



Large-Scale Graph Traversal Algorithms on Multi-core Clusters

Speaker: Huiwei Lu

Advisor: Prof. Ninghui Sun



Argonne National Laboratory, Feb. 2013

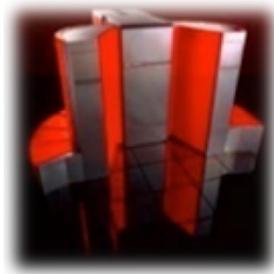
lvhuiwei@gmail.com



HPC is Driven by Applications

Gordon Bell Prize Winners

- 1 GFlop/s; 1988
 - Static finite element analysis
- 1 TFlop/s; 1998
 - Modeling of metallic magnet atoms
- 1 PFlop/s; 2008
 - Superconductive materials



Cray Y-MP: 8 Processors



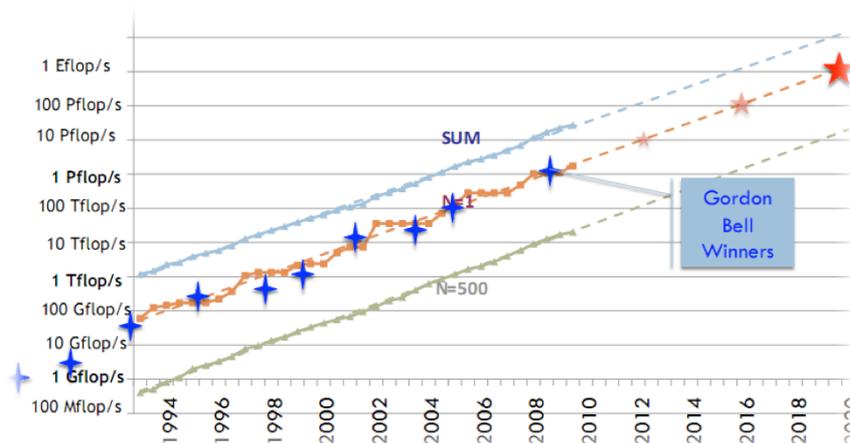
Cray T3E: 1024 Processors



Cray XT5: 1.5×10^5 Processors

- **1 EFlop/s; 2018**

- 1×10^7 Processors (10^9 threads)
- Machine? Application?



Source: Dongarra SC'09

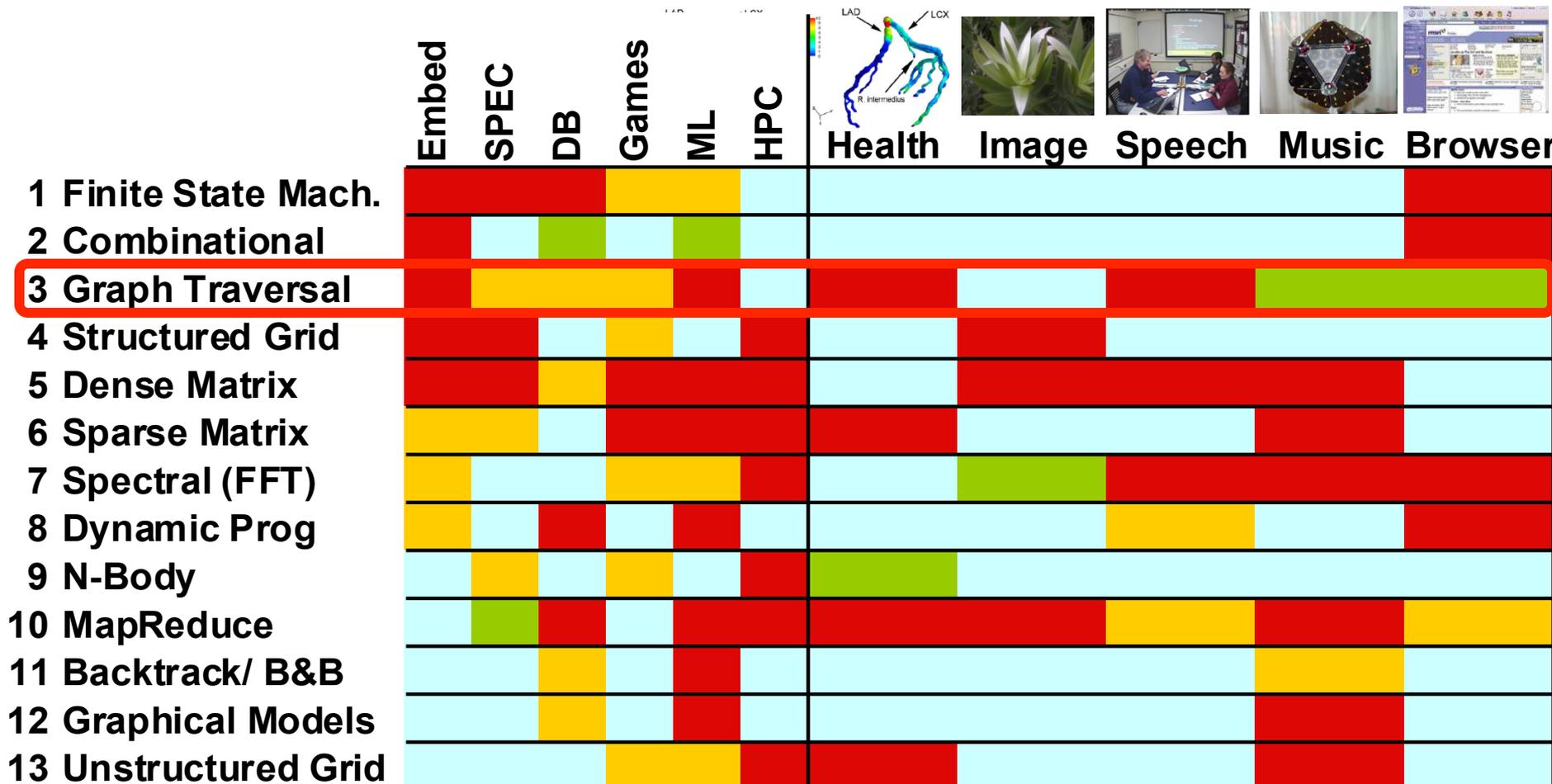


中科院计算所
INSTITUTE OF COMPUTING
TECHNOLOGY

Application Driven Research

Motif/Dwarf: Common Computational Methods

(Red Hot → Blue Cool)



(Source: Demmel 2009)



中科院计算所
INSTITUTE OF COMPUTING
TECHNOLOGY

Application Background of Graph Algorithms

- Applications

- Genome assembly, social network, web search, shortest path
- Social network: community identification, targeted advertising, information spreading
- Biology computation: predicting new interactions, functional annotation of novel proteins, identifying metabolic pathways, identifying new protein complexes

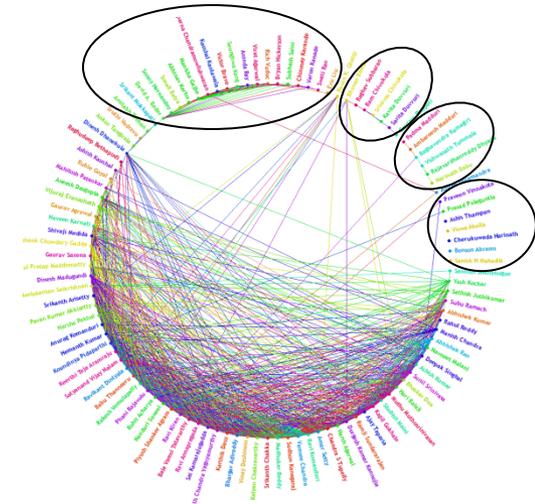


Image Source: Nexus (Facebook application)
(Slides from Kamesh Madduri, CS267/EngC233 Spring 2010)

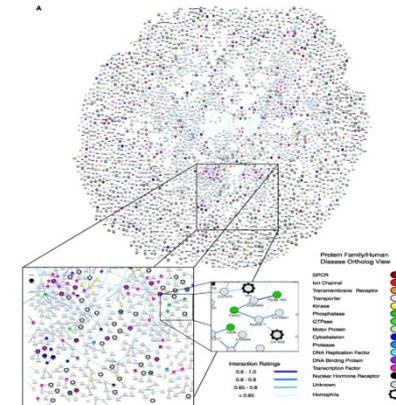
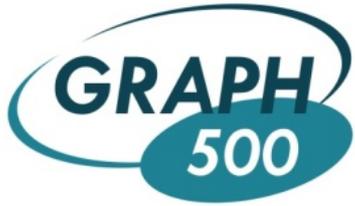


Image Source: Giot et al., "A Protein Interaction Map of *Drosophila melanogaster*",
Science 302, 1722-1736, 2003.

(Slides from Kamesh Madduri, CS267/EngC233 Spring 2010)



- Breadth-first search of a scale-free undirected graph
- Data-intensive, augment to Top 500
 - evaluate memory access (TEPS)
- Announced at ISC'10, first list at SC'10

	GTEPS	Site	Machine	Cores	Nodes
Nov 2010	6.6	Argonne National Laboratory		32000	8192
Jun 2011	40.4	Moscow State University	Lomonosov	8192	4096
Nov 2011	236	NNSA and IBM Research, T.J. Watson	BlueGene/Q	65536	4096
Jun 2012	3541	Argonne National Laboratory & LLNL	BlueGene/Q	524288	32768
Nov 2012	15363	DOE/NNSA/LLNL	BlueGene/Q	1048576	15363

Jun 2011	1.3	Rank 15 of 29, Institute of Computing Technology	SuperDragon-1	384	32
Jun 2012	12.1	Rank 20 of 80, National Supercomputing Centre in Shenzhen	Nebulae	6144	512

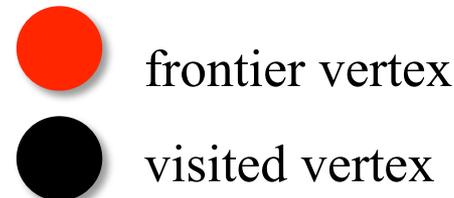
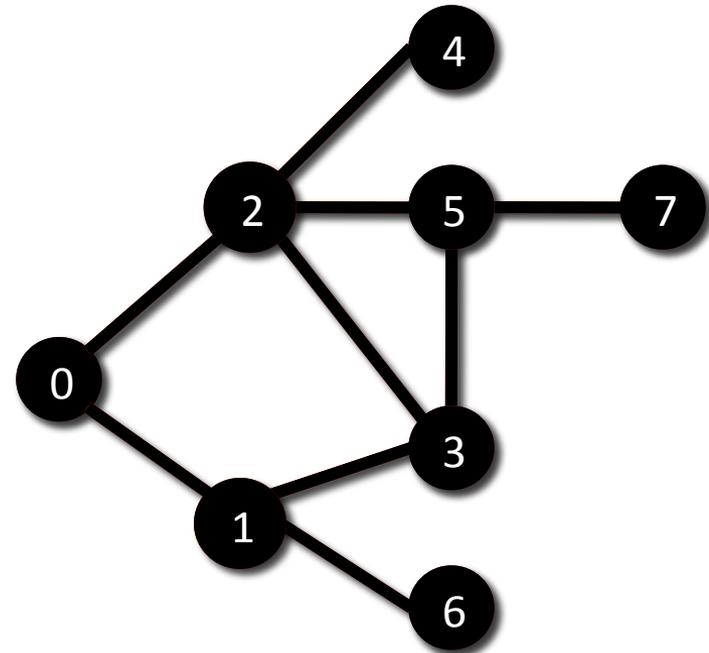


Breadth-First Search as a Building Block



- BFS is a subroutine for many algorithms
 - Betweenness centrality
 - Maximum flows
 - Connected components
 - Spanning forests
- Characteristics of BFS
 - irregular
 - low-arithmetic
 - abundant parallelism

BFS: explores the edges of G to "discover" every vertex that is reachable from s .



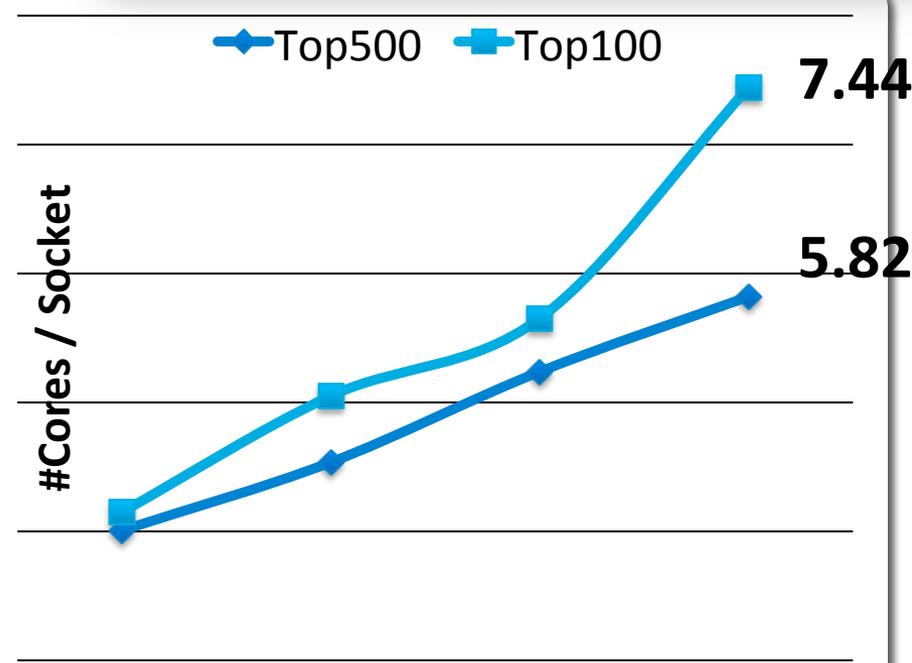
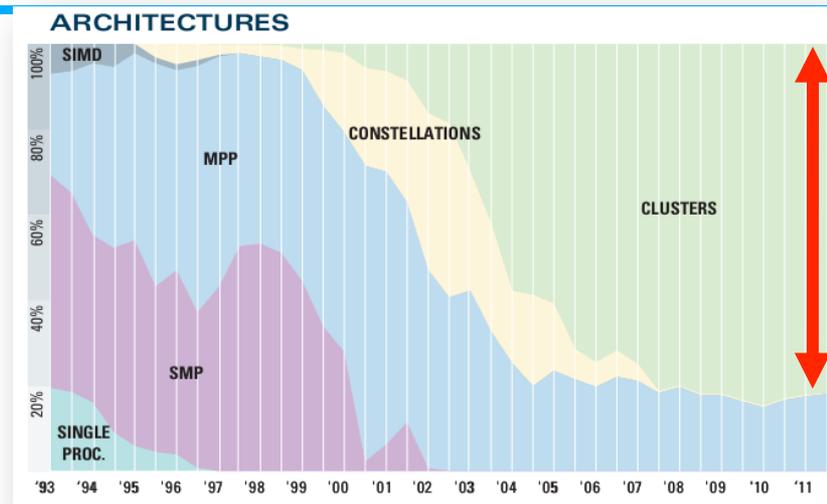


Outline

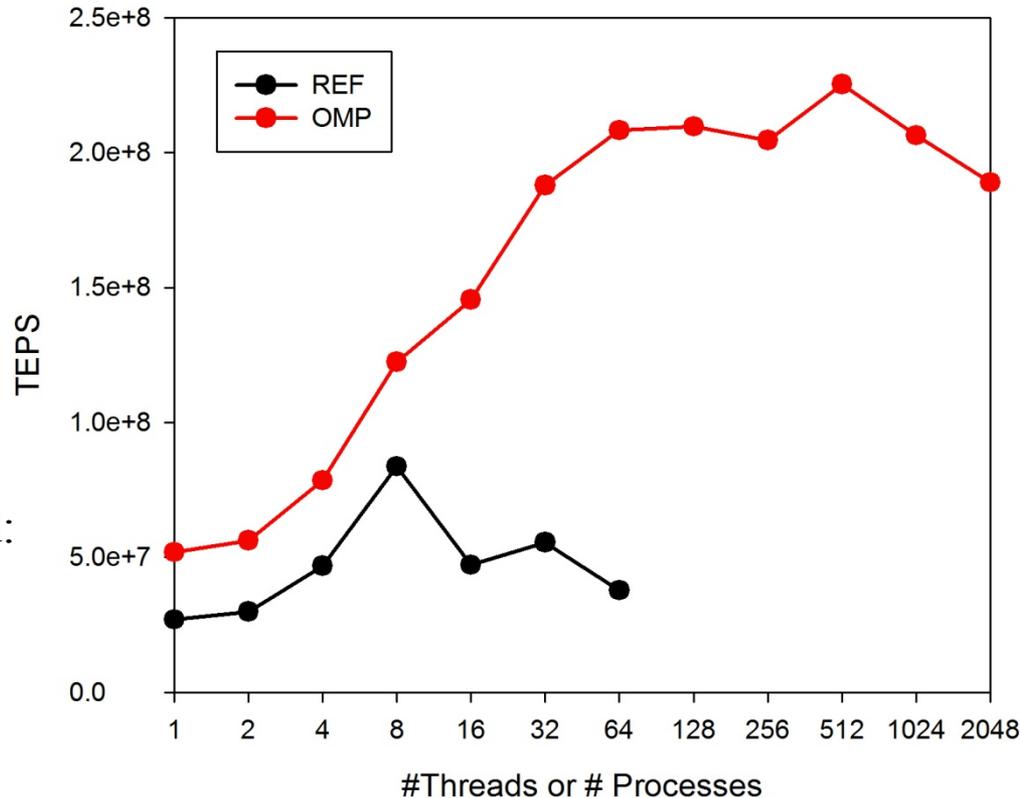
- Background
- **Algorithm 1: Hybrid MPI/Pthreads Parallel BFS**
- Algorithm 2: Reducing Communication in Distributed BFS
- Conclusion & Future Works

Motivation

- Cluster still dominates Top500 and Graph500
- **#Cores/Socket** increases
 - 7.44 (Top100, Nov11)
 - 11.6 (Graph500, Nov11, 28 machines)
- **Memory Parallelism** in State-of-art Processors [Agarwal SC'10]
 - increase #memory reads by 8x using software pipeline
- A great opportunity for **Hybrid Programming**
 - MPI for inter-node; OpenMP or Pthreads for intra-node



Motivation (cont.)



multi-threaded

MPI

Algorithms Tested:
Graph500 MPI
impl. & OpenMP
impl. on a single
node with two
Westmere CPUs.

- For Breadth-First Search on a single-node, multi-threaded version is more effective than MPI version

Compressed Sparse Row Format

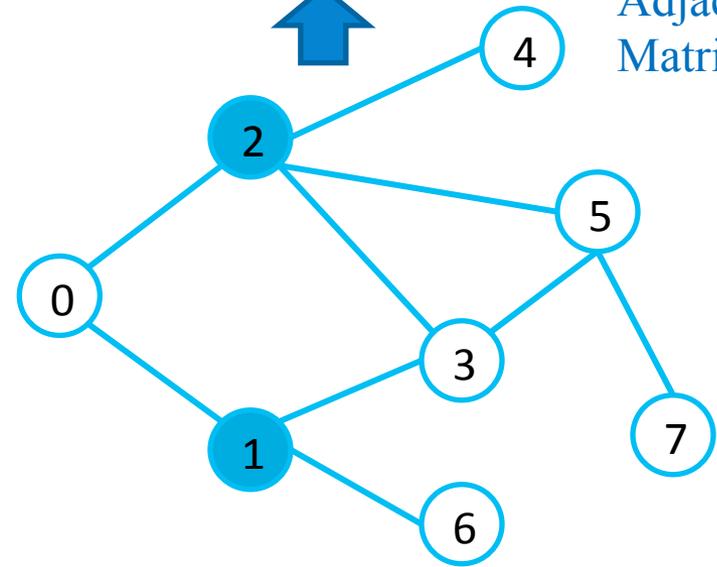
row	0	2	5	9	12	13	16	17	18
column	1	2	0	3	6	0	3	4	5 ...

Compressed
Sparse Row



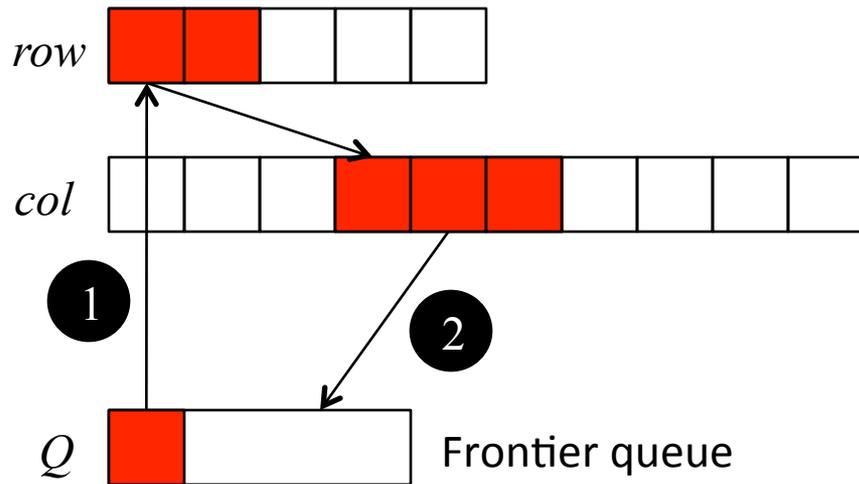
	1	1					
1			1				1
1			1	1	1		
	1	1			1		
		1					
		1	1				1
	1						
					1		

Adjacent
Matrix



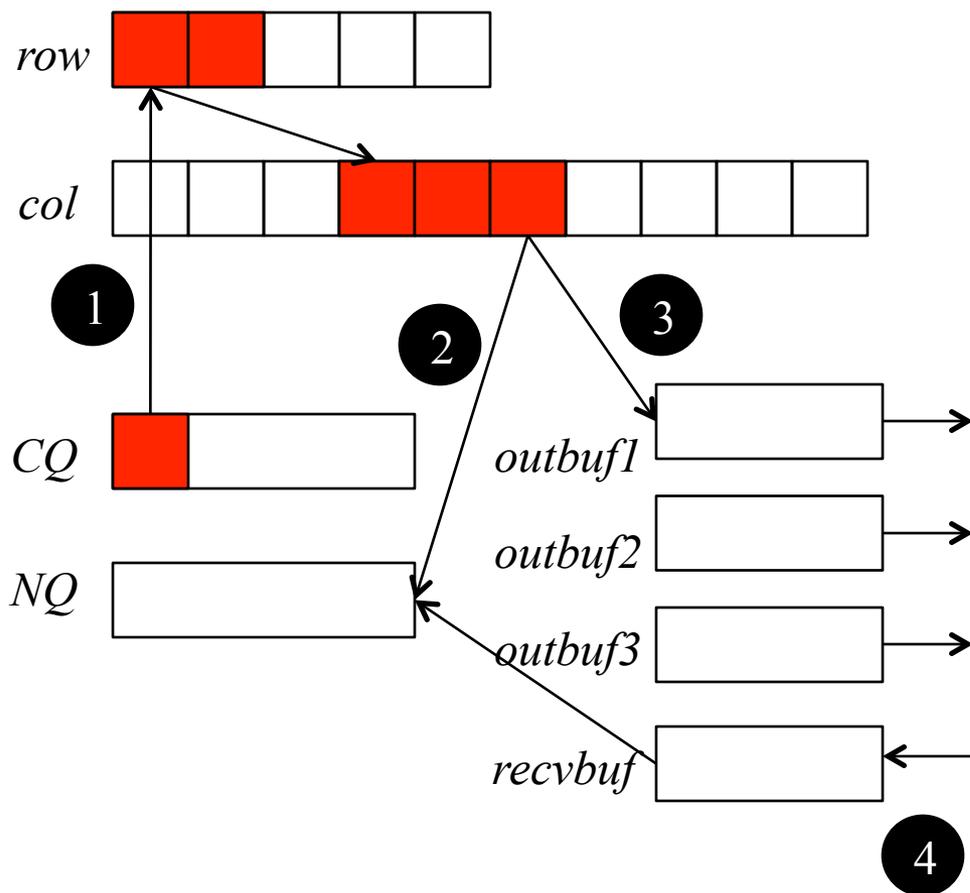
- A compressed form of adjacent matrix; store non-zero edges only
- Row: index, start & end position
- Column: vertex number
- E.g. v_0 's neighbor locates in `column[0, 2)`

Sequential BFS on CSR format



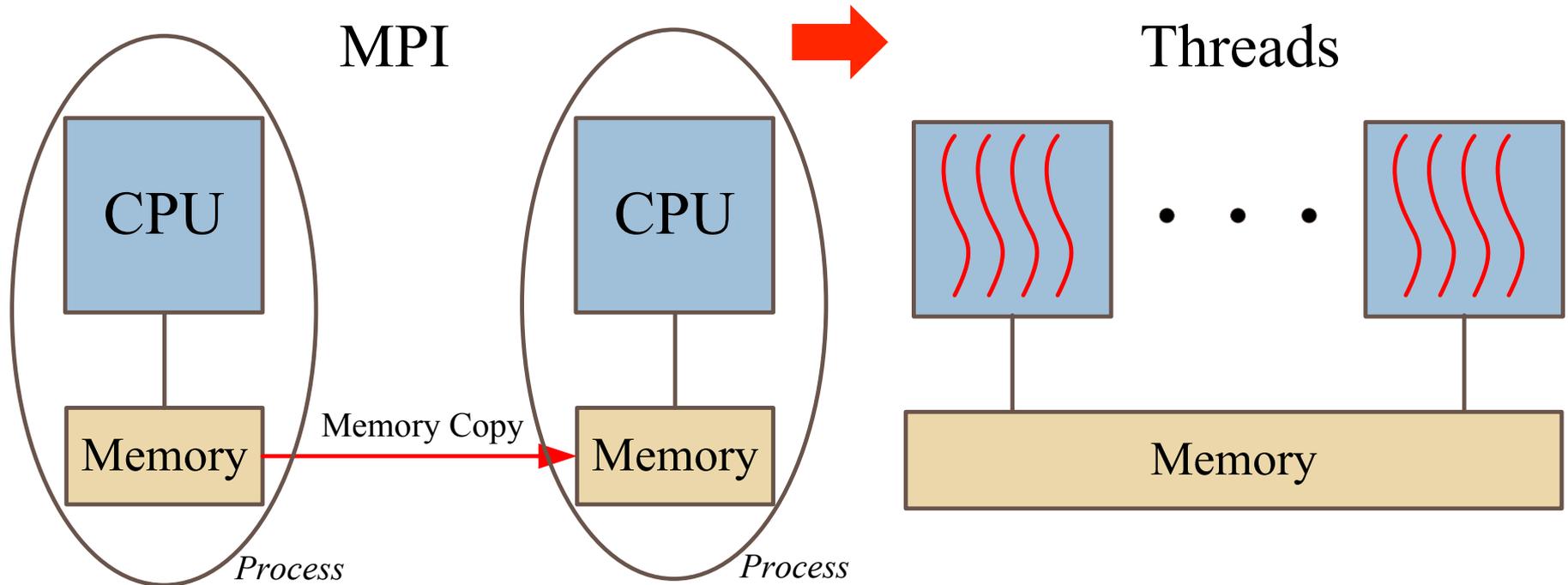
- 1. for all v in Q , find its neighbors N
- 2. insert N into Q
- loop until Q is empty

MPI-only BFS



- 1. for all v in CQ , find its neighbors N
- for each u in N
 - 2. insert to NQ
 - 3. insert to $outbuf$
- 4. send $outbuf$, check $recvbuf$
- exchange CQ with NQ , loop until CQ is empty

MPI v.s. Threads on a share memory node



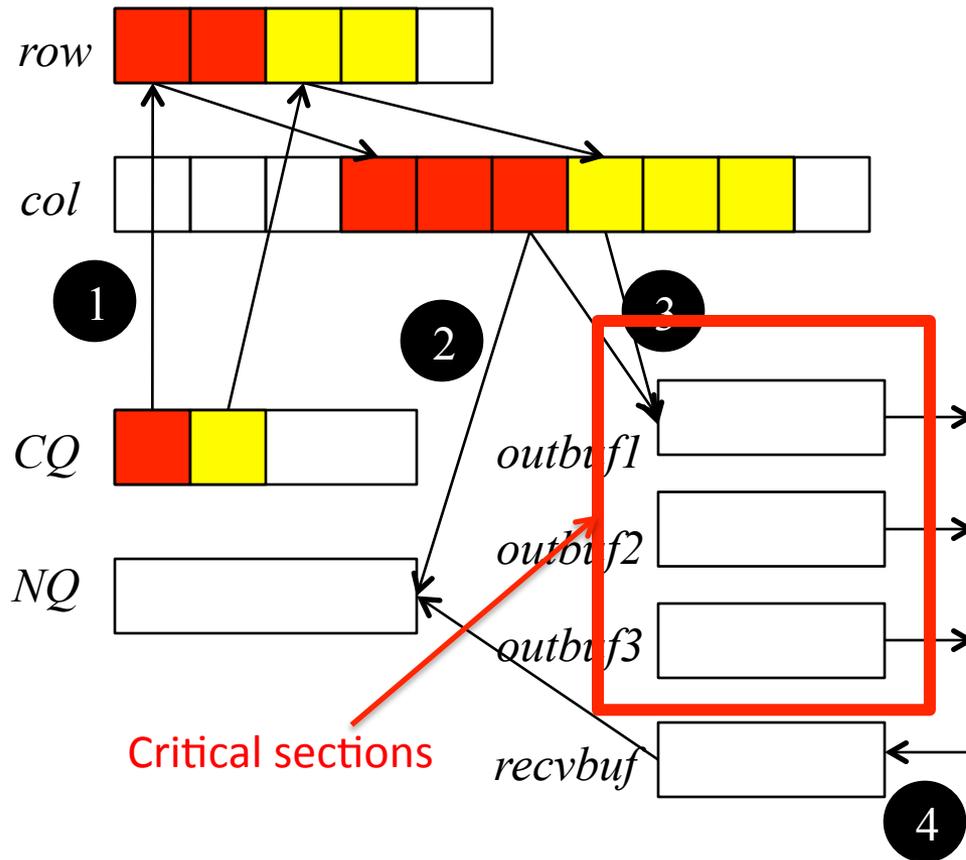
- Threads

- from process to more light-weighted threads → **increased bandwidth** (memory-level parallelism)
- **remove extra memory copy** in MPI

Hybrid MPI/Pthreads BFS

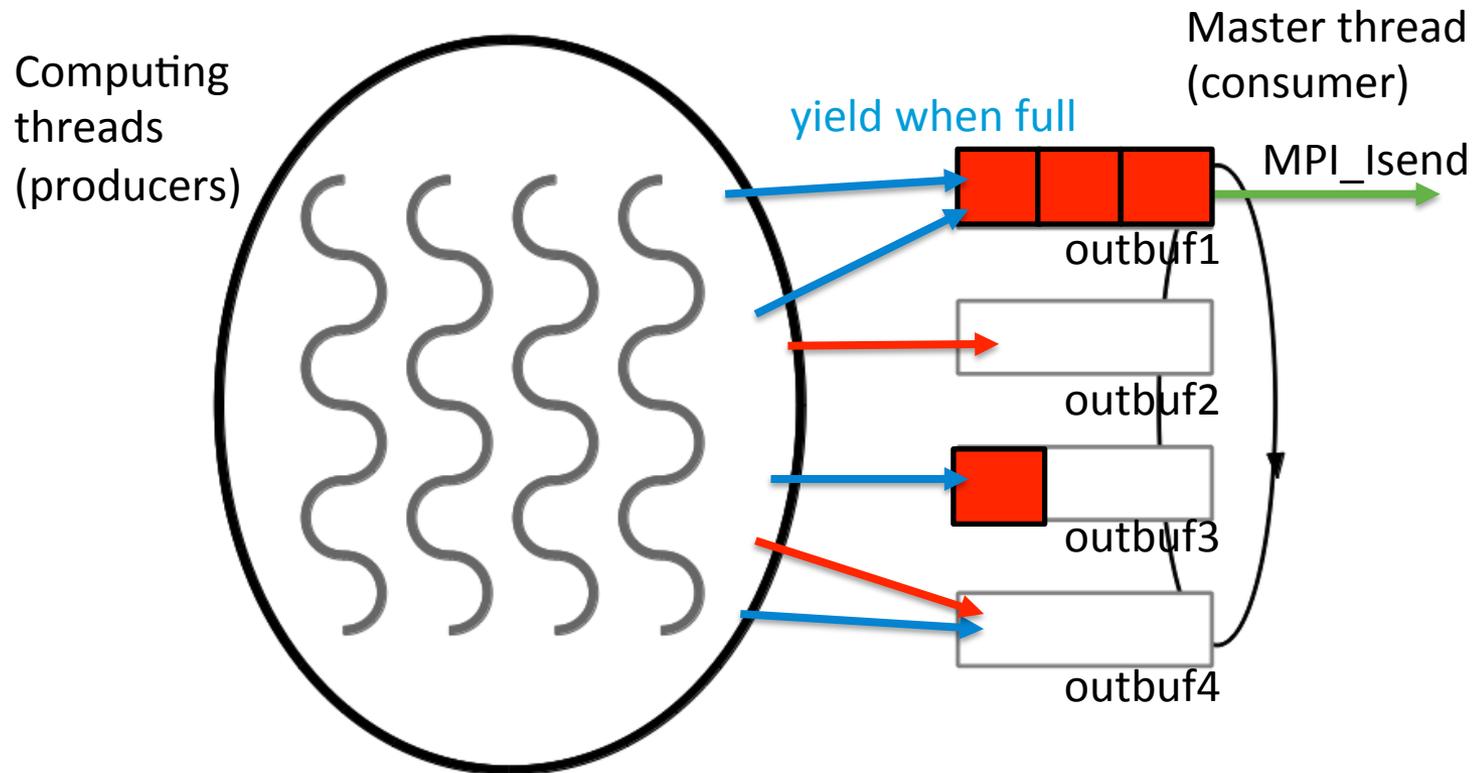


In parallel



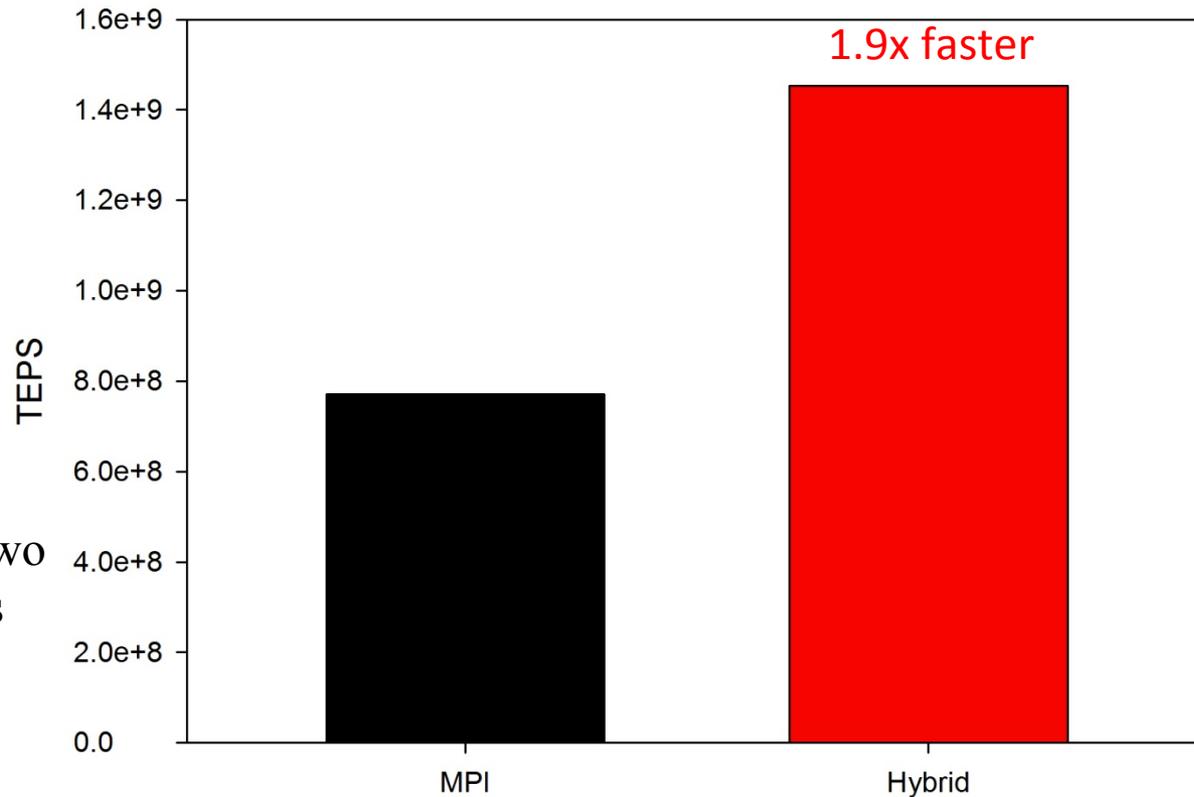
- 1. for all v in CQ , find its neighbors N
- for each u in N
 - 2. insert to NQ
 - 3. insert to $outbuf$
- 4. send $outbuf$, check $recvbuf$
- exchange CQ with NQ , loop until CQ is empty

Lock-free Multi-Producer Single-Consumer Queue



- Lock-free queue
 - Producers: use *fetch_add* to get index, then insert data
 - Consumer: round-robin check all the buffers, communicate with other processes

Hybrid v.s. MPI-only, 32 nodes



32-node cluster, for each node there are two Intel Westmere CPUs connected using QPI. Nodes are connected using InfiniBand

MPI-only V.S. Hybrid

- Hybrid is **1.9x** faster than MPI-only at 32 nodes

Hybrid v.s. CombBLAS, 512 nodes

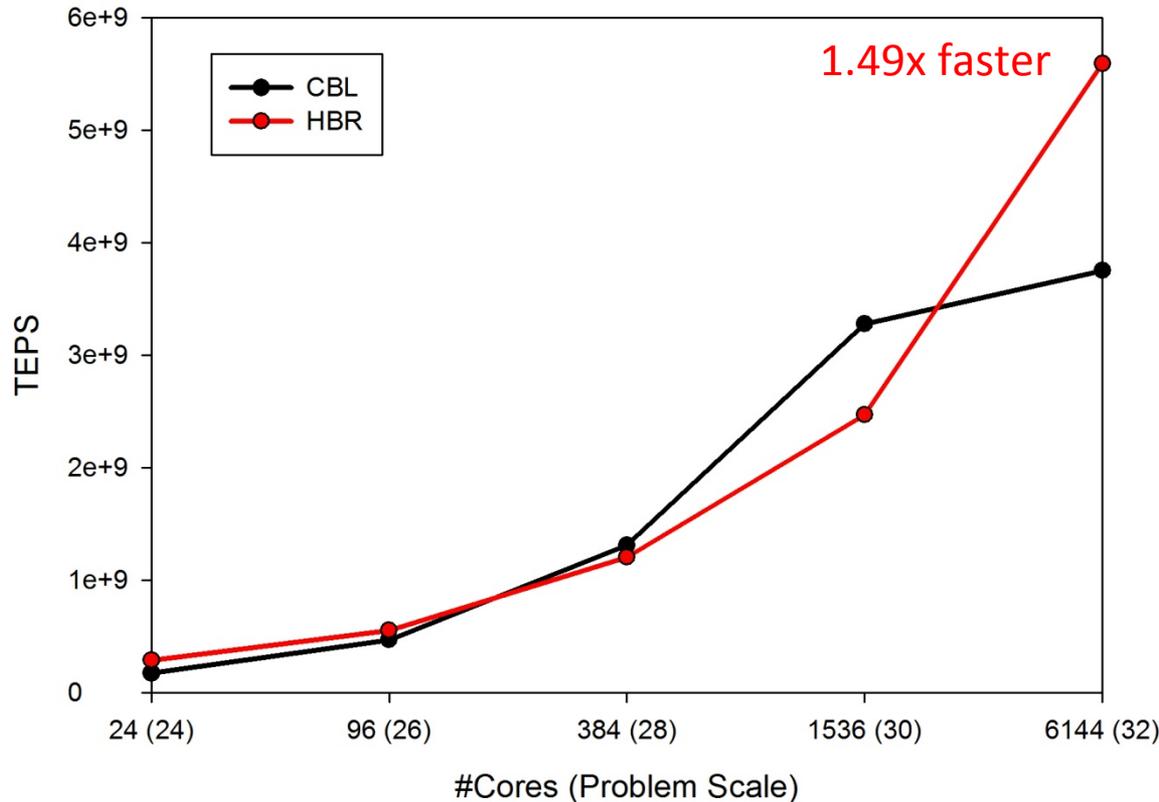
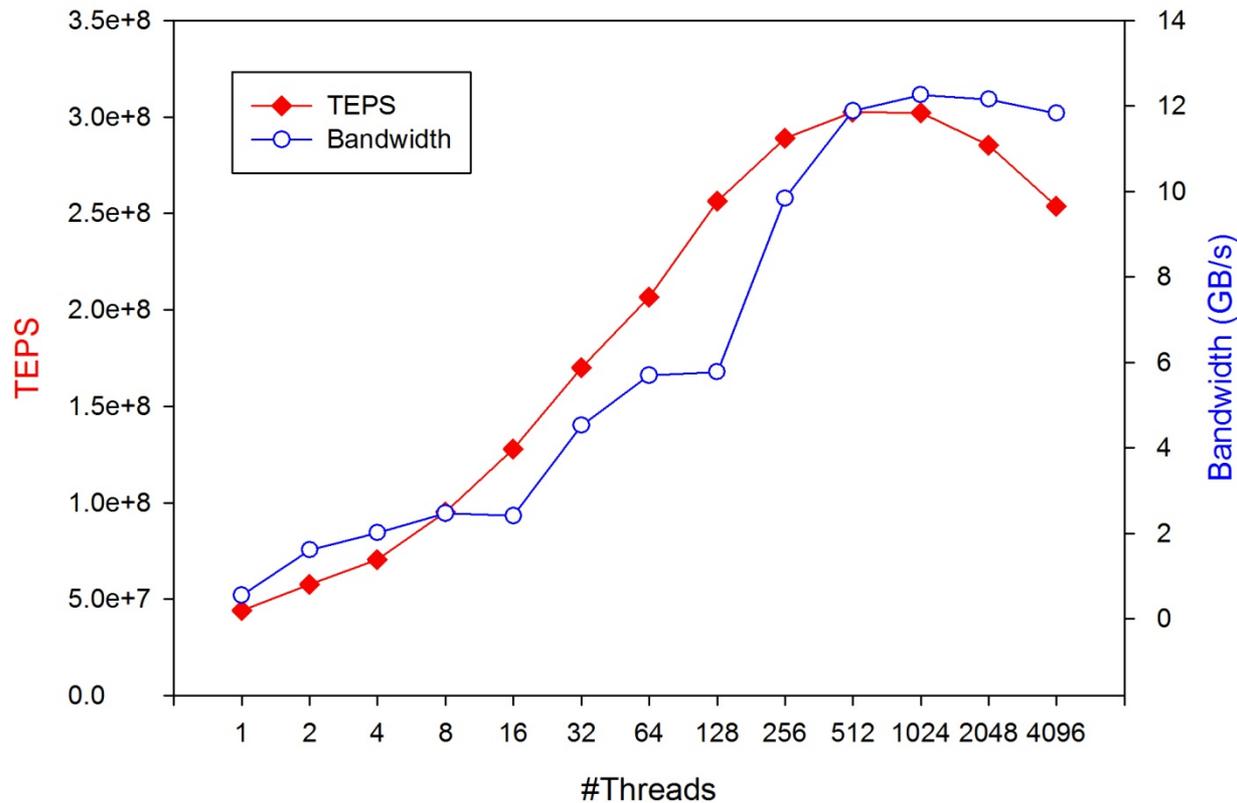


Fig. Weak scaling performance comparison of CombBLAS MPI-only and our hybrid BFS algorithm. Fixed problem size per node is used.

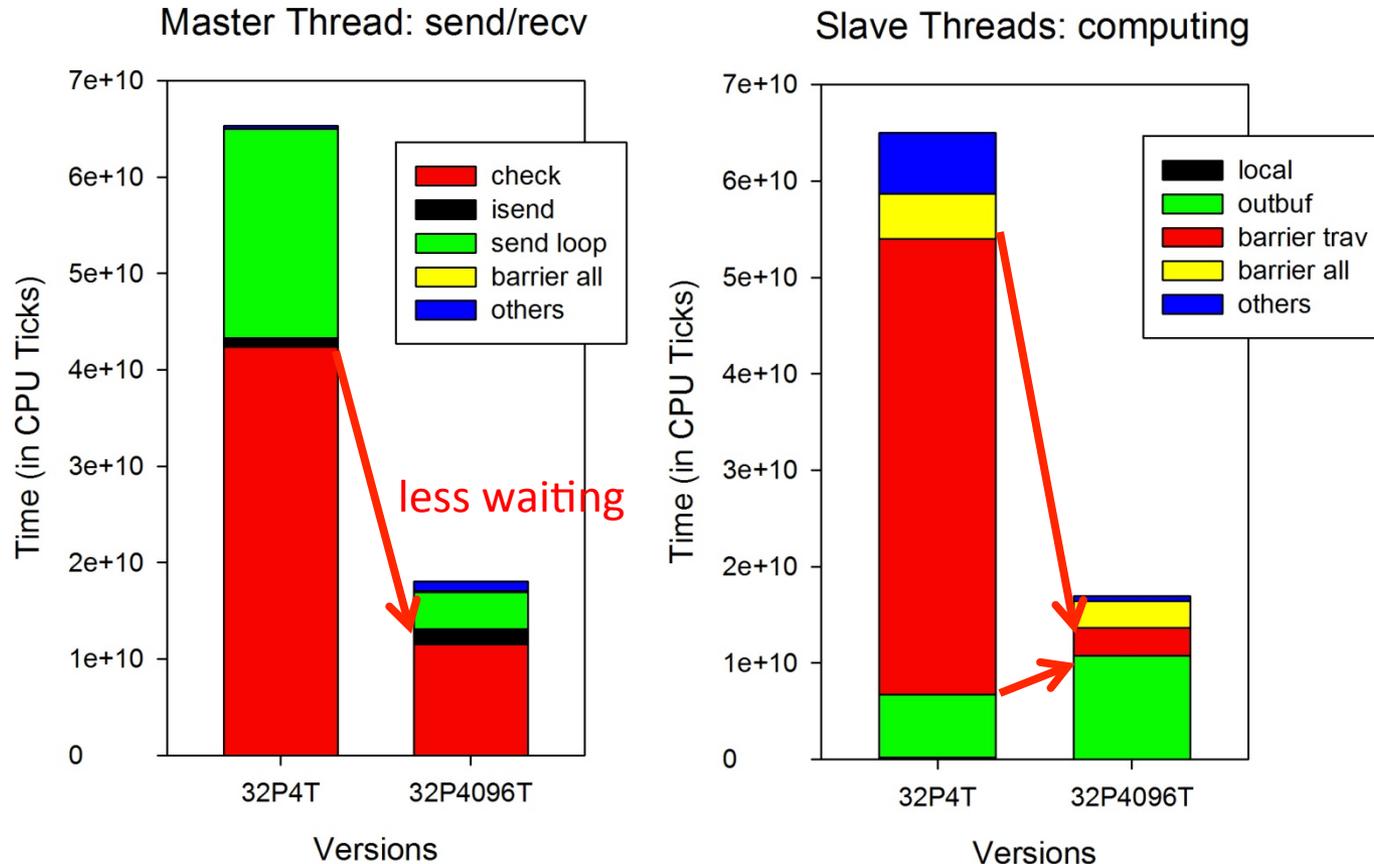
- Hybrid is **1.49x** faster than MPI-only CombBLAS BFS Algorithm at 6,144 cores. (Graph 500 MPI-only version does not finish because out of memory)

Benefit 1: Better Bandwidth



- Increased bandwidth improves BFS performance
- Benefits from multi-threading

Benefit 2: Better load balancing



- Increased #threads improve load balancing
- Aggregated waiting time of master thread reduced by 73%
- Aggregated waiting time of slave thread reduced by 94%

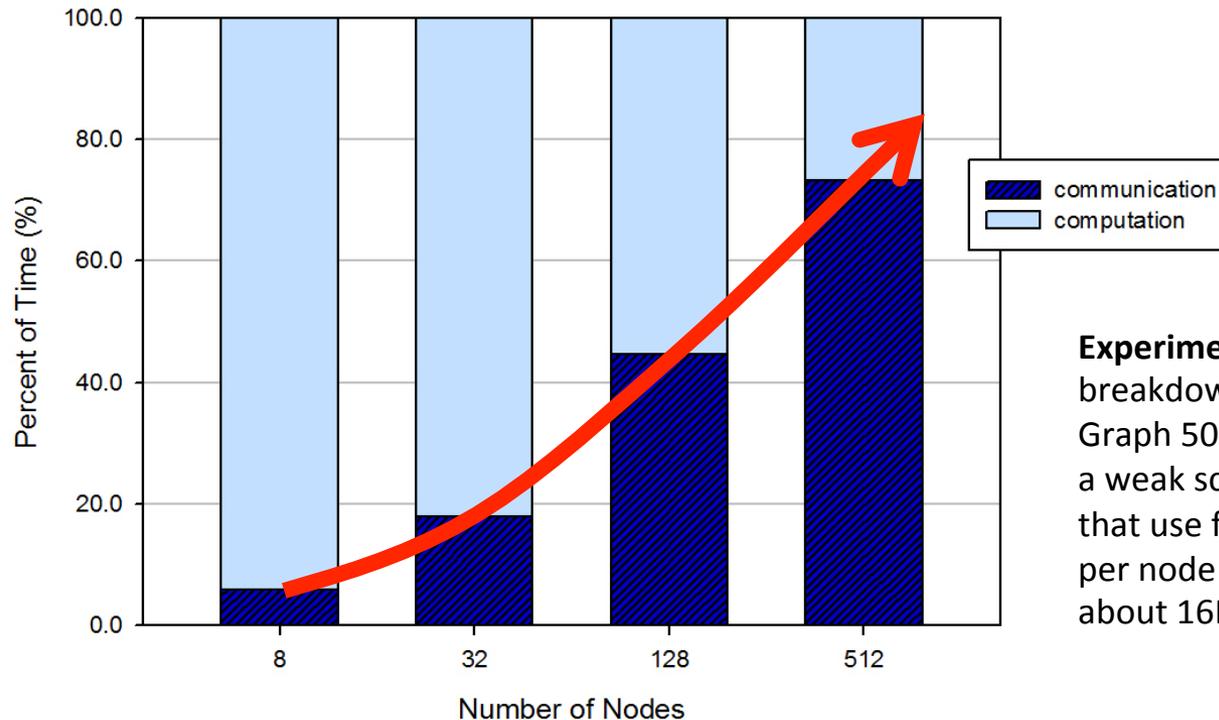


Outline

- Background
- Algorithm 1: Hybrid MPI/Pthreads Parallel BFS
- **Algorithm 2: Reducing Communication in Parallel BFS**
- Conclusion



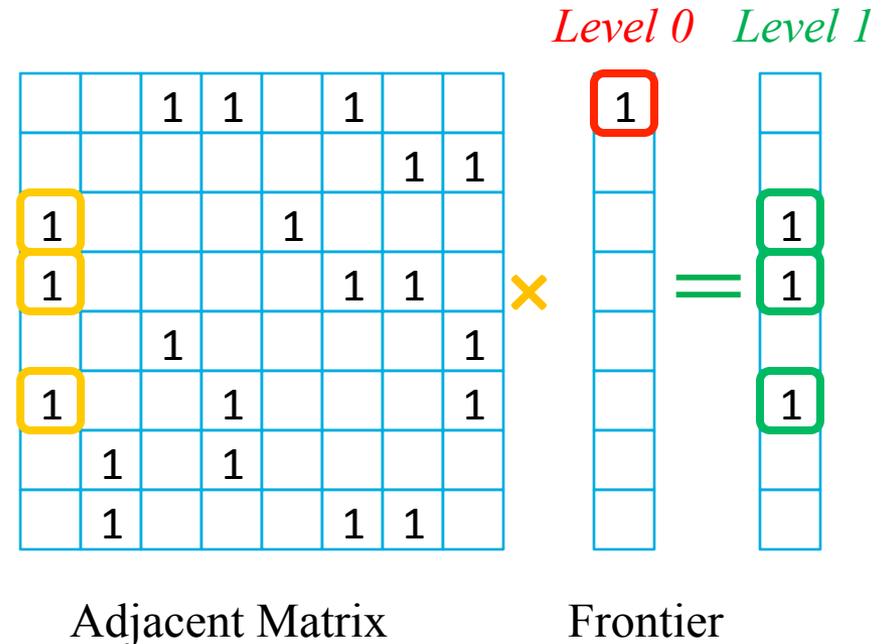
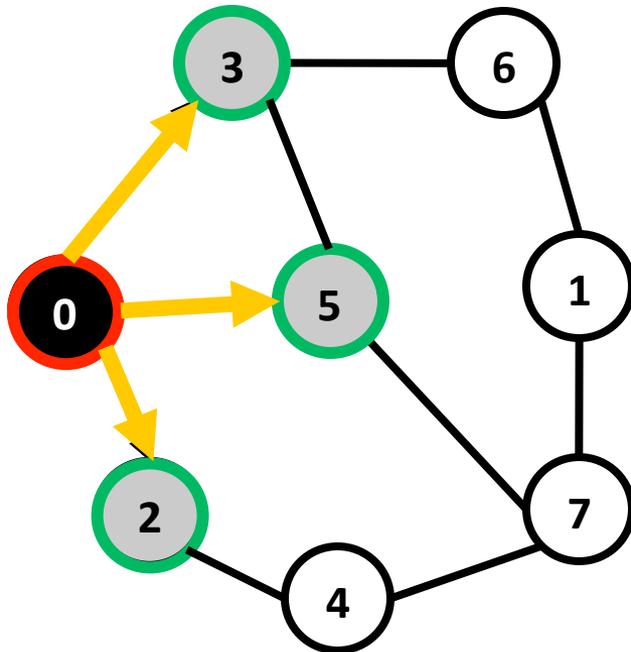
Motivation



Experiment: Time breakdown of a baseline Graph 500 BFS algorithm in a weak scaling experiment that use fixed problem size per node (each node has about 16M vertices).

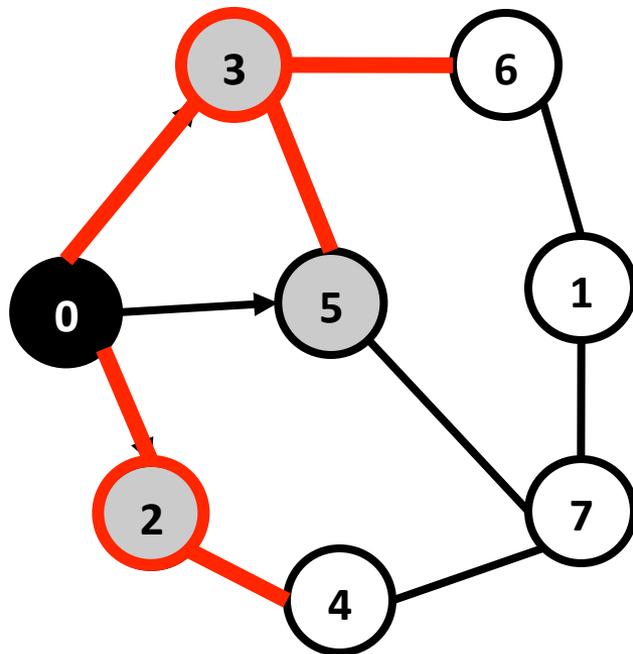
- Replicated-csr BFS in Graph 500, weak scaling
- ~70% time spent on **communication**

Breadth-First Search (BFS) Described in Linear Algebra



- Traversing one level as one Sparse Matrix-Vector Multiplication (SpMV)

Graph Partitioning in Distributed BFS



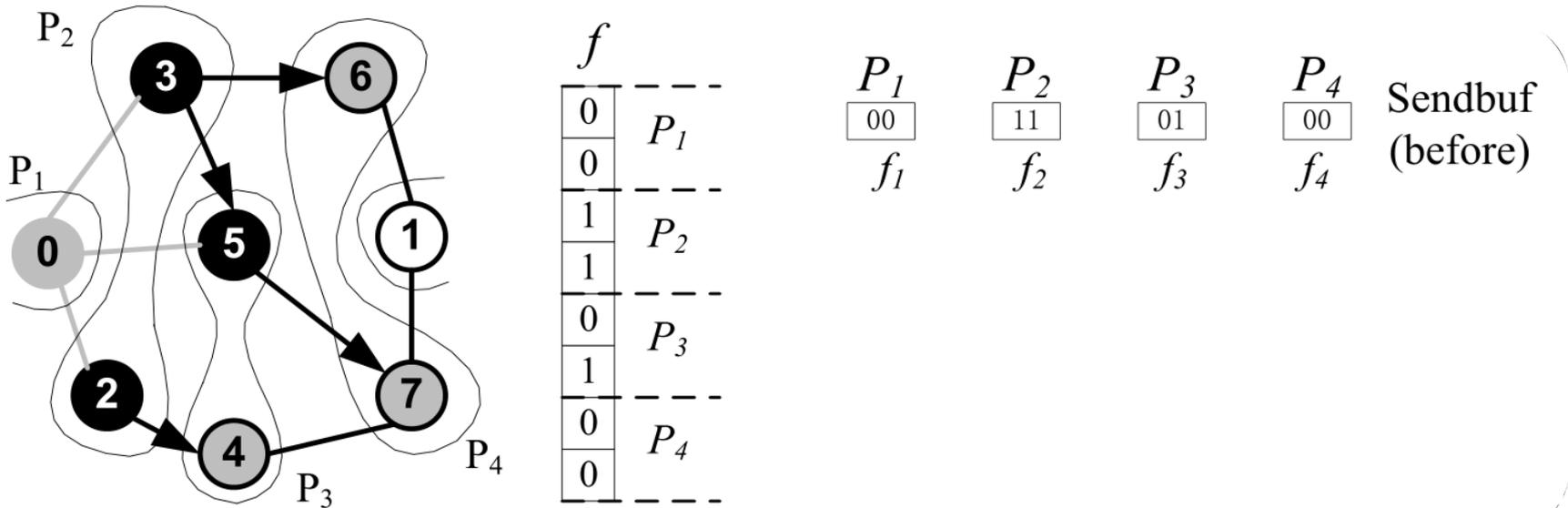
Level 0

		1	1	1			1	$P_1 : \{0,1\}$
1				1				$P_2 : \{2,3\}$
1				1	1			$P_3 : \{4,5\}$
	1						1	
1		1					1	
	1	1						
	1			1	1			$P_4 : \{6,7\}$

Adjacent Matrix Frontier

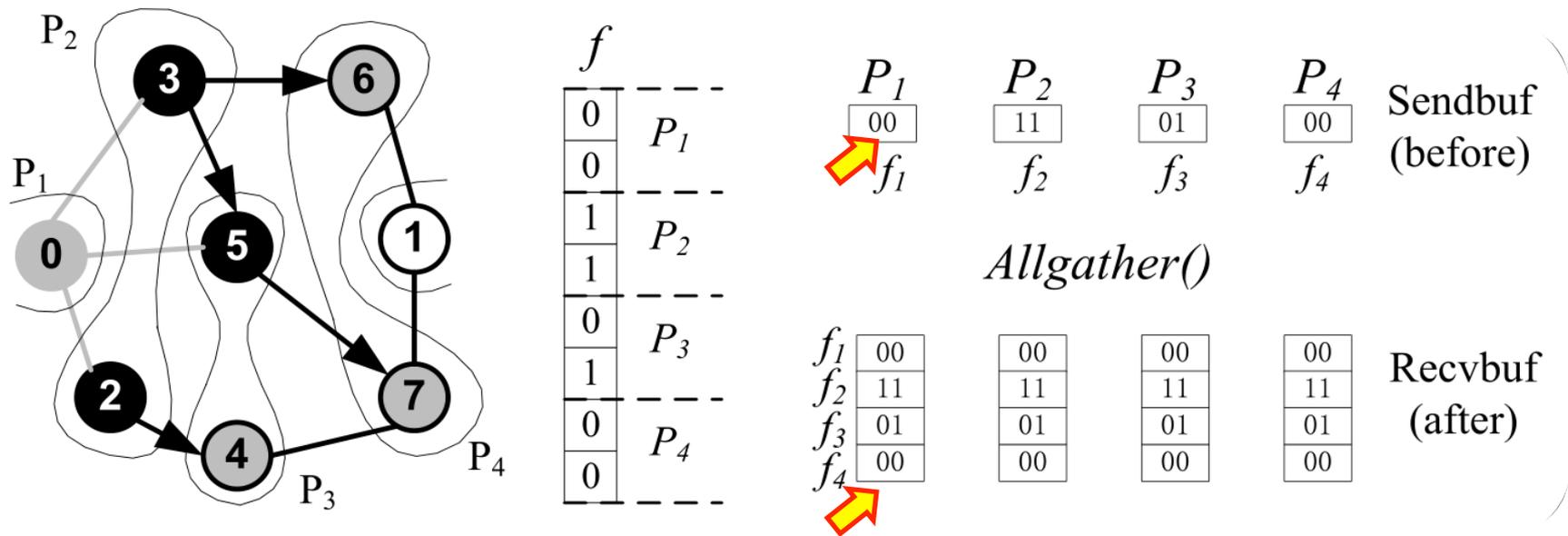
- Vertices, edges and frontier are partitioned among processors
 - for undirected graph, each edge is stored twice
- To get the global frontier information, MPI collective communication such as *MPI_Allgather* is needed

Communication in Baseline BFS



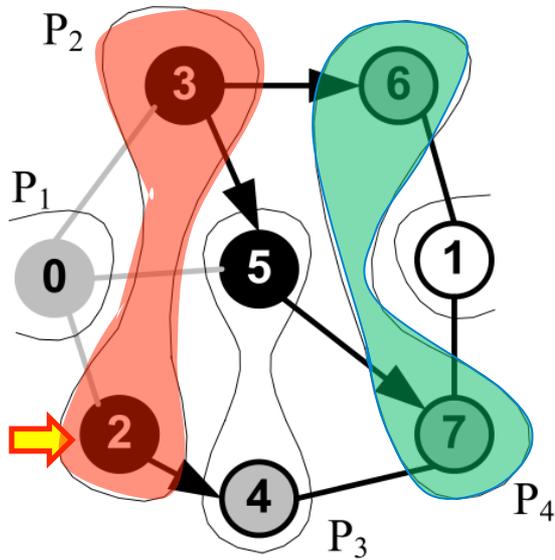
- *MPI_Allgather*

Communication in Baseline BFS

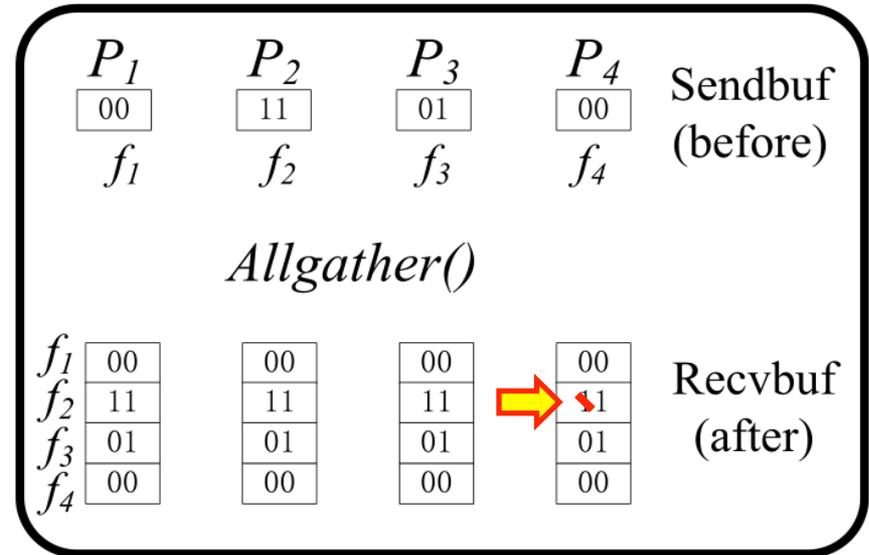


- The frontier vector is stored in bitmap
- *Observation 1: frontier is sparse*
- **Compress it!**

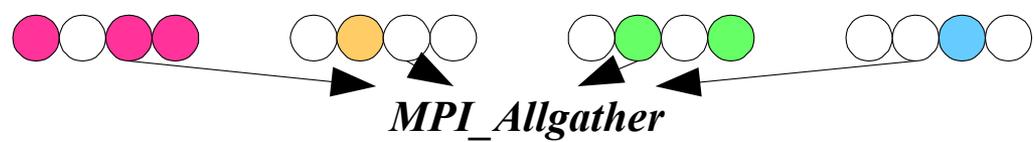
Communication in Baseline BFS



f	
0	P_1
0	
1	P_2
1	
0	P_3
1	
0	P_4
0	

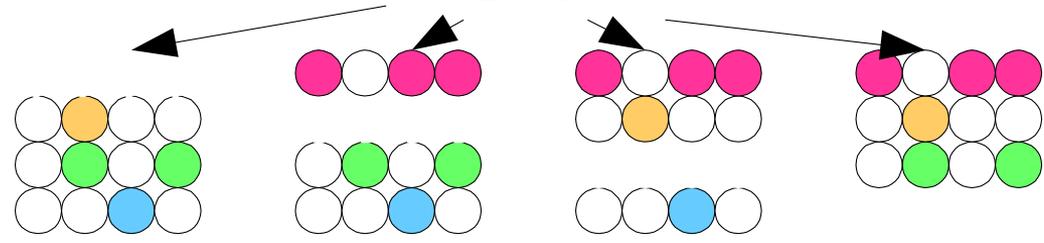


- *Observation 2: broadcast causes waste*
 - e.g. send v_2 to P_4 is unnecessary
- ➔ • **Sieve it before communication!**



Before

MPI_Allgather

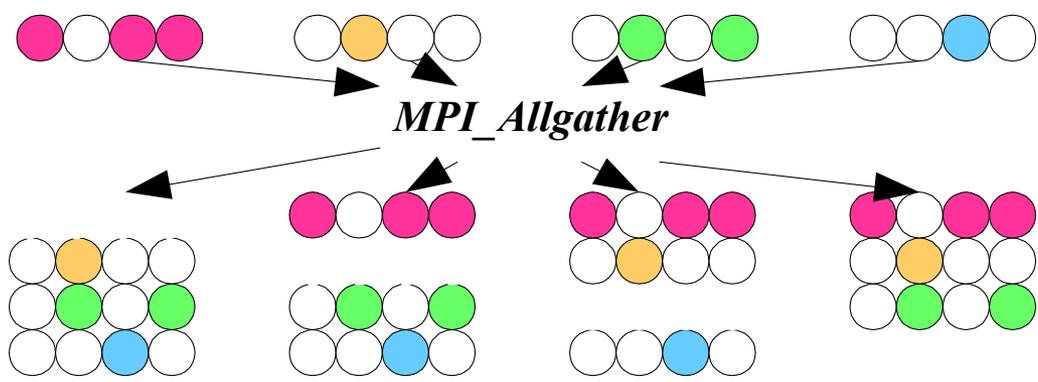


**Average
Communication:**

12

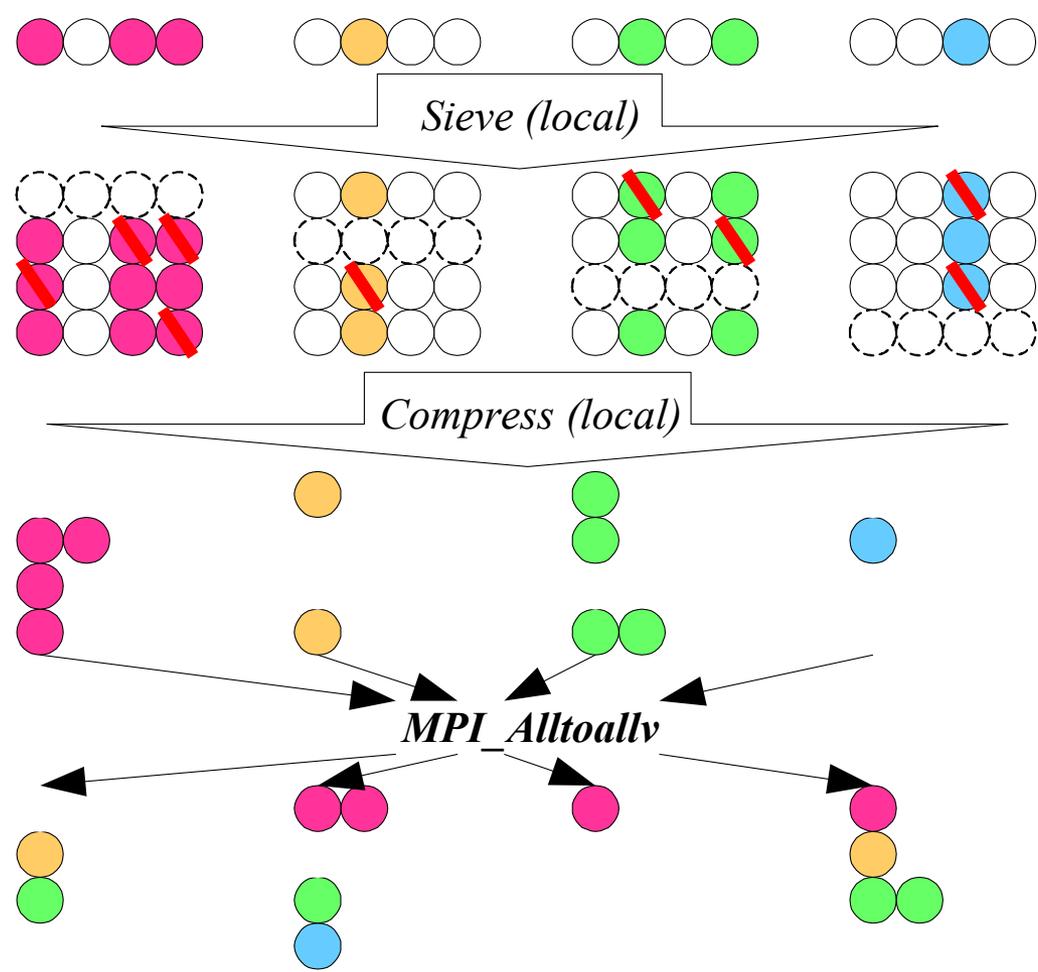


After



Before

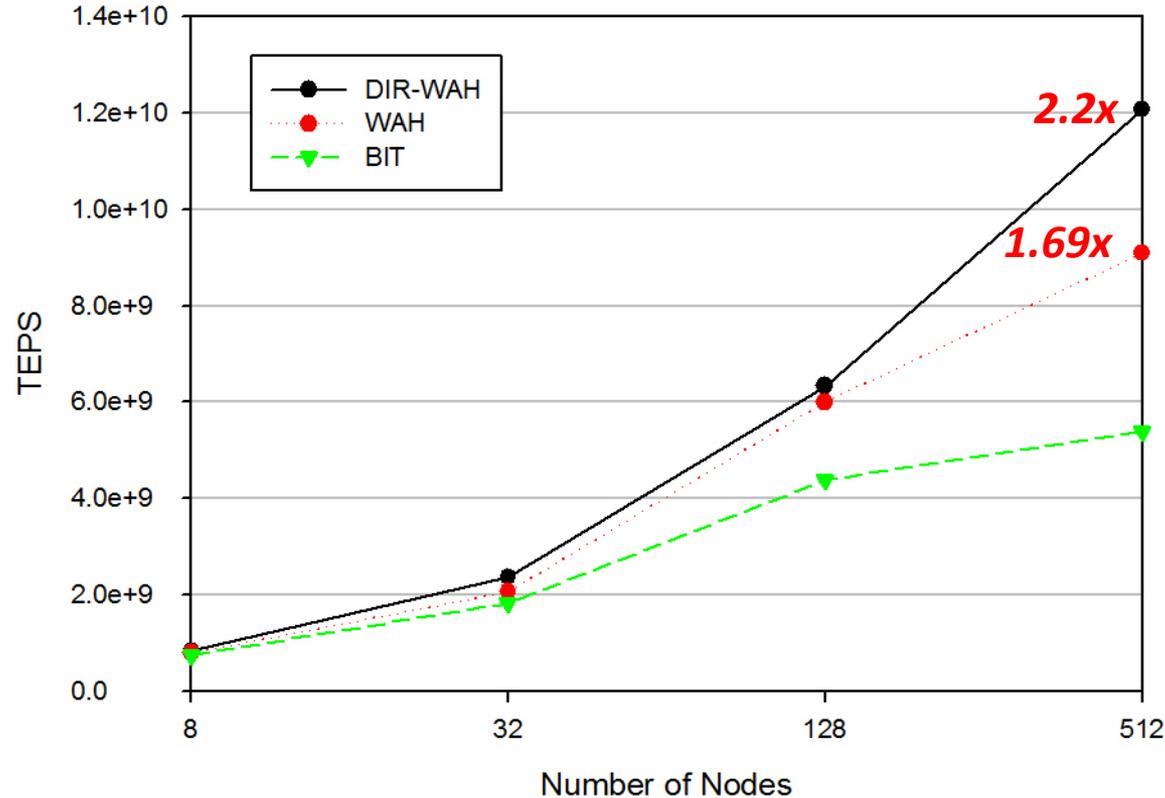
**Average
Communication:
12**



After

**Average
Communication:
2.75**

Weak Scaling Performance

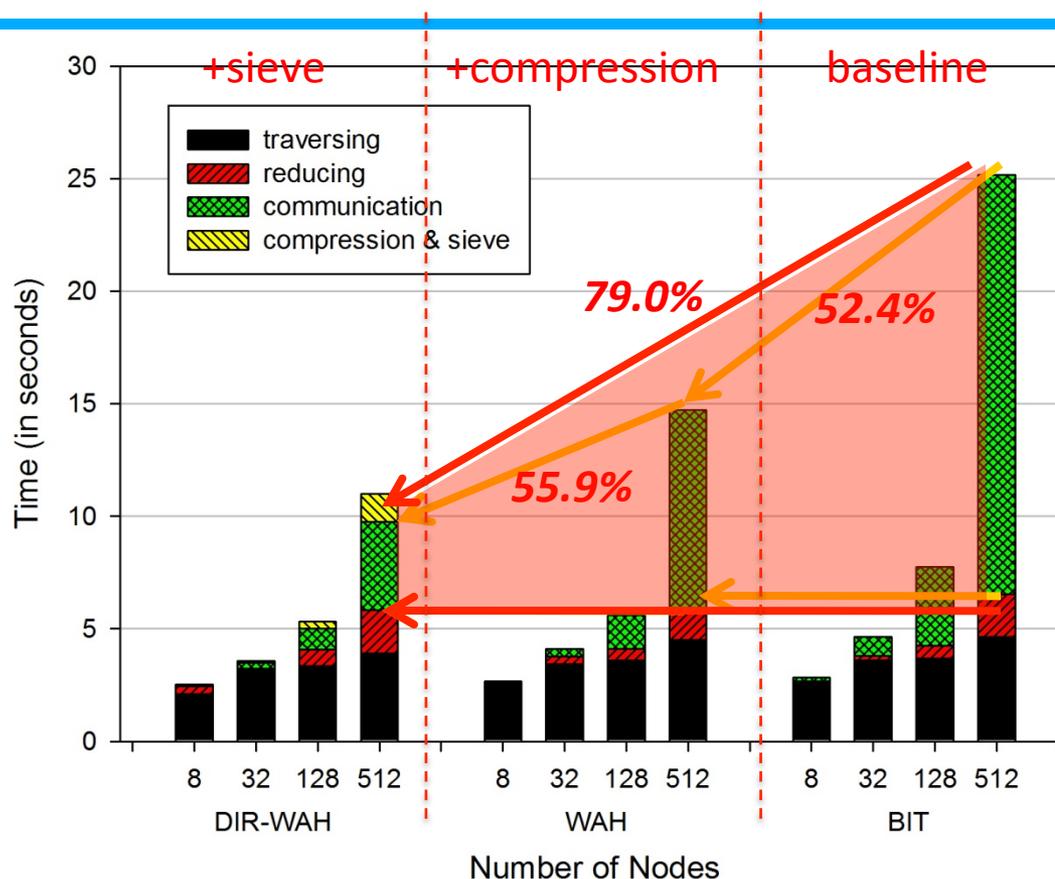


Experiment: fixed problem size per node (each node has about 16M vertices)

- *WAH* is **1.69x** faster than *BIT*
- *DIR-WAH* is **1.33x** faster than *WAH*, **2.2x** faster than *BIT*
- More benefits for larger scale



Time Breakdown



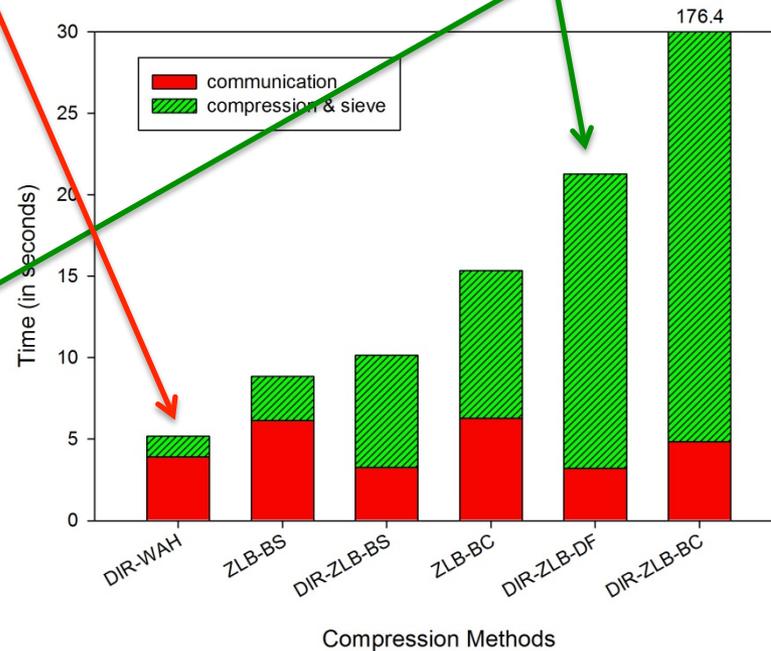
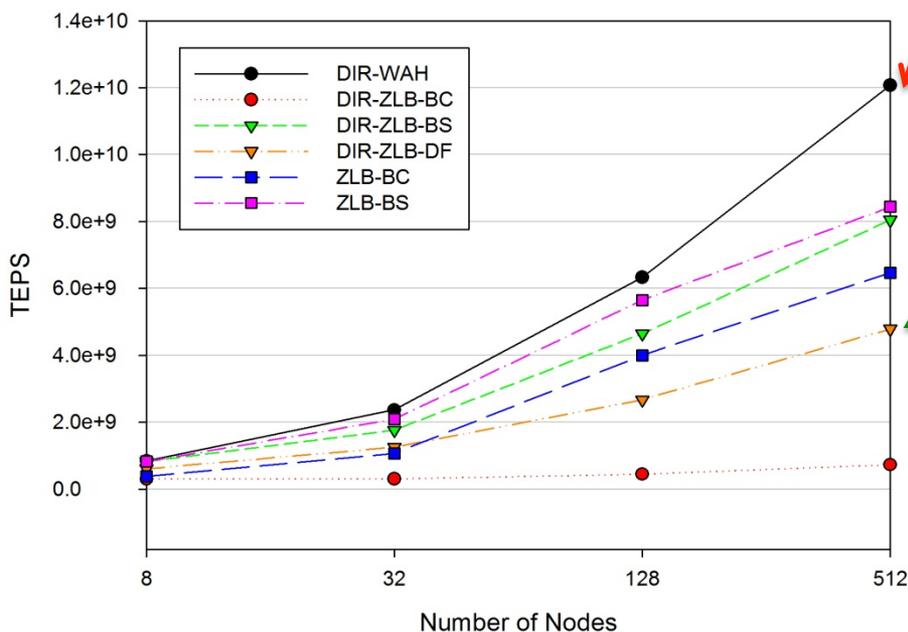
- *WAH* reduce communication time by **52.4%** compared to *BIT*
- *DIR-WAH* reduce communication time by another **55.9%** compared to *WAH*, a total **79.0%** reduction compared to *BIT*

Difference compression methods



good and fast compression

best compression ratio but slow



- Both compression and sieve trade computation for communication
- Zlib default(DIR-ZLB-DF): best compression ratio but its performance is not best
- WAH has the best performance because it has the best tradeoff between compression time and ratio

Conclusion



- Hybrid MPI/Pthreads BFS
 - explores core-level, memory-level and pipeline-level parallelism
 - better bandwidth and load balance
 - **1.9x** v.s. Graph 500 MPI-only on 32 nodes, **1.49x** v.s. CombBLAS MPI-only on 512 nodes
- Compression and Sieve help
 - trade computation for communication
 - reduce communication time by **79.0%**, improve performance **2.2x** on 512 nodes

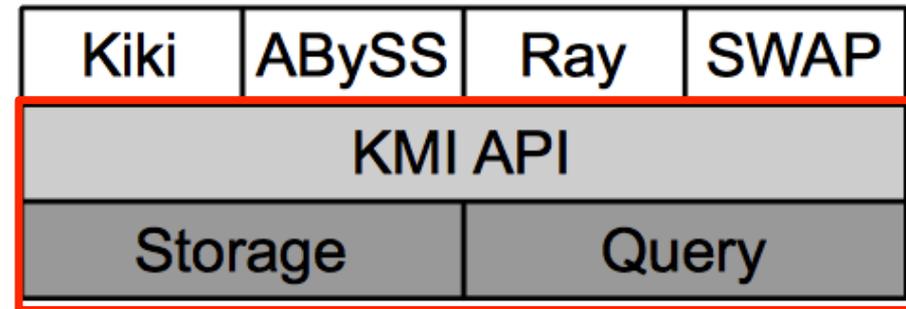
Ongoing Work

- KMI: K-mer Matching Interface for Genome Assembly in Terascale
 - HuiweiLu, Fangfang Xia and Pavan Balaji
- Goal:
 - Terascale genome assembly on Bluegene/P in 2 hours
 - Petascale genome assembly on Mira in 2 days (long term)



Motivation

- Terabytes of genome datasets need distributed storage and processing
- The fundamental operation is k-mer matching
 - k-mer: DNA sequence of length k
- The problem of current genome assemblers
 - Reinventing the wheels: k-mer matching
 - Inefficiency



KMI library

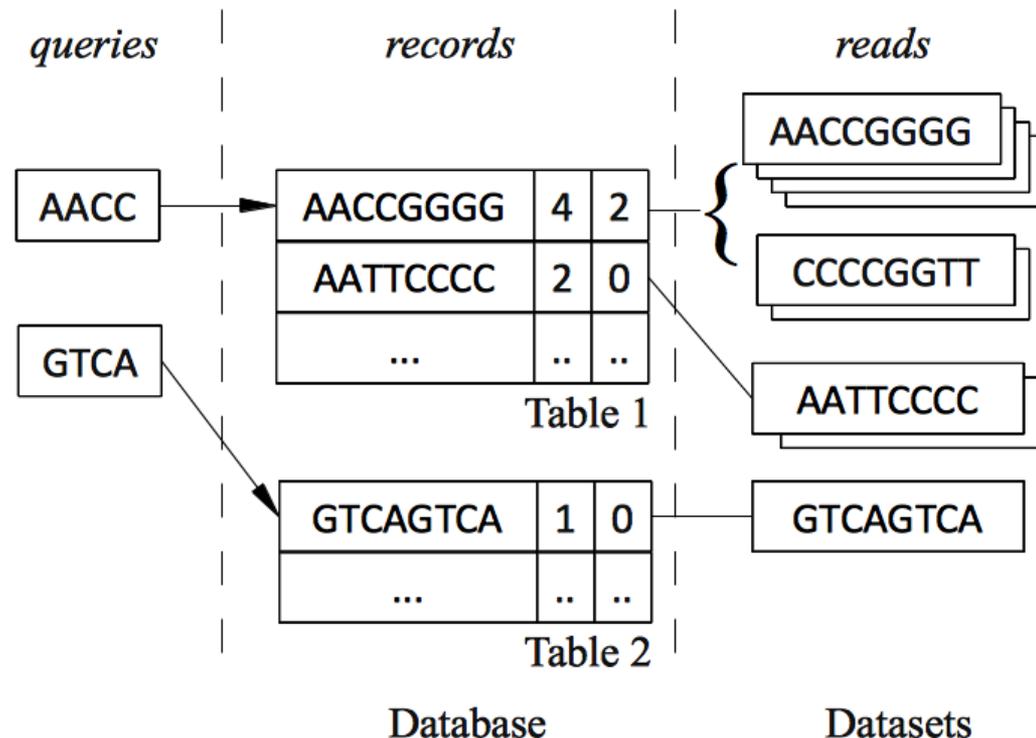


K-mer Matching Interface

- KMI is a distributed query processing system for managing and matching DNA sequence strings at scale.
- Data Model
 - {query : string} → {read: string, count : int, rc count : int}
 - Reads from the datasets are stored in a database. A query search the database.

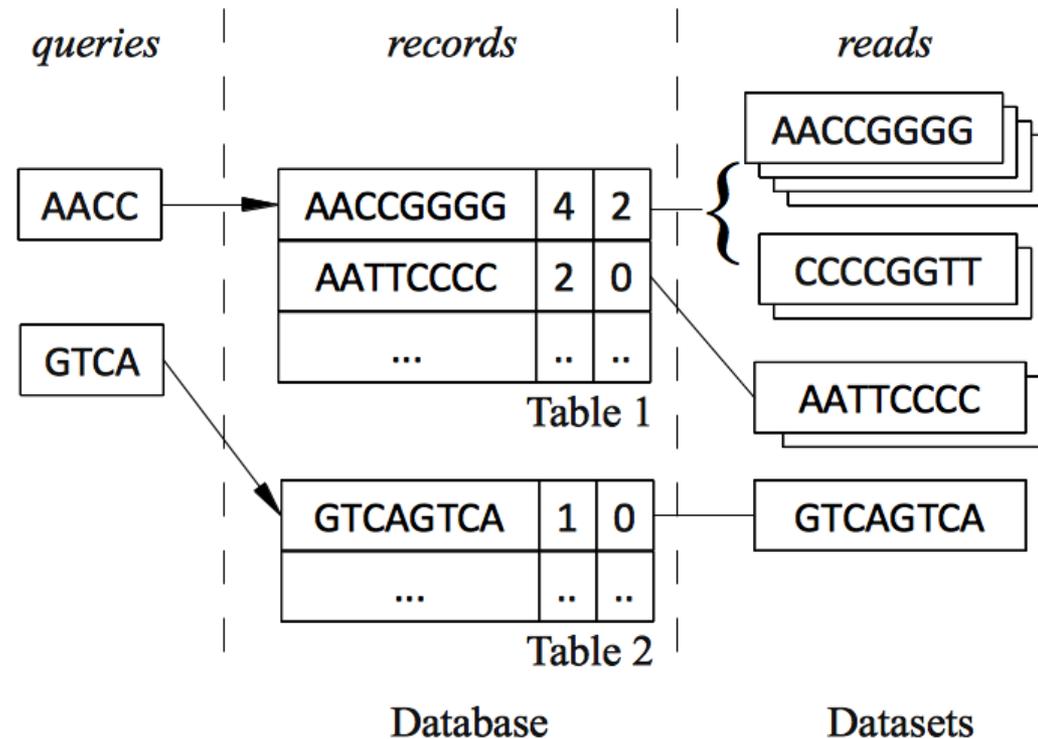
Example:

Search all the strings with prefix “AACC”



Levels of abstraction

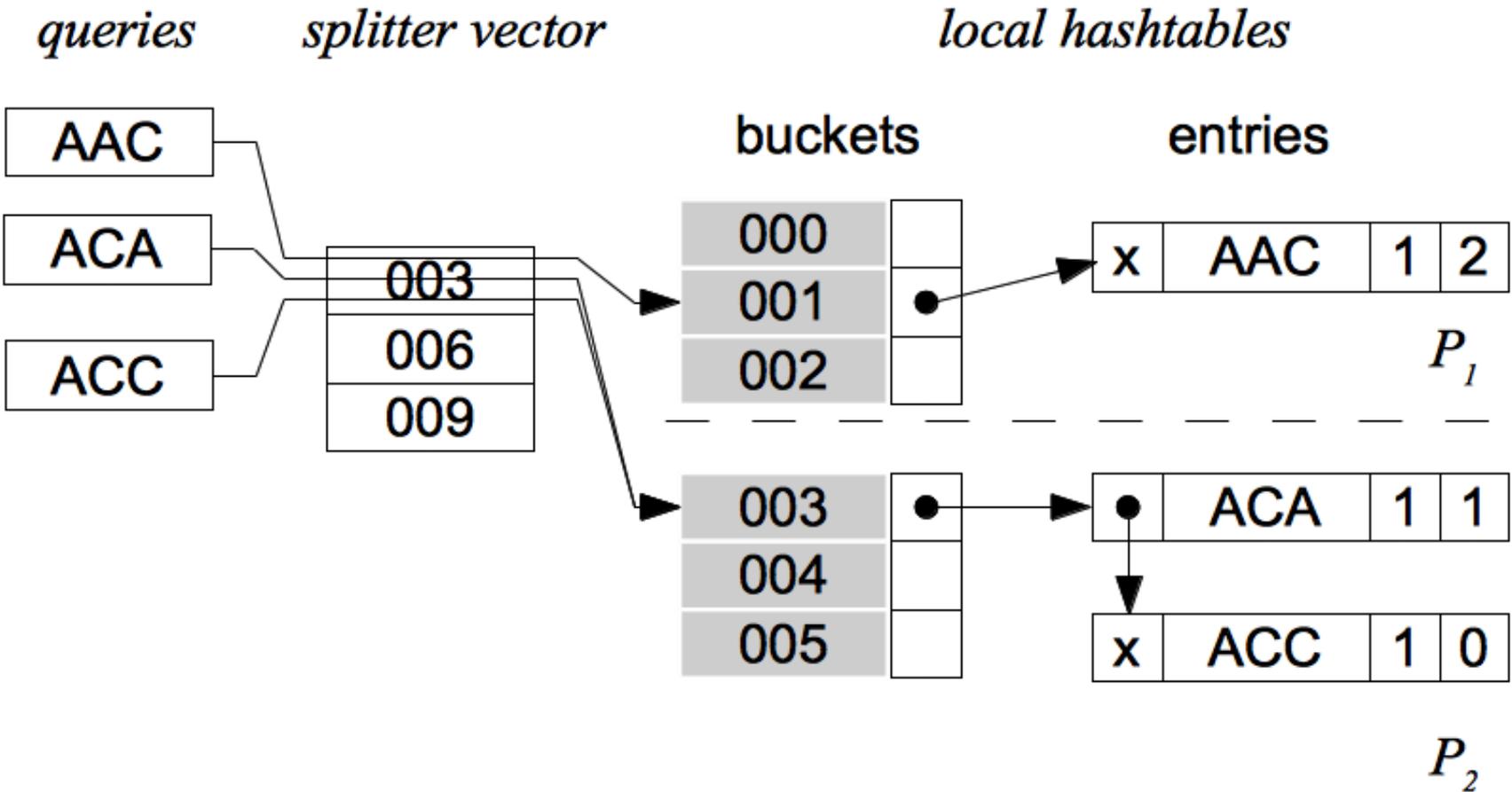
- Records
 - The unit for rebalance and query
- Tables
 - User can organize their data into tables, e.g., different species can have different tables
- Database



Query

Step 1: locate the processor

Step 2: locate the record





Huiwei Lu

Advisor: Prof. Ninghui Sun



中国科学院计算技术研究所

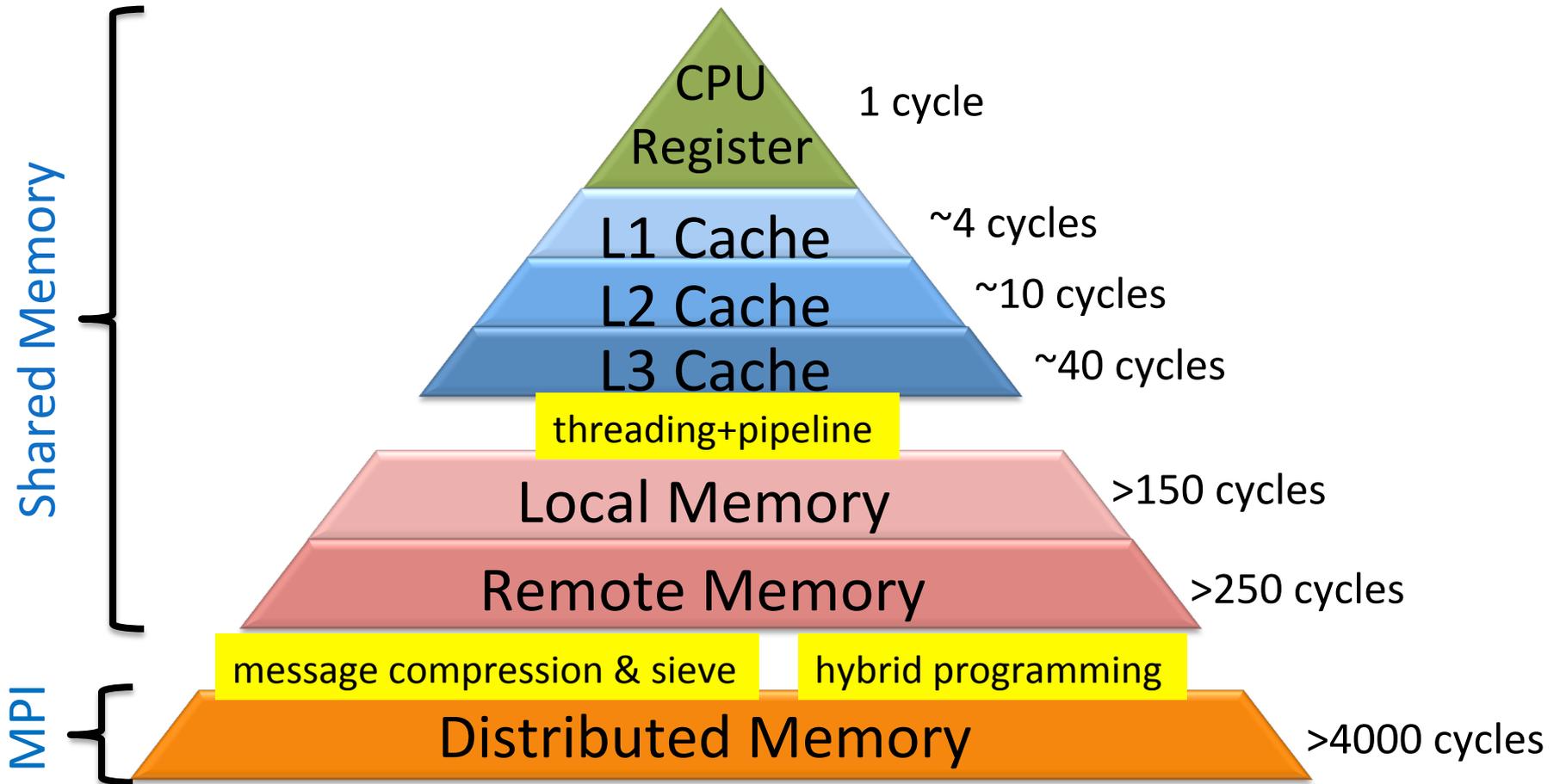
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

lvhuiwei@gmail.com

Backup Slides



Mapping BFS to Memory Hierarchy



- Multi-threading & pipeline: hide memory latency
- Compression & Sieve: trading comp. for comm.

Publications



- *Understanding Parallelism in Graph Traversal on Multi-core Clusters*, ISC'12
- *Reducing Communication in Parallel Breadth First Search on Distributed Memory Systems*. (In Submission)
- *P-GAS: Parallelizing a Cycle-Accurate Event-Driven Many-Core Processor Simulator Using Parallel Discrete Event Simulation*. PADS 2010
- *Simulation of Many-core Processor and Many-core Clusters*. JCRD. 2013
- Co-author:
 - *Evaluation and Optimization of Breadth-First Search on NUMA Cluster*. IEEE Cluster 2012
 - *HPPNetSim: A Parallel Simulation of Large-scale Interconnection Networks*. SpringSim '09
 - *SimK: A Large-Scale Parallel Simulation Engine*. JCST 2009

Awards



- **Institute Chief Award (Top 3%), 2011**
- Outstanding research assistant of the Computer Architecture Laboratory, ICT, 2010
- Triple-A outstanding student of the Chinese Academy of Sciences, 2012, 2009, 2008
- Outstanding class leader of the Chinese Academy of Sciences, 2008
- Social Activities
 - President of the Open Source Software Community of the Chinese Academy of Sciences (One of Top 10 Outstanding Communities of the Year), 2008