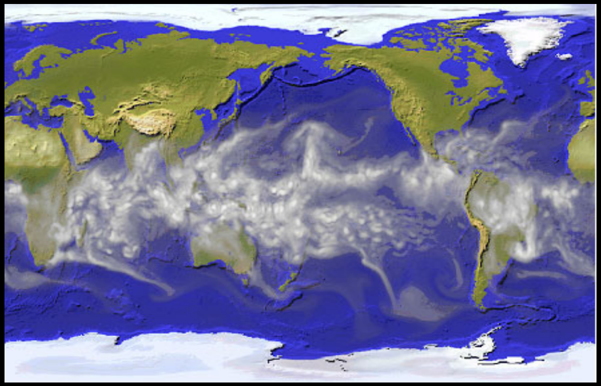
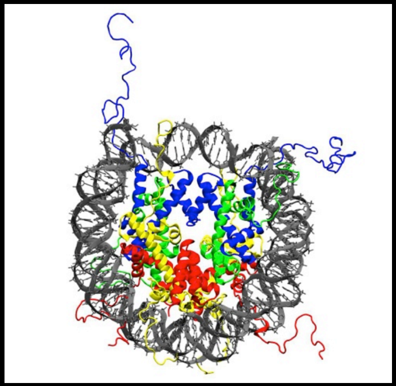
OPTIQ: A Data Movement Optimization Framework for Data-centric Applications on Supercomputers

Huy Bui
Electronic Visualization Lab.
Dept. of Computer Science
University of Illinois at Chicago, USA

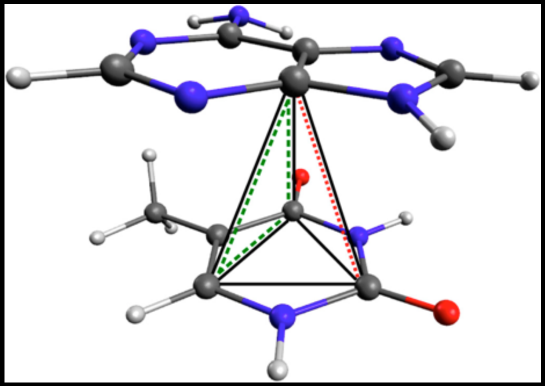
Supercomputers and their applications



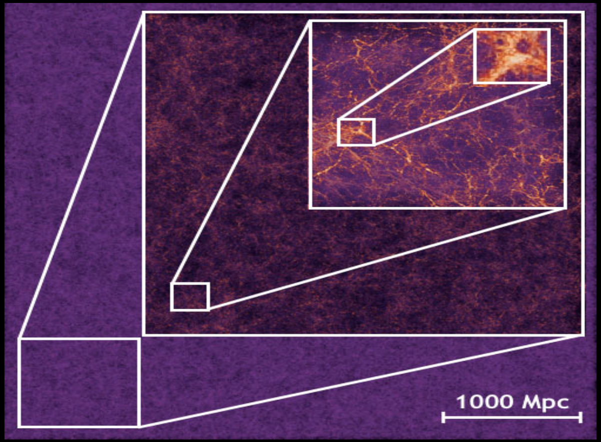
Attributing Changes in the Risk of Extreme Weather and Climate (150 Million Core-Hours)



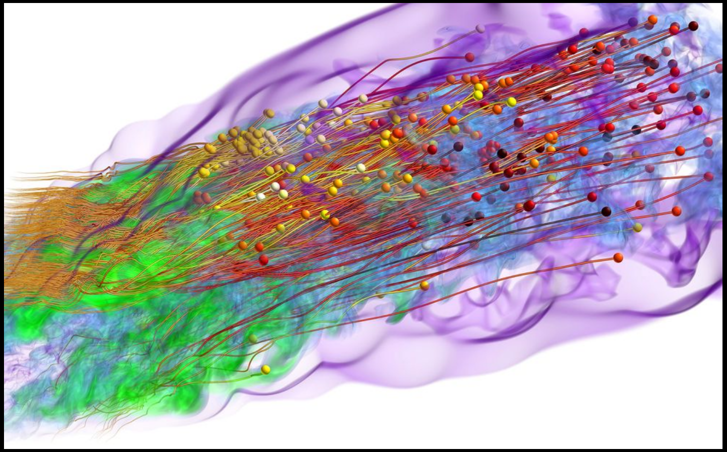
Computational Studies of Nucleosome Stability (20 Million Core-Hours)



Toward Crystal Engineering from First Principles (12 Million Core-Hours)



Computing the Dark Universe (40 Million Core-Hours)



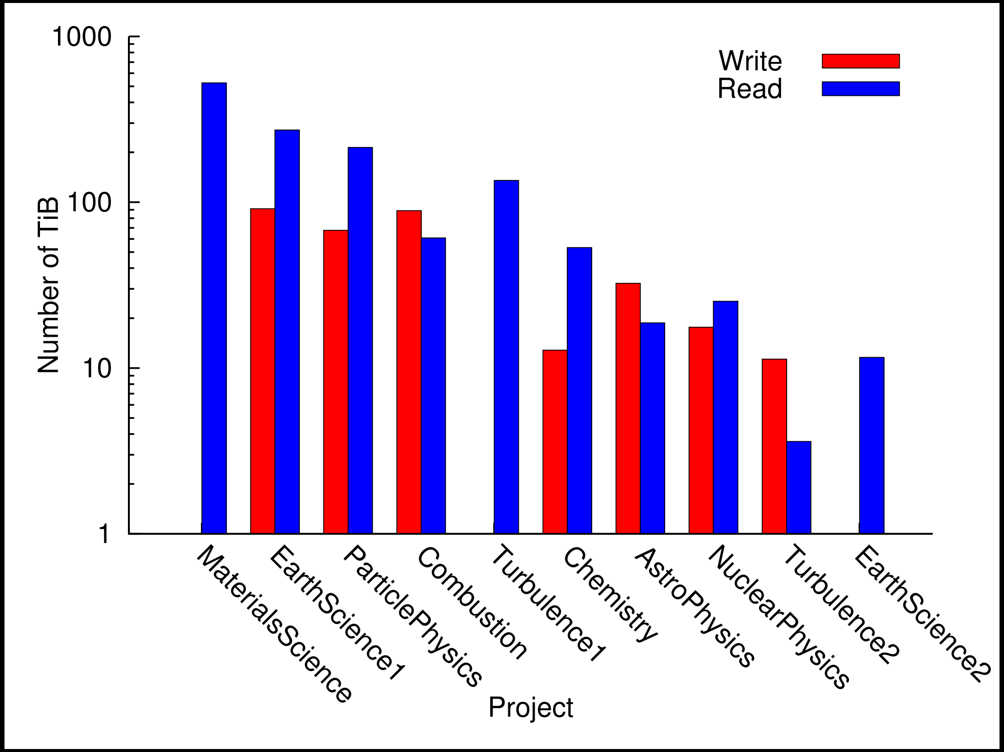
Simulation of combustion engine (113 Million Core-Hours)

evl

Electronic Visualization Laboratory, University of Illinois at Chicago

Data-centric applications on supercomputers

- Most of the applications are data-centric i.e. generating a large amount of data.



Total amount of read/write data per simulation in some applications.

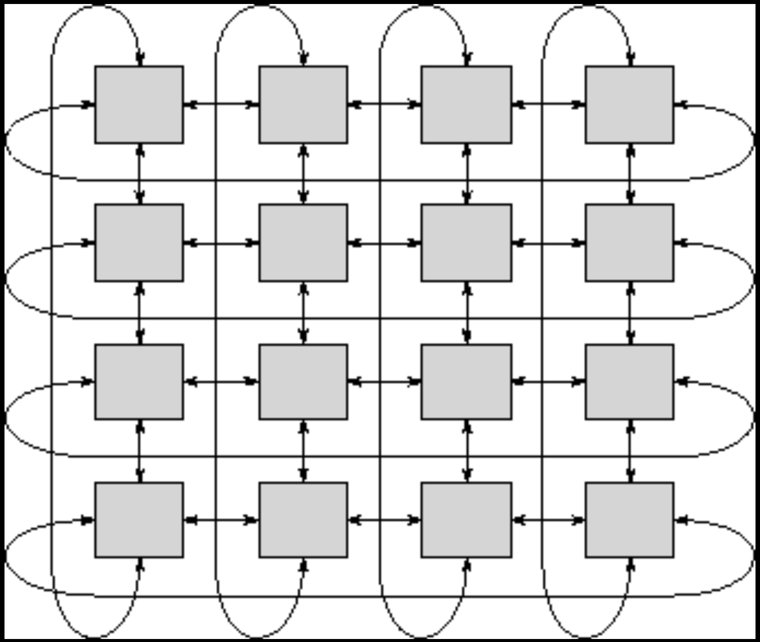
Supercomputers meet Big Data

- From *Exascale(10^{18} PFLOP/s) computing study: Technology challenges in achieving exascale and systems* and *Synergistic Challenges in Data-Intensive Science and Exascale Computing* reports:
  - HPC has been compute-intensive, but is shifting toward data-centric computing.

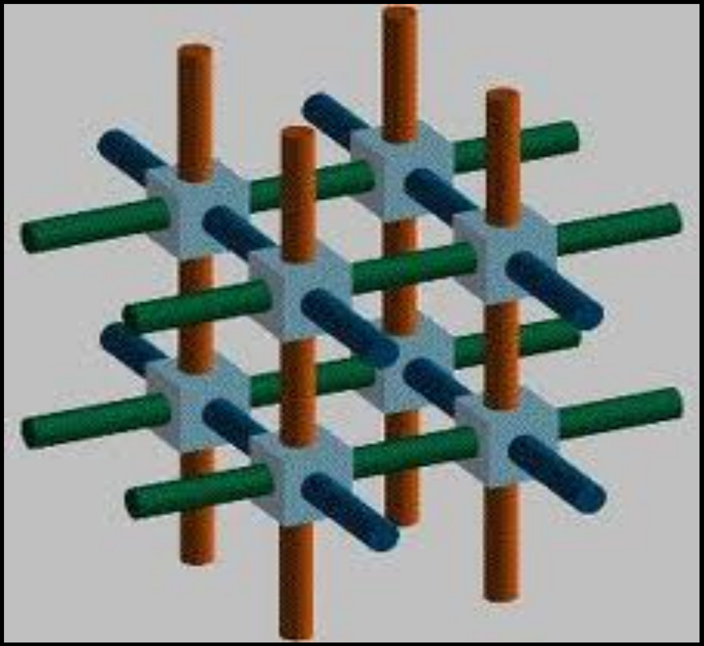
  ⇒Data is a big challenge in supercomputing.

Supercomputer's Interconnection Network

- A supercomputer includes ten thousands compute nodes and high throughput and low latency interconnect network.



2D Torus Topology



3D Topology

Data Movement and Optimization Input in Supercomputers

Layers	Data movements	Optimization Inputs			
		Topology	System routing	Comm. routines	Comm. patterns
Applications	Data flows from sources to destinations	No	No	No	Yes
Middleware (MPI, PGAS...)	Communication routines such as MPI_Send, MPI_Broadcast	No	Yes		No
Systems	Packets/ messages routing	Yes		Limited	No

Data movement and optimization inputs at different layers

Information lost when data is moved between layers.

Data Movement in Data-Centric Application

- Separate optimizations produce local optimization.
- This thesis proposes solutions to improve data movement performance for data-centric applications in supercomputing systems.
  - Optimizing data flows: holistic approach to take system routings, interconnection topology and application communication patterns into formulation.
  - Realizing in to Data Movement Optimization framework (OPTIQ).

Current status of data movement optimization and proposed solutions

Desired Features	Optimization: Input parameters			
Layers	Interconnect topology	System routing	Comm. routines	Comm. patterns
Applications	No	No	No	Limited
Middleware libraries	Limited, system-specific	Yes (wrappers)	No	No
System	Yes		Limited	No
OPTIQ	Yes	Yes	No	Yes

Data movement and optimization inputs at different layers and OPTIQ

OPTIQ takes more inputs promising higher throughput.

Related Work

Related work

- Optimizing data movement.
  - System routing.
    - Static routing: pre-compute paths to move data.
      - Mathematical model based optimization.
      - Heuristic approach.
      - Pros: Optimized for certain routines, fast at runtime.
      - Cons: not optimized for flows, not adapt to state-of-the-art traffic.
    - Adaptive routing: compute paths instantly based on current status of traffic at local regions (sources and destination).
      - Randomized routing, Minimal routing.
      - Pros: fast to adapt to current traffic.
      - Cons: Not globally optimal.

Related work

- Optimizing data movement.
  - General work.

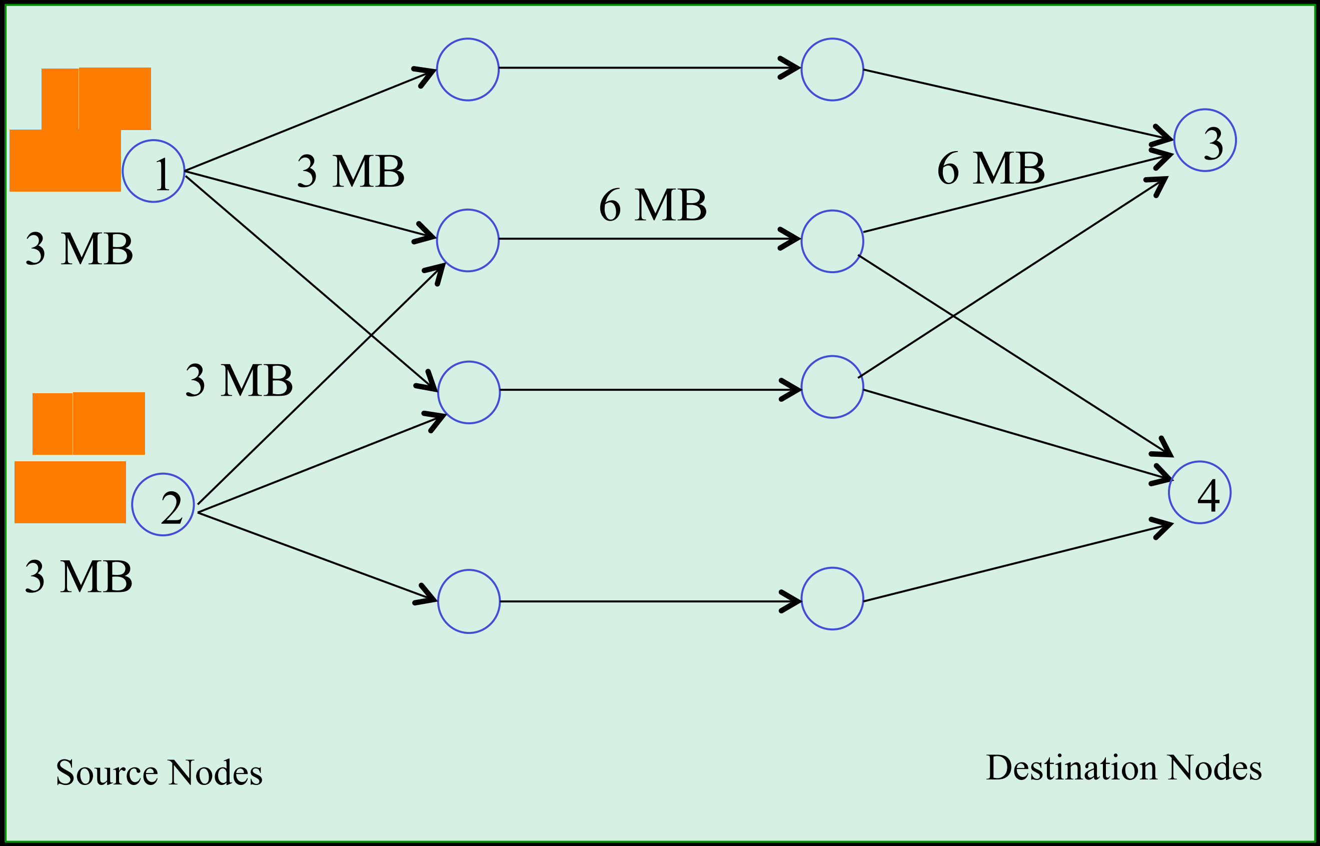
Works	Static	Dynamic	Network Domain	Optimizationt echnique	@Layer	Scale
Valiant'81		Yes	General	Randomized	System	Large
Khana'08	Yes		Grid	Liner Programming	Libraries	Small
Rodriguez'09		Yes	Fat Tree	Heuristics	System	Small
Prisacari'13a	Yes		Fat Tree	Heuristics	System	
Prisacari'13b	Yes		Generalized Fat Tree	Integer Linear Programming	System	x1000

Related work

- Optimizing data movement.
  - Related work on recent supercomputers.

System	Static	Dynamic	Network Domain	Opt technique	@Layer	Scale
BG/L,P	Yes		3D Torus	Heuristics	System	Large
BG/Q	Small net	Large net	5D Torus	Heuristics	System	Large
Cray		Yes	Fat Tree	Heuristics	System	Large
BG/Q	Yes		5D Torus	Heuristics	Middleware	Large

Inefficiencies of data movement in current systems and solutions



Multi-paths data movement can improve performance.

OPTIQ Framework

OPTIQ Framework

Communication demand: sources, destinations, data (sizes, buffers). Graph of nodes and links.

OPTIQ

1. Path generation
   k shortest paths for each pair of communication
   — Reduce search space

2. Path searching
   - Optimization
   - Heuristics
   — Search paths and assign data to path

3. Schedule
   Split and put messages into queues to be sent out

4. Transport
   Transfer data from sources to destinations

OPTIQ Framework (cont.)

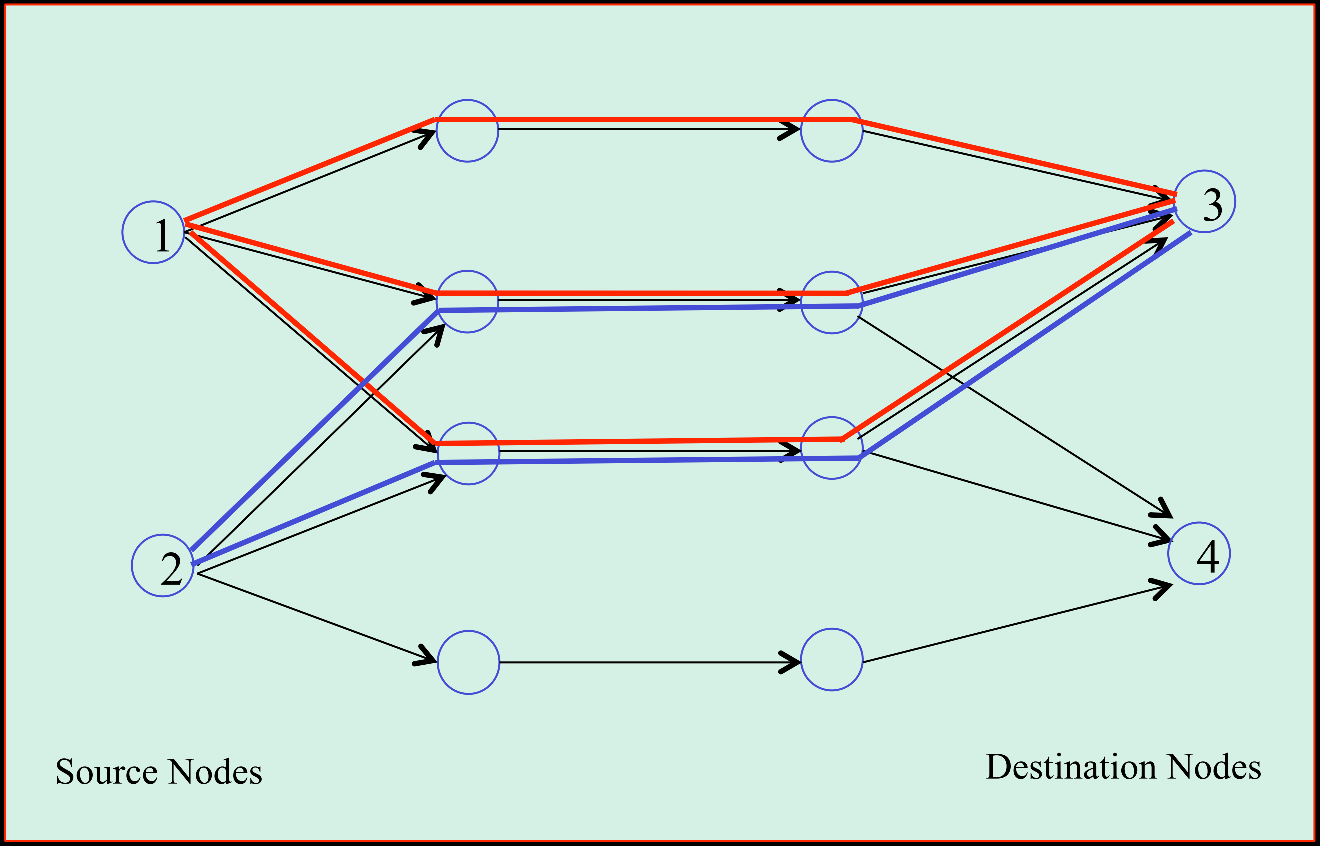
- OPTIQ framework:
  - The framework exposes a simple API that can be used in applications with minimal changes.
  - New features can be added easily: algorithm to search for paths, scheduling, transport.
  - It is also extensible to different systems.

Multi-path Data Movement

Path Generation

- Using any k-shortest paths algorithms to generate k shortest paths between a pair of source and destination.
- Pruning paths with length more than a certain number of hops e.g. diameter of partition of compute nodes.

Path Generation (cont.)

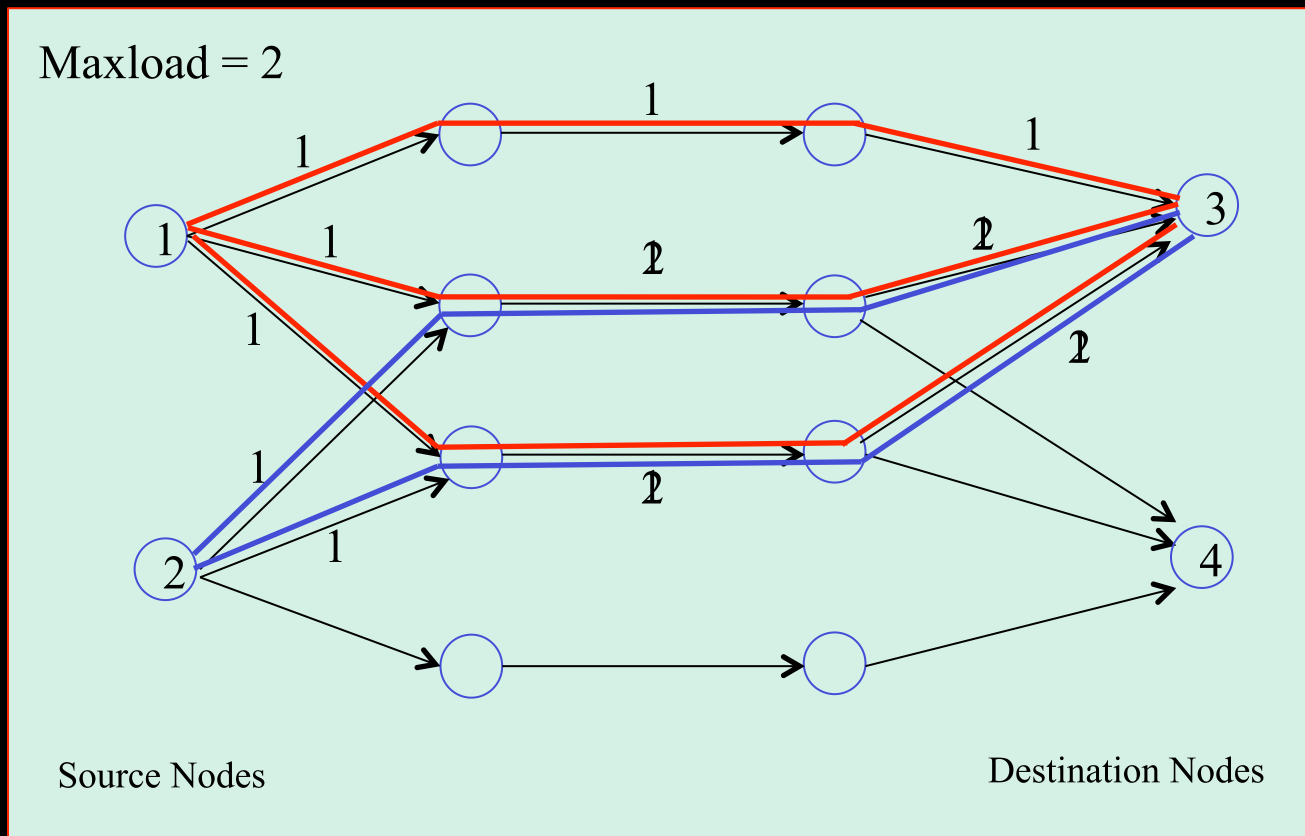


Heuristic Algorithm 1

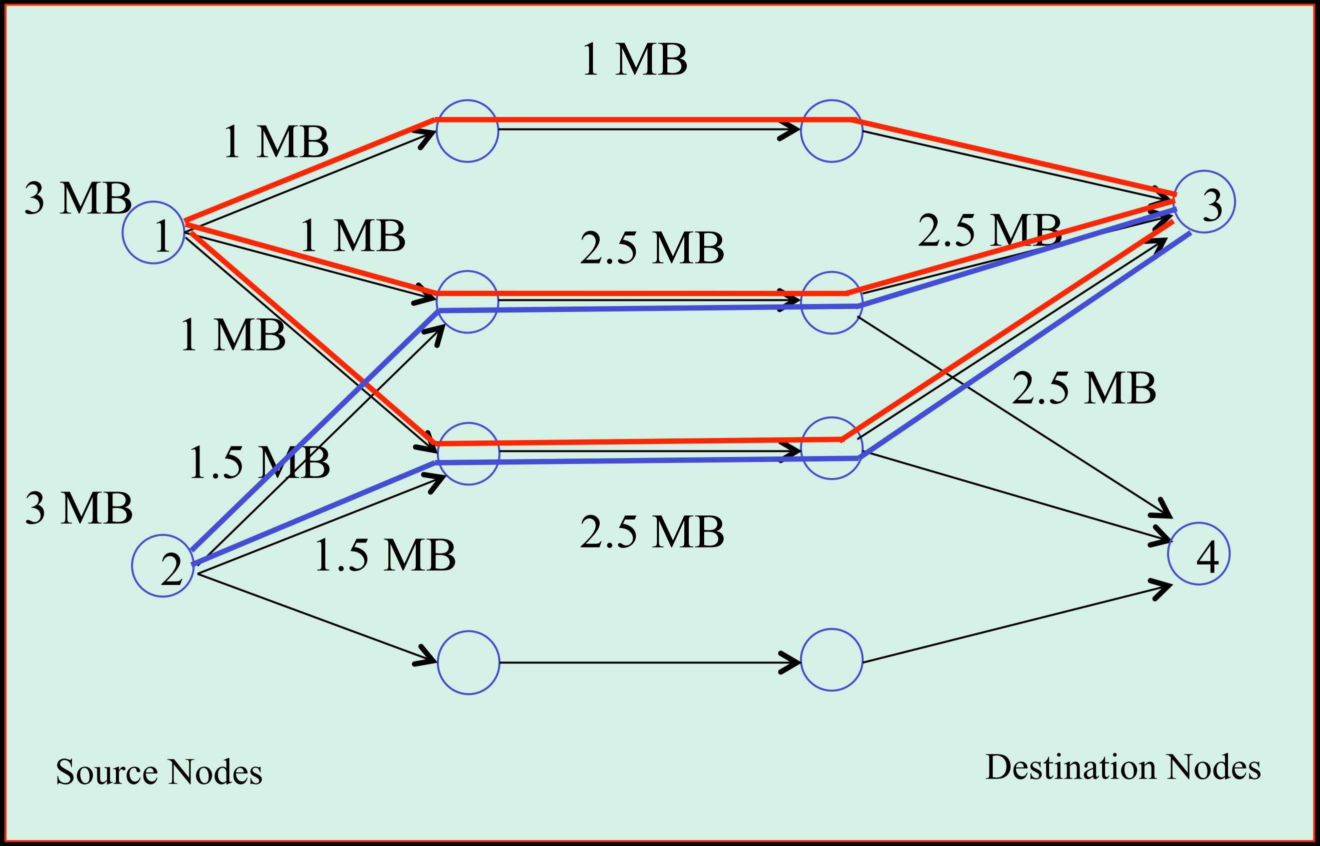
- Assumptions:
  - All pairs of communication have similar number of paths and similar data size per path.
- Goal:
  - Limit the maximum number of paths per link by a given *maxload* value.

  $\Rightarrow$ Limit the data passing through any link.
- Algorithm:
  - Iterate through all pairs.
    - Pick one path per pair at a time.
    - Update load of links and selected paths.
    - If any load on links is over *maxload* value, terminate.

Heuristic Algorithm 1 (cont.)



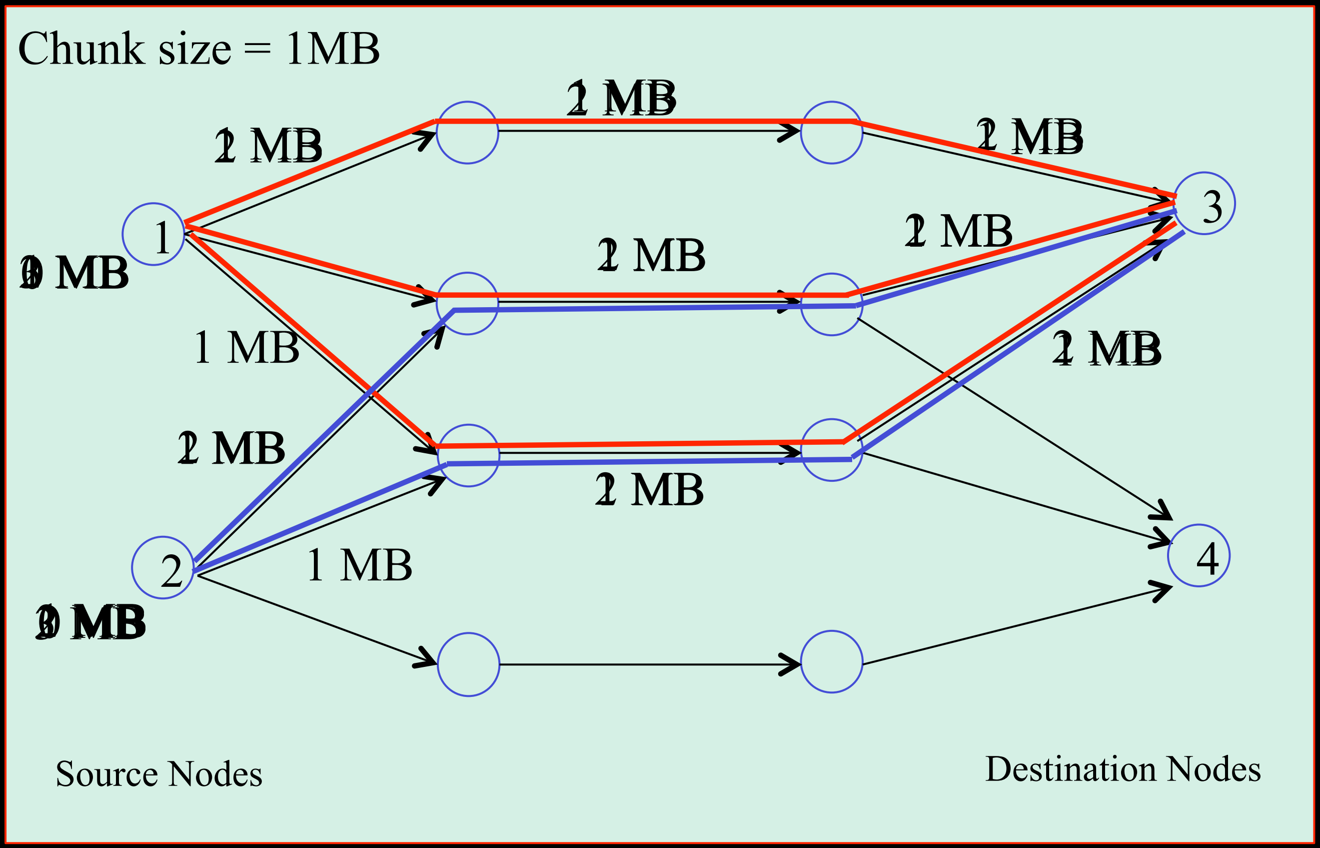
Heuristic Algorithm 1 (cont.)



Heuristic Algorithm 2

- Assumptions:
  - Data size can be different. Number of shortest paths per pair can be different.
- Goal:
  - Minimize the maximum data passing through any link.
- Algorithm:
  - Sort all pairs by data size (demand). The job with largest demand at top.
  - Pick the job at top, assign part of its demand (chunk) to one of its paths with lowest demand. Update the demand of the jobs and loads on links.
  - Repeat until all demands are assigned.

Heuristic Algorithm 2 (cont.)



Heuristic 1 vs. Heuristic 2

Comparison factor	Heuristic 1	Heuristic 2
Load on physical link	Number of paths that used a link.	Actual amount of data passing through a link.
Pair iteration	Each pair 1 time to get 1 path.	The pair that has the largest amount of remaining data.
Need to know amounts of data in advance	No	Yes

Comparison of Heuristic 1 vs. Heuristic 2.

Model-based Data Movement Optimization

- Problem modeling:
  - Given a set of Jobs and set of Paths for each job in Jobs. Each jobs has Demand[job], flow on path p flow[job, p].
  - Each edge (i,j) has capacity c(i, j).
  - Objective function:
    - Minimize the transfer time t.
  - Capacity constraint:

$$\sum_{\forall job \in Jobs} \sum_{\forall p \in kpaths_{job}} flow[job, p]_{i,j} \leq c(i, j)$$

  - Throughput constraint:

$$\sum_{\forall p \in kpaths_{job}} flow[job, p] = \frac{Demand[job]}{t}$$

Evaluations

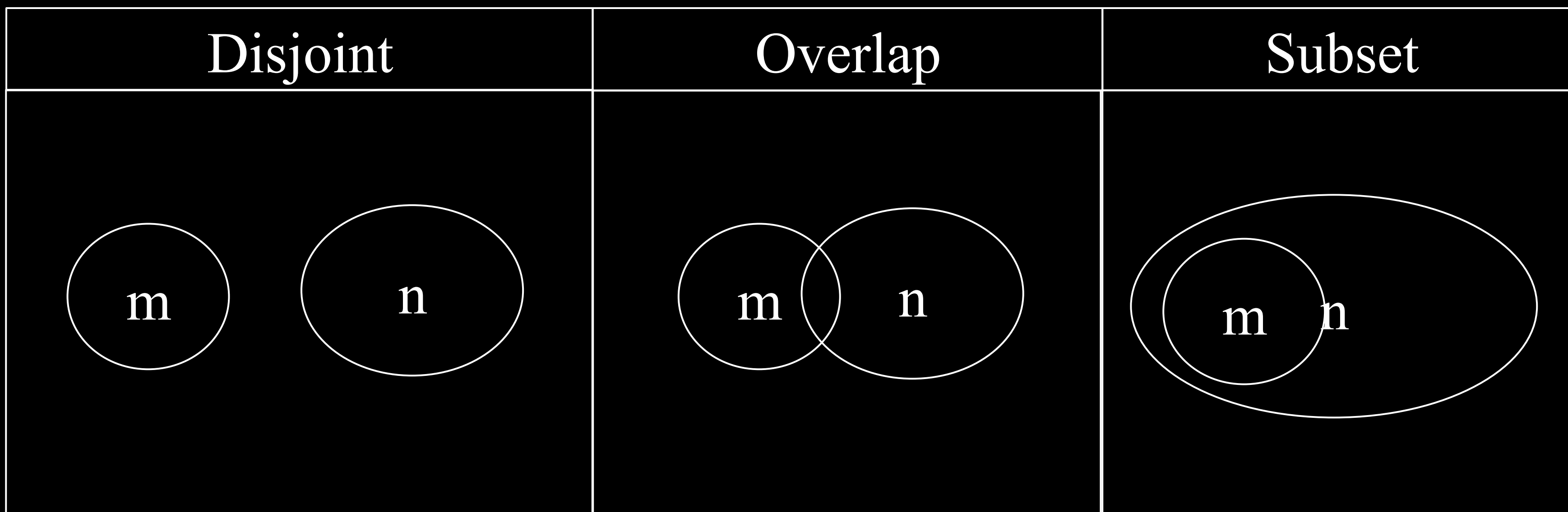
Blue Gene/Q supercomputers

- Mira: 5th in top 500, 48K nodes, 10PF/S, 5D torus.

Mira - a Blue Gene/Q supercomputer at Argonne National Laboratory

Communication Patterns

- 3 main communication patterns.



3 main communication patterns that used by most of applications.

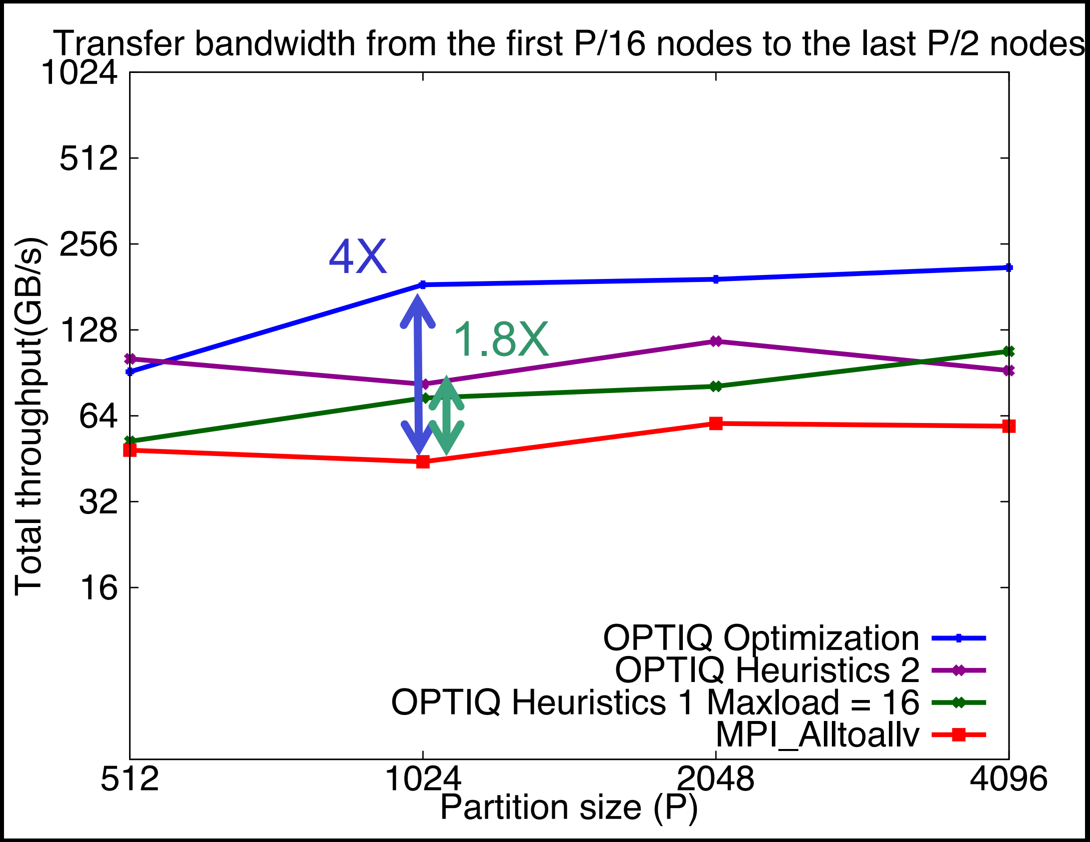
- 91 experiments with Optimization, Heuristics 1 & 2 and MPI default mechanism MPI_Alltoallv.

Set of Experiments

- Scaling total number of nodes.
- Varying sources-destinations distance.
- Varying sources/destinations ratio.
- Random sources-destinations pairing.
- Paths searching time.
- Experiments on 2 applications.

Scaling total number of nodes

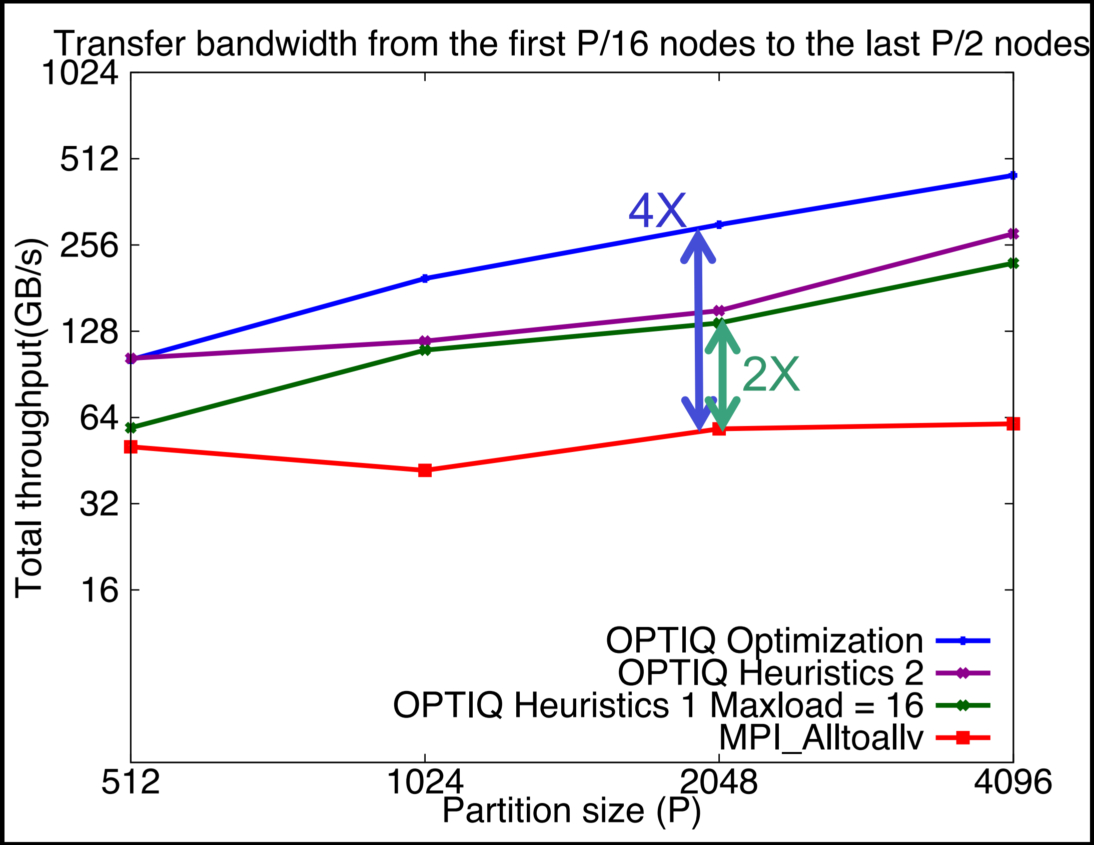
- Same message size – Disjoint pattern.



OPTIQ outperforms MPI at scale for disjoint pattern.

Scaling total number of nodes (cont.)

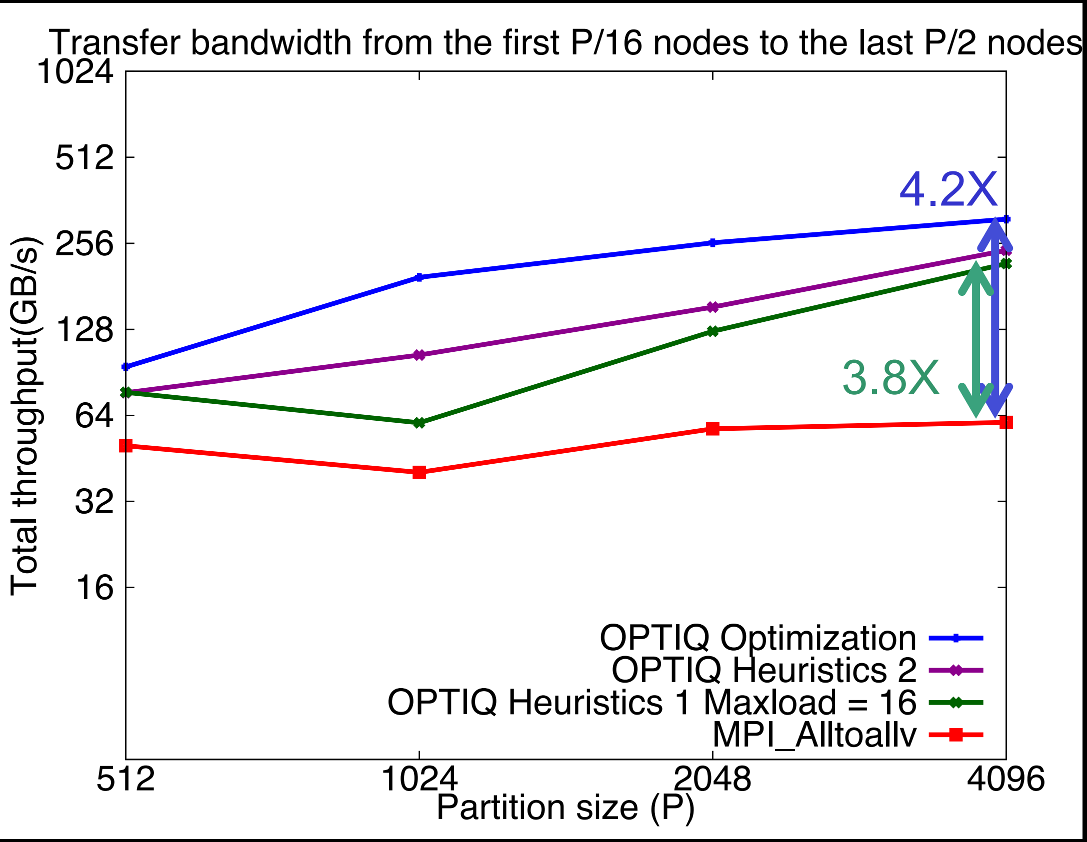
- Same message size – Overlap pattern.



OPTIQ outperforms MPI at scale for overlap pattern.

Scaling total number of nodes (cont.)

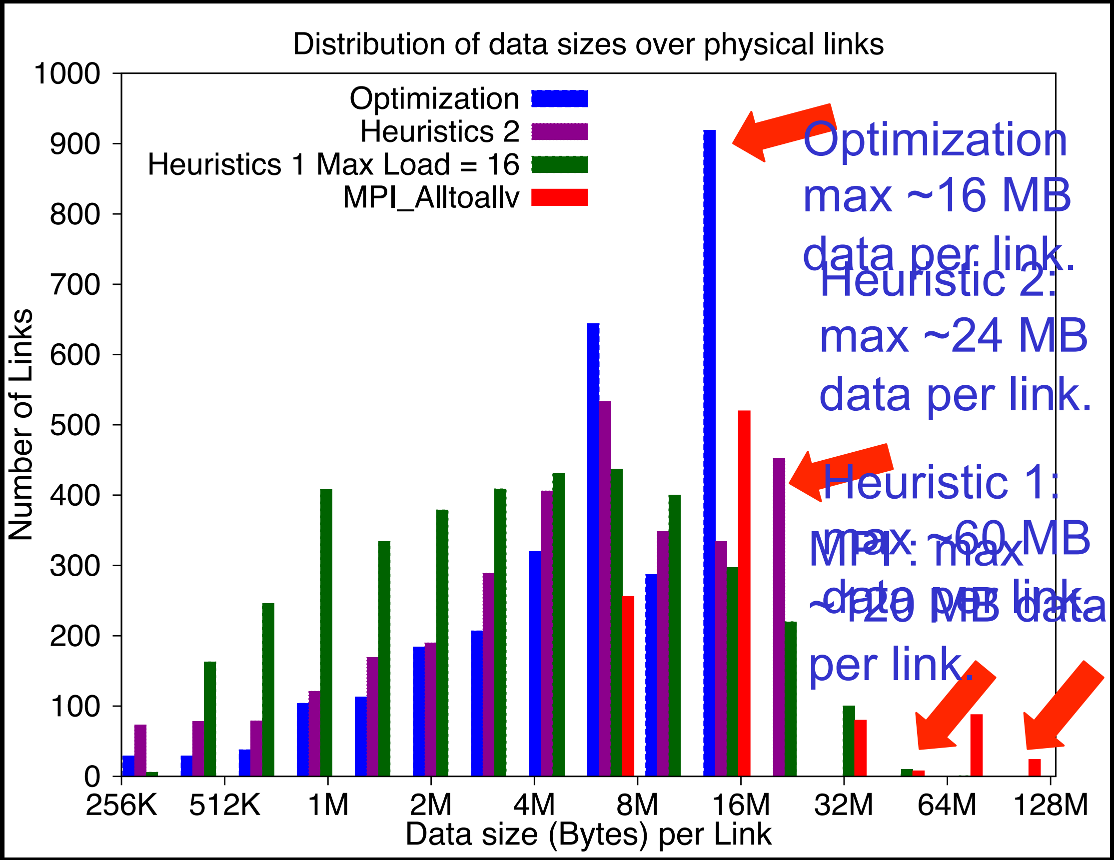
- Same message size – Subset pattern.



OPTIQ outperforms MPI at scale for subset pattern.

Scaling total number of nodes (cont.)

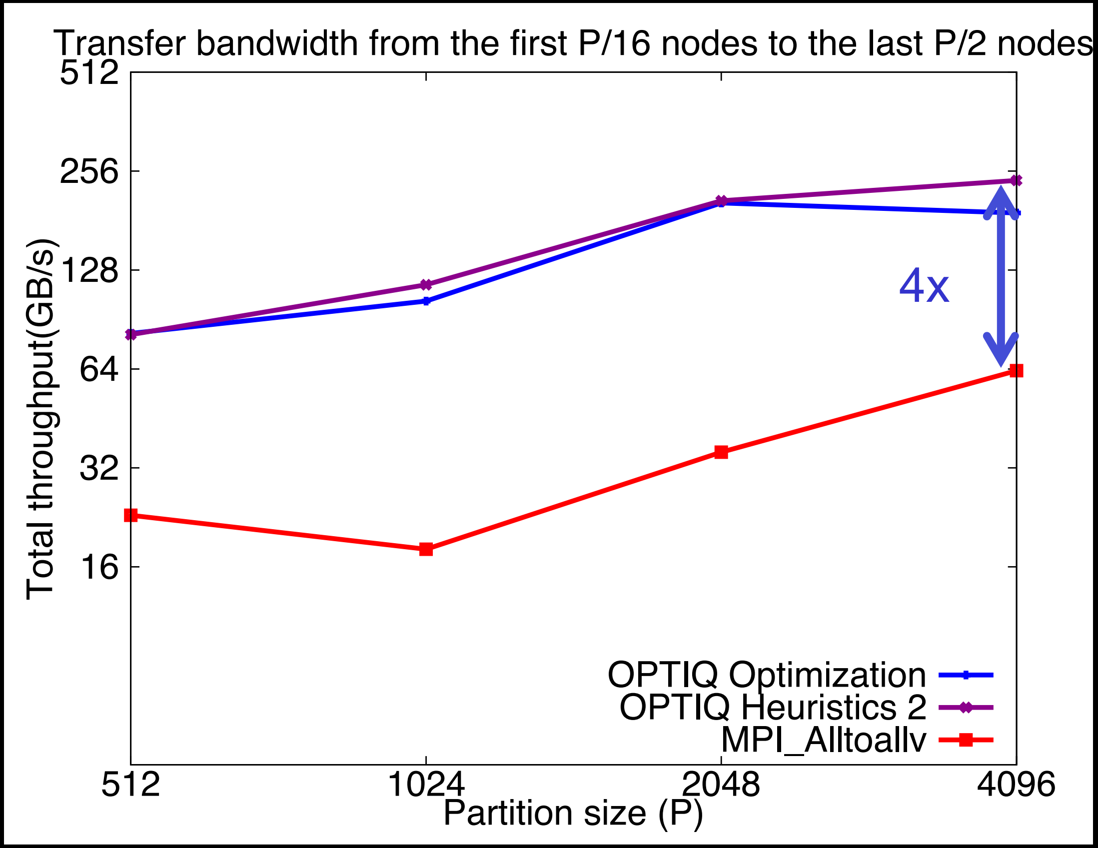
- Same message size – Data distribution on physical network links.



The better data distribution the higher performance.

Scaling total number of nodes (cont.)

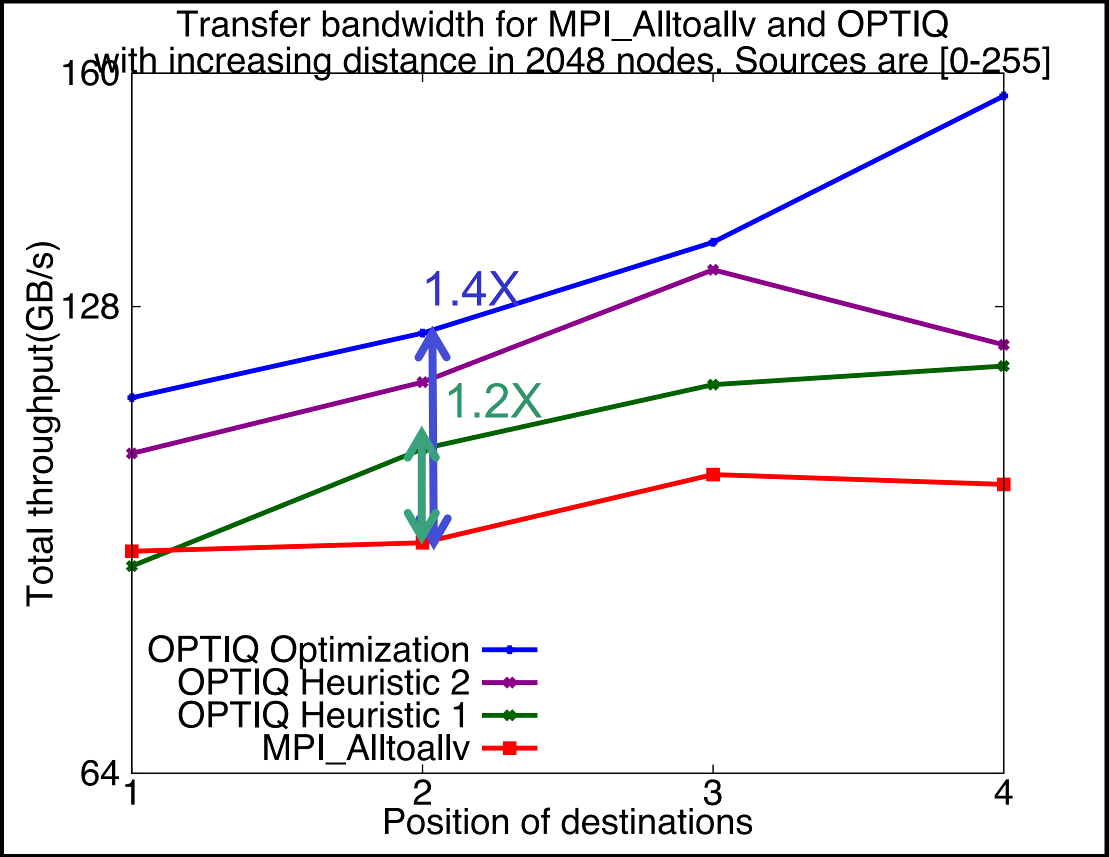
- Random message sizes: Overlap pattern.



OPTIQ outperforms MPI with various message sizes.

Varying Source-Destination Distance

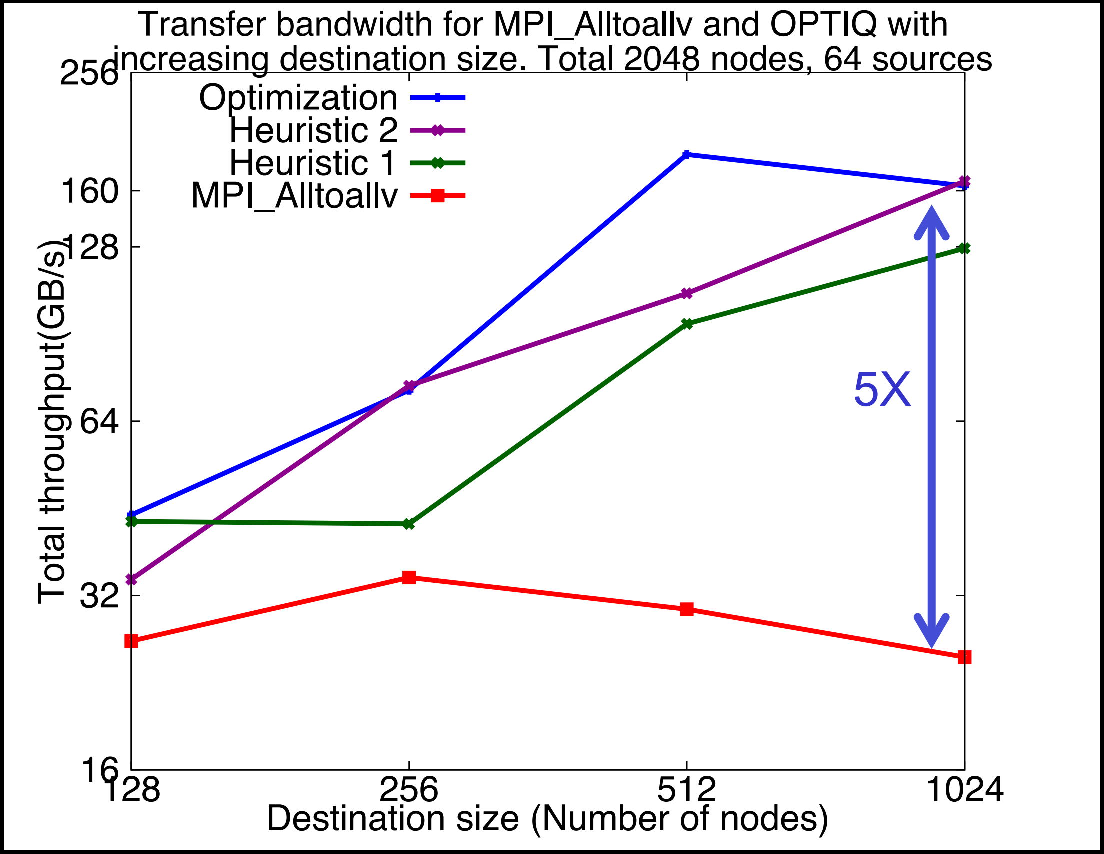
- Increase source-dest distance: Disjoint pattern.



Increasing distance can improve throughput.

Varying Source-Destination Ratio

- Increasing source/destination ratio – Disjoint pattern.



OPTIQ outperforms MPI when increasing ratio.

Random Source-Destination Pairing

- Experiment on 512 nodes: Random pairing sources and destinations for disjoint patterns (32 source nodes to 256 destination nodes) for 5 times. Collect and report average performance.

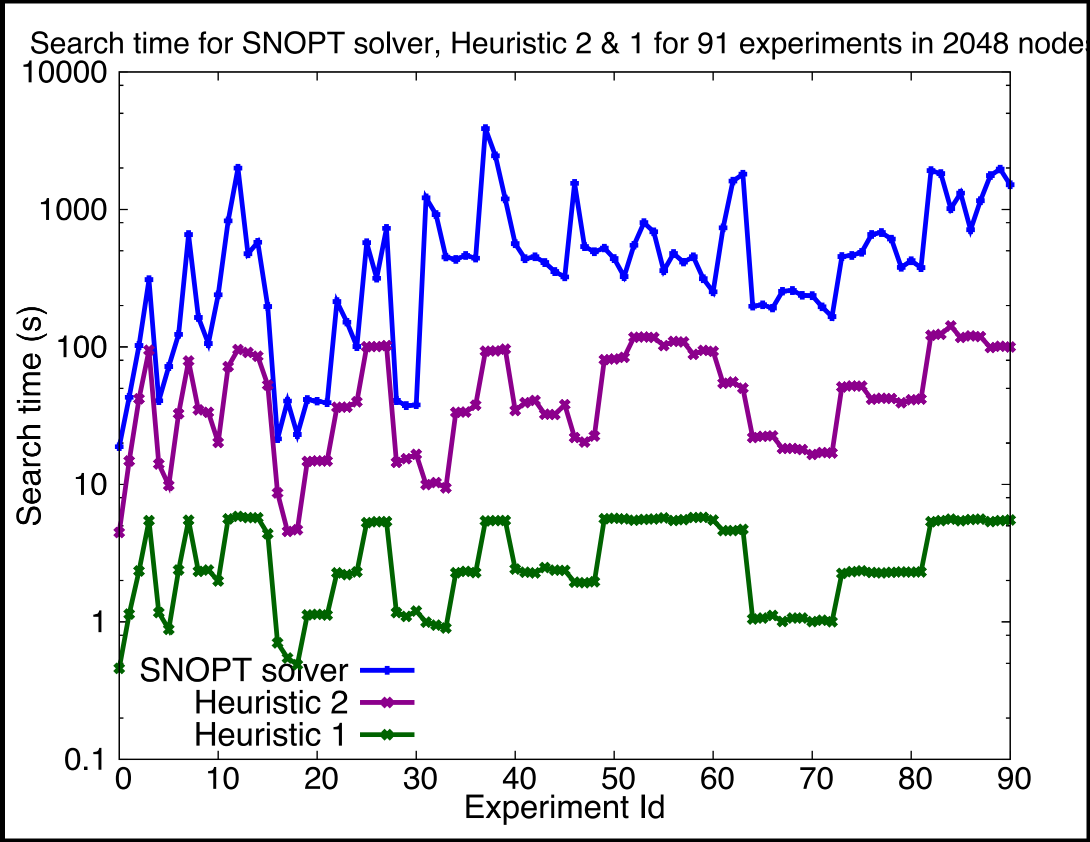
Pattern	Average			
	OPT	HEU 2	HEU 1	MPI
Disjoint	193	206	116	56
Overlap	208	210	124	58
Subset	207	207	124	55

Performance (GB/s) when random source-destination pairing.

OPTIQ still outperforms well with random pairing.

Paths searching time

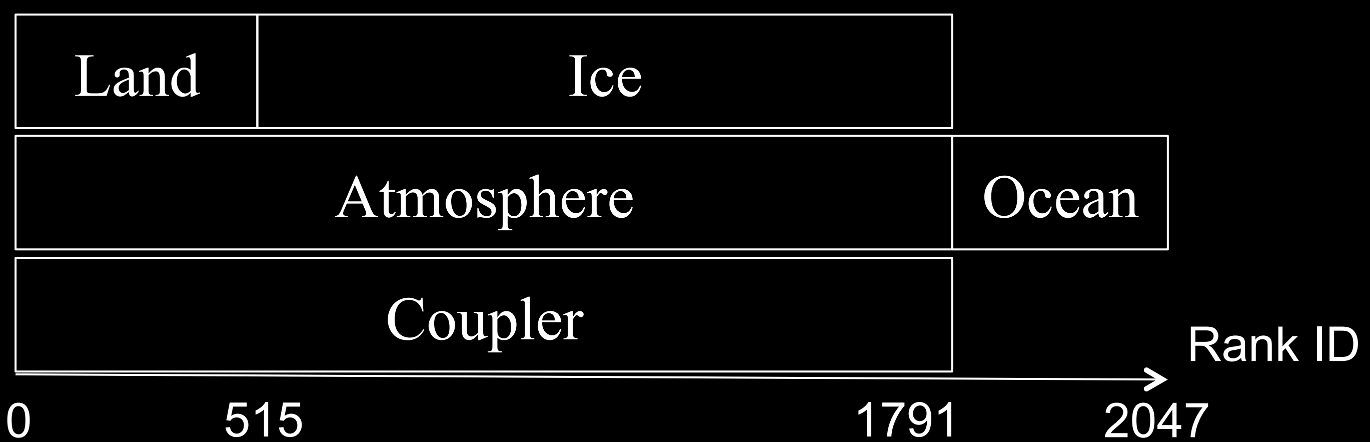
- Search times are significantly different (~10X)



Trade-off between search time and search quality.

Community Earth System Model (CESM)

- Simulate the earth climate. There are 4 models: Atmosphere, Ocean, Ice, Land.
- Coupling models: 4 models communicate via Coupler.
- 512 nodes, 4 ranks/node (total 2048 ranks).



Physical layout of models (by Rank ID)

CESM has 3 subset and 1 disjoint patterns.

Community Earth System Model (CESM) (cont.)

- Data movement throughput: 512 nodes, 32KB to ~2MB/node.

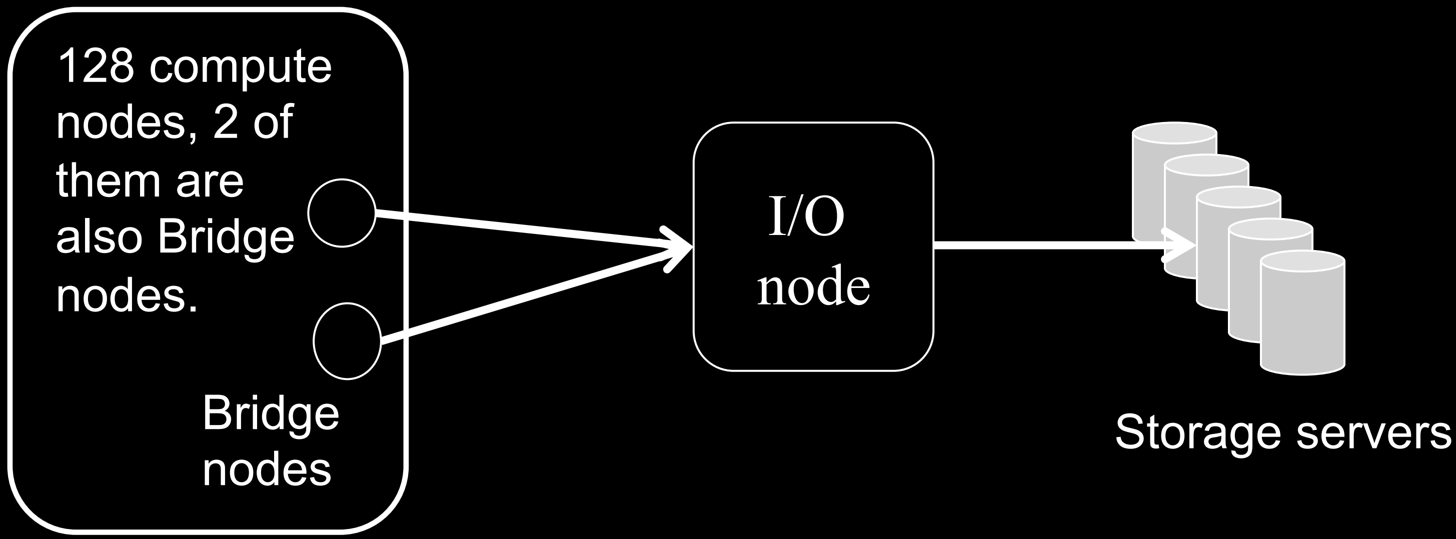
OPTIQ outperforms MPI (20%-46%) with different message sizes, random pairing.

Coupling	Type	Performance	Improv %
CPL-ATM	OPT	352	46%
	HEU 2	350	45%
	MPI	241	
CPL-LND	OPT	343	23%
	HUE 2	332	20%
	MPI	278	
CPL-OCN	OPT	135	30%
	HEU 2	136	30%
	MPI	104	

Performance (GB/s) of data movement between models.

Hardware/Hybrid Accelerated Cosmology Code (HACC)

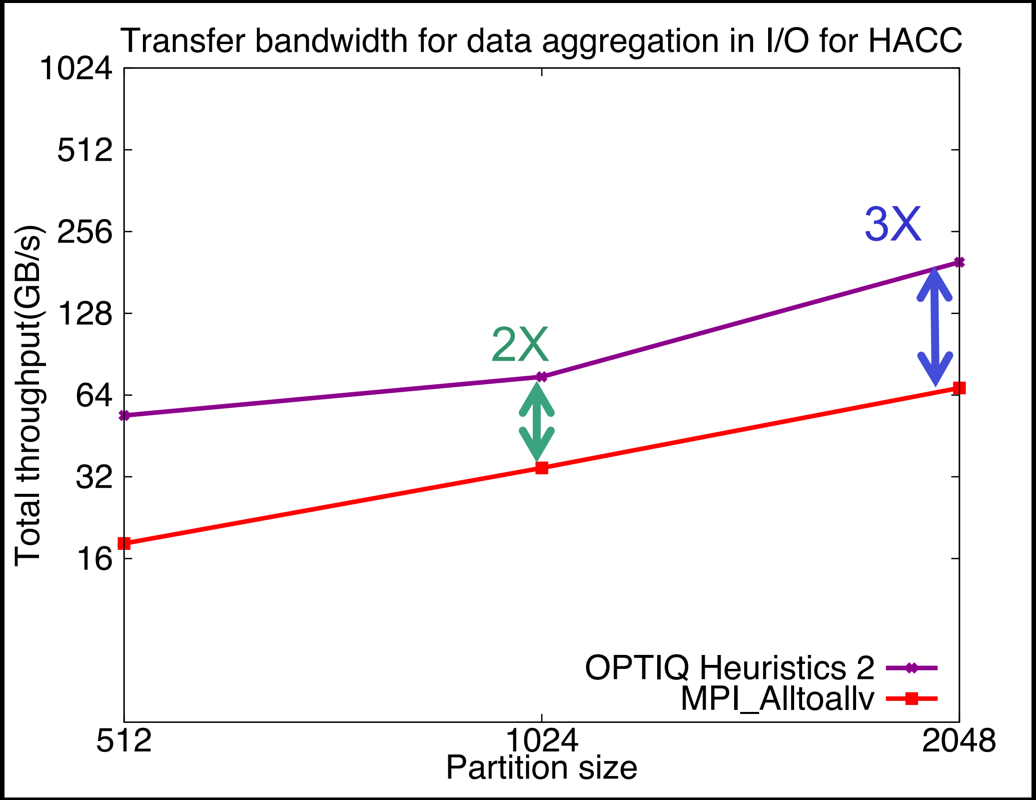
- Simulating universe from beginning ~15 billion years ago. Writing data to storage after each phase.



- In this experiment:
  - Aggregate data from compute nodes to bridge nodes.
  - Data size ~ 6MB/node.

Hardware/Hybrid Accelerated Cosmology Code (HACC) (cont.)

- Scaling number of compute nodes.



OPTIQ outperforms MPI at scale.

Conclusions

Dissertation Contributions

- Proposed a holistic approach to improve data movement for data-centric applications on supercomputers.
- Realized the approach in a framework OPTIQ with an easy to use API, requiring minimal changes to integrate into applications.
- Provided multi-path data movement with a number of algorithms.
- Implemented and demonstrated results on BGQ supercomputer with 2 applications, from ~2X-3X up to 5X improvement.

Publications/Submissions

- Submissions:
  - H. Bui, E. Jung, V. Vishwanath, A. Johnson, J. Leigh, M. E. Papka. Improving Sparse Data Movement Performance Using Multiple Paths on the Blue Gene/Q Supercomputer. International Journal of Parallel Computing (PARCO).
  - H. Bui, P. Malakar, V. Vishwanath, T. Muson, E. Jung, A. Johnson, M. E. Papka, J. Leigh. Improving Communication Throughput by Multipath Load Balancing on Blue Gene/Q. Supercomputer. High Performance Computing (HiPC).
  - H. Bui, R. Jacob, P. Malakar, V. Vishwanath, A. Johnson, M. E. Papka, J. Leigh. Multipath Load Balancing for M × N Communication Patterns on the Blue Gene/Q Supercomputer Interconnection Network (HiPINEB).
- Publications:
  - V. Vishwanath, H. Bui, M. Hereld, M. E. Papka. High Performance Parallel I/O (Chapter 18 (GLEAN)) Oct. 2014.
  - Huy Bui, Eun-Sung Jung, Venkatram Vishwanath, Jason Leigh, Michael E. Papka. Improving Data Movement Performance for Sparse Data Patterns on Blue Gene/Q Supercomputer. ICPPW 2014.
  - Huy Bui, Hal Finkel, Venkatram Vishwanath, Salman Habib, Katrin Heitmann, Jason Leigh, Michael E. Papka, Kevin Harms: Scalable Parallel I/O on a Blue Gene/Q Supercomputer Using Compression, Topology-Aware Data Aggregation, and Subfiling. PDP 2014.

Thank you!

Efficacy of *maxload* value

- Partition of 1024 nodes, 64 sources and 512 destinations, 1 rank/node, 8 MB/pair.
- Varying the maxload value: 1, 2, 4, 8, 16, 32.

Pattern	MPI	*Maxload* value					
		1	2	4	8	16	32
Disjoint	45	31	32	32	63	75	78
Overlap	42	66	66	66	125	112	89
Subset	74	69	70	69	114	110	96

Performance (GB/s) when increasing *maxload* value.

In general, OPTIQ performs best with *maxload* = 16.

Efficacy of Number of Shortest Paths Fed into Solvers

- Partition: 2048 nodes, 128 sources and 1024 destinations, 1 rank/node, 8 MB/pair.
- Number of shortest paths: 4, 16, 32, 50.

Pattern	MPI	Number of paths			
		4	16	32	50
Disjoint	61	29	84	104	197
Overlap	59	82	192	224	308
Subset	111	99	163	168	172

Performance (GB/s) when increasing number of paths fed into solvers.

OPTIQ outperforms better with more paths fed.

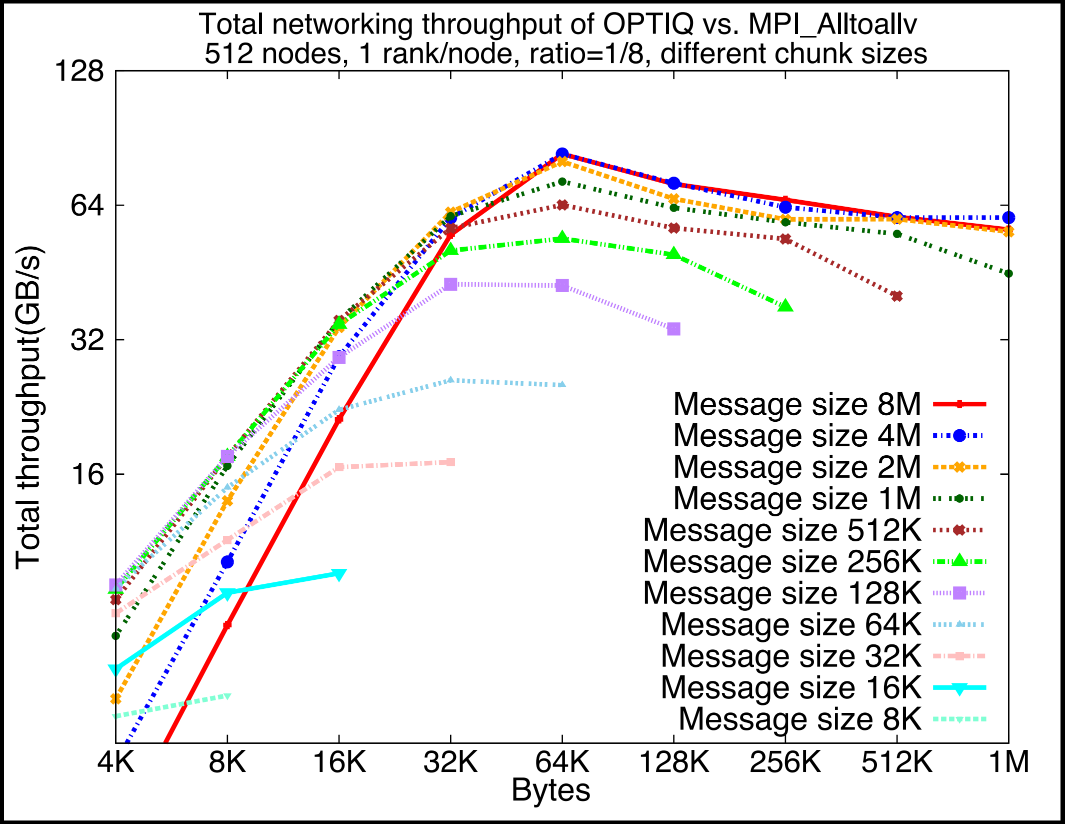
Future work

- Further improve performance by investigating multiple solutions produced by solvers.
- Extend the work to different supercomputers.
- Reduce the solving time of solvers by graph-partitioning approaches.
- Provide the QoS for the supercomputers.

Efficacy of Number of Ranks per Node

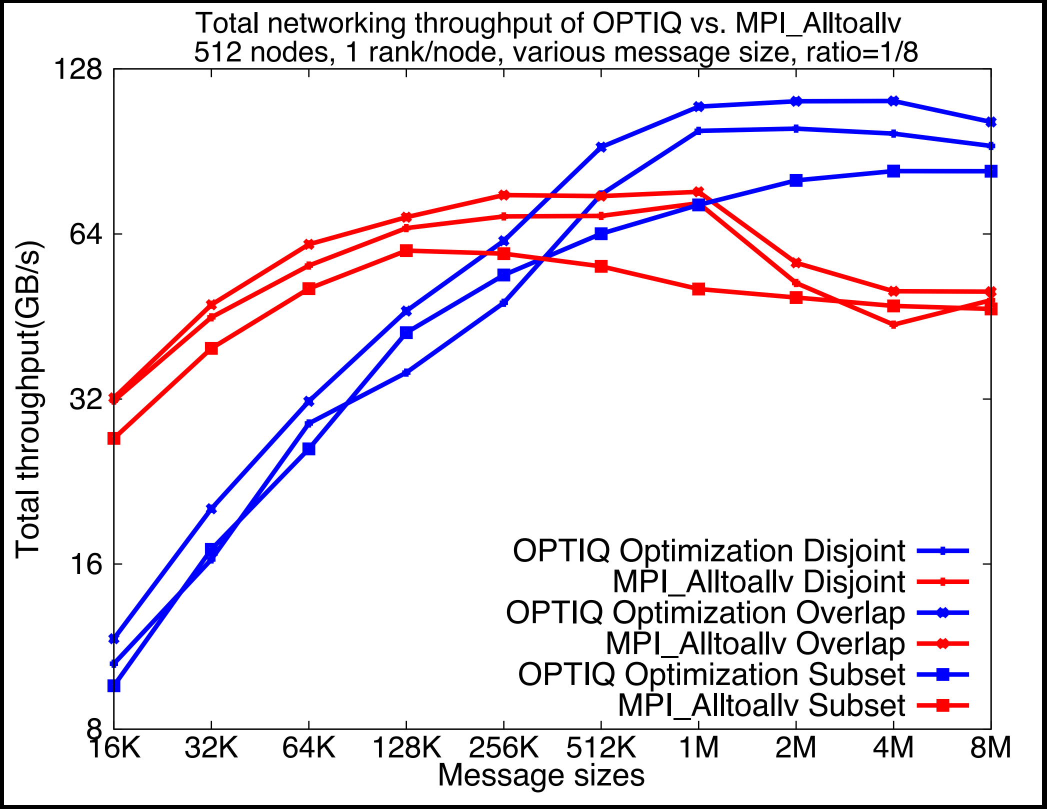
Efficacy of Chunk Size

- Data is split into chunk to send out.
- Chunk size can affect performance.



Efficacy of Message Size

- Due to overheads, OPTIQ shows better performance for messages larger than 512KB.



Efficacy of Solvers

- SNOP vs. CPLEX at 2K, 91 experiments.
- Measure AMPL time, solving time and throughput.

Quality of Service (QoS) via Multiple Routing Classes

- The systems treat all packets in the same ways i.e. best-effort routing/first come first serve. Thus, different data flows have the same priority/no priority at all. But:
  - Some flows are more critical in time/bandwidth.
  - Application developers/scientists can further optimize their applications given their understanding of communication patterns.

⇒Provide capability of giving priorities for different data flows.

Quality of Service (QoS) via Multiple Routing Classes

- Still best-effort data movement.
- For each data pattern we create a routing class.
- Each routing class has its own routing priority.
- Users can assign priority for each class (class-based) or each flow (flow-based).
- Reserve resources for classes based on priority.
- Dynamically control scheduling priorities.

Quality of Service (QoS) via Multiple Routing Classes

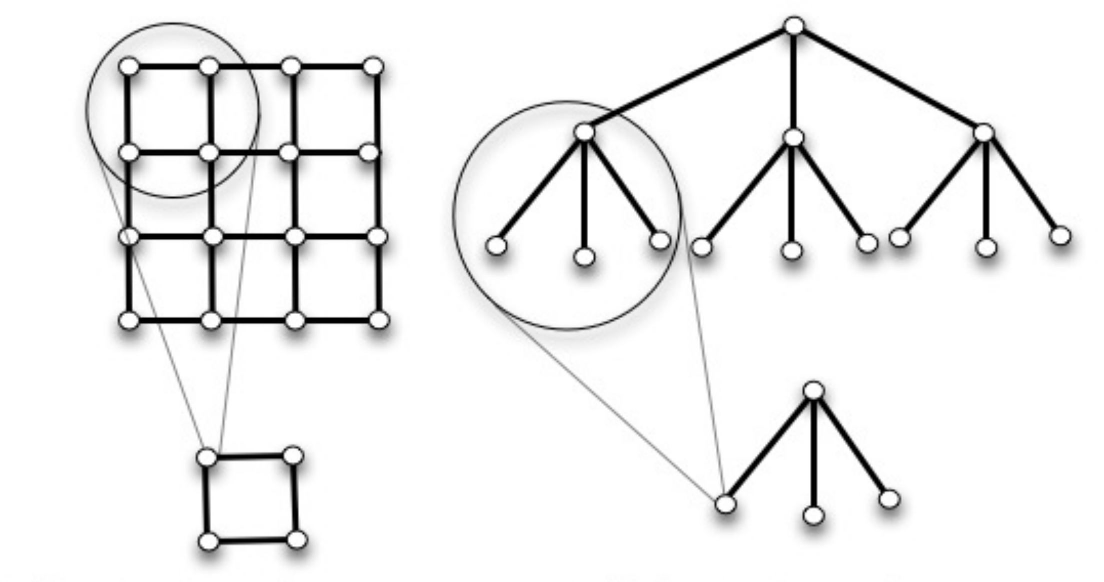
- *l* routing classes index *i* = 1 to *l* and each requires having *αi* portion of the total achievable bandwidth. We add 2 constraints for
  - Total throughput is 1: $\sum_{i=1}^{l} \alpha_i = 1$
  - Total throughput of class *i* with flows *j*:

$$\sum_{j=1}^{h} f_j(s_j, w) = \alpha_i * T$$

  with *j* is index of flows in routing class *i* and *T* being the total throughput that we need to maximize.
  - α_i can be assigned by users or dynamically adapt by the framework.

Optimizing Data Movement at Scale

- Proposed directions:
  - Symmetry of interconnect
  - Multiple level graph.
  - Giving options (offline, online, combined) depends on each problem.



(a) Graph reduction for a simple 2D mesh (b) Graph reduction for a simple tree

Model-based Data Movement Optimization

- Problem modeling:
  - Given a set of Jobs and set of Paths for each job in Jobs. Each jobs has Demand[job], flow on path p flow[job, p].
  - Each edge (i,j) has capacity c(i, j).
  - Objective function:
    - Minimize the transfer time t.
  - Capacity constraint:

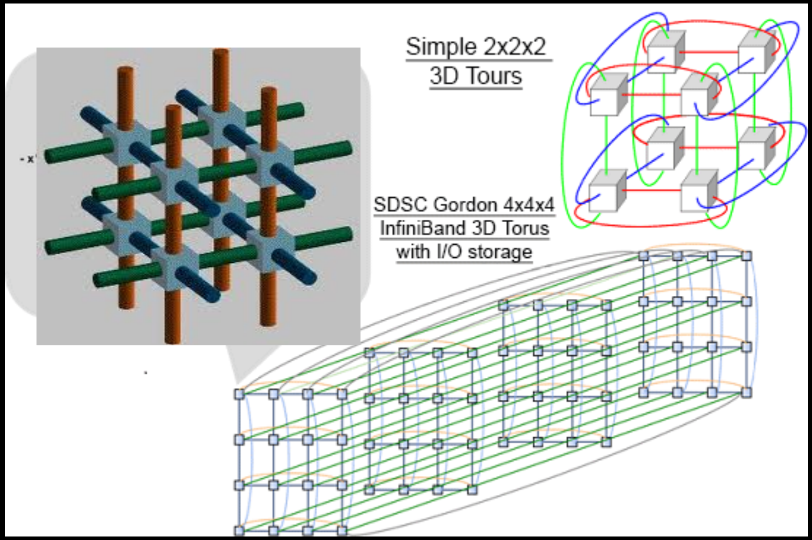
$$\sum_{\forall job \in Jobs} \sum_{\forall p \in kpaths_{job}} flow[job,p]_{ij} \leq c(i,j)$$

  - Throughput constraint:

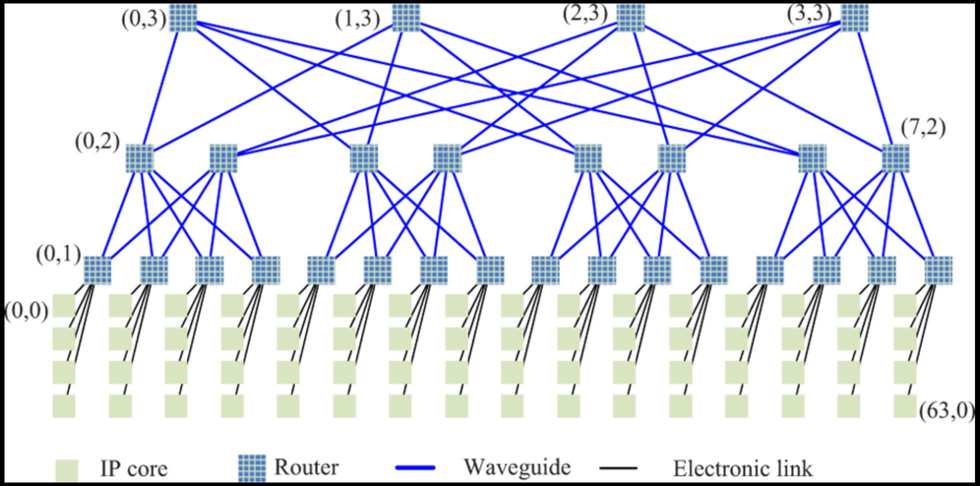
$$\sum_{\forall p \in kpaths_{job}} flow[job,p] = \frac{Demand[job]}{t}$$

Supercomputer's Interconnection Network

- A supercomputer includes ten thousands compute nodes and high throughput and low latency interconnect network.



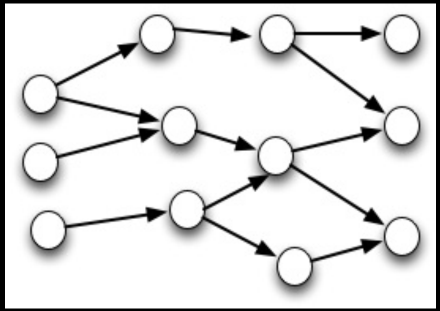
3D Torus

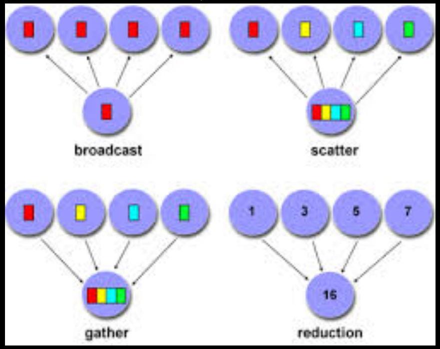


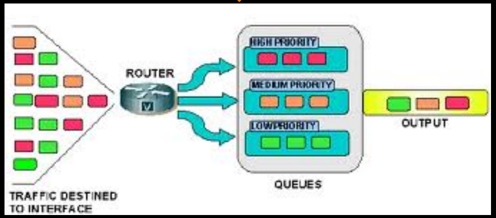
Butterfly Fat tree

Data movement in supercomputers

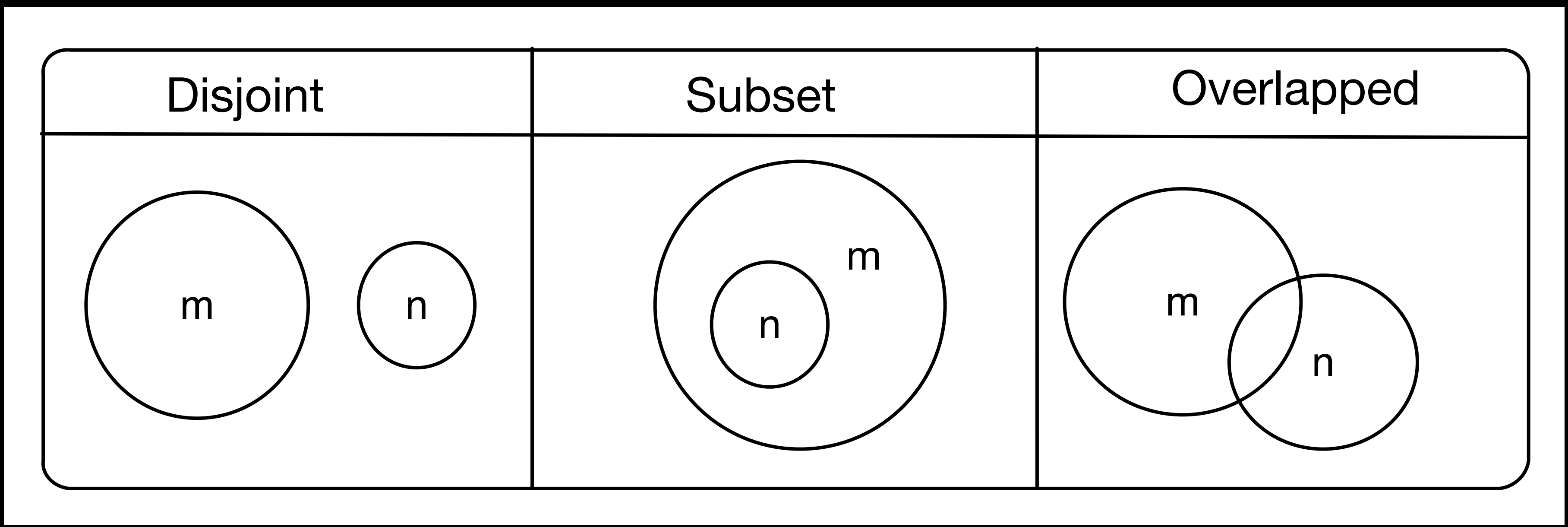
Layers	Data movements
Applications	Data flows from sources to destinations
Middleware (MPI, PGAS...)	Communication routines such as MPI_Send, MPI_Recv, MPI_Broadcast, MPI_Gather ….
Systems	Packets/messages routing





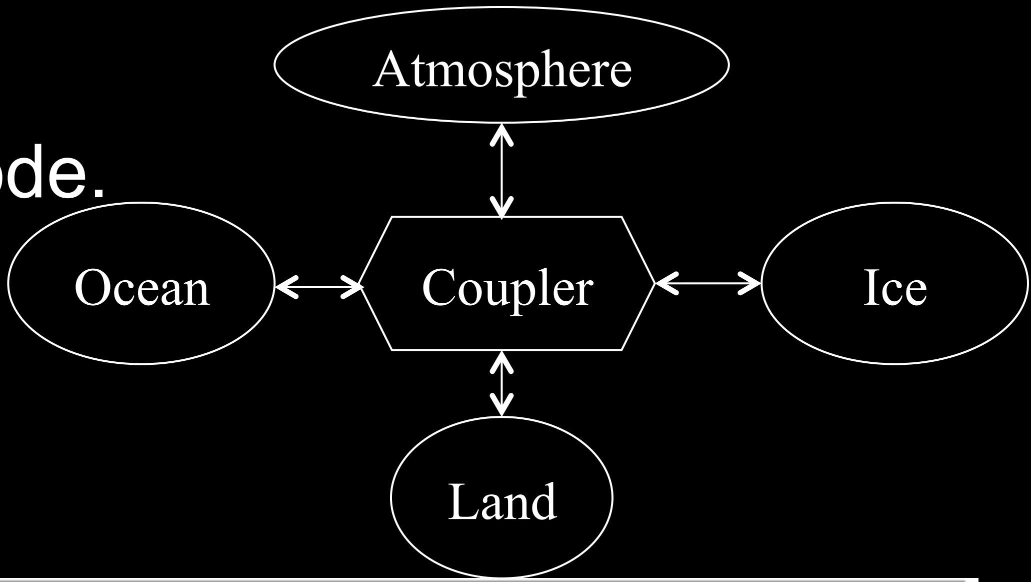


Communication Patterns



Community Earth System Model

- Coupling models
- 512 nodes, 4 ranks/node.



Model	Location	Num. of pairs of communication with Coupler	
		Between Ranks	Between Nodes
ATM	0 - 1791	5227	2233
LND	0 - 515	10390	2911
ICE	516 - 1791	3018	765
OCN	1792 - 2047	2001	502
CPL	0 - 1791		