BLUE WATERS

SUSTAINED PETASCALE COMPUTING

Generic Topology Mapping Strategies for Large-scale Parallel Architectures

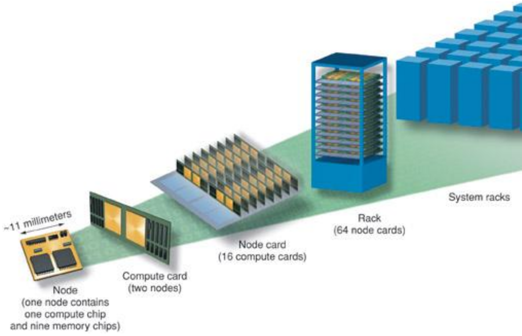
Torsten Hoefler and Marc Snir

Scientific talk at ICS'11, Tucson, AZ, USA, June 1st 2011,

GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION

Hierarchical Sparse Networks are Ubiquitous

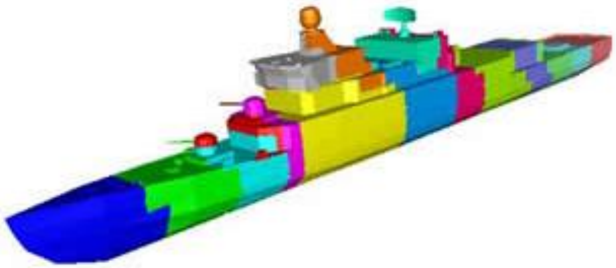
- Large-scale systems are built with low-dimensional network topologies
  - E.g., 3d-torus Jaguar (18k nodes), BG/P (64k nodes)



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

Application Communication Patterns

- Scalable communications are sparse!
  - E.g., 2d FFT decomposition
  - Most patterns have spatial locality



- Trivial 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)

Topology Mapping - State of the Art

- 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 general problem has not been studied much
  - We show that it's NP-complete
  - Also consider heterogeneous networks [PERCS'10]

Terms and Conventions

- Application communication pattern is modeled as weighted graph $\mathcal{G} = (V_{\mathcal{G}}, \omega_{\mathcal{G}})$
  - $V_{\mathcal{G}}$ is the set of processes
  - $\omega_{\mathcal{G}}$ represents the communication volume
- Physical network is $\mathcal{H} = (V_{\mathcal{H}}, C_{\mathcal{H}}, c_{\mathcal{H}}, \mathcal{R}_{\mathcal{H}})$
  - $V_{\mathcal{H}}$ set of physical nodes
  - $C_{\mathcal{H}}(u)$ number of PEs in node $u \in V_{\mathcal{H}}$
  - $c_{\mathcal{H}}(u,v)$ link capacity (bandwidth) of link $(u,v) \in V_{\mathcal{H}} \times V_{\mathcal{H}}$
  - $\mathcal{R}_{\mathcal{H}}$ set of routes (may be multiple routes from u to v)

Topology Mapping Metrics

- Topology Mapping $\Gamma : V_{\mathcal{G}} \to V_{\mathcal{H}}$
- Average dilation (|p| = length of path p)
  - $Dilation(uv) = \sum_{p \in \mathcal{P}(\Gamma(u)\Gamma(v))} \mathcal{R}_{\mathcal{H}}(\Gamma(u)\Gamma(v))(p) \cdot |p|$
  - $Dilation(\Gamma) = \sum_{u,v \in V_{\mathcal{G}}} \omega_{\mathcal{G}}(uv) \cdot Dilation(uv)$
  - "average path length through the network"
- 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"

Meaning of the Metrics

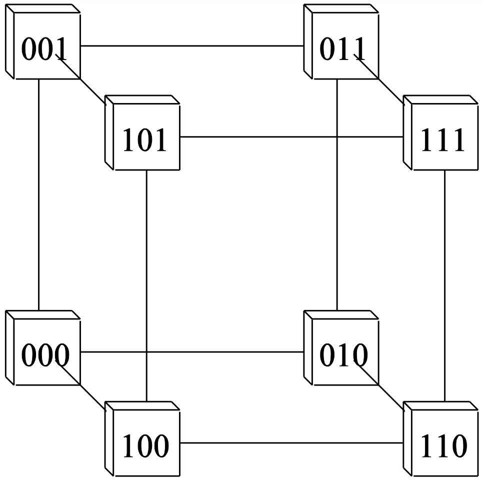
- Worst-case congestion (or just “congestion”)
  - Lower bound on the communication time
  - Measure of performance
- Average dilation (or just “dilation”)
  - Number of transceivers involved in communication
  - Measure for power consumption

Assumptions and Practical Issues

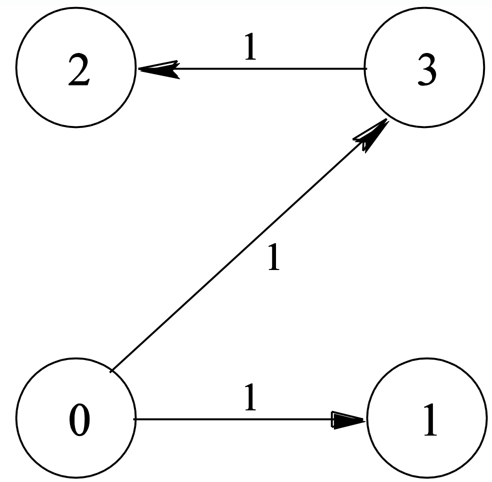
- Assumptions:
  1. Infinite bandwidth for intra-node communication
  2. Dilation=0 for intra-node communication
  3. Nonblocking (full-bandwidth) switches
  4. Oblivious routing with fixed routing algorithm
- Practical Issues:
  1. Process communication pattern is defined once
  2. Processes are mapped once

Example Mappings

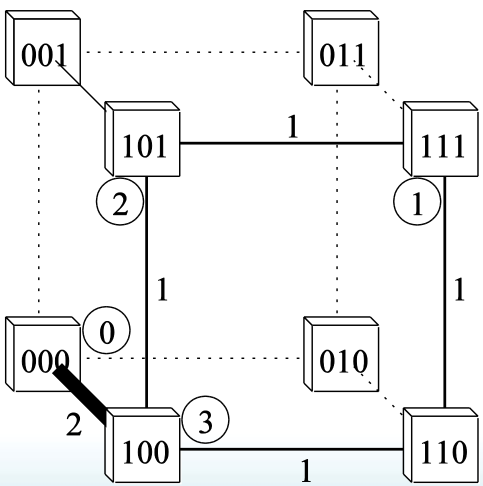
Physical Topology:



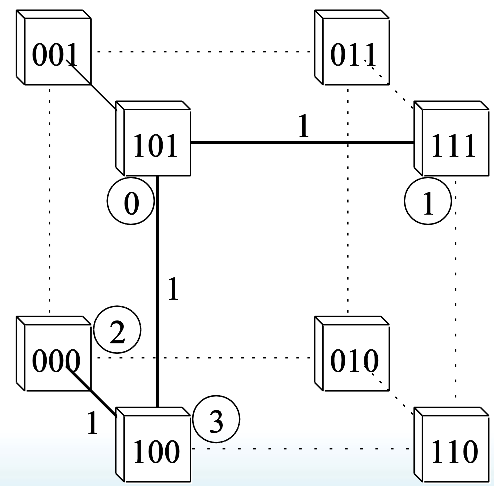
Application Topology:



Mapping 1:



Mapping 2:

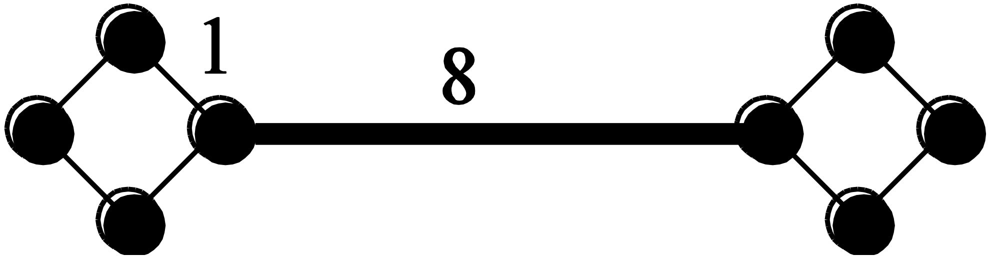


Topology Permutation Mapping

- Application topologies $\mathcal{G}$ are often only known during 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

Topology Mapping is NP-complete

- Reduction to MINIMUM CUT INTO BOUNDED SETS [ND17 in Garey&Johnson]
- Intuition:
  - Assume host graph is "dumbbell"



  - Any mapping defines a partition of the application graph into two equal sizes
  - Must minimize the edge-cut for optimal congestion

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: $\mathcal{O}(|E_\mathcal{G}| \log(|V_\mathcal{G}|) + |E_\mathcal{H}| \cdot |V_\mathcal{G}|)$

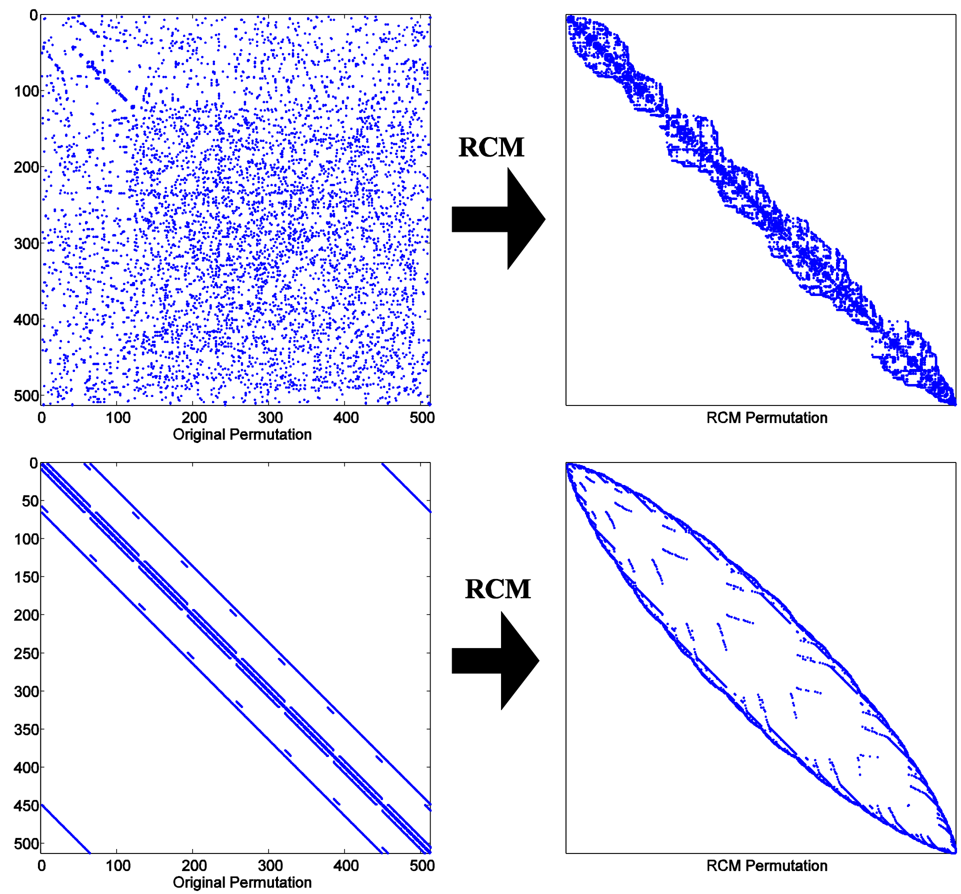
Mapping Heuristics (2/3)

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)



Mapping Heuristics (3/3)

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

Practical Issues – A TopoMapper Library

1. Getting the network topology statically
   - Query each network, generate adjacency list file
   - Key is the hostname (must be unique)

Interconnection Network (API)	**Topology Query Tool(s)**
Myrinet (MX)	`fm_db2wirelist`
InfiniBand (OFED)	`ibdiagnet & ibnetdiscover`
SeaStar (Cray XT)	`xtprocadmin & xtdb2proc`
BlueGene/P (DCMF)	`DCMF API`

2. Querying the topology and location
   - Only supported on BlueGene (DCMF personality)

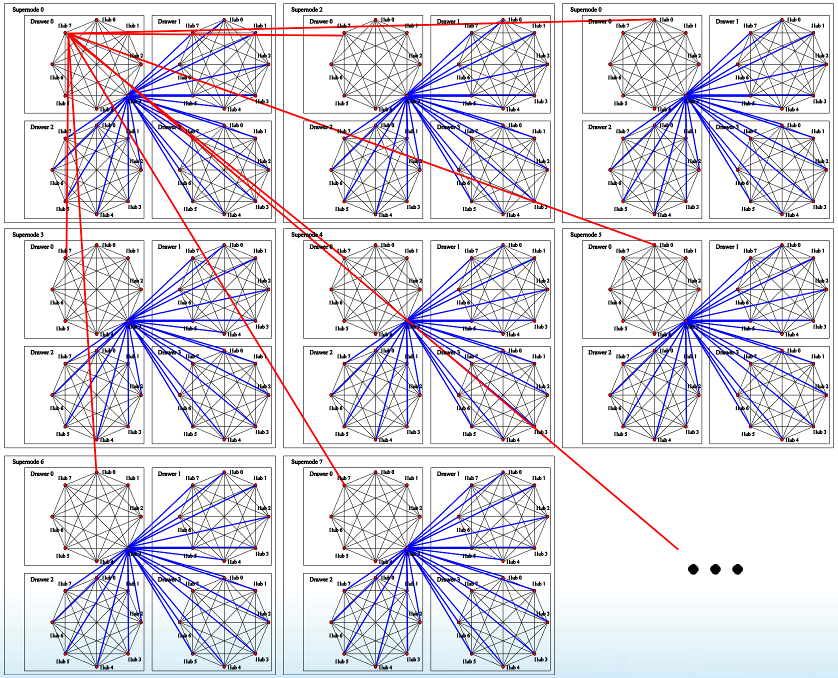
Experimental Evaluation - Methodology

- 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)

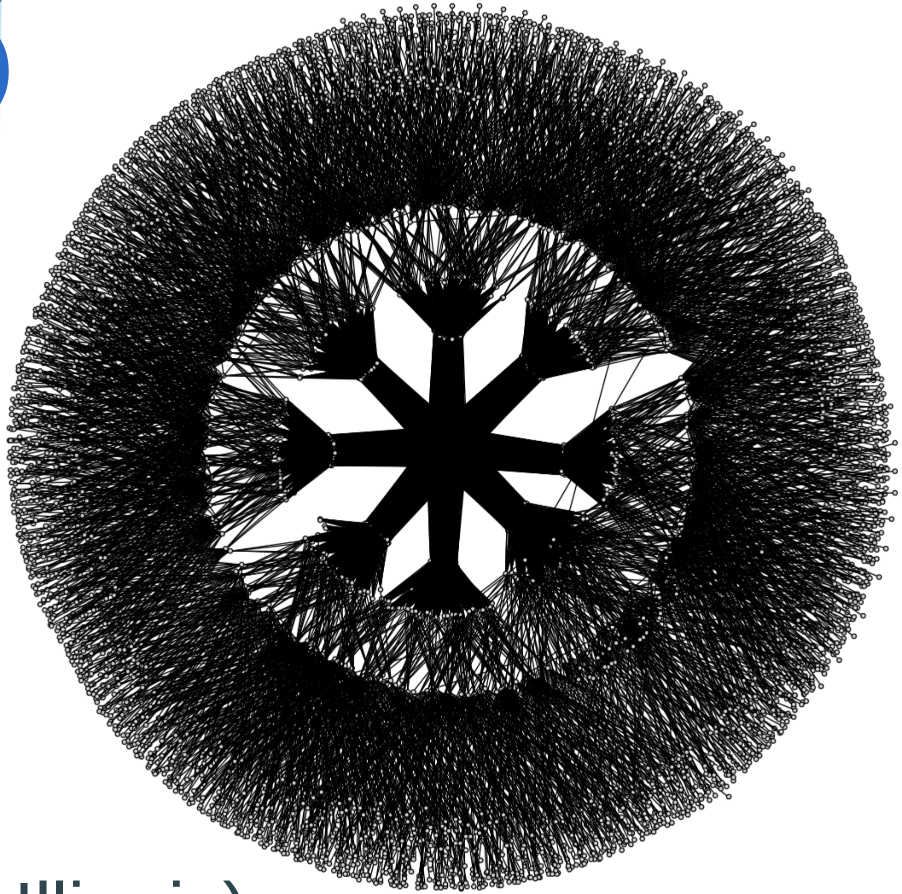
Matrix Name	Rows and Columns	NNZ (sparsity)
F1	343,791	26,837,113 $(2.27 \cdot 10^{-4}\%)$
audikw_1	943,695	39,297,771 $(4.4 \cdot 10^{-5}\%)$
nlpkkt240	27,993,600	401,232,976 $(5 \cdot 10^{-7}\%)$

Network Topologies (1/2)

1. 3d Torus
   - x=y=z for maximum bisection
   - 3^3 to 12^3 maximum map file size: 31.2 kiB
2. PERCS
   - 7 LL, 24 LR links
   - Assuming 9 D-links
   - Total of 9248 nodes
   - Map file size: 1455 kiB

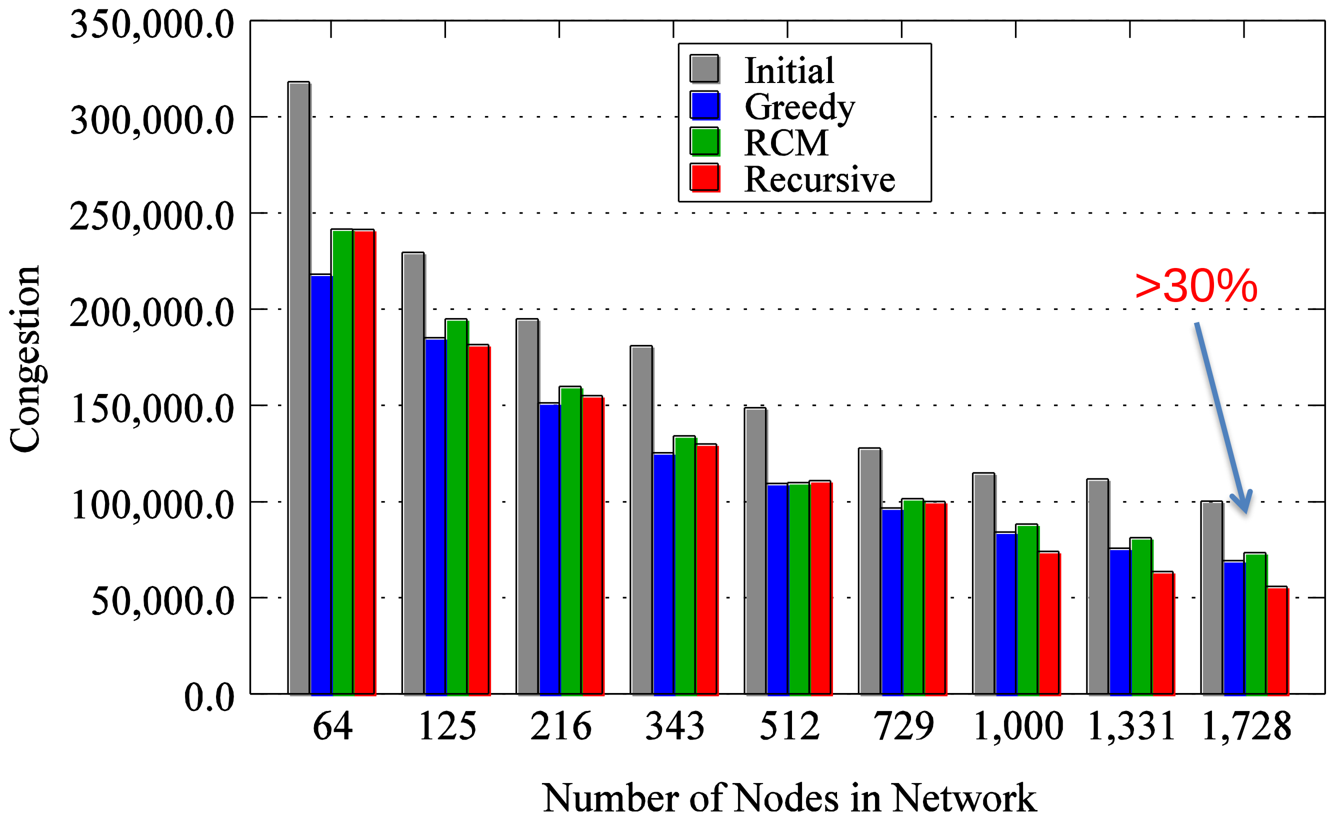


Network Topologies (2/2)



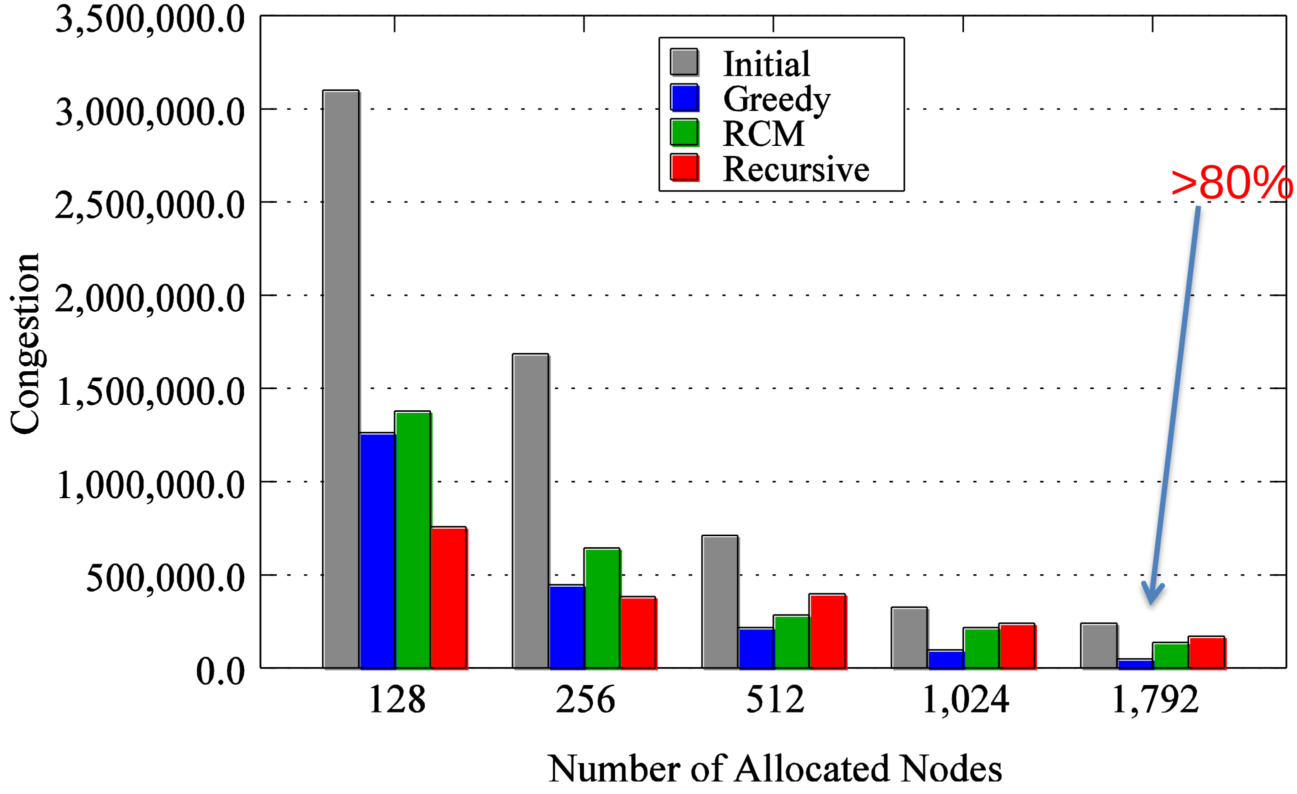
- Juropa (JSC, Germany)
  - 3292 endpoints
  - Map file size: 87 kiB
- Ranger (TACC, Texas)
  - 4081 endpoints
  - Map file size: 134 kiB
- Surveyor (BG/P, Argonne, Illinois)
  - Queried during runtime

Congestion on Torus Networks



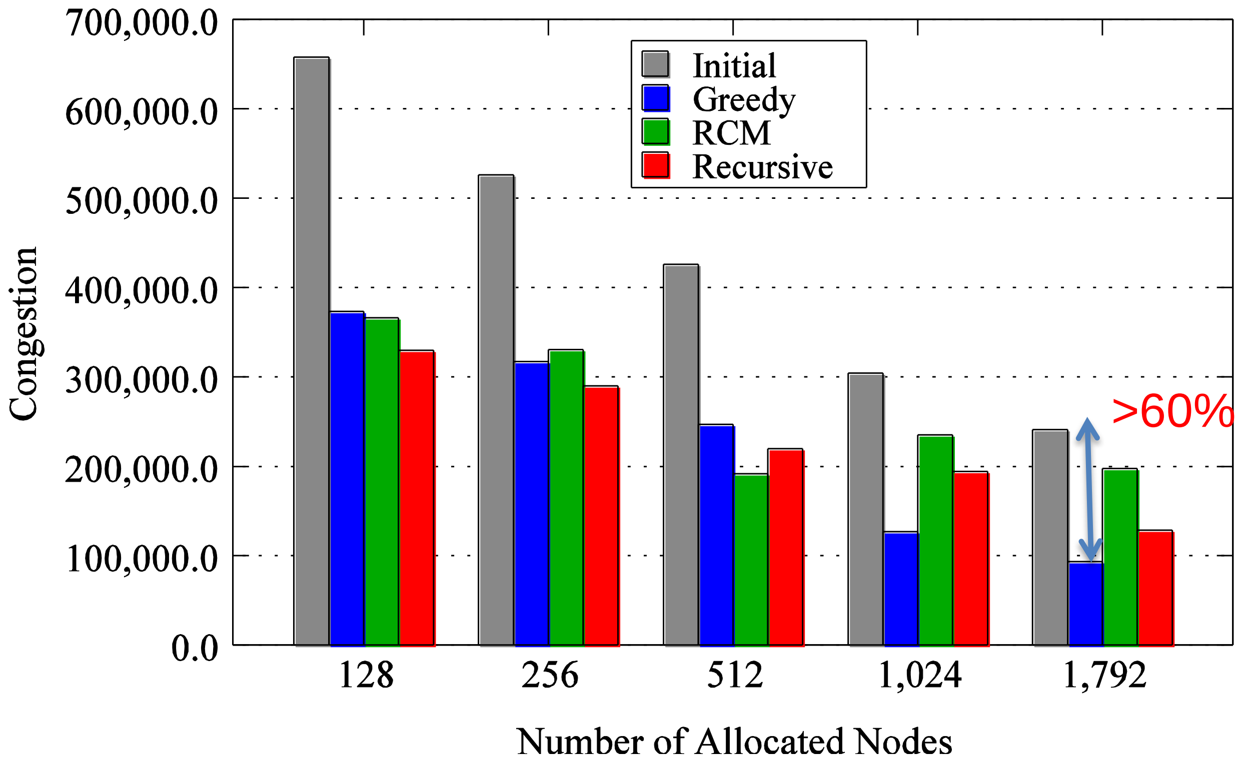
- nlpkkt240, dilation for 12^3: 9.0, 9.03, 7.02, 4.5
- Times for 123: <0.01s, 1s, 1s, 10 min

Congestion on PERCS



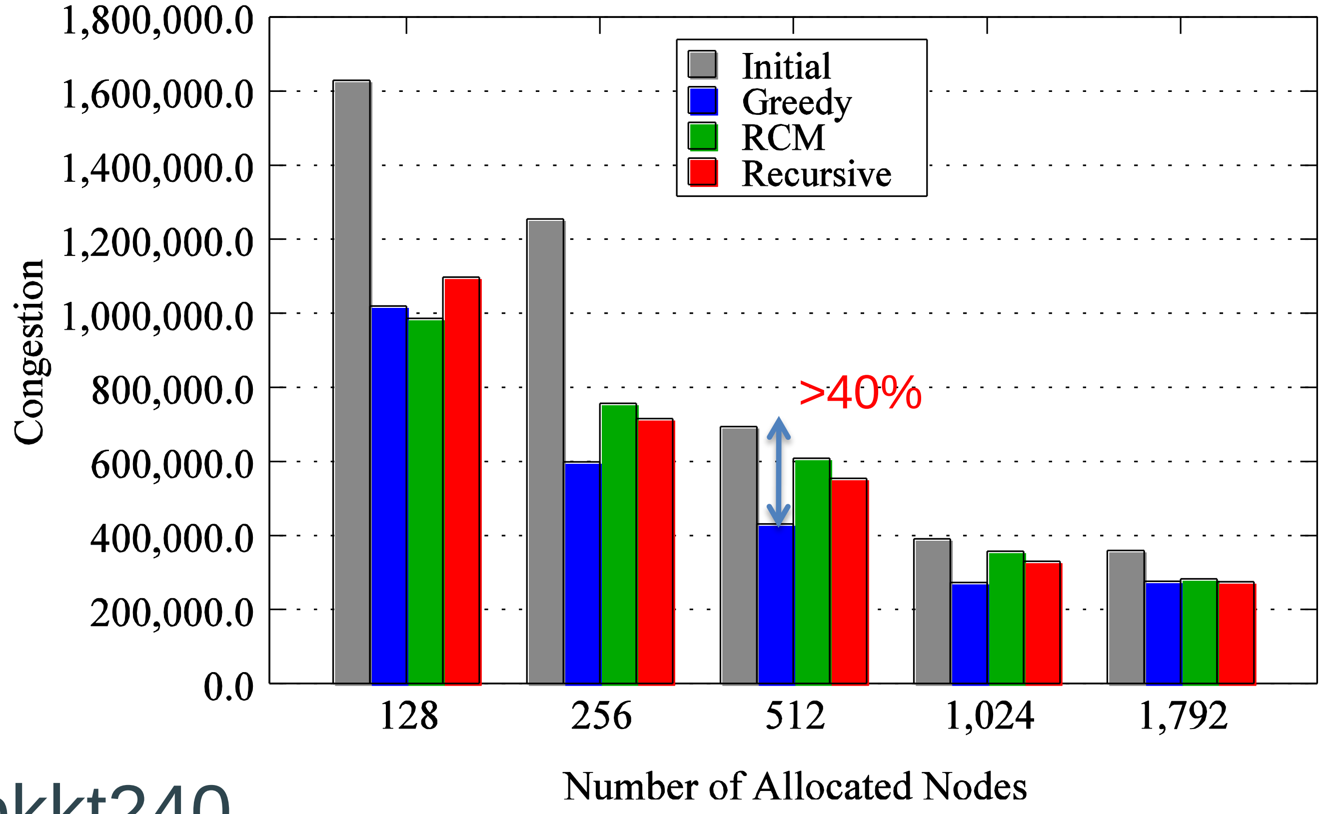
- nlpkkt240, packed allocation, dilation: all ~2.5
- Times: <0.01s, 0.8-22s, 4.5-7.5s, >41 min

Congestion on Juropa



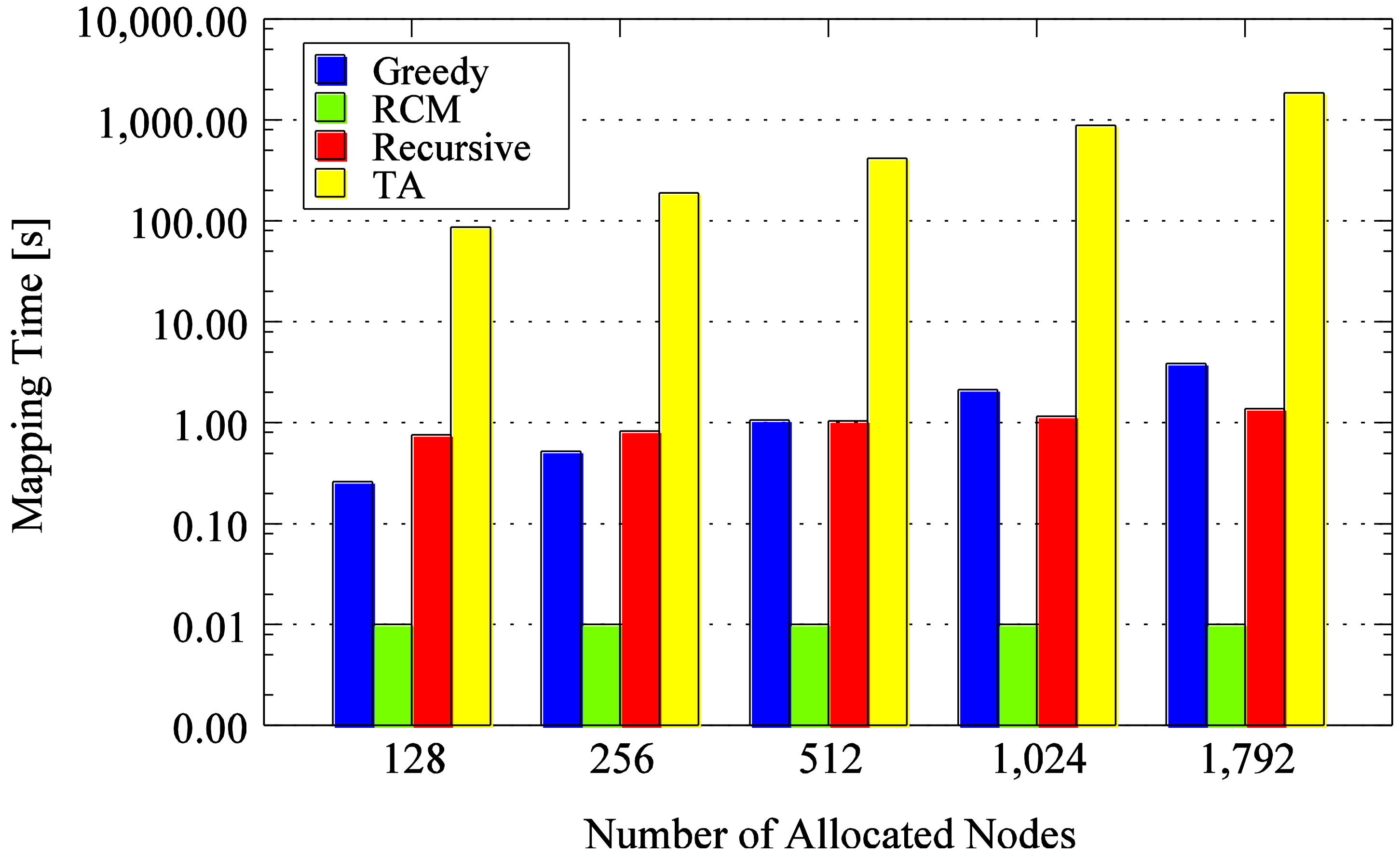
- audikw_1, dilation: 5.9, 5.8, 4.45, 5.13
- Times: <0.01s, 0.16-2.6s, 0.63-1.21s, 9 min

Congestion on Ranger



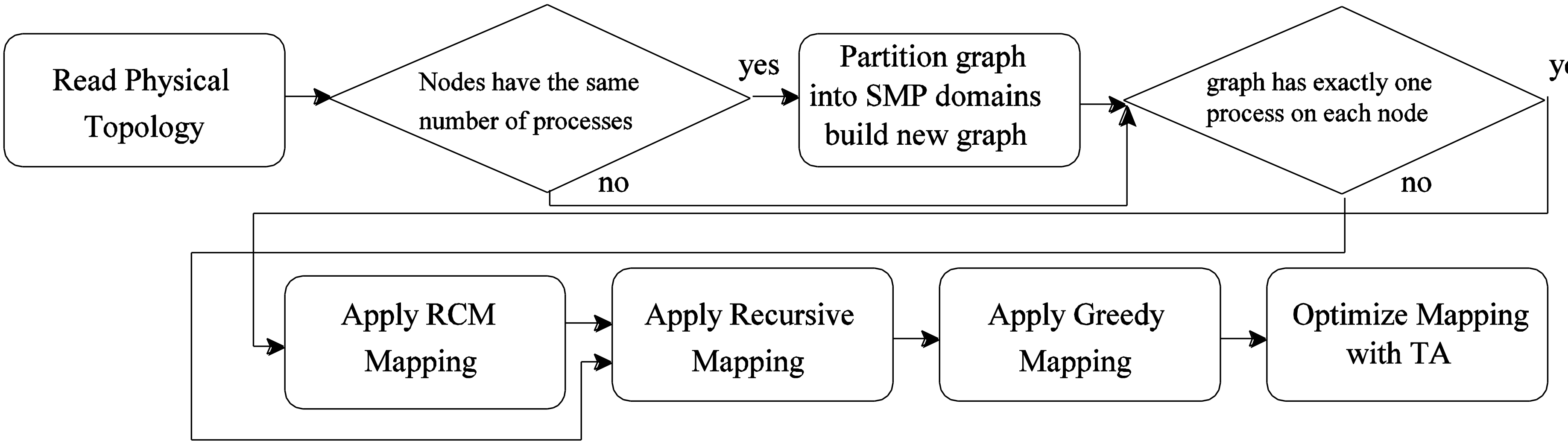
- nlpkkt240
- times: <0.01s, 0.26-3.85s, 0.76-1.5s, 0.5-14 min

Mapping Times Scaling (Ranger)



A Practical Mapping Strategy

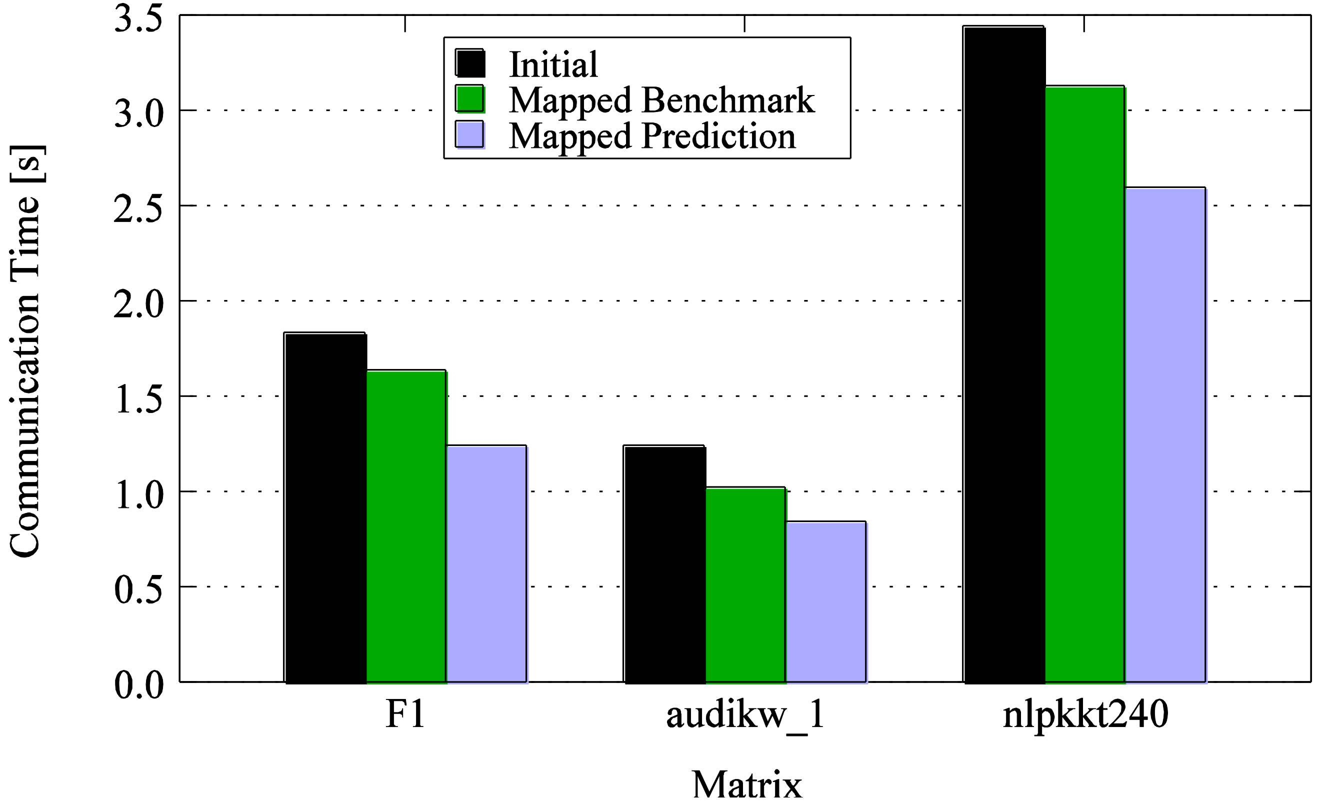
- P cores are available – use all of them!
  - All heuristics in parallel, greedy varies start processes



Real-World Benchmarks

- Used Surveyor which has close-to-optimal routing
- Load matrix, partition with ParMETIS
  - Construct MPI-2.2 distributed graph toplogy
  - Apply topology mapping
  - Re-distribute data
- Perform timed sparse MatVec
  - Report time for 100 communication phases
  - Maximum time across all ranks

Benchmark Results



- 512 nodes, up to 18% improvement measured

Thanks and try it at Home!

- LibTopoMap (download tools and library)
  http://www.unixer.de/research/libtopomap
- Conclusions
  - Topology mapping is feasible at large scale
  - Good heuristics exist and should be implemented in MPI-2.2
- Future Work
  - Exploit structure for faster optimal algorithms (e.g., Cartesian)
  - Consider intra-node costs