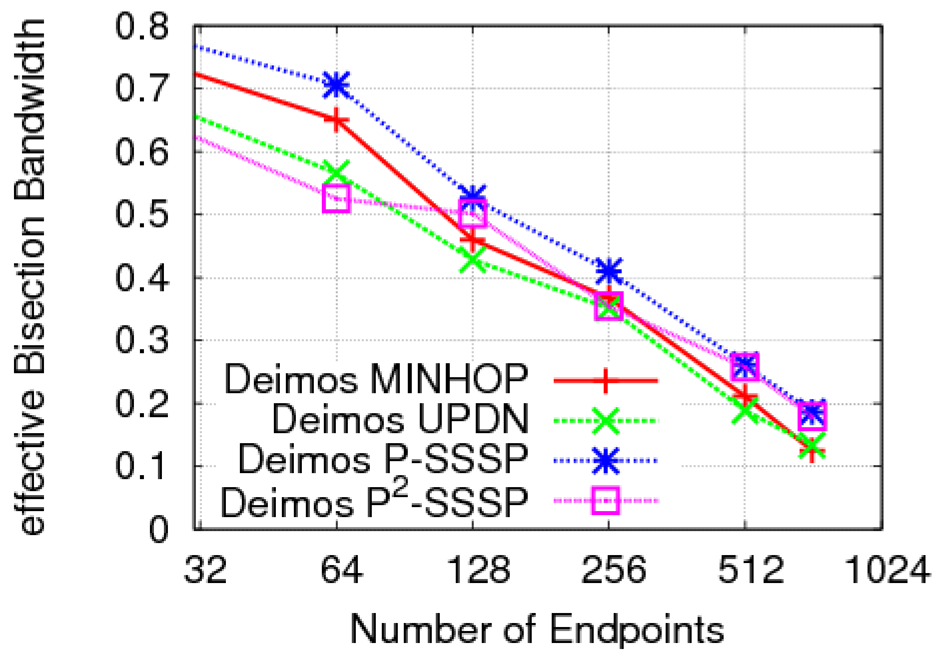
EVALUATION - DEIMOS

Simulation



Simulation predicts 23% improvement

Benchmark
(Netgauge Pattern eBB)



Benchmark shows 40% improvement!

T. Hoefler, T. Schneider and A. Lumsdaine: Optimized Routing for Large-Scale InfiniBand Networks

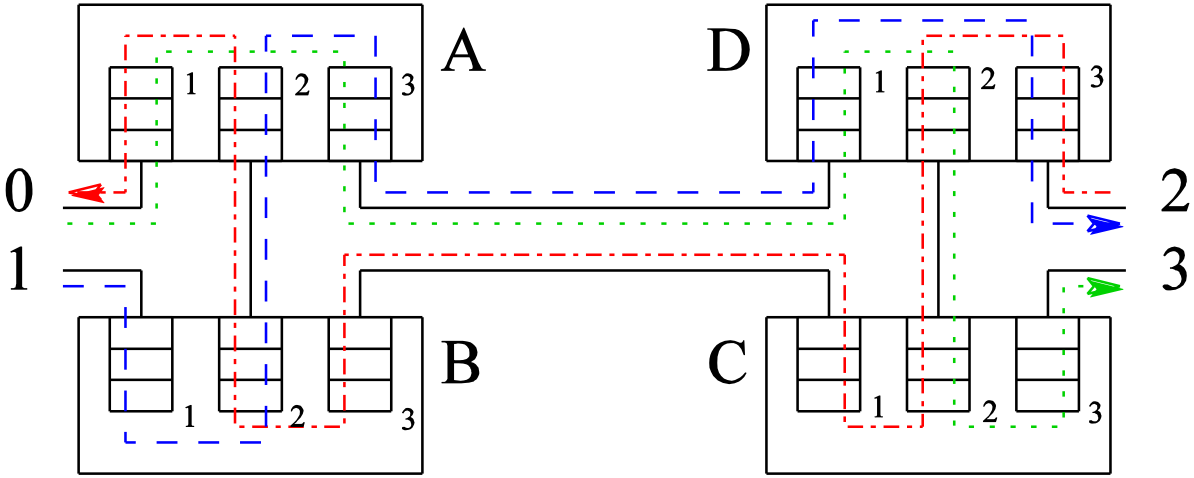
IT WORKS, IS THAT ALL? JUST SSSP?

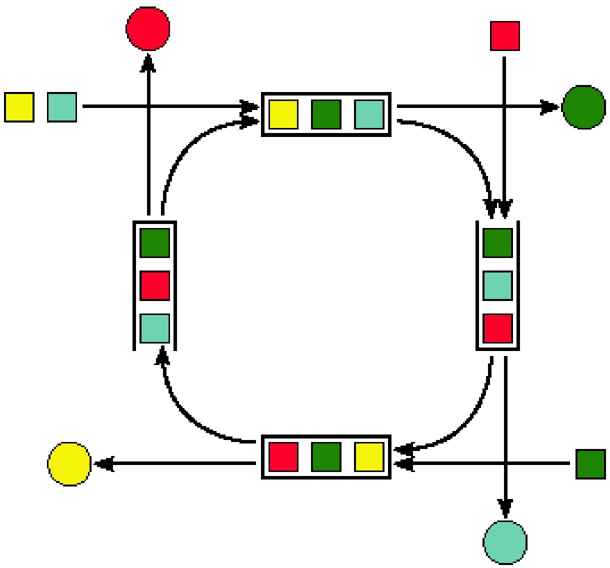
- Shown to run well on real systems in practice
  - Odin (128 nodes, 23% eBB speedup)
  - Deimos (~700 nodes, 40% eBB speedup)
  - Lomonosov[1] (~4.5k nodes, ~10-20% Graph500 speedup)
- Unfortunately not!
- SSSP Routing may create loops
  - On certain topologies
  - To be proven if some topologies are loop-free
  - Problematic in production environments (and interesting in theory ☺)

[1] *Lomonosov experiments were executed by Anton Korzh and Alexander Naumov*

WHAT ARE CREDITS AND WHY DO THEY LOOP?

- IB uses credit-based p2p flow-control
  - egress messages sent only if receive-buffer available



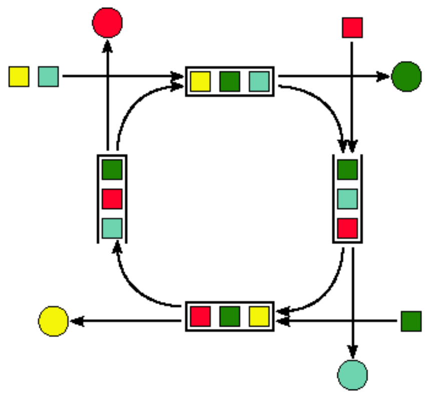


- very similar to deadlocks in wormhole-routed systems

Domke, Hoefler, Nagel: Deadlock-Free Oblivious Routing for Arbitrary Topologies

Deal With Credit Loops

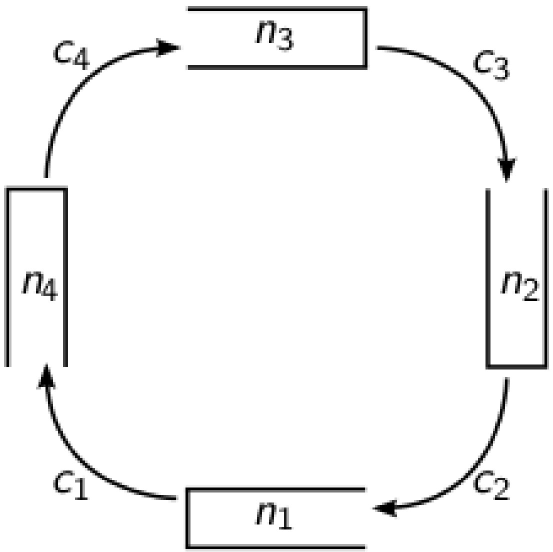
- Prevent (UP*/Down*, turn-based routing)
  - Limits routing options
- Resolve (LASH, use VLs to break cycles)
  - Consumes additional buffers
- Ignore (MINHOP, DOR)
  - Potential resolution: packet timeouts
  - Discouraged by IB specification
- Others: Bubble Routing etc.
  - Not supported by current devices

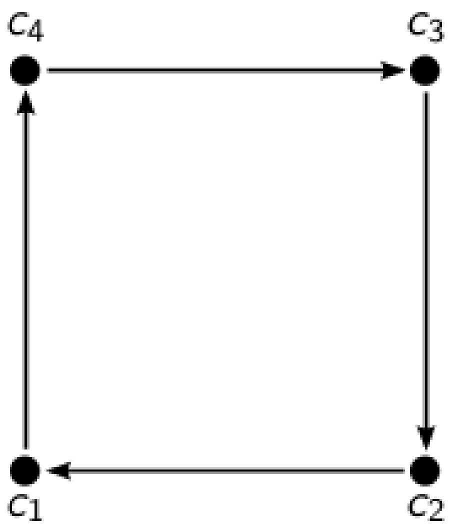


Domke, Hoefler, Nagel: Deadlock-Free Oblivious Routing for Arbitrary Topologies

Using VLs to Avoid Deadlocks

- Pioneered with LASH, example:



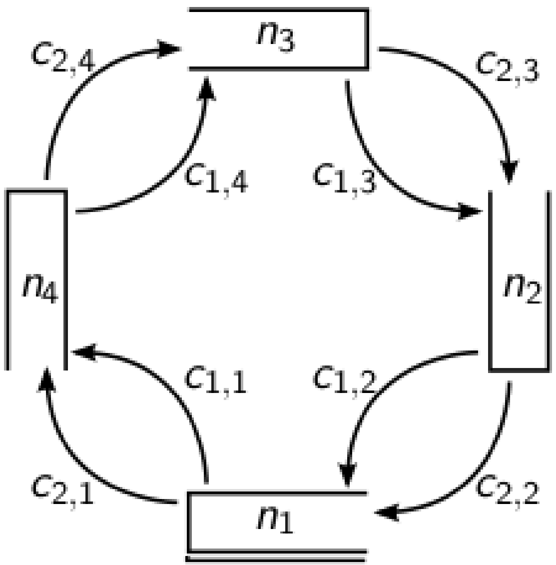


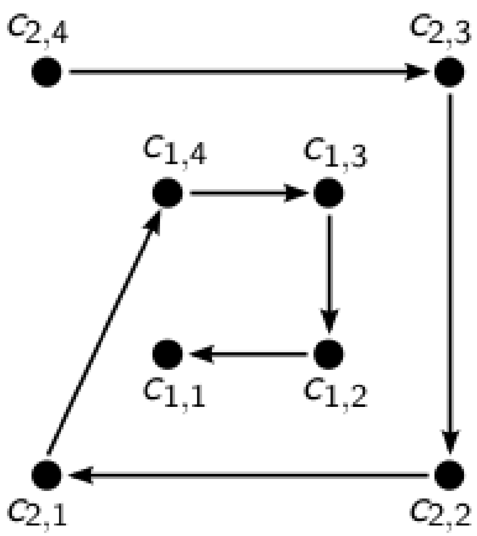
- Deadlock!

Domke, Hoefler, Nagel: Deadlock-Free Oblivious Routing for Arbitrary Topologies

Using VLs to Avoid Deadlocks

- Pioneered with LASH, example:



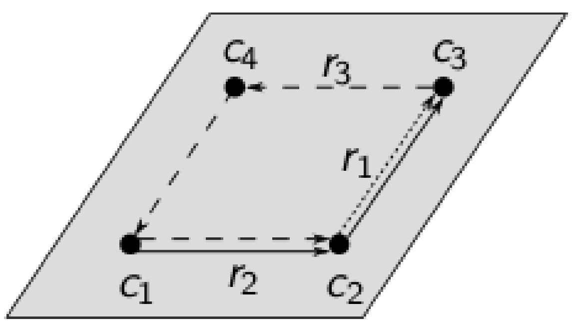


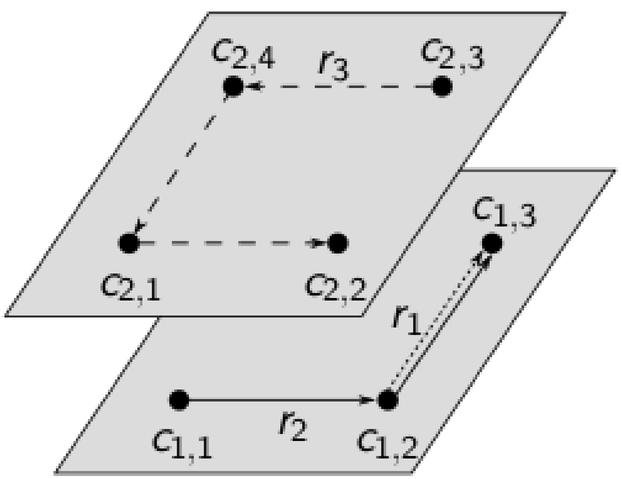
- 2 VLs resolve deadlock

Domke, Hoefler, Nagel: Deadlock-Free Oblivious Routing for Arbitrary Topologies

DEADLOCK-FREE SSSP ROUTING

- Perform normal SSSP
- Detect cycles
  - "Break" cycle by adding new VL, rinse, repeat
- VLs are expensive, how many do we need?





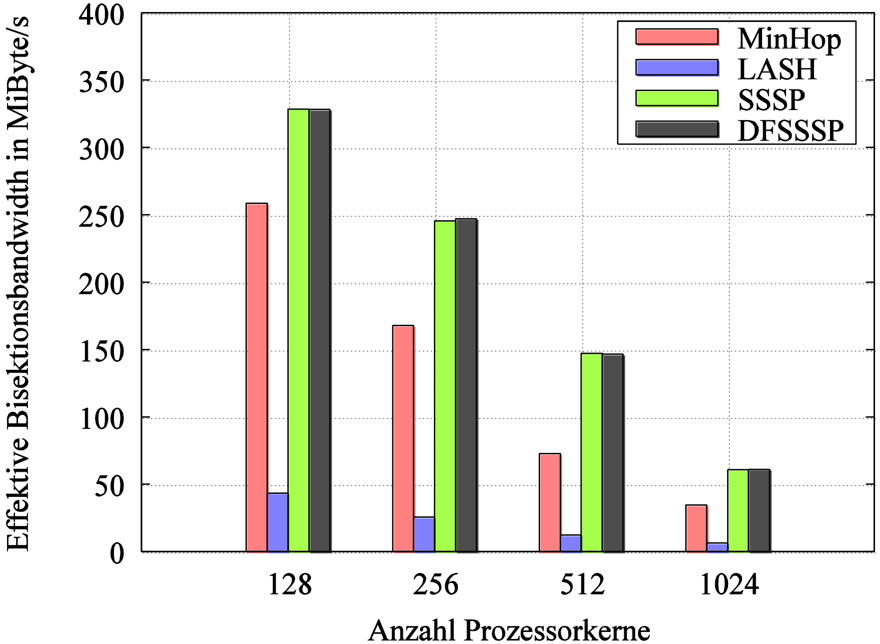
Domke, Hoefler, Nagel: Deadlock-Free Oblivious Routing for Arbitrary Topologies

The Acyclic Path Partitioning Problem

- Abstract formulation: ”acyclic path partitioning” problem (APP)
  - Split a set of paths into subsets which produces acyclic channel dependency graphs
  - We proved NP completeness ☹
  - Reduction of graph k-colorability to APP
- Heuristics:
  - Random edge
  - Heaviest edge (max e(l) in cycle)
  - Lightest edge (min e(l) in cycle) → performed best

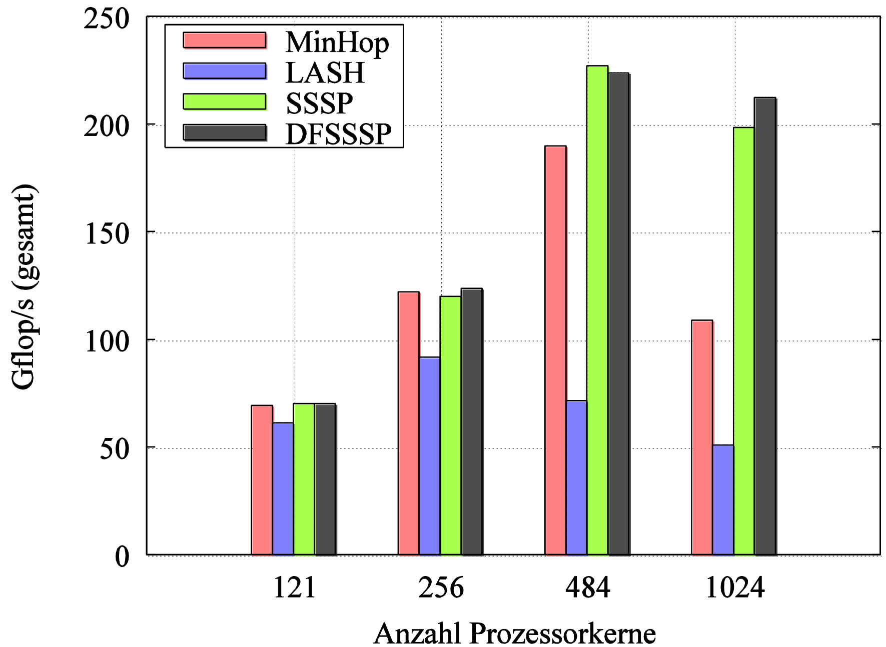
Domke, Hoefler, Nagel: Deadlock-Free Oblivious Routing for Arbitrary Topologies

EBB AND NAS BECHMARKS



Netgauge, eBB

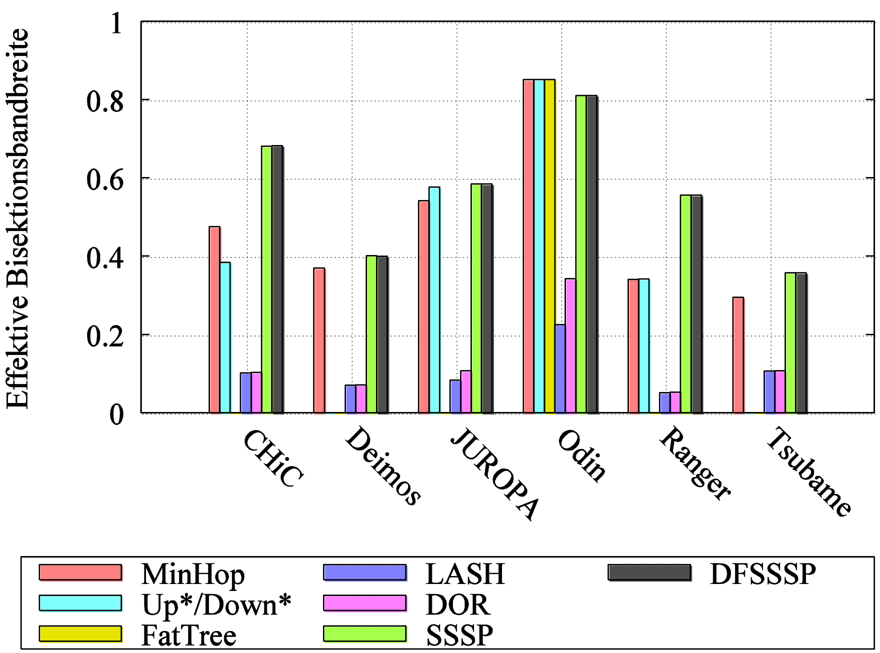
BT, class C – solver

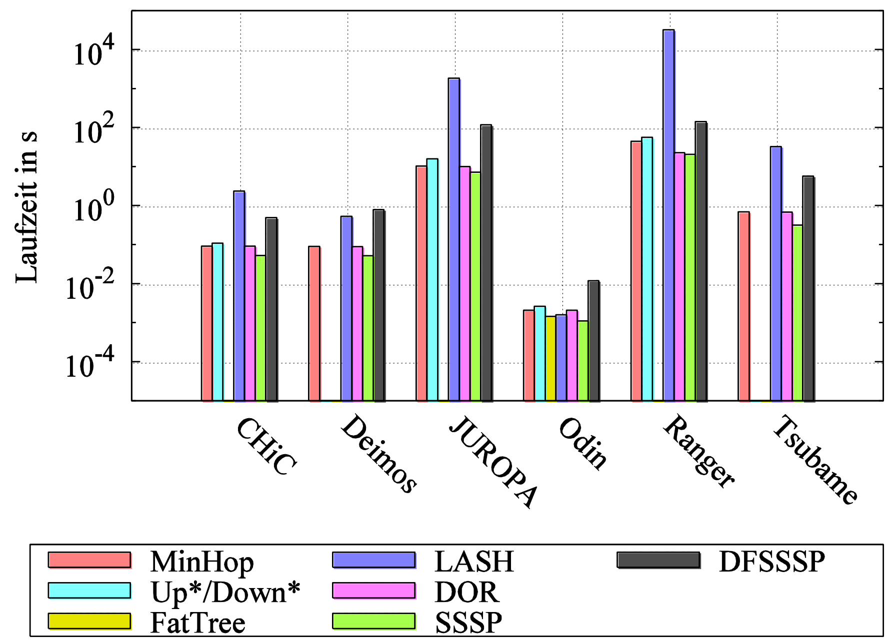


Domke, Hoefler, Nagel: Deadlock-Free Oblivious Routing for Arbitrary Topologies

Is it Practical? What about Exascale?

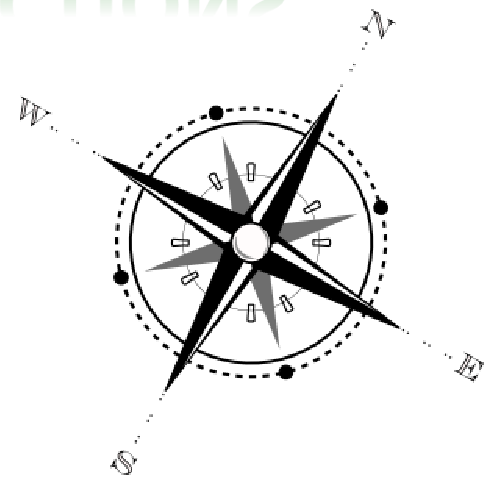
- Merged into OFED (v3.3.14)
  - Runtime is an issue!





Domke, Hoefler, Nagel: Deadlock-Free Oblivious Routing for Arbitrary Topologies

Possible Future Directions

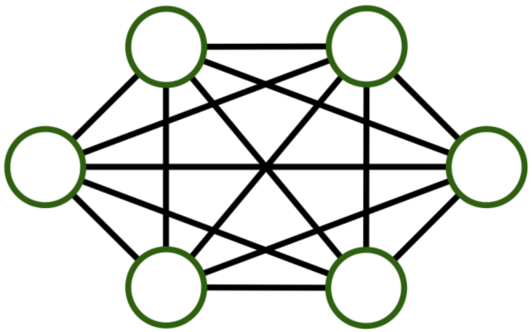


- Better Heuristics
  - Higher bandwidth
  - Lower number of VLs
- Fault tolerance
  - Analyze behavior with failing links
  - Online re-routing (no re-computing from scratch)
- Adaptive routing
  - Extensions possible (interesting!)
  - Also subset-random routing
- Application-specific
  - Modeling/Co-design[1]!

[1] *Hoefler, Gropp, Snir and Kramer: Performance Modeling for Systematic Performance Tuning*

Do I care? What else Can I do?

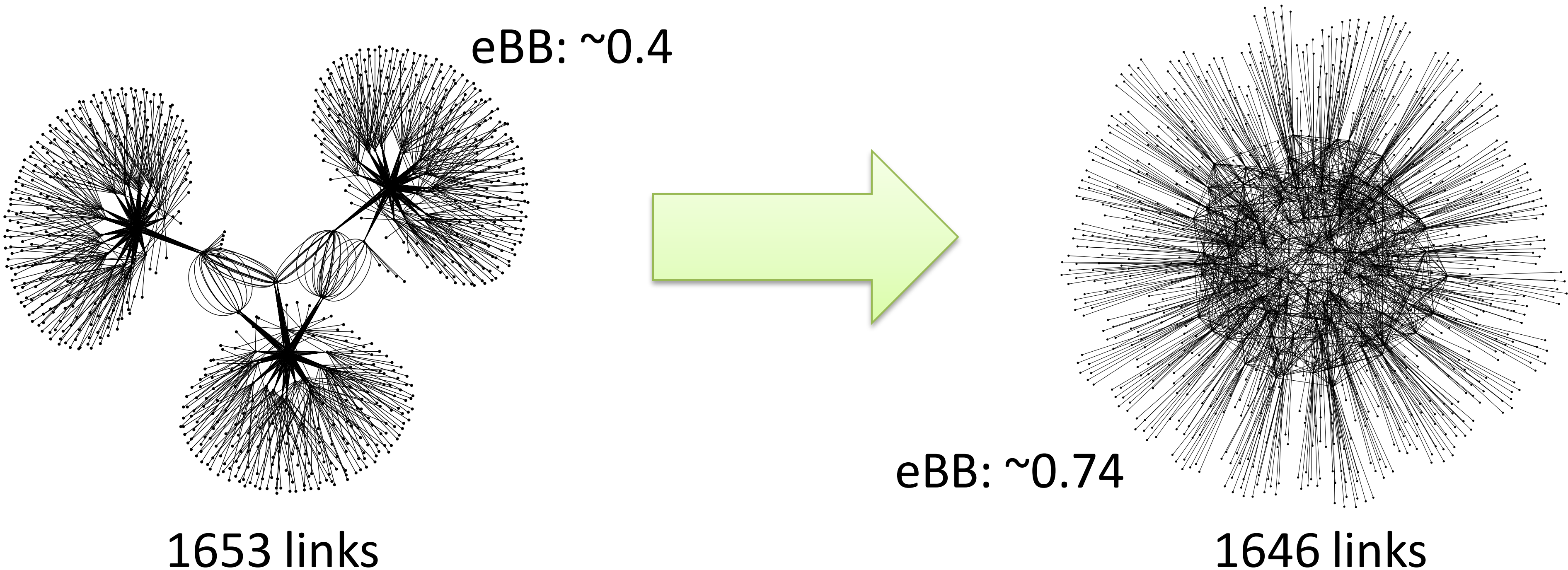
- Opinion 1: This is great! I am computing alltoalls and love this!
  - Graph computations
  - Spectral methods



- Opinion 2: I don't care about global bandwidth, my halo communication is local
  - Well, you think so?
  - Irregular stencils are often badly mapped!

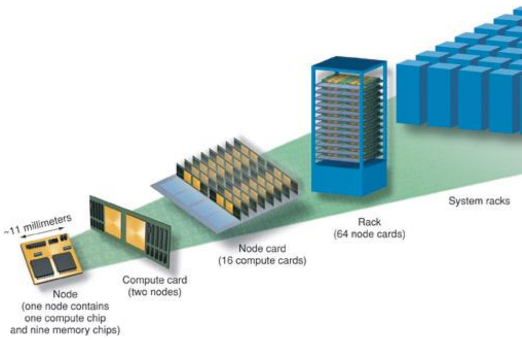
Opinion 1: Optimize Global Bandwidth!

- Maybe use a different topology (Co-Design ☺)
- For example: Deimos vs. Dragonfly



Option 2: Optimize Locality!

- Large-scale systems are built with low-dimensional network topologies
  - E.g., 3d-torus Jaguar (18k nodes), BG/P (64k nodes)



- Number of nodes grows (~100k-1M for Exascale)
  - Will rely on fixed arity switches
  - ➢ Diameter increases
  - ➢ Bisection bandwidth decreases (in relative terms)

Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

The need for Topology Mapping

- Default mapping of processes to nodes often fails to take advantage of locality
  - E.g., linear mapping of a 3d grid onto a hierarchical (e.g., multicore) network
    (should use sub-cubes)
- Problem has been analyzed for mapping Cartesian topologies [Yu'06,Bhatele'09]
  - But communication network might have complex structure (failed links, "naturally grown")
  - And application likely to be non-Cartesian too (AMR)

The Problem and Metrics

- The general mapping problem $\Gamma : V_{\mathcal{G}} \to V_{\mathcal{H}}$
  - We showed that it's NP-complete
- Average dilation
  - "average path length through the network"
  - Number of transceivers involved → power
- Worst-case congestion (cf. paper for equation)
  - "congestion of a link is ratio of traffic to bandwidth"
  - "worst-case congestion is the maximum congestion on any link in the network"
  - Bound on the communication time → performance

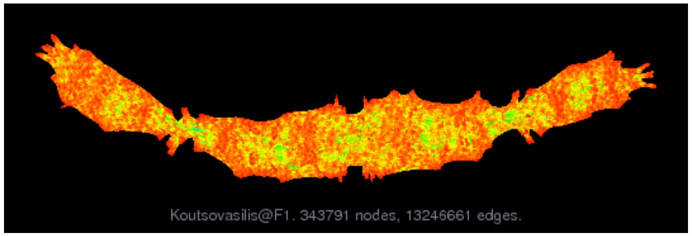
Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

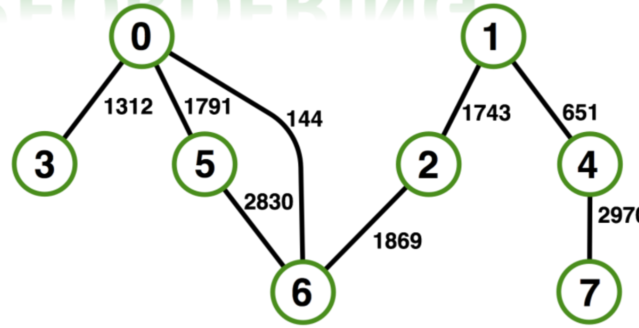
An MPI Interface to TopoMap

- Application topologies are often only known at runtime
  - Prohibits mapping before allocation
  - Batch-systems also have other constraints!
- MPI-2.2 defines interface for re-mapping
  - Scalable process topology graph
  - Permutes ranks in communicator
  - Returns “better” permutation π to the user
  - User can re-distribute data and use π

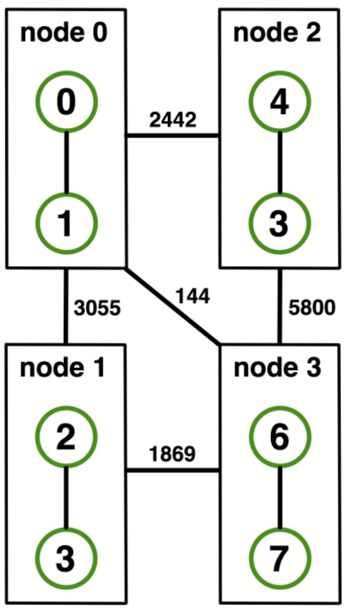
Hoefler et al.: The Scalable Process Topology Interface of MPI 2.2

On-Node Reordering



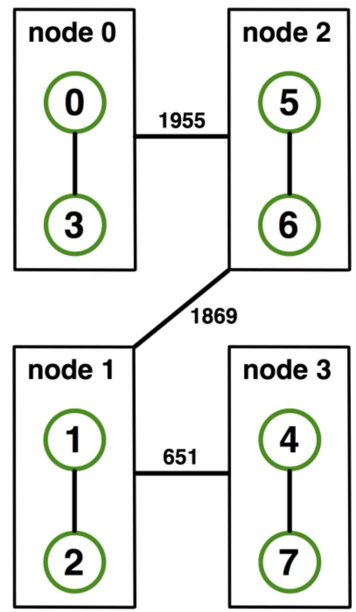


Naïve Mapping



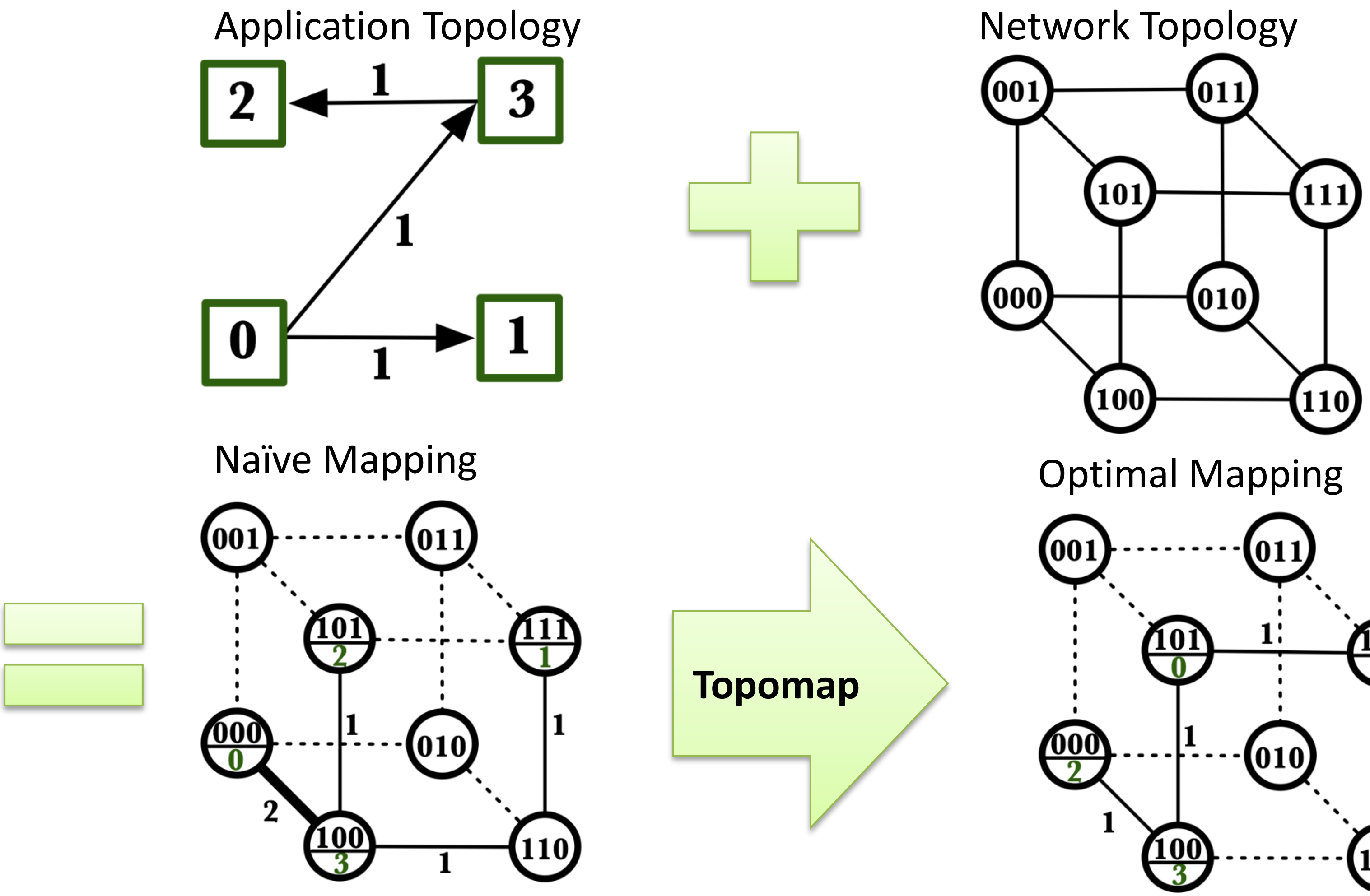
Topomap

Optimized Mapping



Gottschling and Hoefler: Productive Parallel Linear Algebra Programming with Unstructured Topology Adaption, CCGrid 2012

Off-Node (Network) Reordering



Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

Composable Mapping Heuristics (1/3)

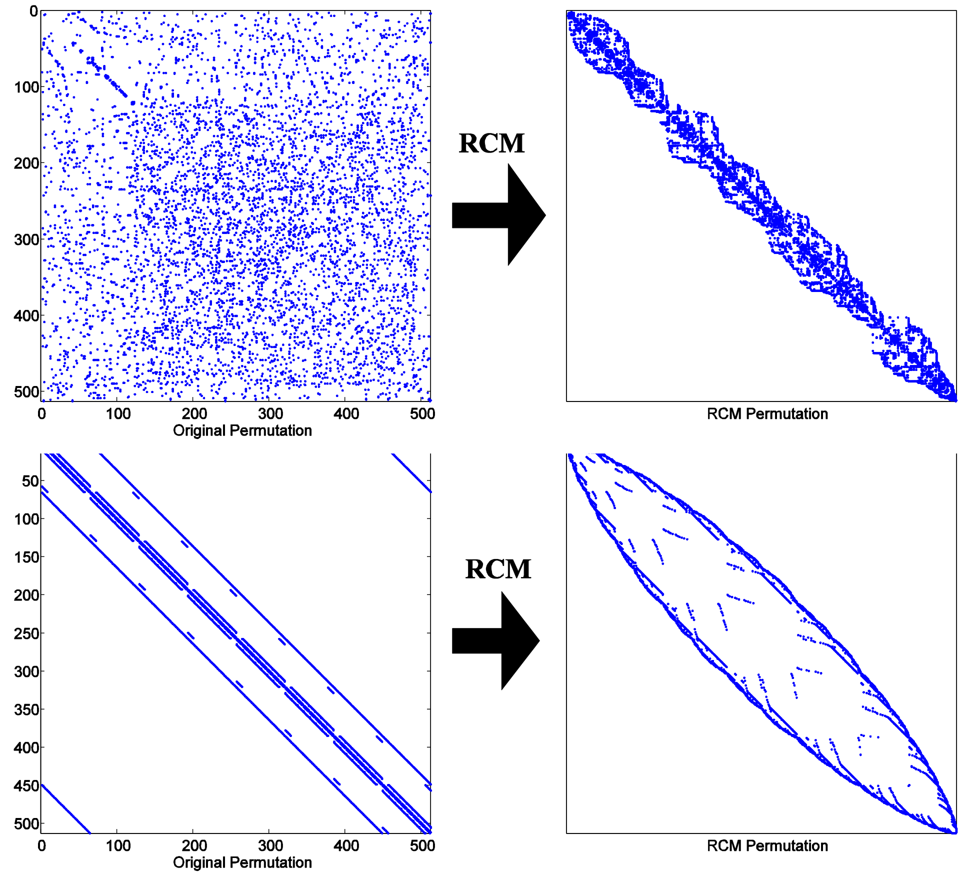
1. Simple Greedy
   - Start at some vertex in $\mathcal{H}$
   - Map heaviest vertex in $\mathcal{G}$ as “close” as possible
   - Runtime: $\mathcal{O}(|V_\mathcal{G}| \cdot (|E_\mathcal{H}| + |V_\mathcal{H}| \log |V_\mathcal{H}| + |V_\mathcal{G}| \log |V_\mathcal{G}|))$
2. Recursive Bisection
   - Recursively cut $\mathcal{H}$ and $\mathcal{G}$ into minimal bisections
   - Map vertices in $\mathcal{G}$ to vertices in $\mathcal{H}$
   - Runtime: $\mathcal{O}(|E_\mathcal{G}| \log(|V_\mathcal{G}|) + |E_\mathcal{H}| \cdot |V_\mathcal{G}|)$

Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

Composable Mapping Heuristics (2/3)

3. Graph Similarity
   Cuthill McKee

- Apply RCM to $\mathcal{H}$ and $\mathcal{G}$
- Map resulting permutations
- Runtime:
  $\mathcal{O}(m_{\mathcal{H}} \log(m_{\mathcal{H}})|V_{\mathcal{H}}|)$
  + $\mathcal{O}(m_{\mathcal{G}} \log(m_{\mathcal{G}})|V_{\mathcal{G}}|)$
  (m = max degree)



Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

Composable Mapping Heuristics (2/3)

3. Hierarchical Multicore Mapping
   - Assuming $C(v) = p \; \forall v \in \Gamma(V_{\mathcal{H}})$
   - Partition $\mathcal{G}$ into P/p balanced partitions
   - Using METIS for $(\mathrm{k}, 1+\epsilon)$-balanced partitions
     - Might need corrections!
4. Simulated Annealing / Threshold Accepting (TA)
   - SA was proposed as heuristic [Bollinger&Midkiff]
   - Using TA to improve found solution further

Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

EVALUATION

- We assume static routing with load spread evenly
- Real-world MatVec from Florida Sparse Matrix Coll.
  - F1, audikw_1: symmetric stiffness matrices, representing automotive crankshafts
  - nlpkkt240: nonlinear programming (3d PDE, constrained optimization problem)

Matrix Name	Rows and Columns	NNZ (sparsity)
F1	343,791	26,837,113 $(2.27 \cdot 10^{-4}\%)$
audikw_1	943,695	39,297,771 $(4.4 \cdot 10^{-5}\%)$
nlpkkt240	27,993,600	401,232,976 $(5 \cdot 10^{-7}\%)$

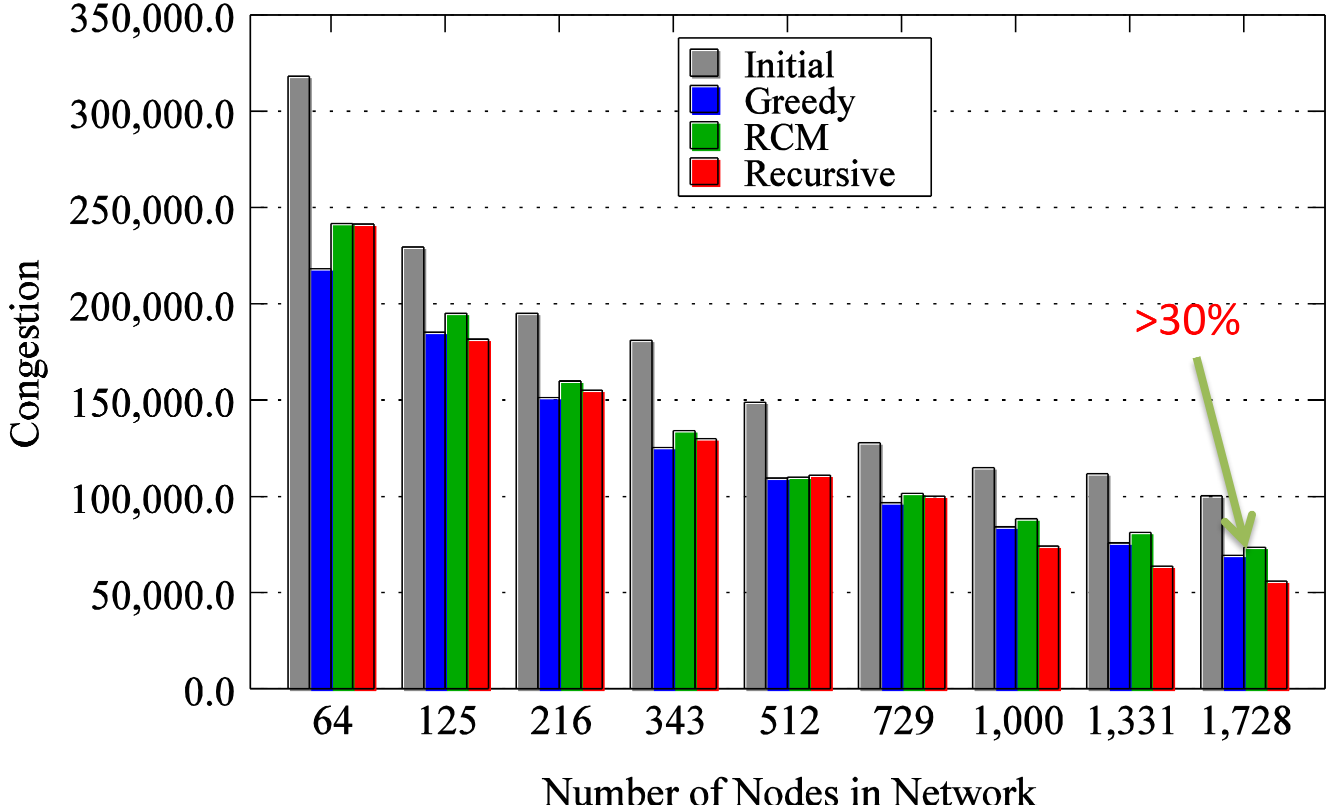
Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

Experimental Verification

- Load matrix, partition with ParMETIS
  - Construct MPI-2.2 distributed graph topology
  - Apply topology mapping
  - Re-distribute data
- Assess quality:
  - Simulate congestion and dilation
    - Simple counting, assumes idealized routing!
  - Run a timed benchmark
    - Report time for 100 communication phases
    - Maximum time across all ranks

Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

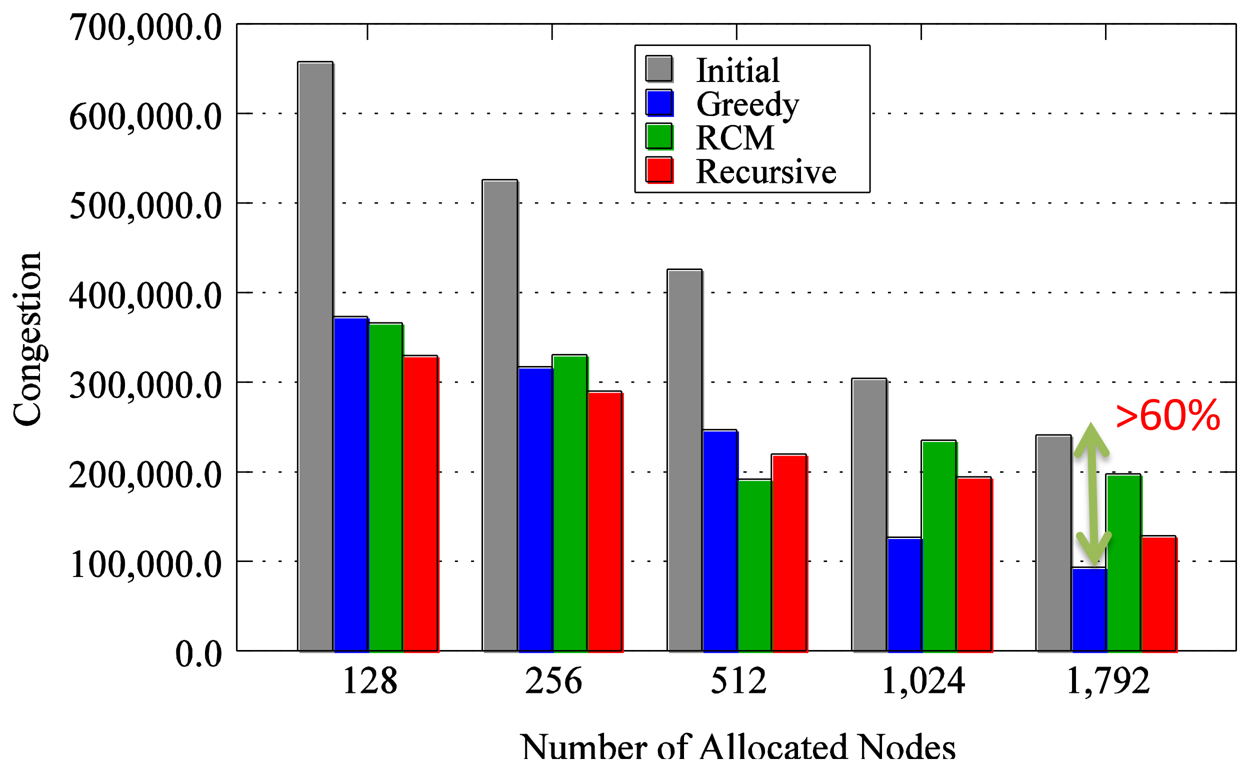
Simulation: 3D Torus Networks



- nlpkkt240, dilation for 12^3: 9.0, 9.03, 7.02, 4.5
- Times for 123: <0.01s, 1s, 1s, 10 min

Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

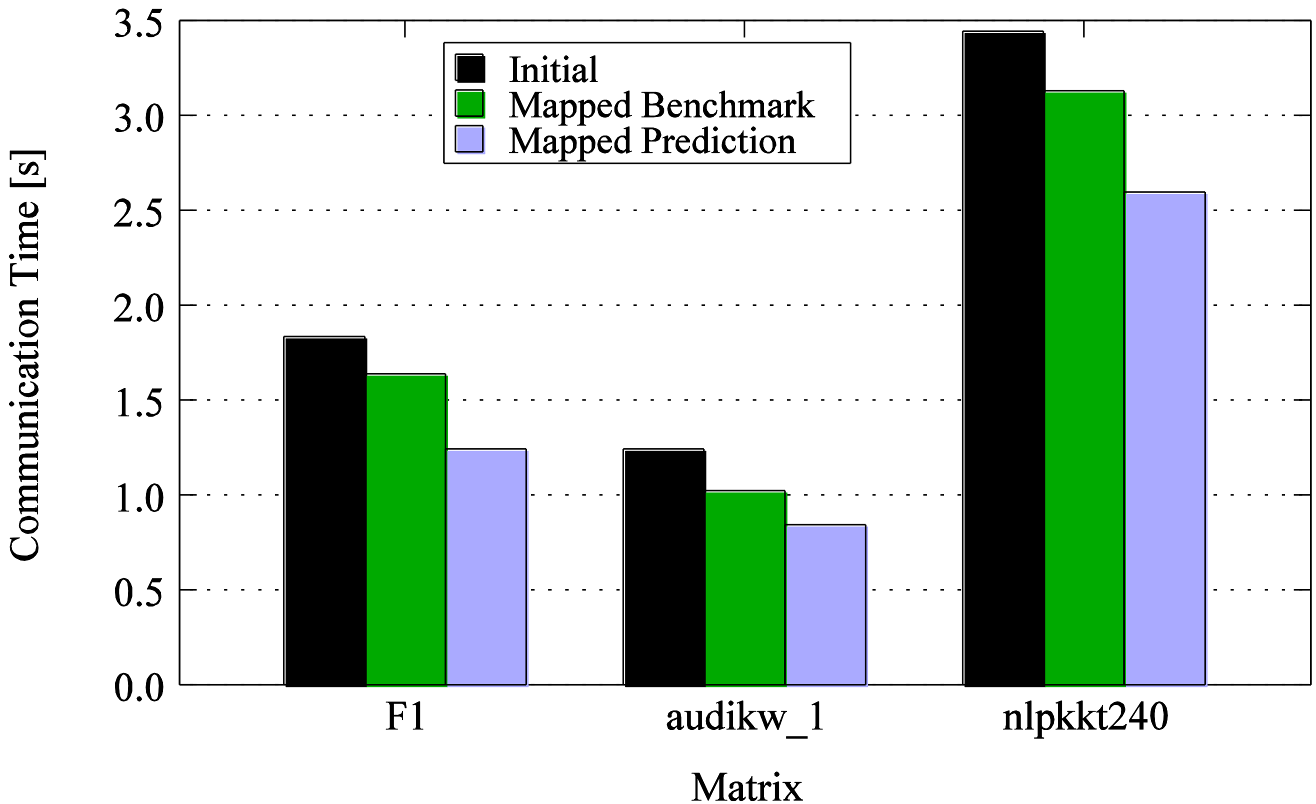
SIMULATION: JUROPA - INFINIBAND



- audikw_1, dilation: 5.9, 5.8, 4.45, 5.13
- Times: <0.01s, 0.16-2.6s, 0.63-1.21s, 9 min

Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

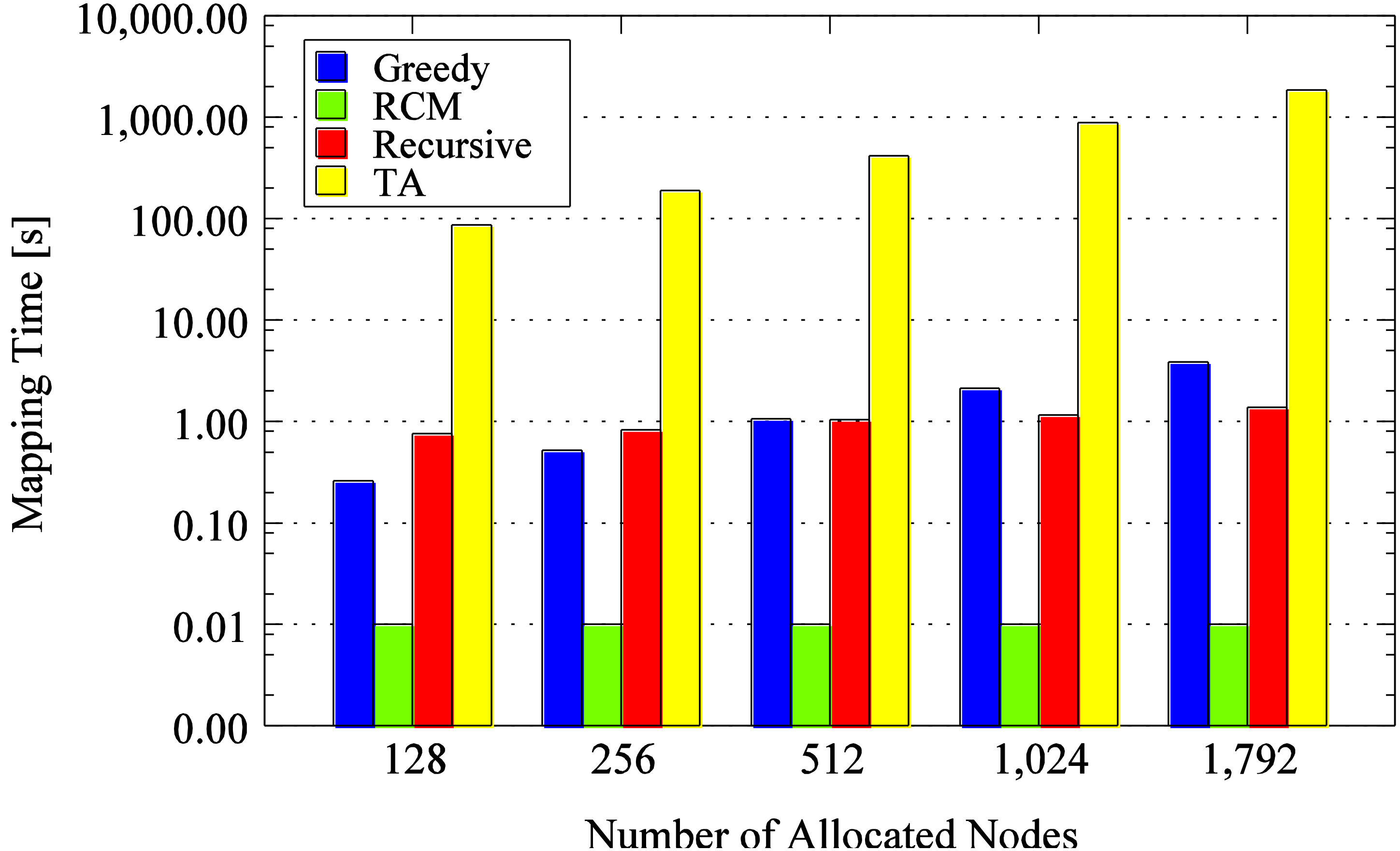
BENCHMARK: BLUEGENE/P



- 512 nodes, up to 18% improvement measured
  - BG/P has good routing

Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

MAPPING TIMES



- Topology: Ranger, InfiniBand, ~4k nodes

Hoefler and Snir: Generic Topology Mapping Strategies for Large-scale Parallel Architectures

Topomap Problems and Directions

- The endless search for better heuristics
  - Topology-specific
  - Exascale? Parallelize topomap, improve speed
- The routing metric is artificial (idealized)
  - Simulated predictions are inaccurate
  - Target metric can be improved
- Combine topology mapping and routing
  - Application-specific mapped routing

Summary & Conclusions

- Optimized SSSP Routing works
  - http://www.unixer.de/research/dfsssp (in OFED 3.3.14)
- ORCS – Congestion/Routing Simulation
  - http://www.unixer.de/research/orcs/ (research quality)
- LibTopoMap – Generic Topology Mapping
  - http://www.unixer.de/research/mpitopo/libtopomap/
- LogGOPSim – full MPI Simulator
  - http://www.unixer.de/research/LogGOPSim/
  - Can be integrated with topology (research quality)
- Sponsors: