Large Scale Graph Analysis

Erik Saule

HPC Lab
Biomedical Informatics
The Ohio State University

March 11, 2013
UMass Boston

Outline

Data in the Modern Days

Facebook

1B active users a month. Each day:

- 2.5B content items shared
- 2.7B Likes
- 300M photos
- 500TB data

Data in the Modern Days

Facebook

1B active users a month. Each day:

- 2.5B content items shared
- 2.7B Likes
- 300M photos
- 500TB data

Twitter

- 500M users
- 340M tweets/day (2,200/sec)
- 24.1M super bowl tweets

Data in the Modern Days

Facebook

1B active users a month. Each day:

- 2.5B content items shared
- 2.7B Likes
- 300M photos
- 500TB data

Twitter

- 500M users
- 340M tweets/day (2,200/sec)
- 24.1M super bowl tweets

Academic networks

- 1.5M papers/year (4,000/day)
- 100,000 papers/year in CS

Data in the Modern Days

Facebook

1B active users a month. Each day:

- 2.5B content items shared
- 2.7B Likes
- 300M photos
- 500TB data

Twitter

- 500M users
- 340M tweets/day (2,200/sec)
- 24.1M super bowl tweets

Academic networks

- 1.5M papers/year (4,000/day)
- 100,000 papers/year in CS

Transportation

- 10M trips in Paris public transportation/day
- 2.5M registered vehicles in LA
- 1.2M used for commuting/day

Data in the Modern Days

Facebook

1B active users a month. Each day:

- 2.5B content items shared
- 2.7B Likes
- 300M photos
- 500TB data

Twitter

- 500M users
- 340M tweets/day (2,200/sec)
- 24.1M super bowl tweets

Academic networks

- 1.5M papers/year (4,000/day)
- 100,000 papers/year in CS

Transportation

- 10M trips in Paris public transportation/day
- 2.5M registered vehicles in LA
- 1.2M used for commuting/day

Compositing

Problems can also come from multiple sources, e.g., identify coauthors in Facebook.

Are these problems new?

"CERN report 1959" about a 1H experiment on the synchrocyclotron

The use of the computer in this sort of measurement is important, not only because of the large amounts of data which must be handled, but because with a modern high speed computer one can search quickly for various systematic errors.

Are these problems new?

"CERN report 1959" about a 1H experiment on the synchrocyclotron

The use of the computer in this sort of measurement is important, not only because of the large amounts of data which must be handled, but because with a modern high speed computer one can search quickly for various systematic errors.

But also...

- Intrusion detection in computer security
- Search engines
- Stock market predictions
- Weather forecast

Are these problems new?

"CERN report 1959" about a 1H experiment on the synchrocyclotron

The use of the computer in this sort of measurement is important, not only because of the large amounts of data which must be handled, but because with a modern high speed computer one can search quickly for various systematic errors.

But also...

- Intrusion detection in computer security
- Search engines
- Stock market predictions
- Weather forecast

Not so new!

So why is it important now?

Ubiquitous

- Scientist (LHC, Metagenomics)
- Big companies (Data companies, Operational marketing)
- Small companies (Website logs, who buys what? where?)
- People (Personal analytics)

So why is it important now?

Ubiquitous

- Scientist (LHC, Metagenomics)
- Big companies (Data companies, Operational marketing)
- Small companies (Website logs, who buys what? where?)
- People (Personal analytics)

In brief, everybody has Big Data problems now!

None of these data can be manually analyzed. Automatic analysis is mandatory.

The Three Attributes of Big Data

Variety	Velocity	Volume
unstructured data	flowing in the system	in high volume

The Three Attributes of Big Data

Variety	Velocity	Volume
unstructured data	flowing in the system	in high volume
Graphs Hypergraphs Conceptual data	Streaming data Temporal data Flow of queries	Millions, Billions, Trillions of vertices and edges

The Three Attributes of Big Data

Variety	Velocity	Volume
unstructured data	flowing in the system	in high volume
Graphs Hypergraphs Conceptual data	Streaming data Temporal data Flow of queries	Millions, Billions, Trillions of vertices and edges

Problems

- Storing and transporting such data
- Extracting the important data and building a graph (or else)
- Analyzing the graph:
  - static analysis
  - recurrent analysis
  - temporal analysis

My Goal

Study Big Data problems and design solutions for them.

My Goal

Study Big Data problems and design solutions for them.

Applications (Source)

Facebook, the**advisor**, twitter, CiteULike, traffic camera, transportation systems

My Goal

Study Big Data problems and design solutions for them.

Applications (Source)

Facebook, the**advisor**, twitter, CiteULike, traffic camera, transportation systems

Algorithms (Analysis)

Page Rank, Random Walk, Traversals, Centrality, Community Detection, Outlier Detection, Visualization

My Goal

Study Big Data problems and design solutions for them.

Applications (Source)

Facebook, the**advisor**, twitter, CiteULike, traffic camera, transportation systems

Middleware

MPI, Hadoop, Pegasus, Graph Lab, DOoC+LAF, DataCutter, SQL, SPARQL

Algorithms (Analysis)

Page Rank, Random Walk, Traversals, Centrality, Community Detection, Outlier Detection, Visualization

My Goal

Study Big Data problems and design solutions for them.

Applications (Source)

Facebook, the**advisor**, twitter, CiteULike, traffic camera, transportation systems

Middleware

MPI, Hadoop, Pegasus, Graph Lab, DOoC+LAF, DataCutter, SQL, SPARQL

Algorithms (Analysis)

Page Rank, Random Walk, Traversals, Centrality, Community Detection, Outlier Detection, Visualization

Hardware

Clusters, Cray XMT, Intel Xeon Phi, FPGAS, SSD drives, NVRAM, Infiniband, Cloud Computing, GPU.

My Goal

Study Big Data problems and design solutions for them.

Applications (Source)

Facebook, the**advisor**, twitter, CiteULike, traffic camera, transportation systems

Middleware

MPI, Hadoop, Pegasus, Graph Lab, DOoC+LAF, DataCutter, SQL, SPARQL

Algorithms (Analysis)

Page Rank, Random Walk, Traversals, Centrality, Community Detection, Outlier Detection, Visualization

Hardware

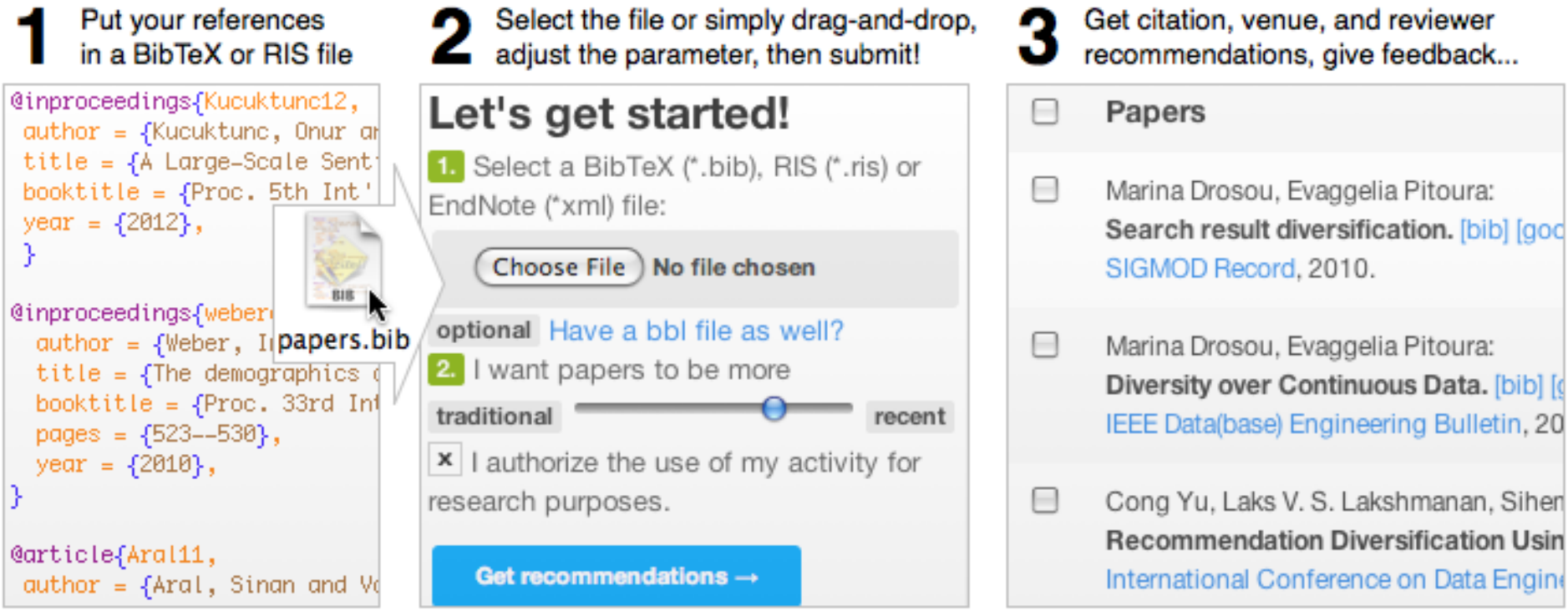
Clusters, Cray XMT, Intel Xeon Phi, FPGAS, SSD drives, NVRAM, Infiniband, Cloud Computing, GPU.

What to use? When to use them? What is missing?

Outline

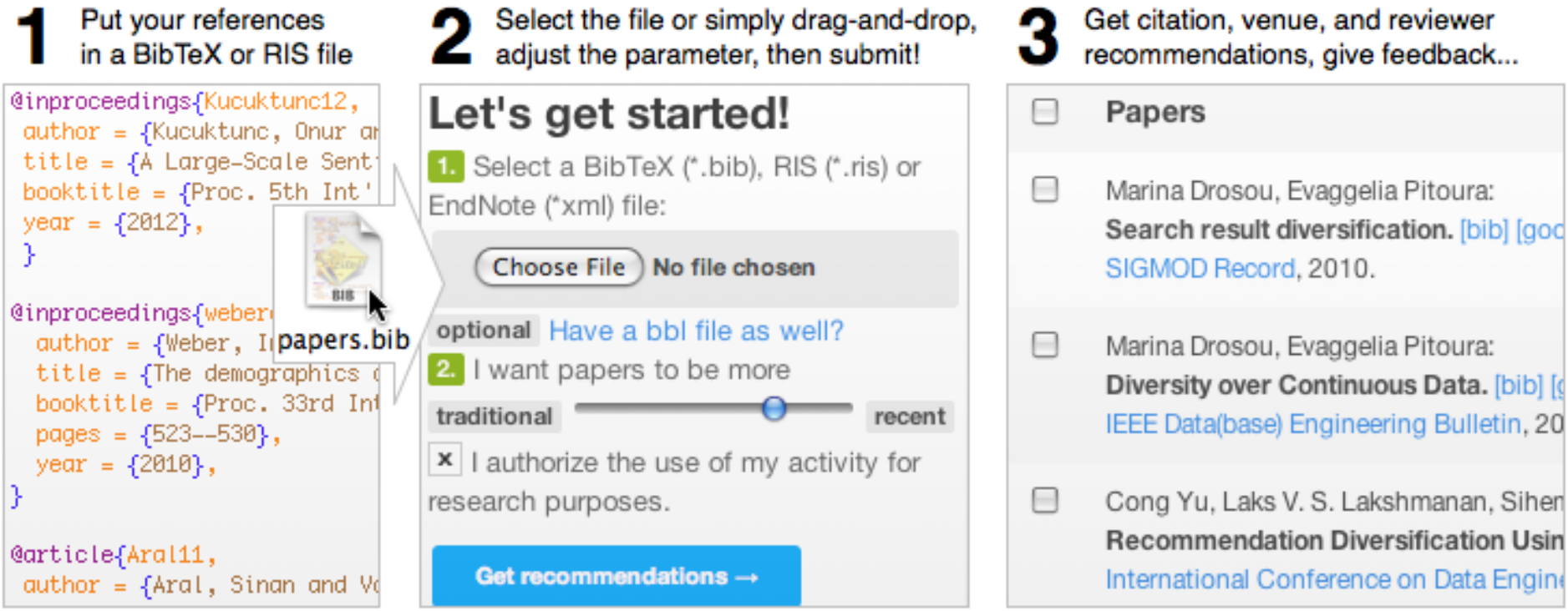
A Use Case

http://theadvisor.osu.edu/



A Use Case

http://theadvisor.osu.edu/



Using the Citation Graph

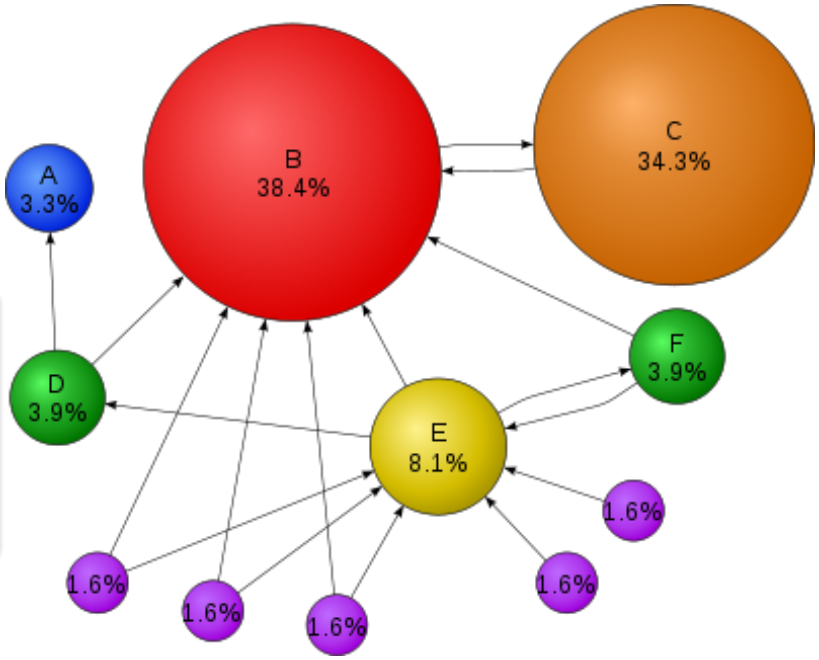
Hypothesis: If two papers are related or treat the same subject, then they will be close to each other in the citation graph (and reciprocal)

Global Approches: PageRank

Let $G = (V, E)$ be the citation graph

Personalized PageRank [Haveliwala02]

$\pi_i(u) = dp^*(u) + (1-d)\sum_{v \in N(u)} \frac{\pi_{i-1}(v)}{\delta(v)}$
with $\sum p^*(u) = 1$.



source: wikipedia

Direction Awareness [DBRank12]

Time exploration

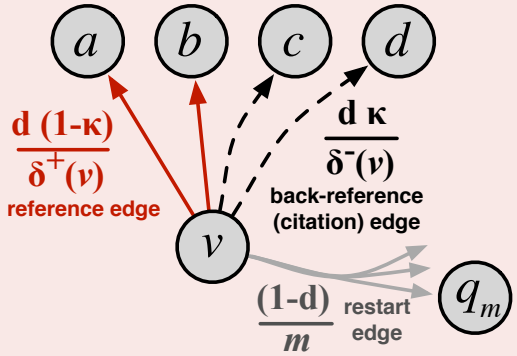
What if we are interested in searching papers per years. Recent papers? Traditional papers?

Let $\mathcal{Q}$ be a set of known relevant papers.

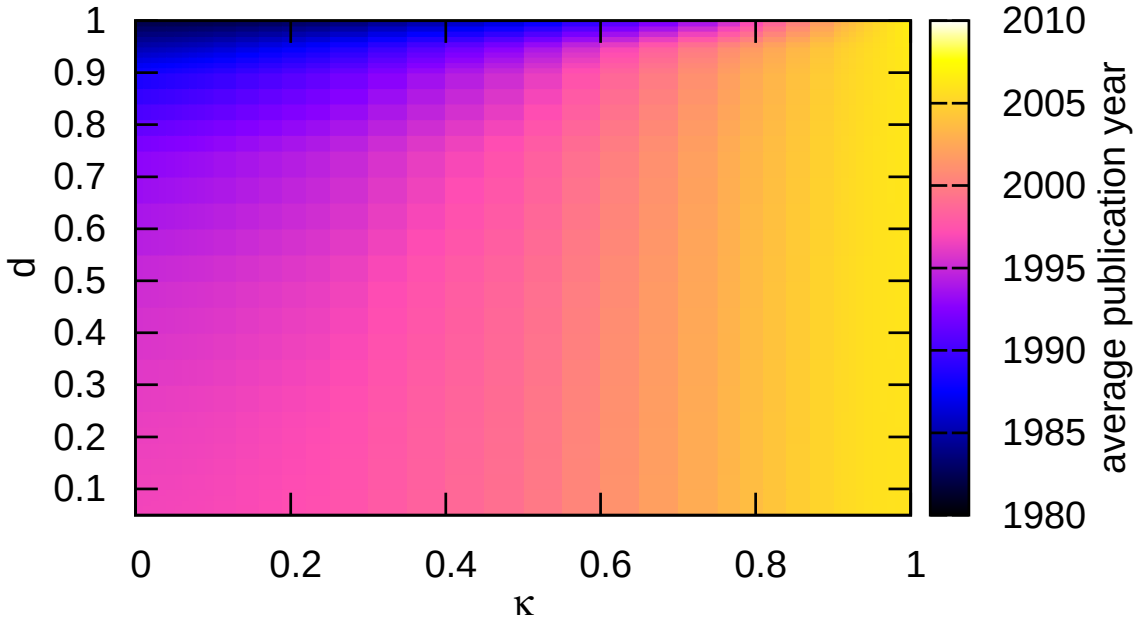
Direction Aware Random Walk with Restart

$$\pi_i(u) = dp^*(u) + (1-d)(\kappa \sum_{v \in N^+(u)} \frac{\pi_{i-1}(v)}{\delta^-(v)} + (1-\kappa) \sum_{v \in N^-(u)} \frac{\pi_{i-1}(v)}{\delta^+(v)})$$

- $d \in (0:1)$ is the damping factor.
- $\kappa \in (0:1)$.
- $p^*(u) = \frac{1}{|\mathcal{Q}|}$, if $u \in \mathcal{Q}$, $p^*(u) = 0$, otherwise



Analysis: Time and Accuracy



	hide random			hide recent			hide earlier		
	mean	interval		mean	interval		mean	interval	
DARWR	48.00	46.80	49.20	**42.22**	**40.95**	**43.50**	**60.64**	**59.48**	**61.80**
P.R.	**56.56**	**55.31**	**57.80**	38.75	37.50	40.00	58.93	57.76	60.10
Katz$_\beta$	46.33	45.16	47.50	34.56	33.42	35.70	44.19	42.97	45.40
Cocit	44.60	43.39	45.80	14.22	13.25	15.20	55.97	54.64	57.30
Cocoup	17.28	16.36	18.20	17.56	16.61	18.50	2.93	2.57	3.30
CCIDF	18.05	17.11	19.00	18.97	17.94	20.00	3.55	3.10	4.00

A Sparse Matrix-Vector Multiplication (SpMV)

Rewriting DARWR

$$\pi_i(u) = dp^*(u) + (1-d)\left(\kappa \sum_{v \in N^+(u)} \frac{\pi_{i-1}(v)}{\delta^-(v)} + (1-\kappa) \sum_{v \in N^-(u)} \frac{\pi_{i-1}(v)}{\delta^+(v)}\right)$$

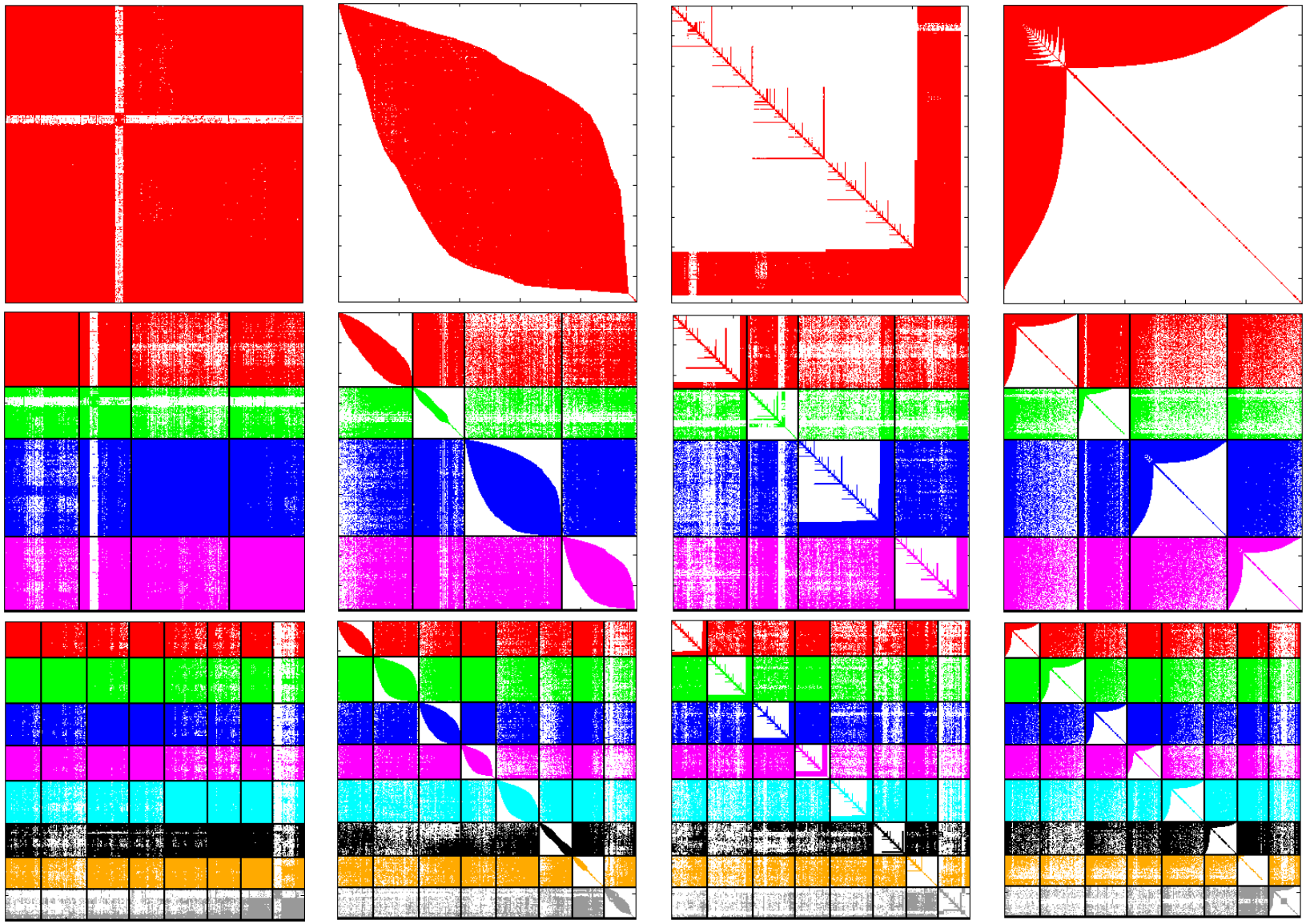
$$\pi_i(u) = dp^*(u) + \sum_{v \in N^+(u)} \frac{(1-d)\kappa}{\delta^-(v)} \pi_{i-1}(v) + \sum_{v \in N^-(u)} \frac{(1-d)(1-\kappa)}{\delta^+(v)} \pi_{i-1}(v)$$

$$\pi_i = dp^* + \mathbf{A}^- \pi_{i-1} + \mathbf{A}^+ \pi_{i-1}$$

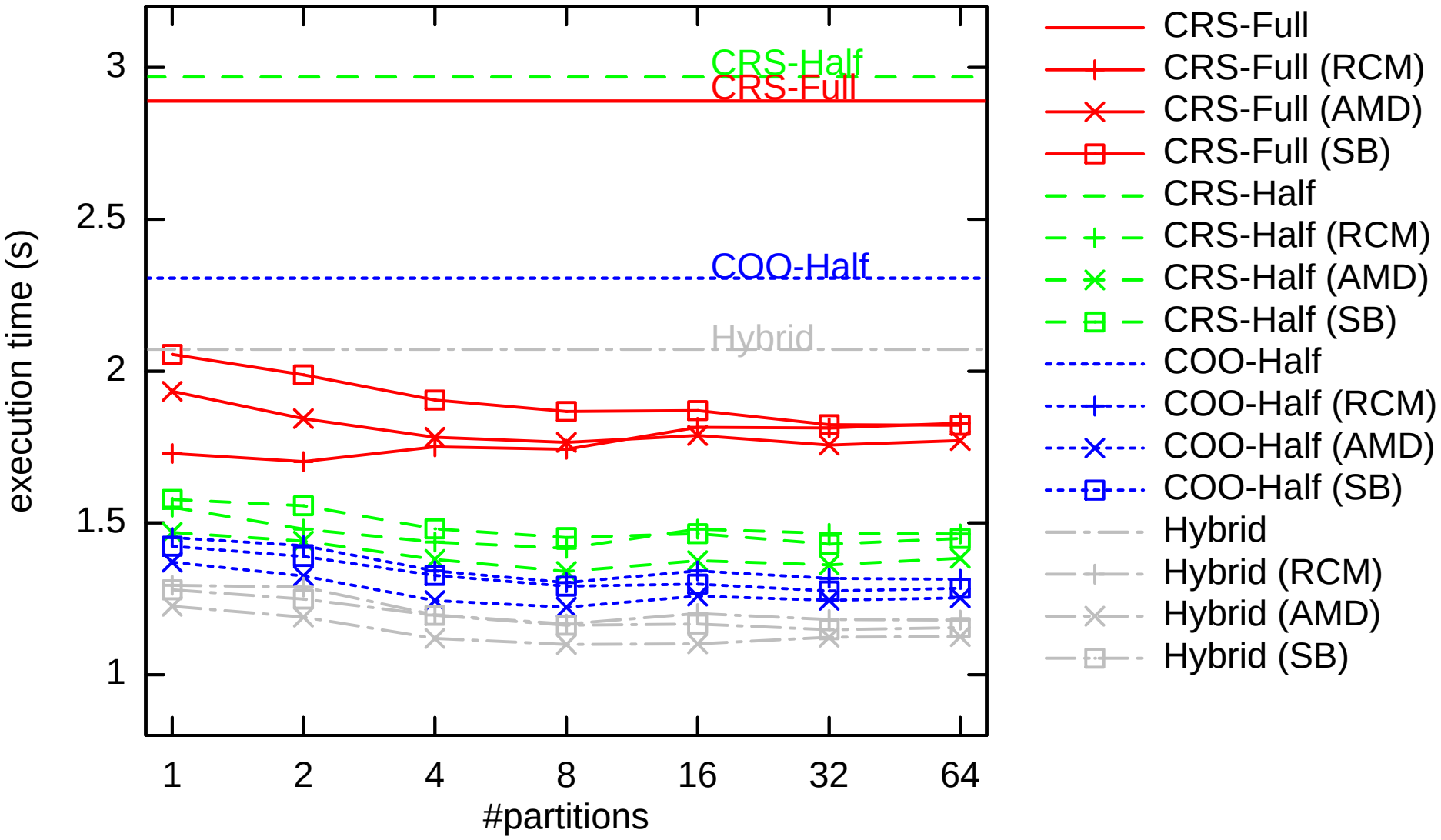
$$\pi_i = dp^* + \mathbf{A} \pi_{i-1} \qquad \textit{(CRS Full)}$$

$$\pi_i = dp^* + \mathbf{B}^- \left(\frac{(1-d)\kappa}{\delta^-} \pi_{i-1}\right) + \mathbf{B}^+ \left(\frac{(1-d)(1-\kappa)}{\delta^+} \pi_{i-1}\right) \qquad \textit{(CRS Half)}$$

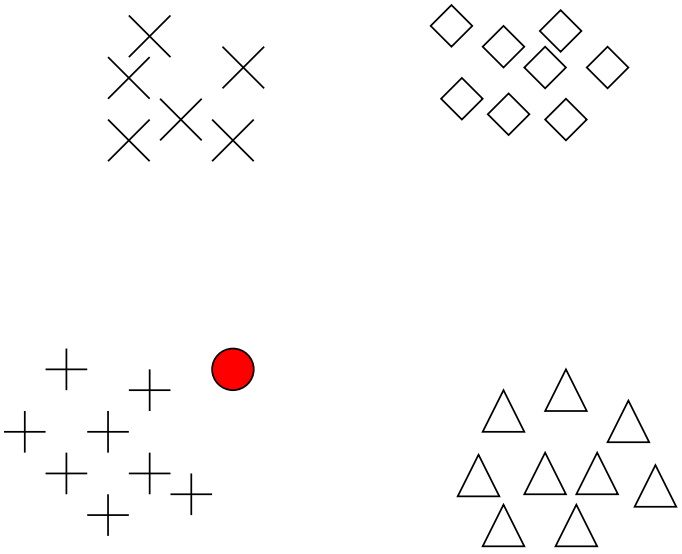
Partitioning and Ordering



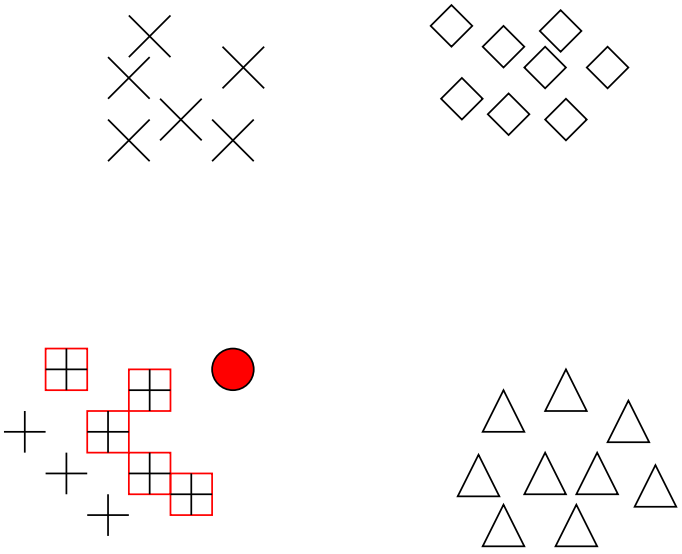
Runtimes - AMD Opteron 2378 [ASONAM12]



Diversification: Principle

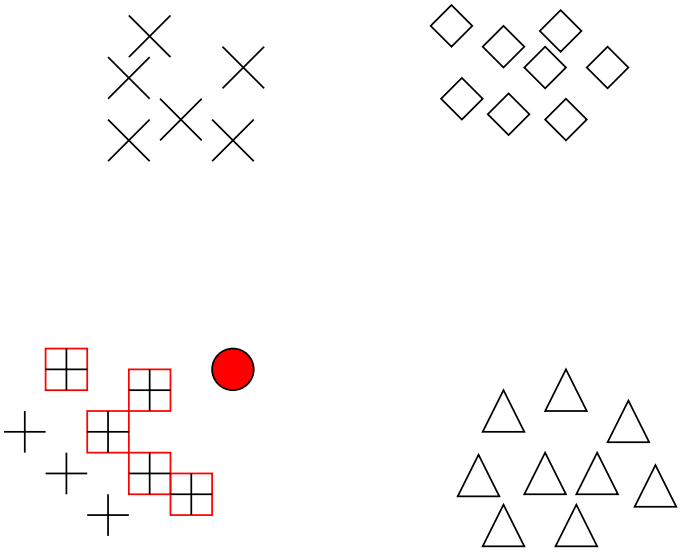


Diversification: Principle

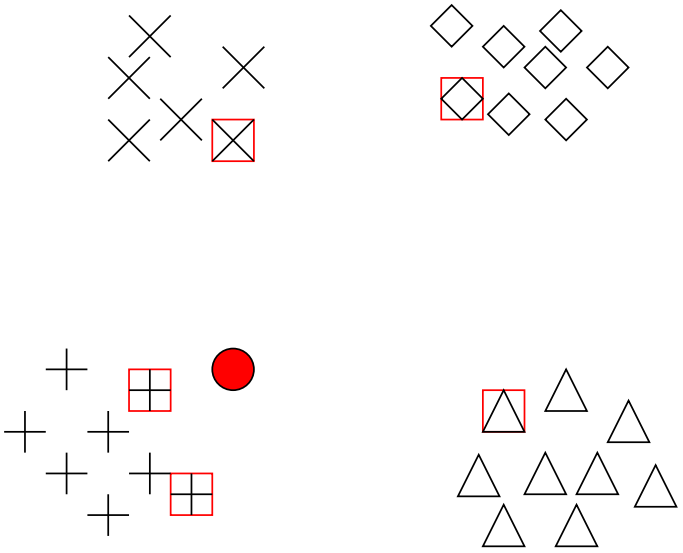


Relevant

Diversification: Principle

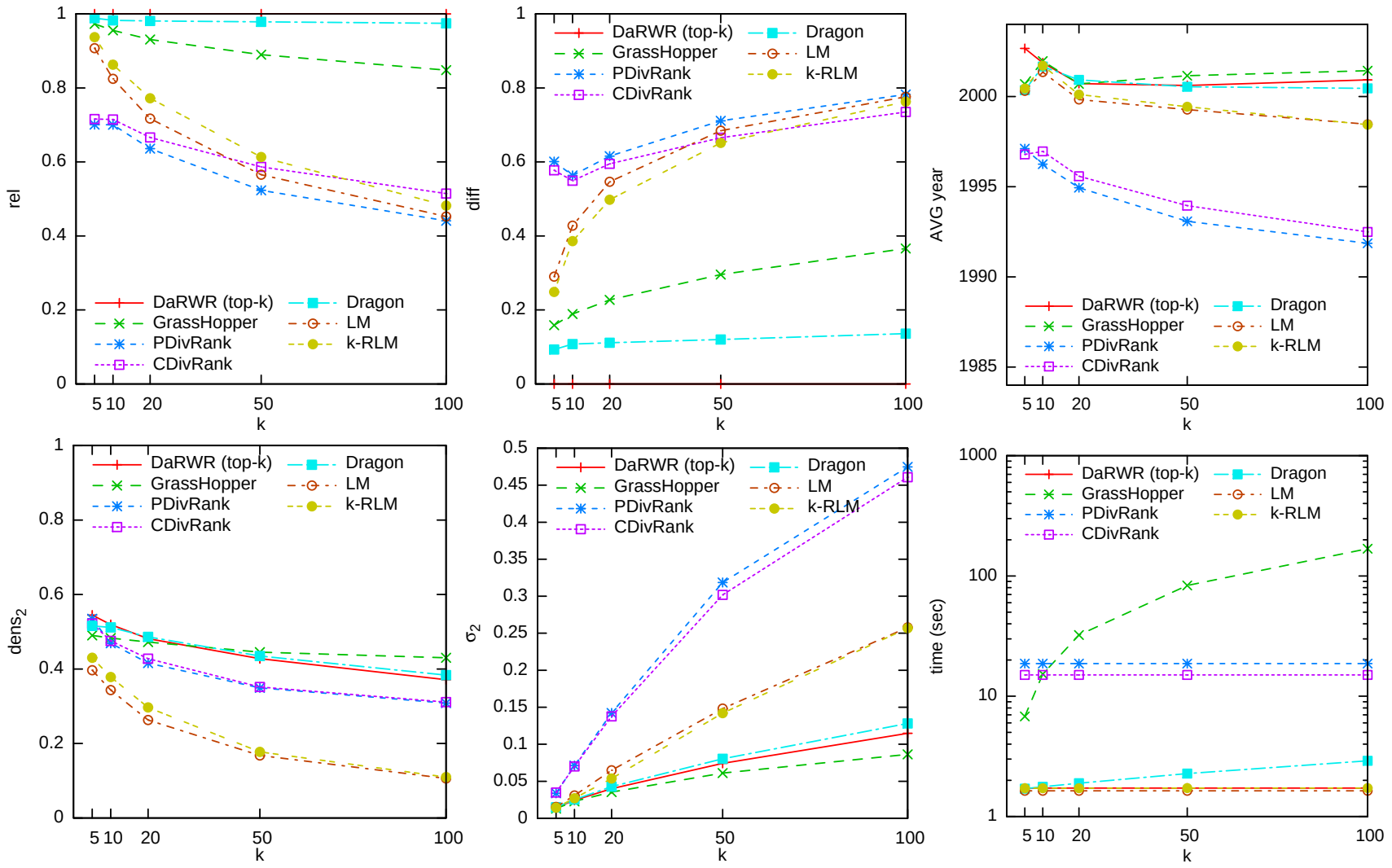


Relevant

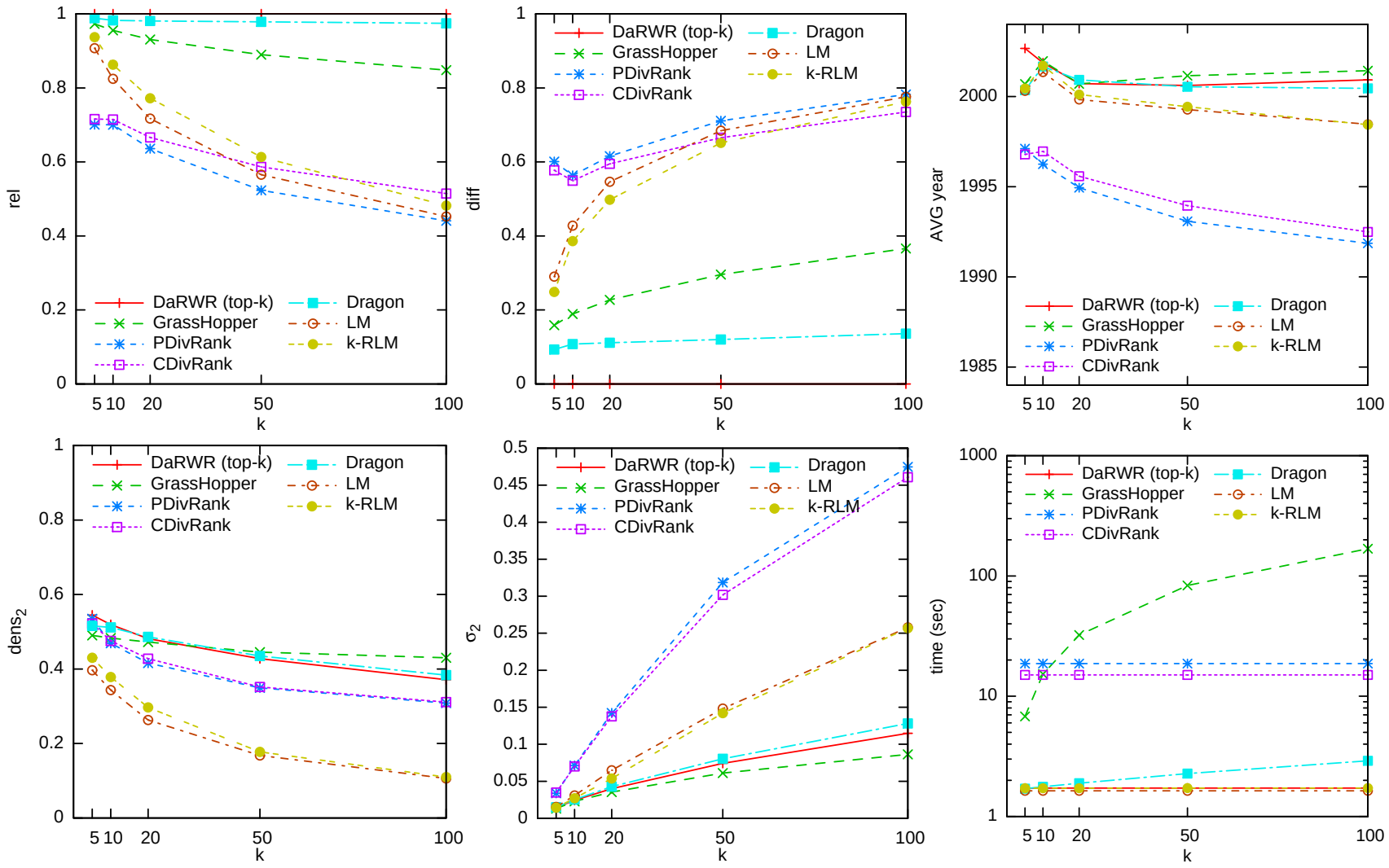


Relevant Diverse

Metrics and Algorithms [CSTA13]

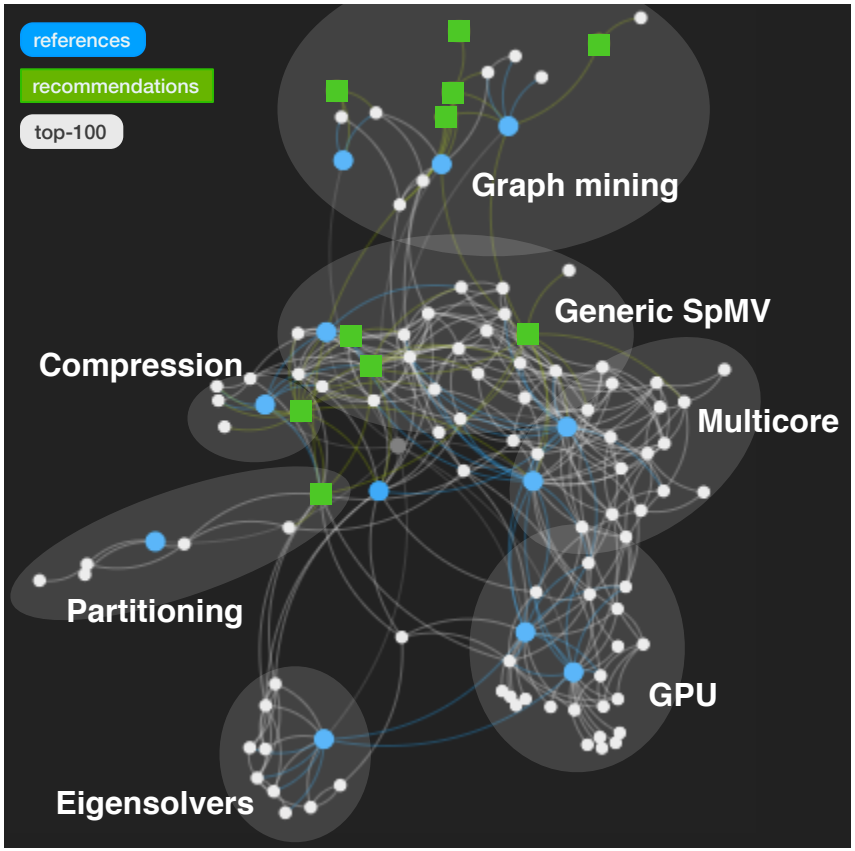


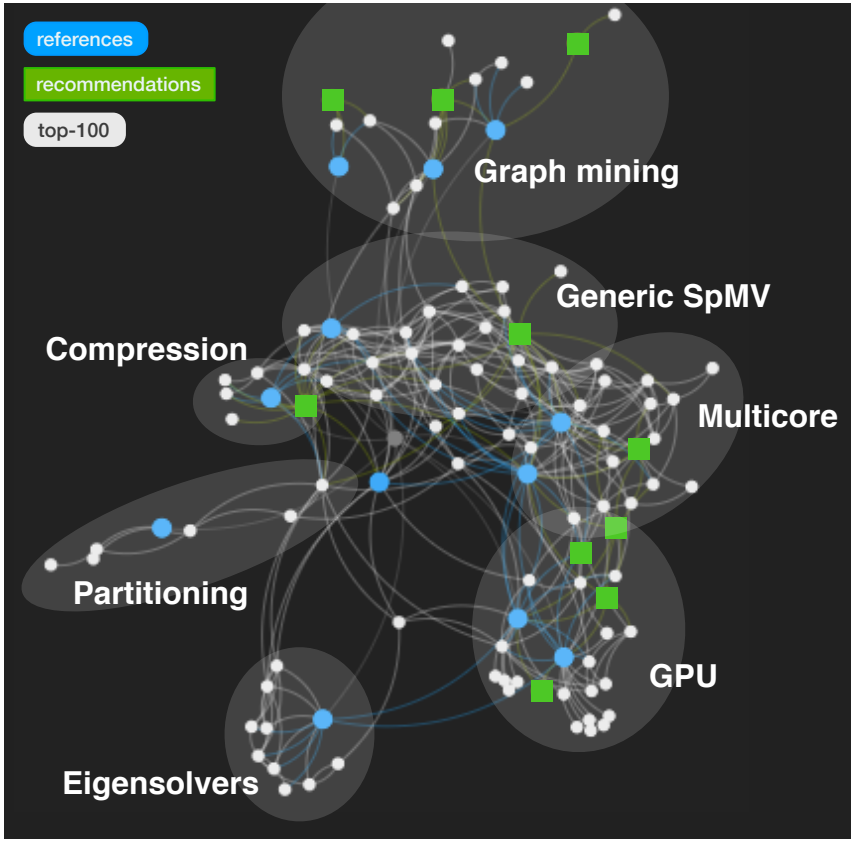
Metrics and Algorithms [CSTA13]



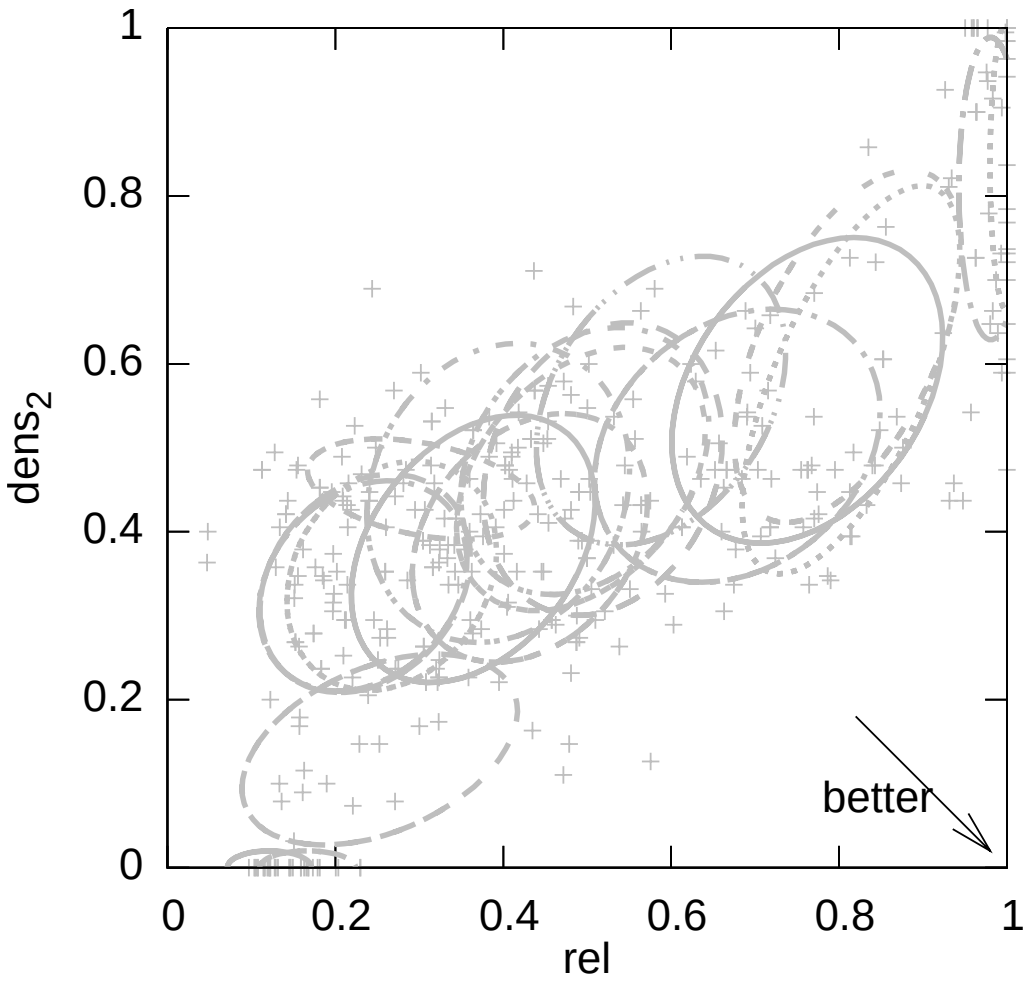
k-RLM is good.

Results


references
recommendations
top-100
Graph mining
Generic SpMV
Compression
Multicore
Partitioning
GPU
Eigensolvers

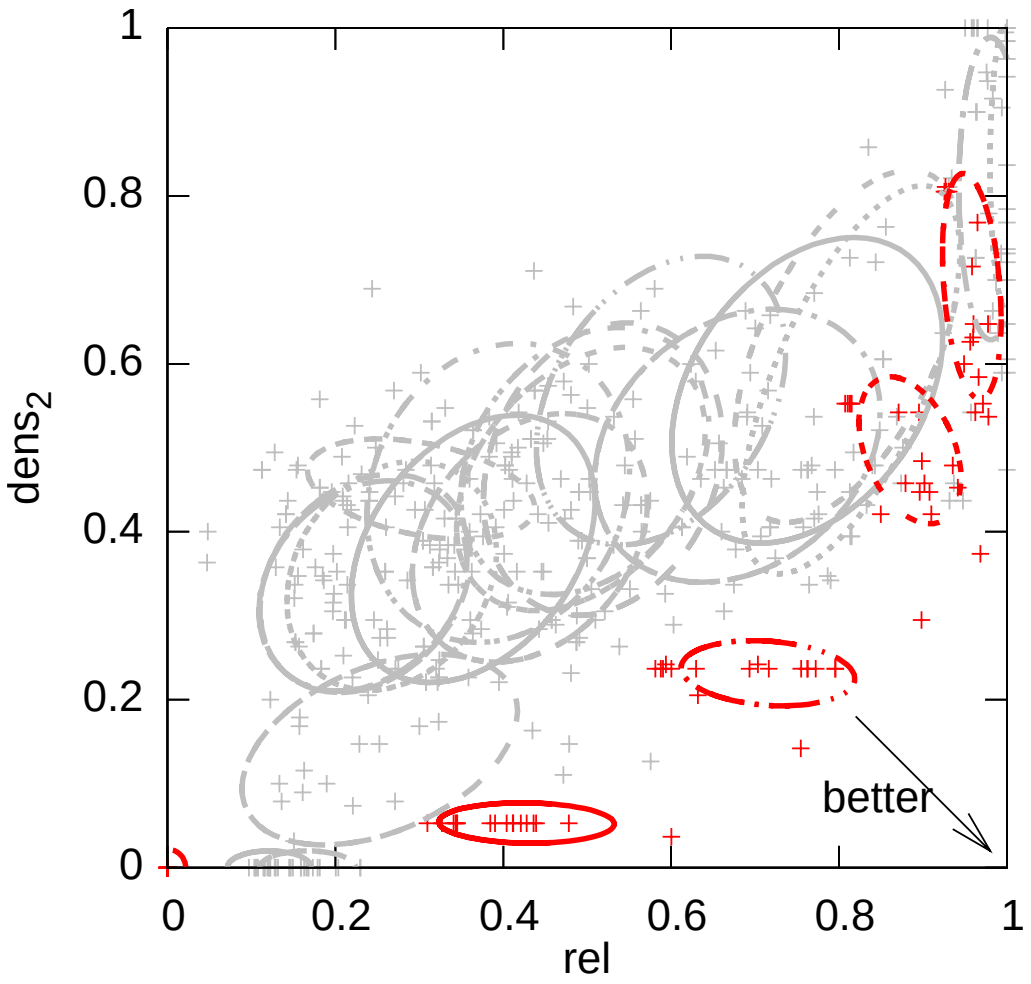

references
recommendations
top-100
Graph mining
Generic SpMV
Compression
Multicore
Partitioning
GPU
Eigensolvers

A Modelization problem [WWW13]



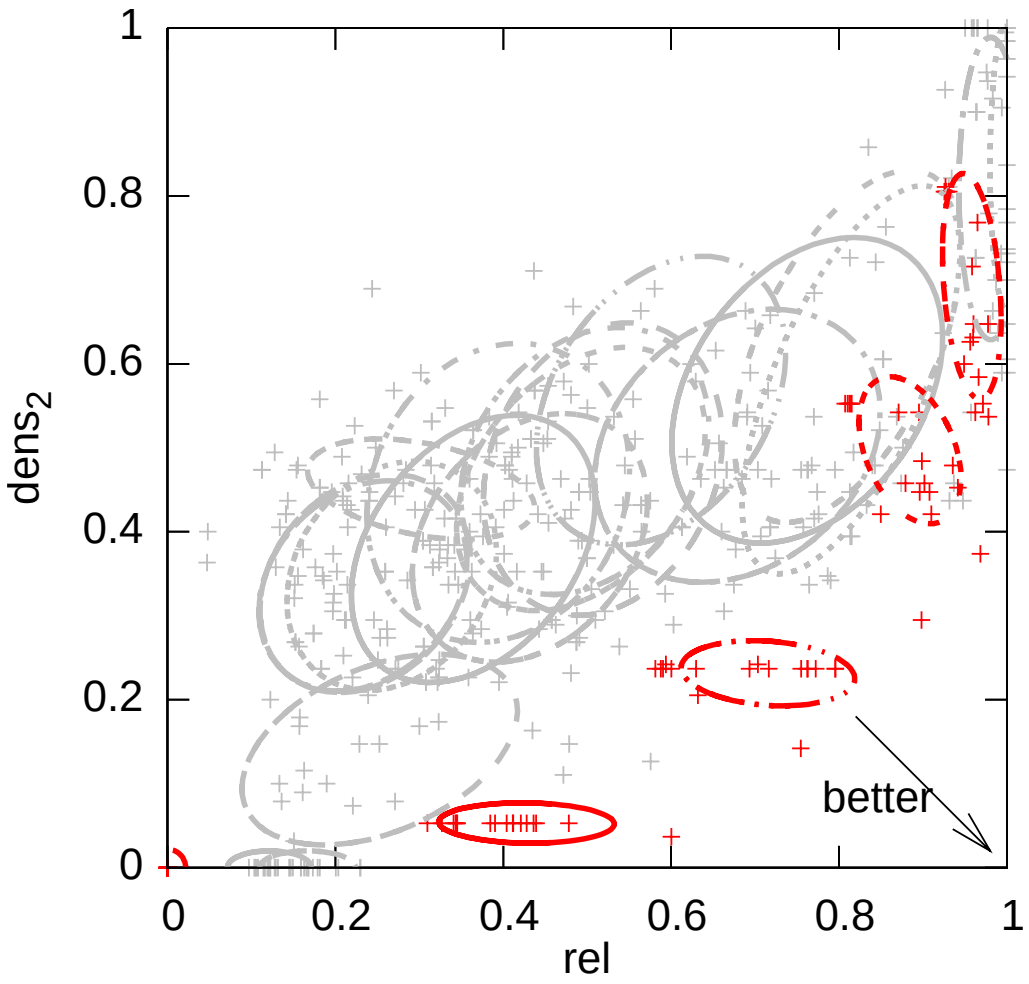
Here is a distribution of known algorithms.

A Modelization problem [WWW13]



Here is a distribution of known algorithms.
Would such an algorithm be of interest?

A Modelization problem [WWW13]



Here is a distribution of known algorithms.
Would such an algorithm be of interest?
That algorithm is query-oblivious!

Expanded Relevance

Sum the relevance of all documents at distance ℓ of a recommendation.

Outline

Centralities - Concept

Answer questions such as

- Who controls the flow in a network?
- Who is more important?
- Who has more influence?
- Whose contribution is significant for connections?

Applications

- Covert network (e.g., terrorist identification)
- Contingency analysis (e.g., weakness/robustness of networks)
- Viral marketing (e.g., who will spread the word best)
- Traffic analysis
- Store locations

Centralities - Definition

Let $G = (V, E)$ be a graph with the vertex set V and edge set E.

- **closeness centrality**: $cc[v] = \frac{1}{far[v]}$, where the farness is defined as $far[v] = \sum_{u \in comp(v)} d(u, v)$. $d(u, v)$ is the shortest path length between u and v.
- **betweenness centrality**: $bc(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the number shortest paths between s and t, and $\sigma_{st}(v)$ is the number of them passing through v.

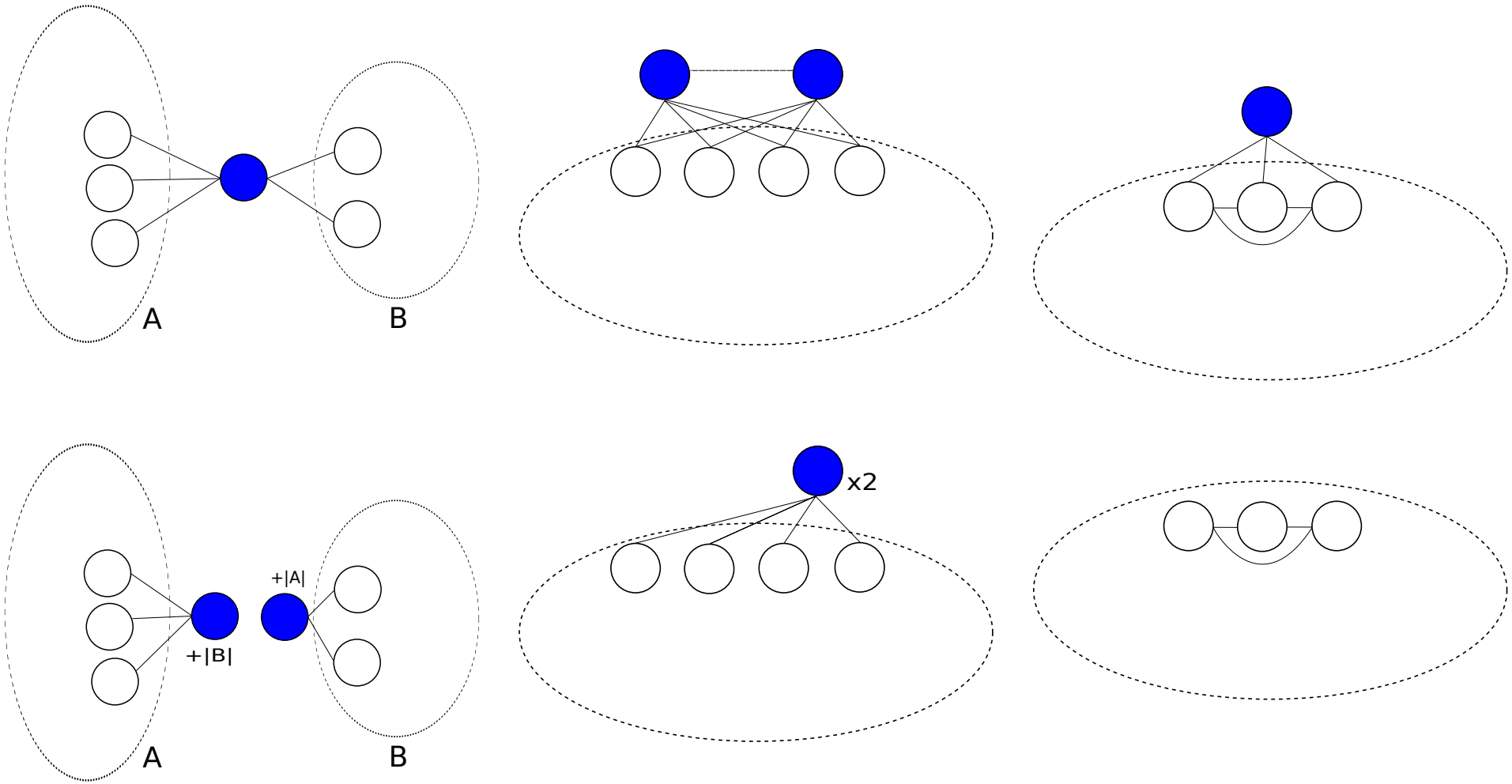
Centralities - Definition

Let $G = (V, E)$ be a graph with the vertex set V and edge set E.

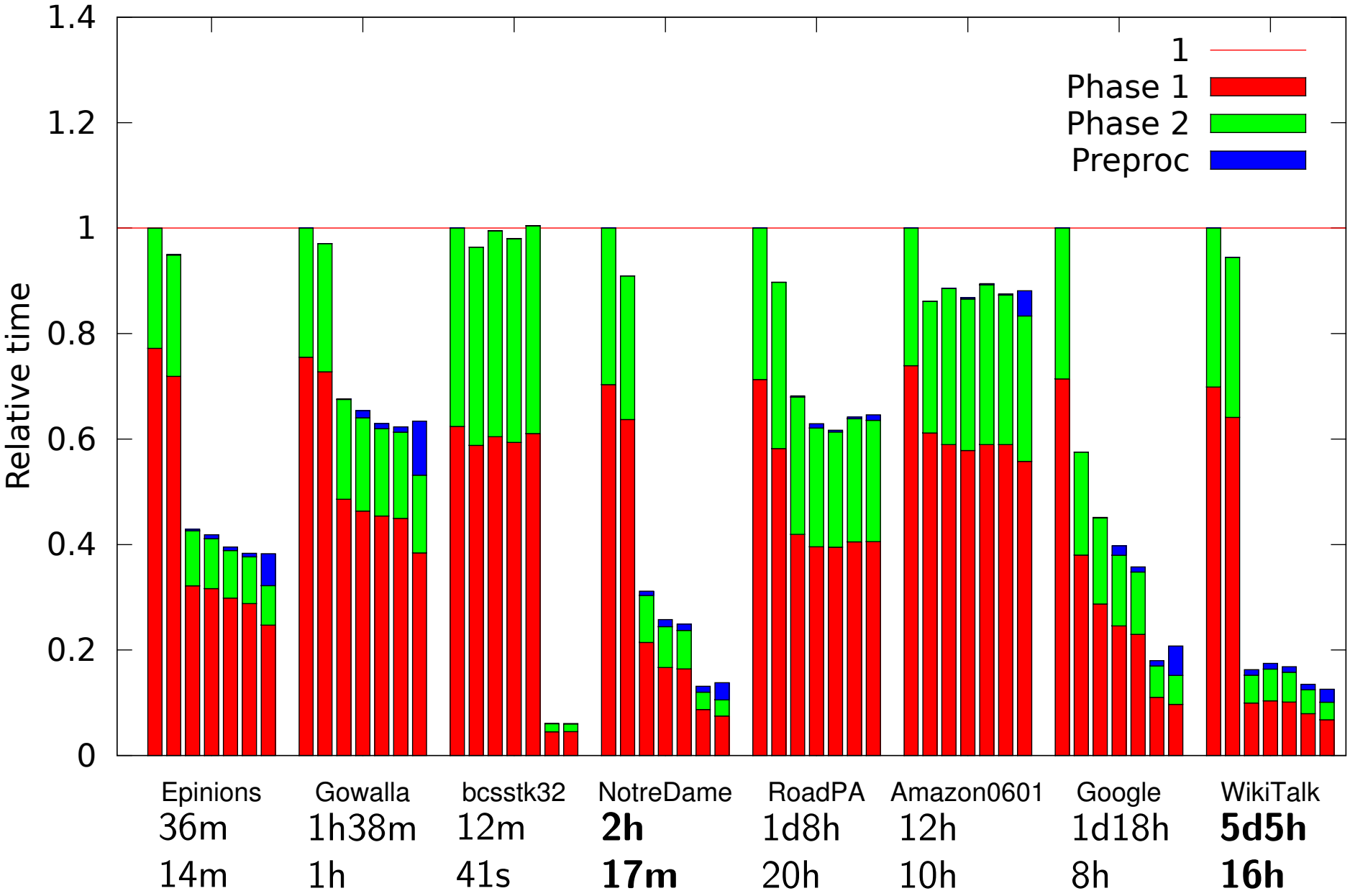
- **closeness centrality**: $cc[v] = \frac{1}{far[v]}$, where the farness is defined as $far[v] = \sum_{u \in comp(v)} d(u, v)$. $d(u, v)$ is the shortest path length between u and v.
- **betweenness centrality**: $bc(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the number shortest paths between s and t, and $\sigma_{st}(v)$ is the number of them passing through v.

Both metrics care about the structure of the shortest path graph.
Brandes algorithm computes the shortest path graph rooted in each vertex of the graph. $O(|E|)$ per source. $O(|V||E|)$ in total.
Believed to be asymptotically optimal [Kintali08].

Compression and Shattering

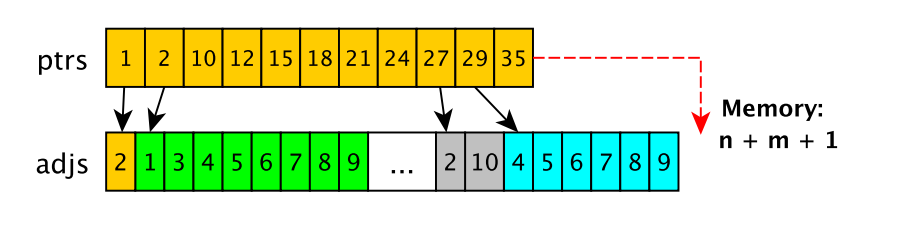


BADIOS [SDM2013]



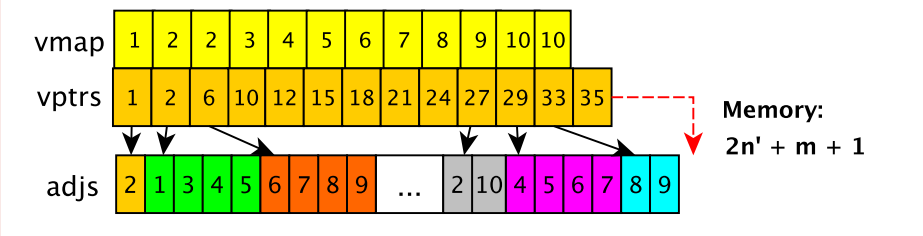
Matrix Representations for GPUs

CRS [Shi11]



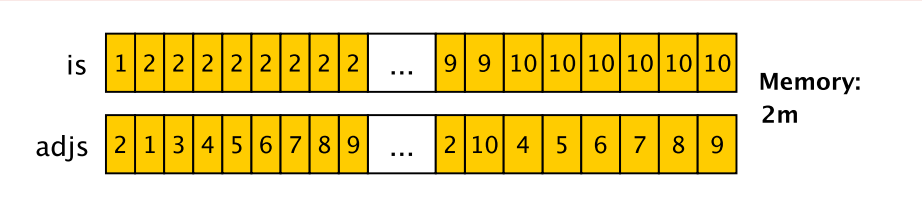
1 thread per vertex: bad load balance

Virtual-vertex



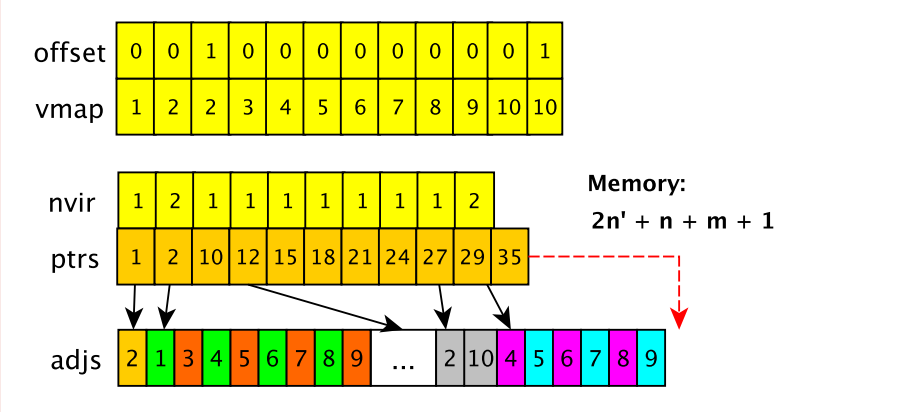
Balances load and limits atomics

COO [Jia11]



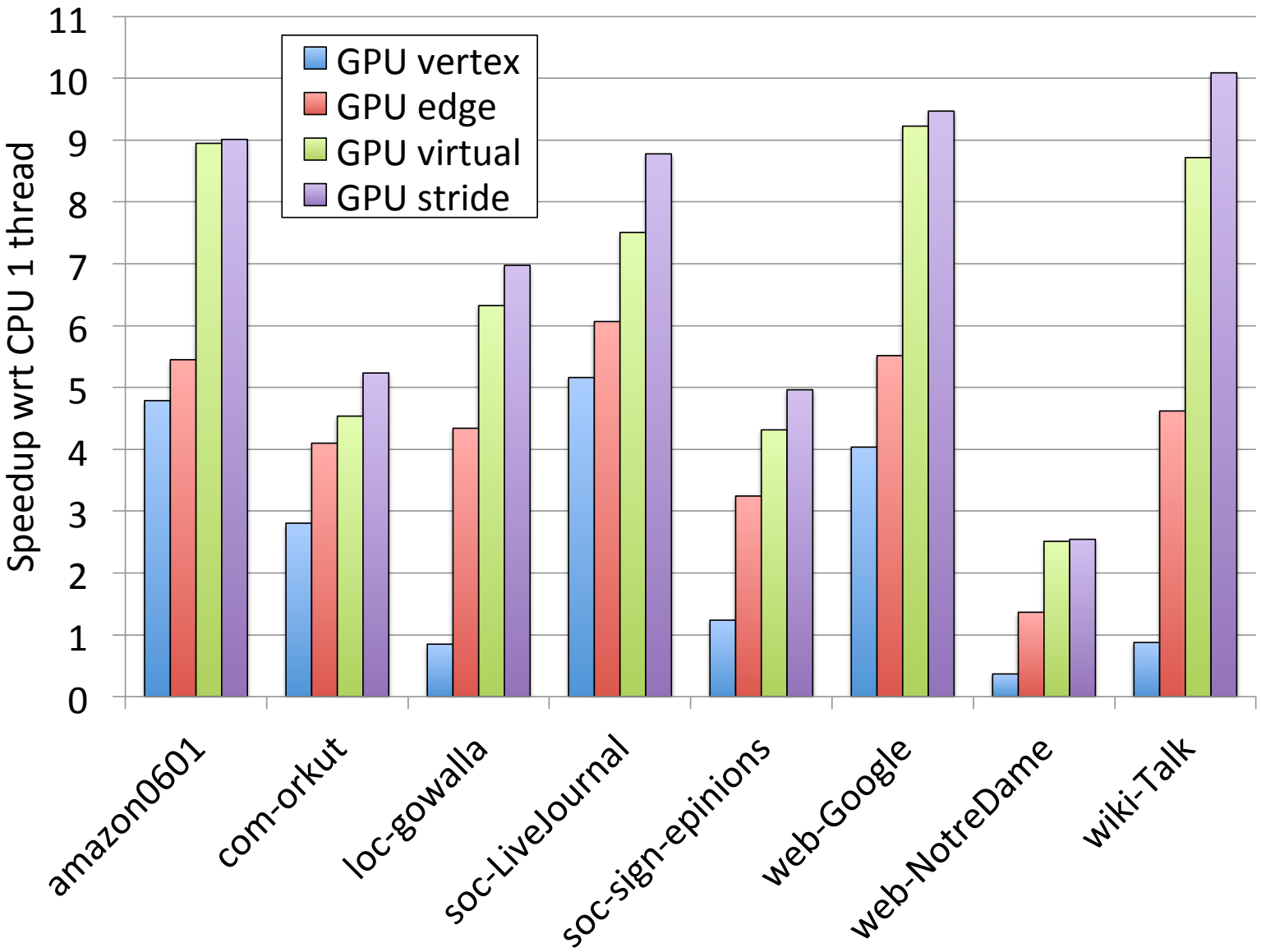
1 thread per edge: too many atomics

Stride



Enables coalesced memory accesses

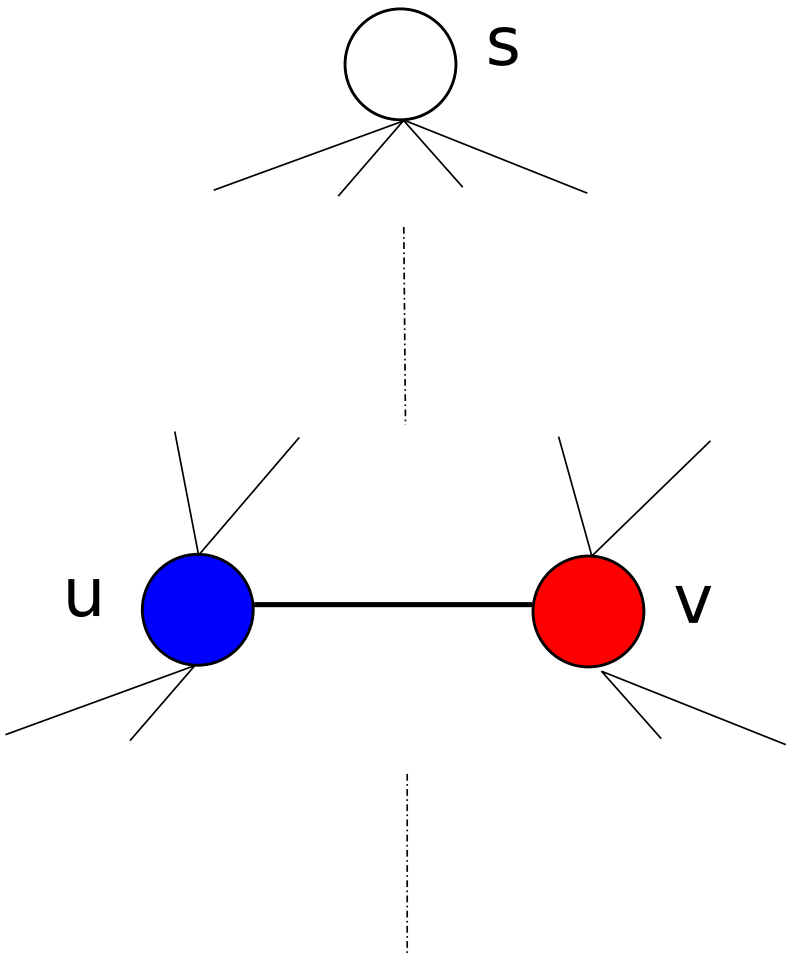
NVIDIA C2050 performance [GPGPU2013]



Edge Insertion for closeness centrality : three cases

If $d(u,s) = d(v,s)$

The shortest path graph does not differ. So the farness of s is correct.



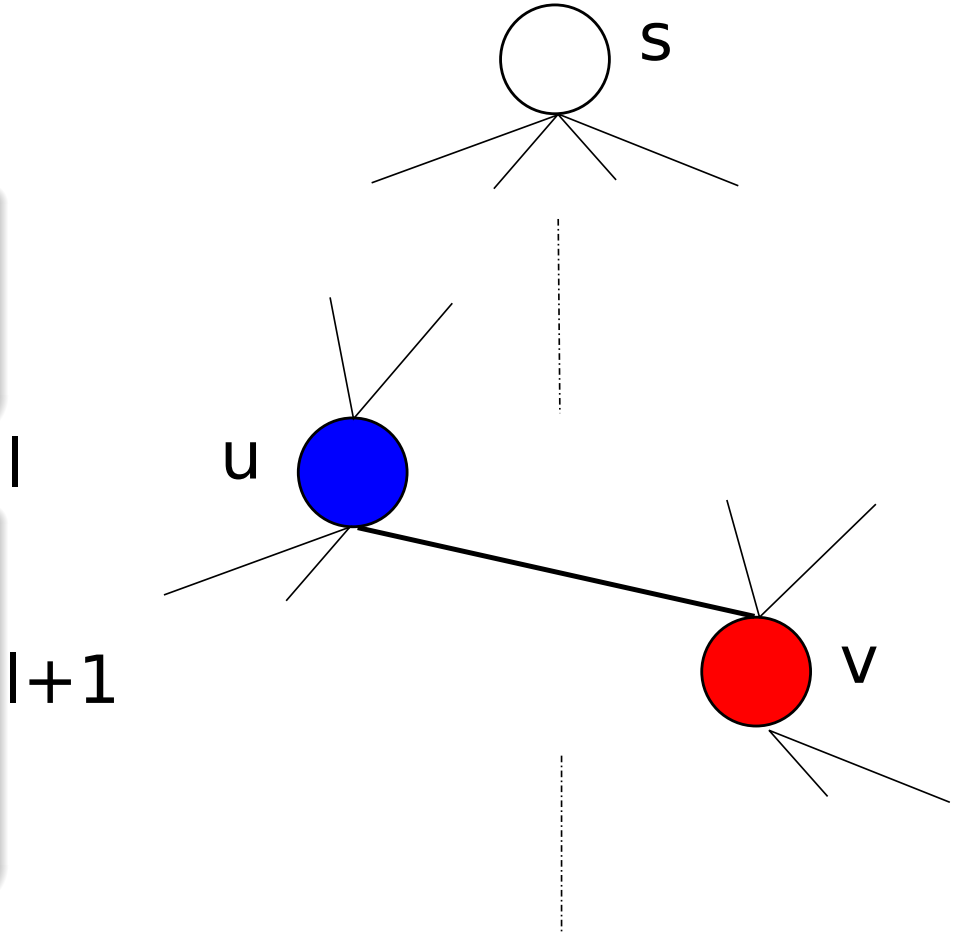
Edge Insertion for closeness centrality : three cases

If $d(u,s) = d(v,s)$

The shortest path graph does not differ. So the farness of s is correct.

If $d(u,s) + 1 = d(v,s)$

The shortest path graph differs by exactly one edge. The levels stay the same. So the farness of s is still correct.



Edge Insertion for closeness centrality : three cases

If $d(u,s) = d(v,s)$

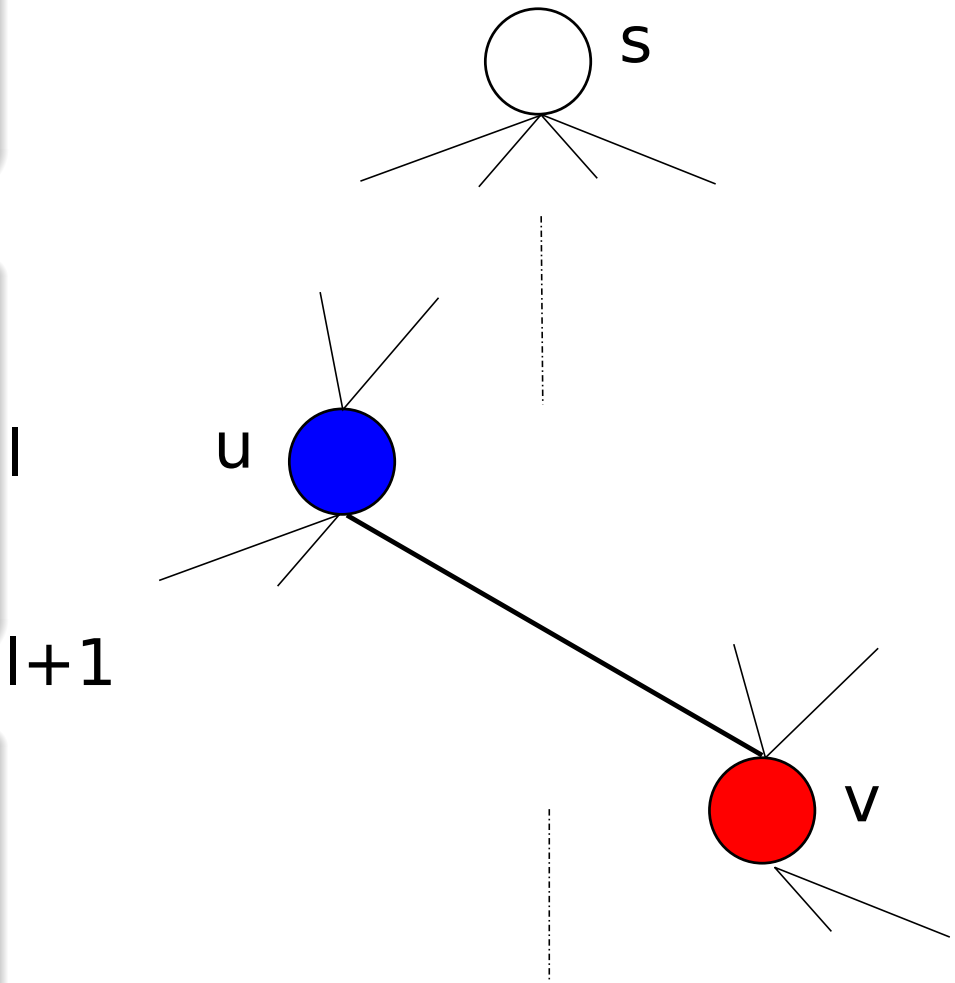
The shortest path graph does not differ. So the farness of s is correct.

If $d(u,s) + 1 = d(v,s)$

The shortest path graph differs by exactly one edge. The levels stay the same. So the farness of s is still correct.

If $d(u,s) + 1 < d(v,s)$

The shortest path graph differs by at least one edge. The level of v changes (and potentially more). So the farness of s is incorrect.



Edge Insertion for closeness centrality : three cases

If $d(u,s) = d(v,s)$

The shortest path graph does not differ. So the farness of s is correct.

If $d(u,s) + 1 = d(v,s)$

The shortest path graph differs by exactly one edge. The levels stay the same. So the farness of s is still correct.

If $d(u,s) + 1 < d(v,s)$

The shortest path graph differs by at least one edge. The level of v changes (and potentially more). So the farness of s is incorrect.

Algorithm

- Upon insertion of (u,v)
- Compute BFS from u and v (before edge insertion)
- For all $s \neq u, v$, if $|d(u,s) - d(v,s)| > 1$, flag s
- Add (u,v) to the graph
- Compute $cc[s]$ for all flagged s

Results : Speedup

Graph	CC-B	CC-BL
soc-sign-epinions	3.0	37.8
loc-gowalla_edges	1.8	17.1
bcsstk32	1.0	5,493.0
web-NotreDame	4.9	23.9
roadNet-PA	1.6	3.0
amazon0601	1.2	27.6
web-Google	3.0	26.6
wiki-Talk	6.8	69.8
Geometric mean	2.39	43.58

Outline

Data Flow Middleware

Applications

- Image processing
- Video surveillance
- MRI Analysis
- Satellite data processing

Complex Hardware

- Accelerators (GPU, Xeon Phi)
- Clusters
- Grid
- Heterogeneous systems

Versatile [CSUR13]

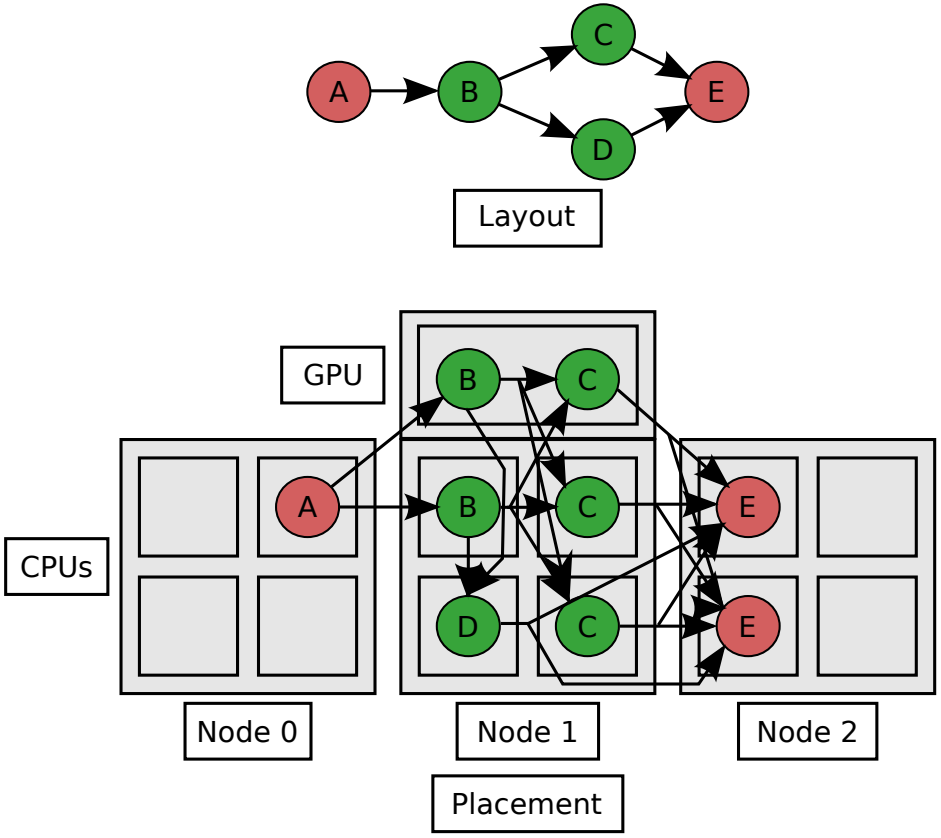
Supports:

- Pipeline parallelism
- Task parallelism
- Replicated parallelism
- Data parallelism

Optimizes:

- Throughput
- Latency
- Energy
- Reliability

DataCutter [Beynon01]



Filter Stream

Programming model that specify the *layout* of an application: a set filters that transform the data streamed through them.

Placement

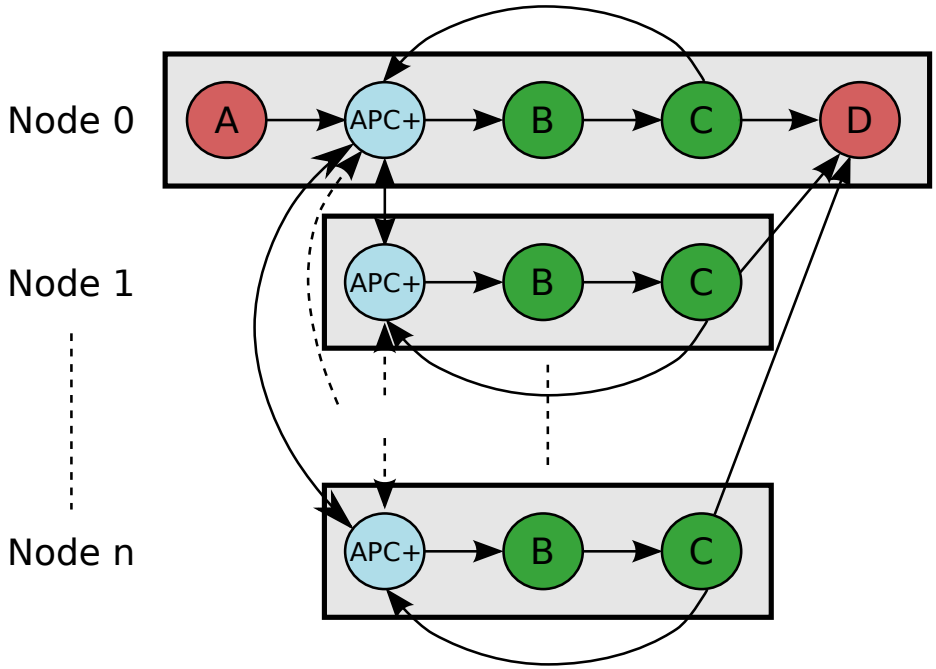
A given layout can be executed in different ways by the programming framework.
Potentially filters can be *replicated*.

APC+

How much to cut the work?

- Small chunks, pay overhead
- Big chunks, pay imbalance

- Adapt to the network
- Adapt to the computing units



Performance Model

Predicts computation time for all size and processing unit.

Workload Partitioner[HiPC10]

Predicts the end of the computation to balance the load.

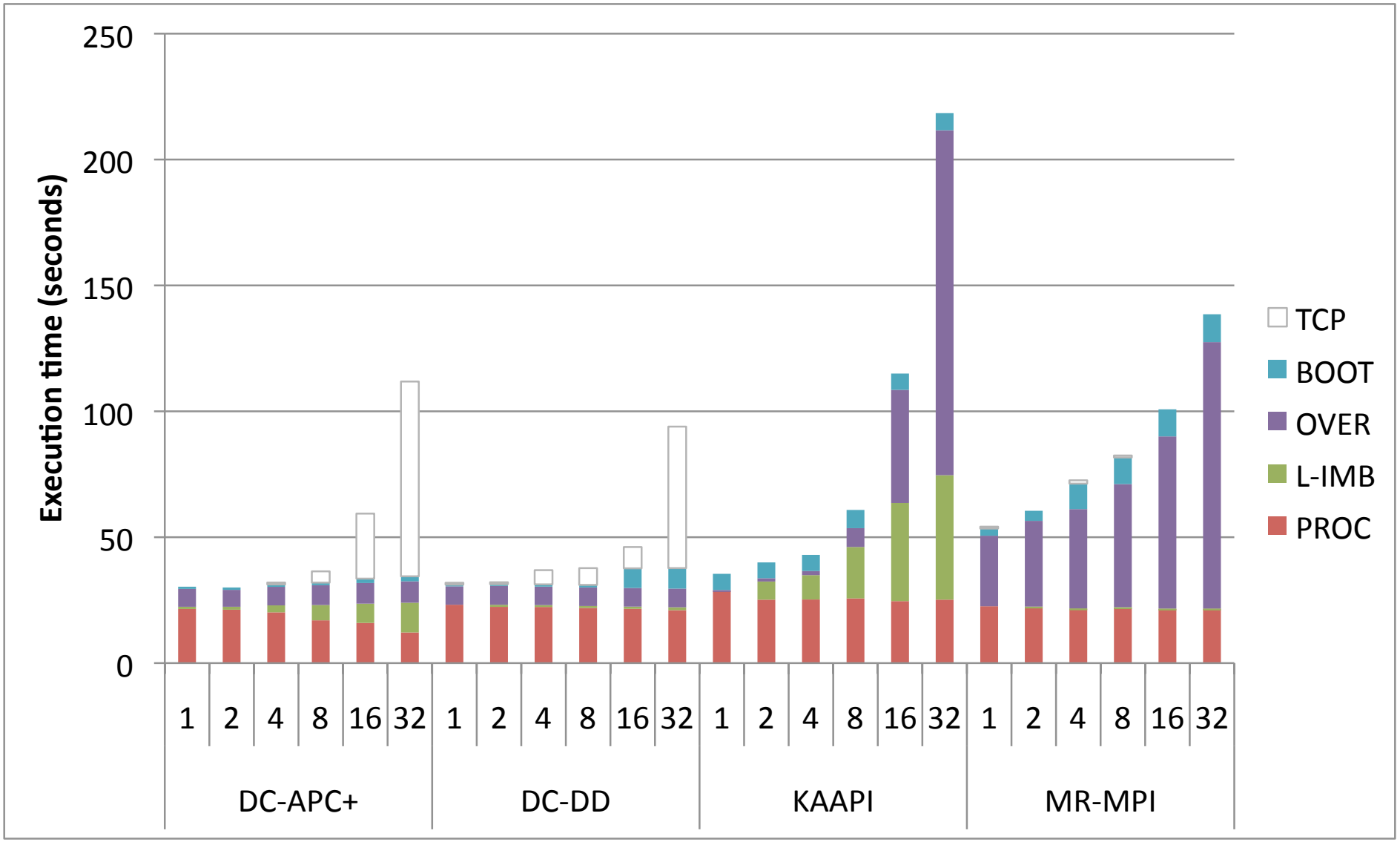
Distributed Work Stealing

Balances the load accros multiple node.

Storage Layer

Schedule data transfers to optimize network performance.

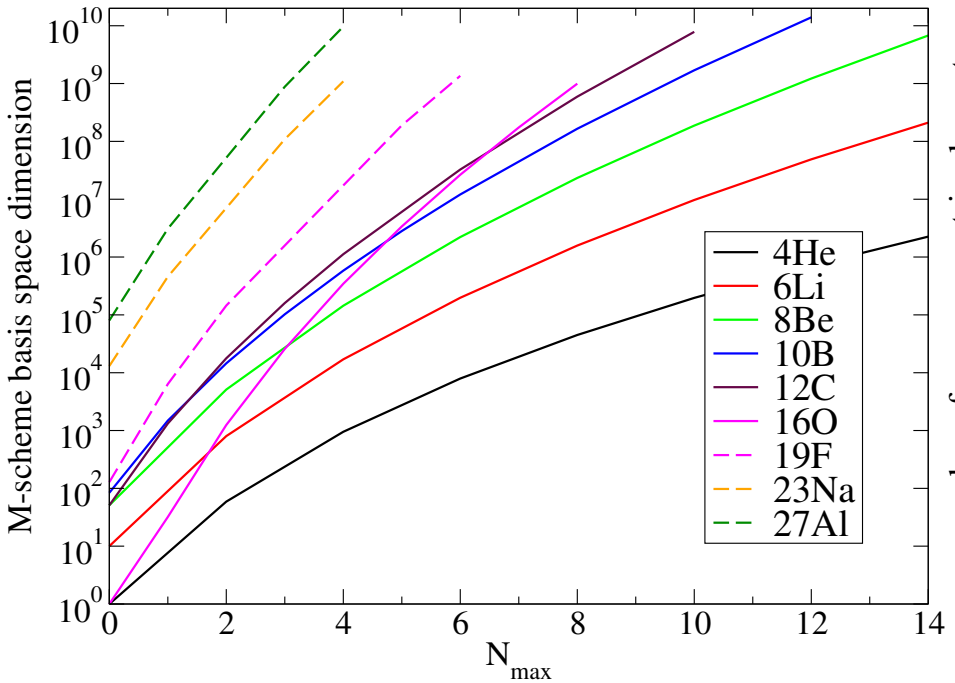
SAR imaging - Weak Scaling - CPU/GPU [Parco12]

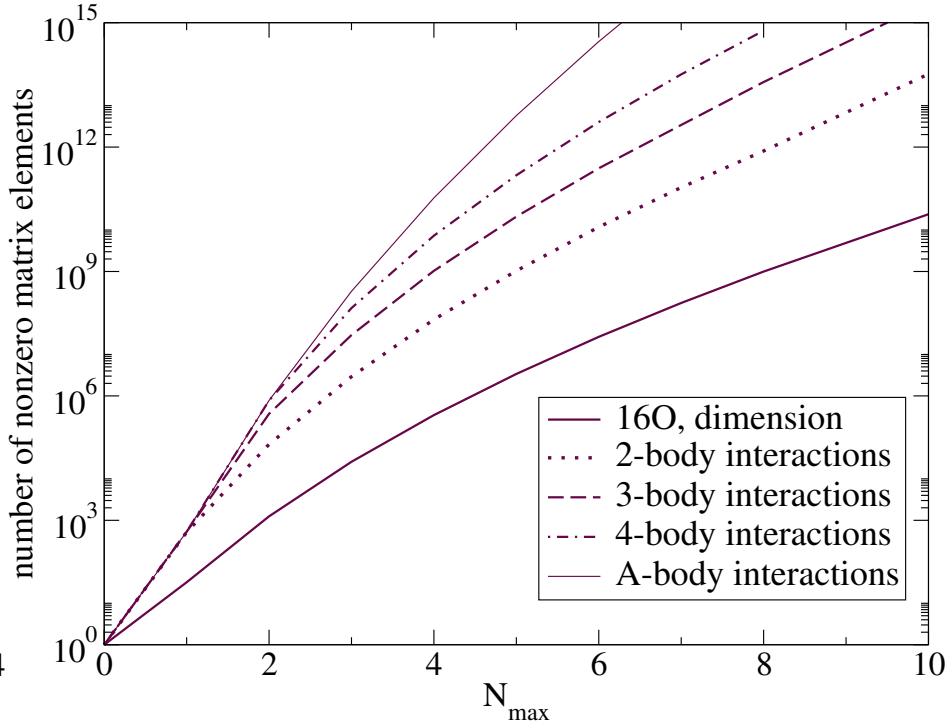


A Peta Scale nuclear physics problem

Extract the lowest eigenpairs of a large Hamiltonian matrix, whose size grows with the number of particles and truncation parameter in the atom.

For Boron 10, with Nmax=8 with 2 body interactions (Toy case): 160 millions of rows, 124 billions of non zero elements.

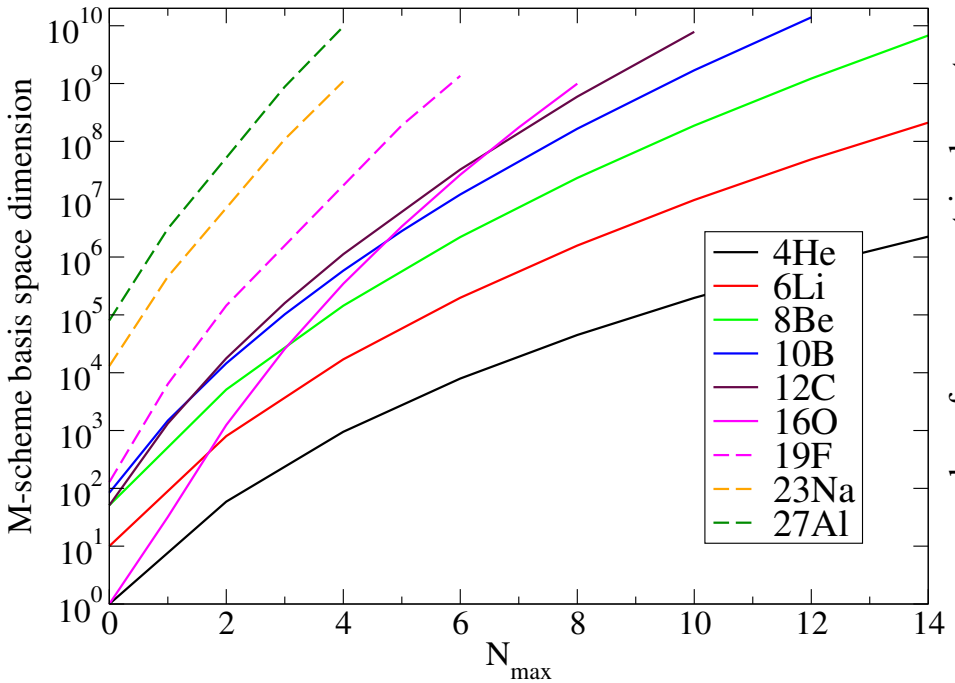


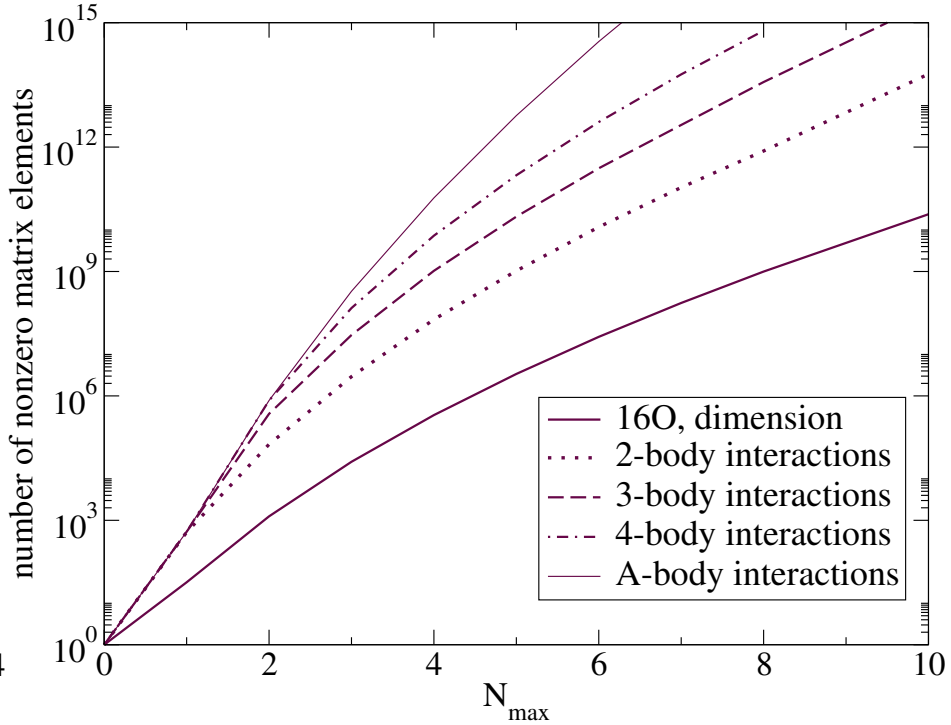


A Peta Scale nuclear physics problem

Extract the lowest eigenpairs of a large Hamiltonian matrix, whose size grows with the number of particles and truncation parameter in the atom.

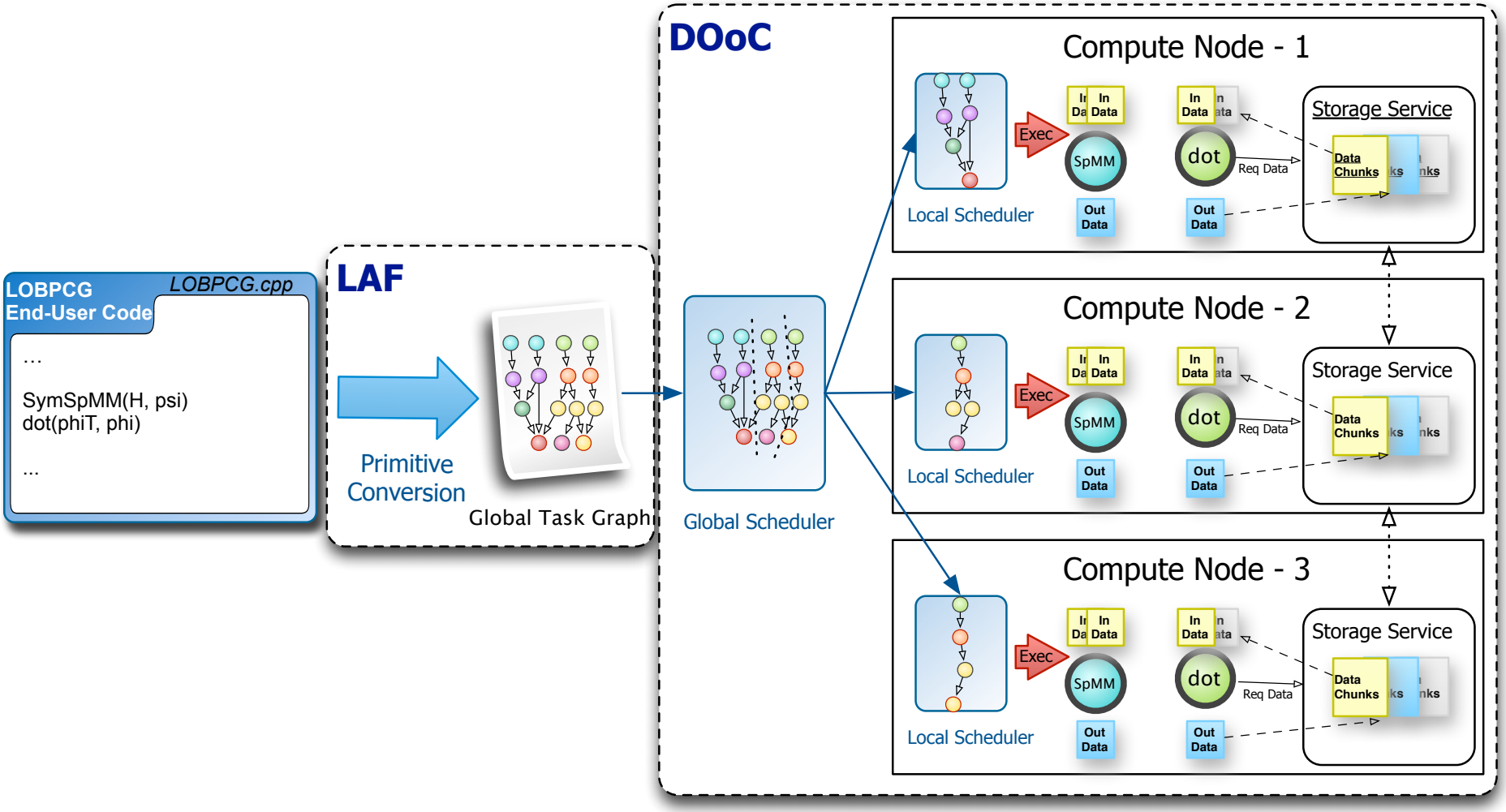
For Boron 10, with Nmax=8 with 2 body interactions (Toy case): 160 millions of rows, 124 billions of non zero elements.





Two options: use a really large machine or use Out-of-Core (SSD).

DOoC+LAF


LOBPCG
End-User Code
LOBPCG.cpp
...
SymSpMM(H, psi)
dot(phiT, phi)
...
LAF
Primitive
Conversion
Global Task Graph
DOoC
Global Scheduler
Compute Node - 1
Compute Node - 2
Compute Node - 3
Local Scheduler
Exec
In Data
SpMM
dot
Req Data
Out Data
Storage Service
Data Chunks

Linear Algebra Frontend

Provides data types and operations for Linear Algebra.

Lanczos

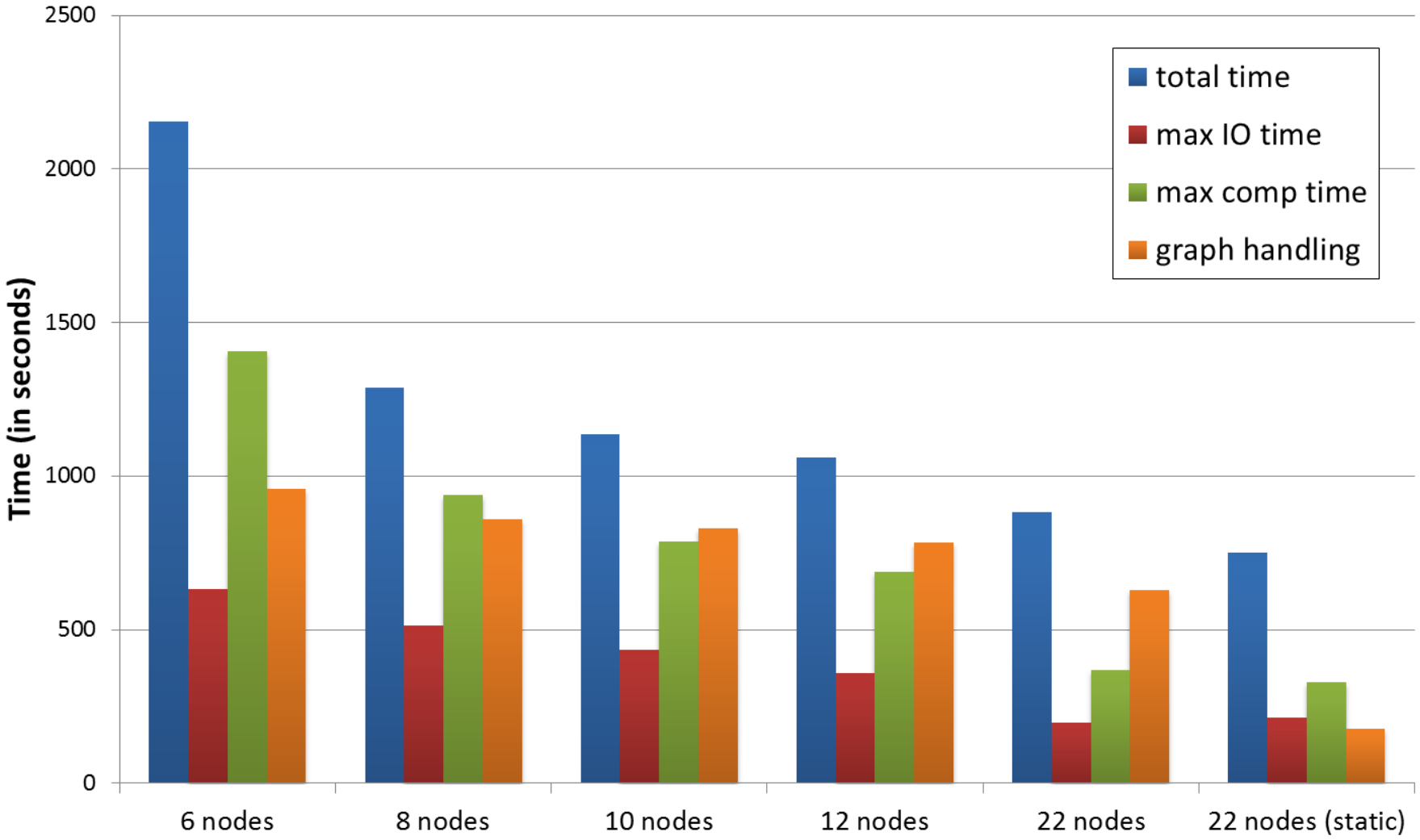
```
Lanczos (v_in, M, a_in, b_in,
         v_out, a_out, b_out) {
 Vector w(out.meta());
 Vector wprime(out.meta());
 Vector wsecond(out.meta());

 symSpMV (w, M, v_in);
 axpyV (wprime, w, v_in, 1, -b_in);
 dot (a_out, wprime, v_in);
 axpyV (wsecond, wprime, v_in, 1, -a_out);
 dot (b_out, wsecond, wsecond);
 vector_scale(v_out, wsecond, 1/b_out);
}
```

Supported Operations

Primitives	Operation
Primitives that creates Matrix	
MM, (Sym)SpMM	$C = AB$
addM	$C = A + B$
axpyM	$C = aA + b$
randomM	$C = random()$
Primitives that creates Vector	
MV, (Sym)SpMV	$y = Ax$
addV	$y = x + w$
axpyV	$y = ax + b$
Primitives that creates scalar	
dot	$a =< x, y >$

5 Lanczos iterations at NERSC [Cluster12]



Outline

Other things I do

Scheduling, Mapping, Partitioning

Areas:

- Application scheduling
- Cluster scheduling
- Pipelined scheduling
- Spatial workload partitioning

Multi objective:

- Makespan
- Throughput
- Fairness
- Latency
- Reliability

Techniques:

- Optimal algorithms
- Approximation algorithms
- Heuristics

Other things I do

Scheduling, Mapping, Partitioning

Areas:

- Application scheduling
- Cluster scheduling
- Pipelined scheduling
- Spatial workload partitioning

Multi objective:

- Makespan
- Throughput
- Fairness
- Latency
- Reliability

Techniques:

- Optimal algorithms
- Approximation algorithms
- Heuristics

Parallel Graph Algorithms

Scalable distributed memory local search for graph coloring. Communication reductions and compression. Hybrid MPI/OpenMP.

Other things I do

Scheduling, Mapping, Partitioning

Areas:

- Application scheduling
- Cluster scheduling
- Pipelined scheduling
- Spatial workload partitioning

Multi objective:

- Makespan
- Throughput
- Fairness
- Latency
- Reliability

Techniques:

- Optimal algorithms
- Approximation algorithms
- Heuristics

Parallel Graph Algorithms

Scalable distributed memory local search for graph coloring. Communication reductions and compression. Hybrid MPI/OpenMP.

Cutting Edge Architecture

Investigated graph algorithms and sparse linear algebra operations on pre-release Intel Xeon Phi.

Conclusions - My Philosophy

Applications

- Analyze data sources
- What are we trying to do?
- What is important?

Conclusions - My Philosophy

Applications

- Analyze data sources
- What are we trying to do?
- What is important?

Algorithms

- Design
- Re-engineer
- Approximate
- Incremental

Conclusions - My Philosophy

Applications

- Analyze data sources
- What are we trying to do?
- What is important?

Algorithms

- Design
- Re-engineer
- Approximate
- Incremental

Middleware

Makes the software:

- Easier to write
- Reusable
- Efficient

Conclusions - My Philosophy

Applications

- Analyze data sources
- What are we trying to do?
- What is important?

Algorithms

- Design
- Re-engineer
- Approximate
- Incremental

Middleware

Makes the software:

- Easier to write
- Reusable
- Efficient

Hardware

- What is suitable?
- How to use it?
- How to improve it?

Conclusions - My Philosophy

Applications

- Analyze data sources
- What are we trying to do?
- What is important?

Middleware

Makes the software:

- Easier to write
- Reusable
- Efficient

Algorithms

- Design
- Re-engineer
- Approximate
- Incremental

Hardware

- What is suitable?
- How to use it?
- How to improve it?

Which is important? All of it!

What's Next?

Applications

- **Multi-graph**
- Author Venue Paper
- Personal analytics
- Cross social network application

Middleware

- High Level Query
- Cluster with Accelerator Graph Middleware
- **The MATLAB of graphs**

Algorithms

- Streaming
- Community detection
- **Temporal analysis**

Hardware

- Cluster with Computational Accelerator (GPU, Xeon Phi)
- Cluster with Storage Accelerator (SSD)
- Both! (Beacon project)

Active Coauthors

The Ohio State University:

- Ümit V. Çatalyürek
- Kamer Kaya
- Onur Küçüktunç
- Ahmet Erdem Saryıüce

Grenoble University, France:

- Denis Trystram
- Grégory Mounié
- Pierre-François Dutot
- Jean-François Mehaut
- Guillaume Huard

INRIA, France:

- Yves Robert
- Anne Benoit
- Emmanuel Jeannot
- Alain Girault

Lawrence Berkely National Lab:

- Esmond G. Ng
- Chao Yang
- Hasan Metin Aktulga

University of Tennessee:

- Jack Dongarra

University of Luxembourg:

- Johnatan Pecero-Sanchez

Vanderbilt University:

- Zhiao Shi

Iowa State University:

- James Vary
- Pieter Maris

Thank you

More information

contact : esaule@bmi.osu.edu

visit: `http://bmi.osu.edu/~esaule`

`http://bmi.osu.edu/hpc/`

Research at HPC lab is supported by