Architecture- and Workload-Aware Graph (Re)Partitioning

Thesis Defense

Angen Zheng

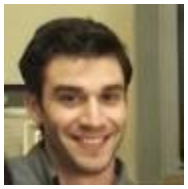
Thesis Committee

Alexandros Labrinidis
CS, Pitt

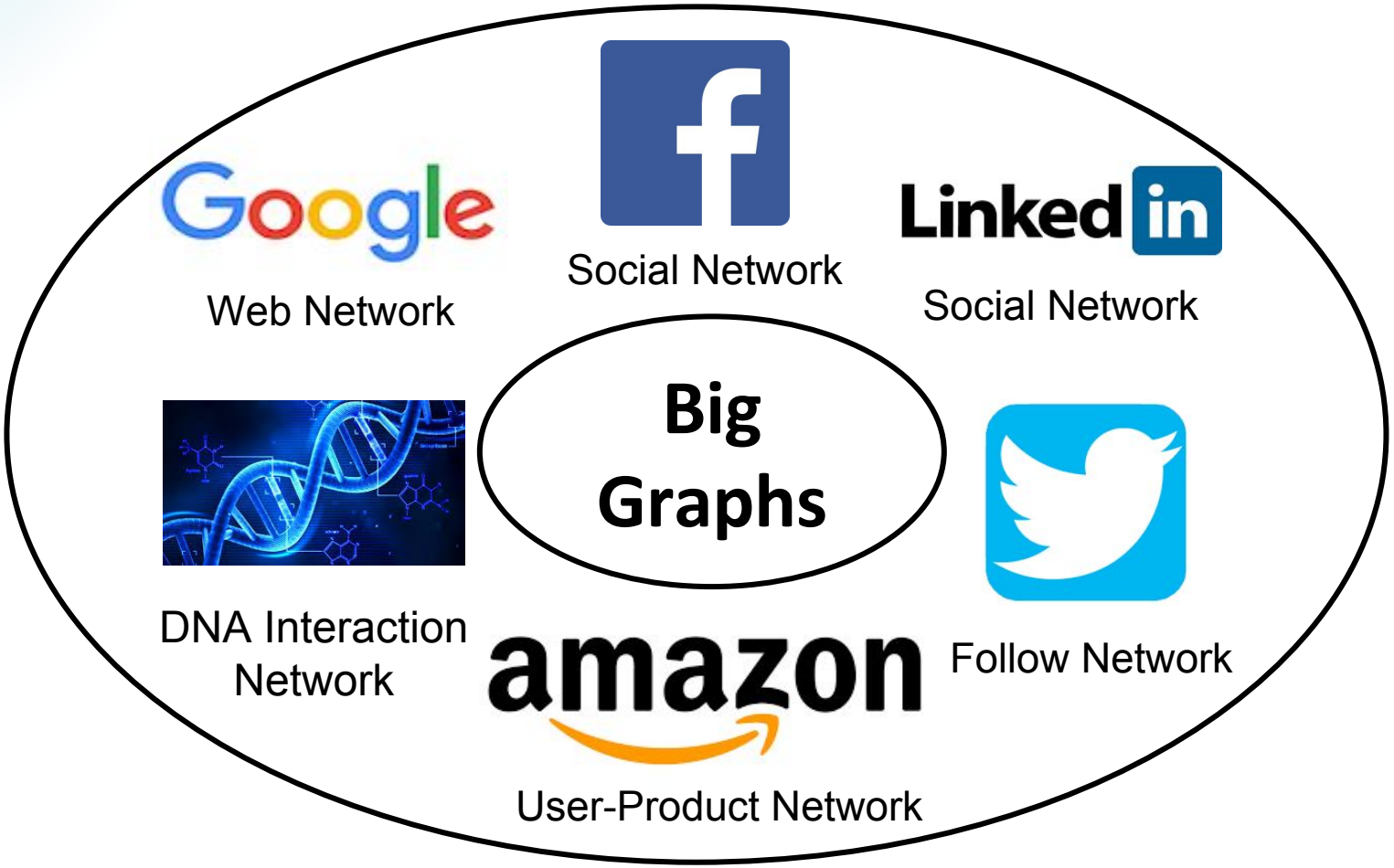
Panos K. Chrysanthis
CS, Pitt

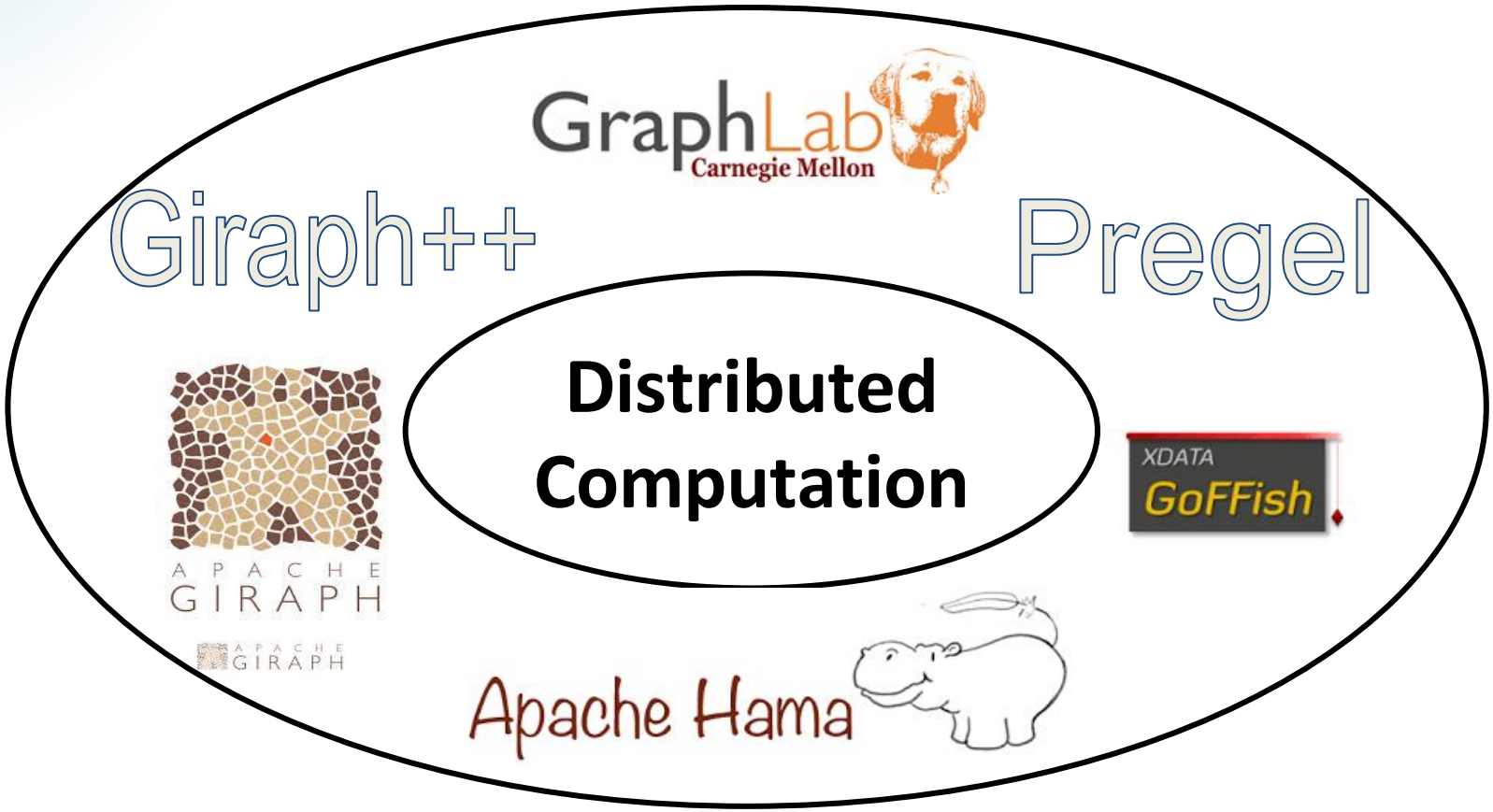
Jack Lange
CS, Pitt

Peyman Givi
SSOE, Pitt



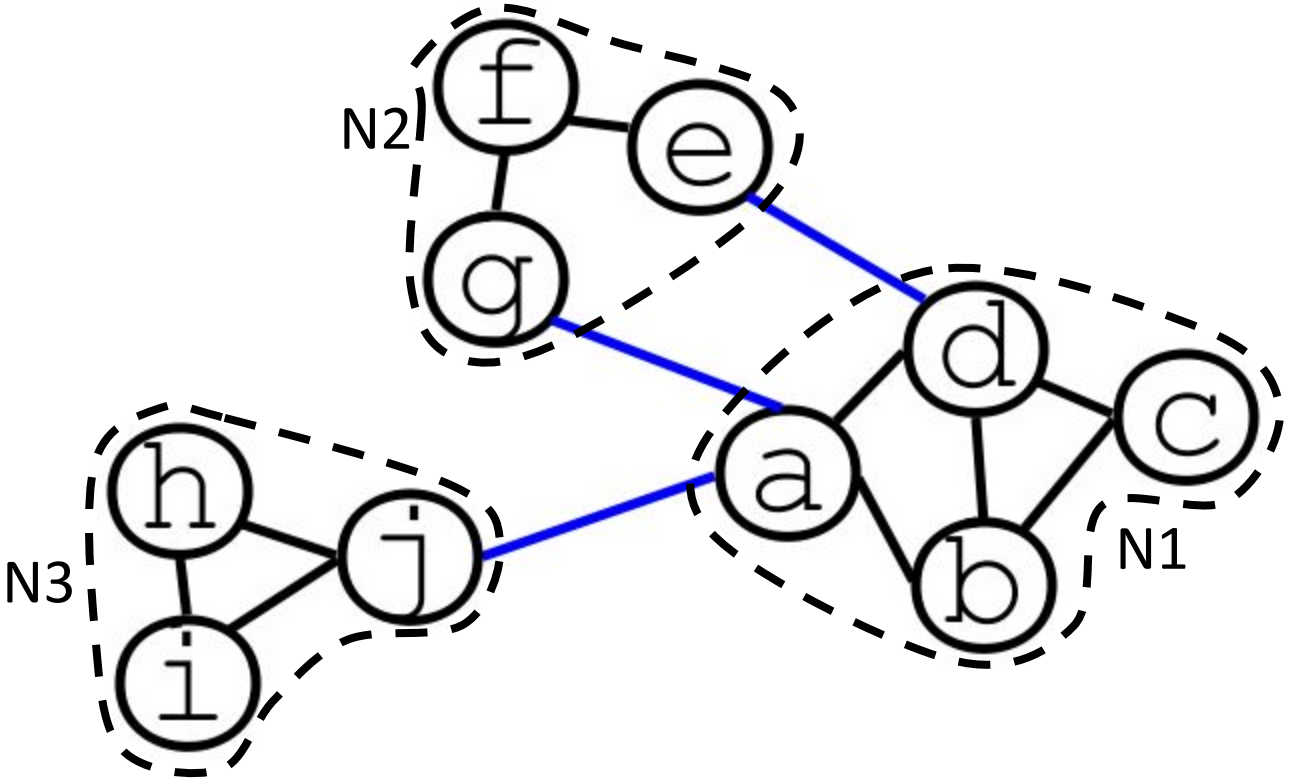
Patrick Pisciuneri
SSOE, Pitt


Google
Web Network
Social Network
LinkedIn
Social Network
Big Graphs
DNA Interaction Network
Follow Network
amazon
User-Product Network


GraphLab
Carnegie Mellon
Giraph++
Pregel
Distributed Computation
APACHE GIRAPH
XDATA
GoFFish
APACHE GIRAPH
Apache Hama

A Balanced Partitioning = Even Load Distribution

Minimal Edge-Cut = Minimal Data Comm



Thesis Statement

- ✓ Nonuniform network comm costs
- ✓ Contentiousness of the memory subsystems

- ✓ Nonuniform comp requirement
- ✓ Nonuniform comm requirement
- ✓ Time-varying skewness

Architecture- and **workload-aware** **graph partitioning** enables efficient distributed graph computation in modern HPC clusters.

Research Overview

Architecture- and Workload-Aware Graph (Re)Partitioning

Aragon [BigGraphs'14]
(small dynamic graphs)

Planar+ [To submit'17]
(large dynamic graphs)

Paragon [EDBT'16]
(median-size dynamic graphs)

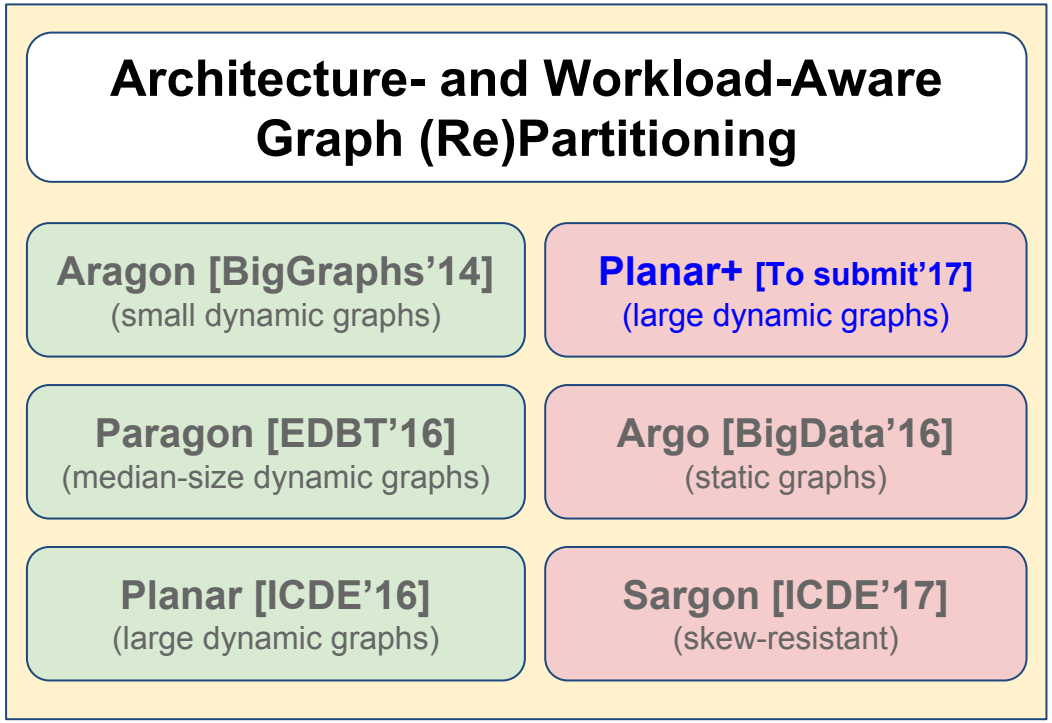
Argo [BigData'16]
(static graphs)

Planar [ICDE'16]
(large dynamic graphs)

Sargon [ICDE'17]
(skew-resistant)

Planar+: Parallel Lightweight Architecture-Aware Graph Repartitioning

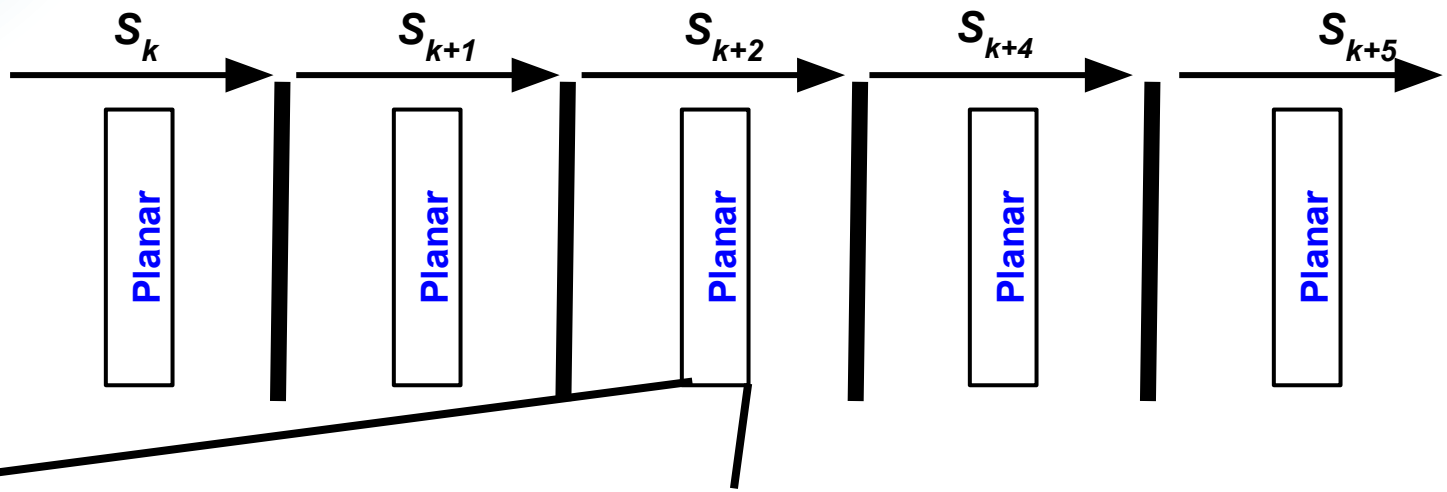
Angen Zheng · Patrick Pisciuneri · Alexandros Labrinidis · Panos K. Chrysanthis · Jack Lange · Peyman Givi



Planar+: Overview

- **Optimizations**
  - Memorization technique
  - Eliminating per step physical vertex migration
  - Optimized network comm cost measurement
  - Optimized vertex gain computation

Planar: Overview



Phase-1: Logical Vertex Migration

- Phase-1a: Minimizing Comm Cost
- Phase-1b: Ensuring Balanced Partitions

★ ***Migration Planning***
- *What vertices to move?*
- *Where to move?*

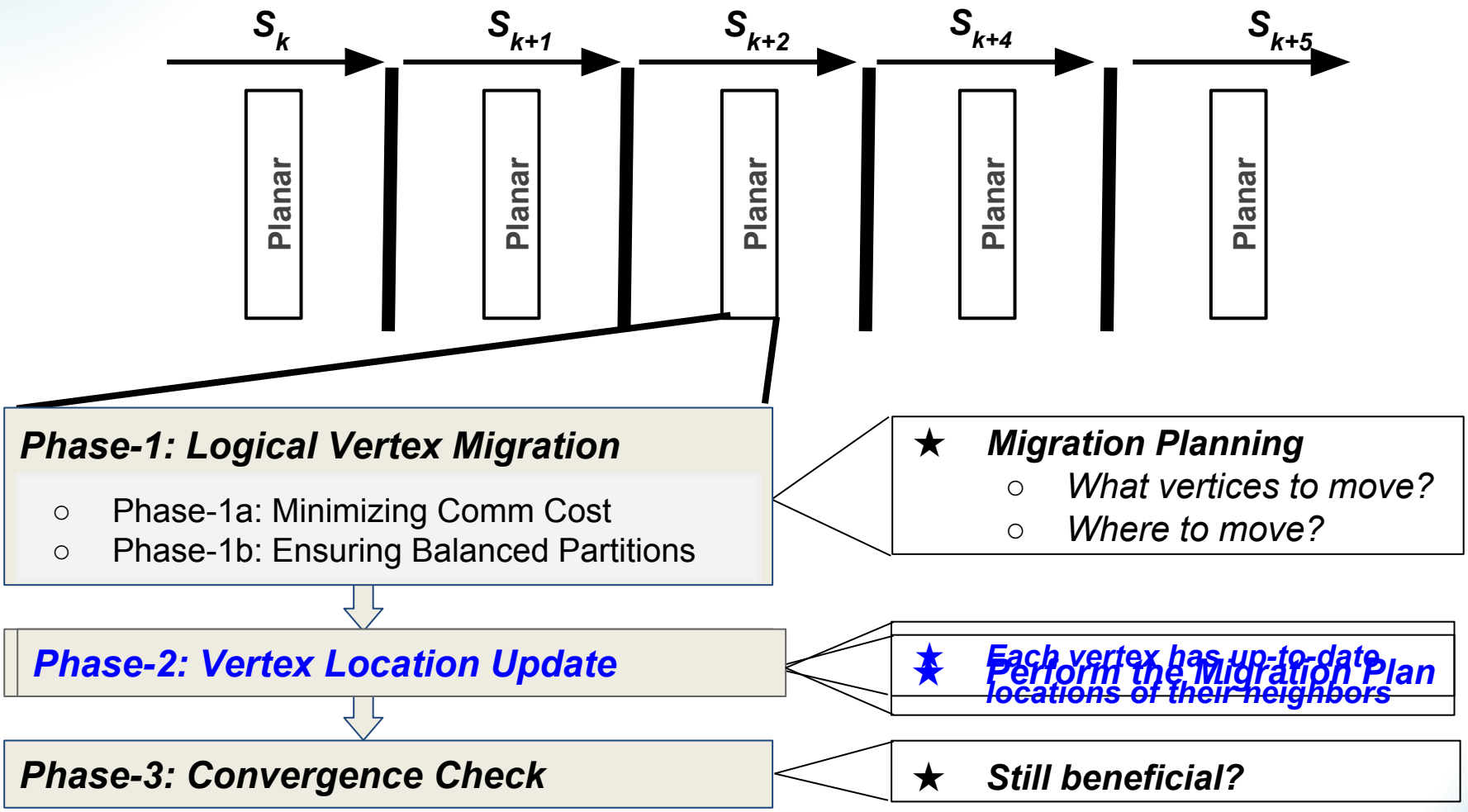
Phase-2: Physical Vertex Migration

★ ***Perform the Migration Plan***

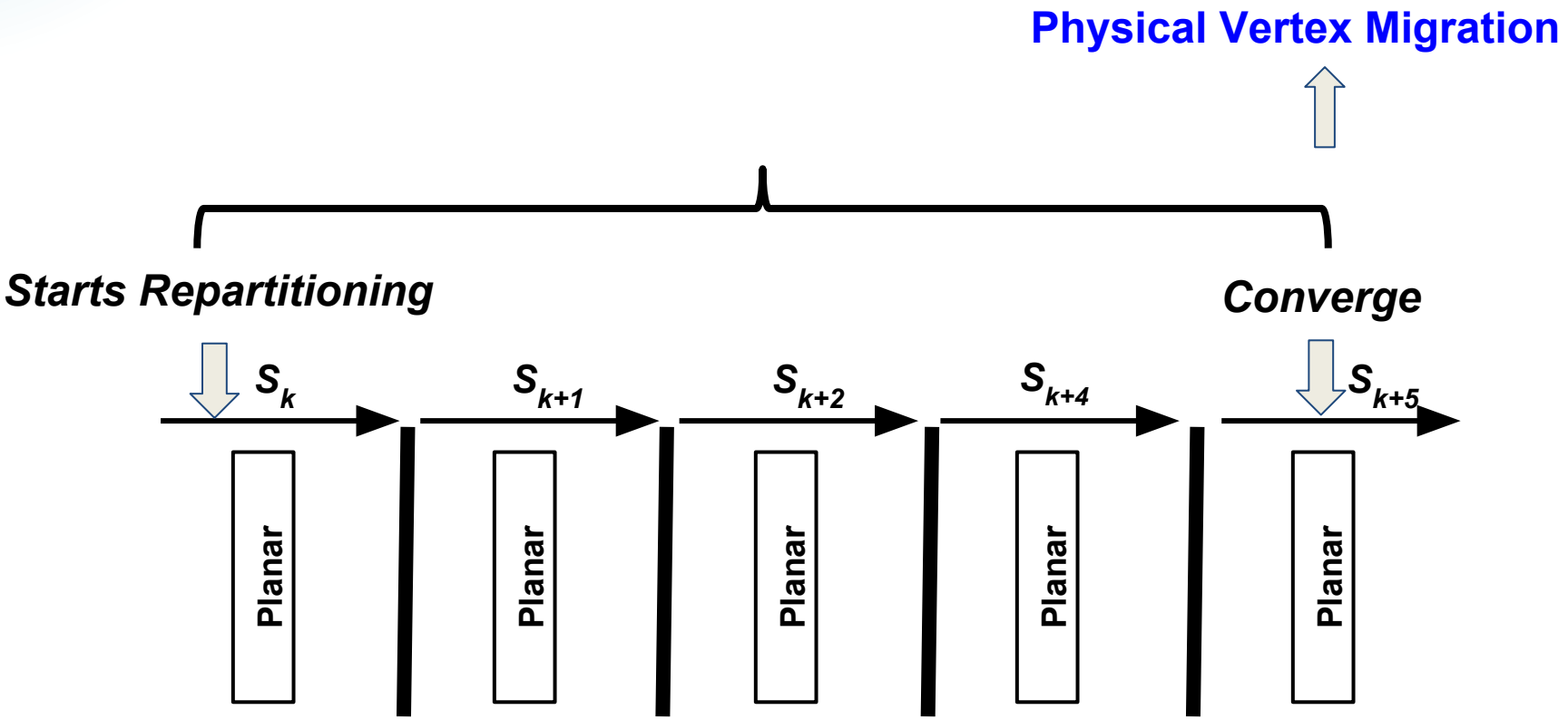
Phase-3: Convergence Check

★ ***Still beneficial?***

Planar+: Eliminating Per Step Physical Vertex Migration



Planar+: Eliminating Per Step Physical Vertex Migration

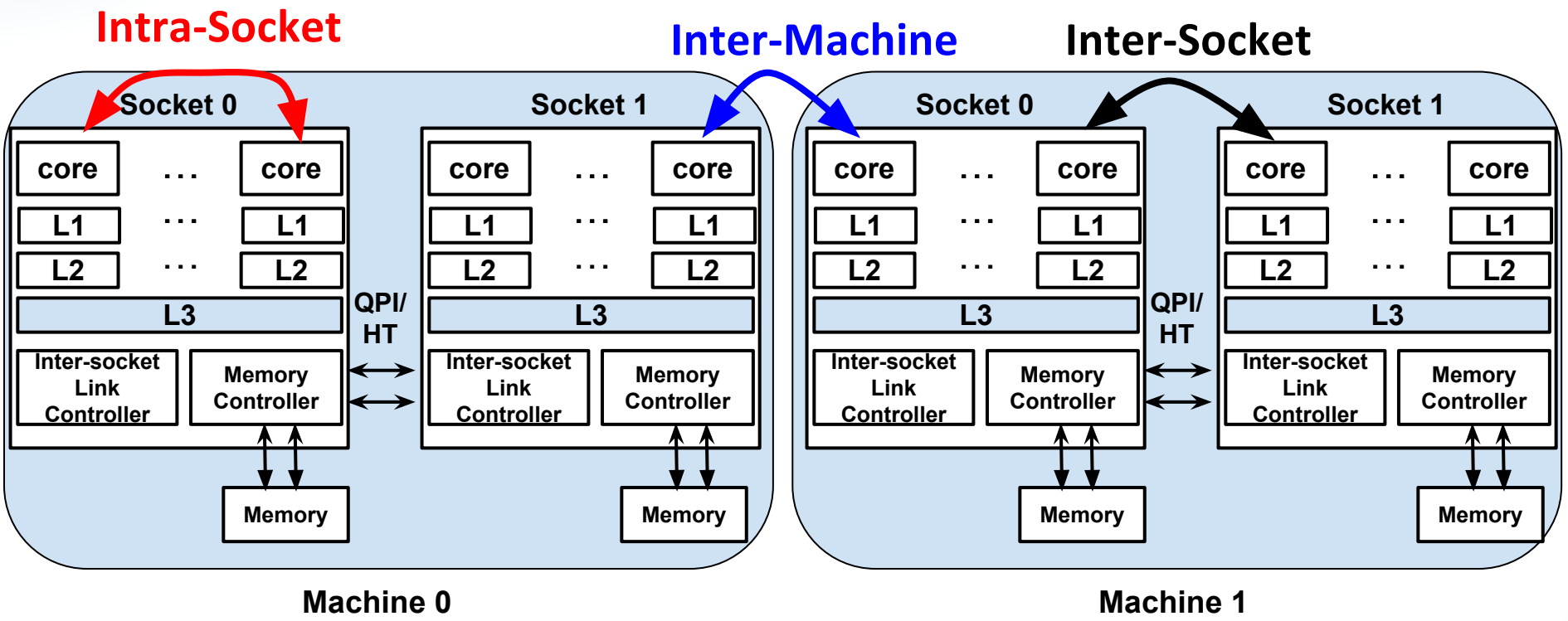


Planar+: Optimized Network Comm Cost Measurement

- five 20-core machines
- one process per core
- two CPU sockets per machine

	Naive Solution (Measuring all pair-wise cost)
# of Pairs Measured	(100 * 99) / 2 = 4950

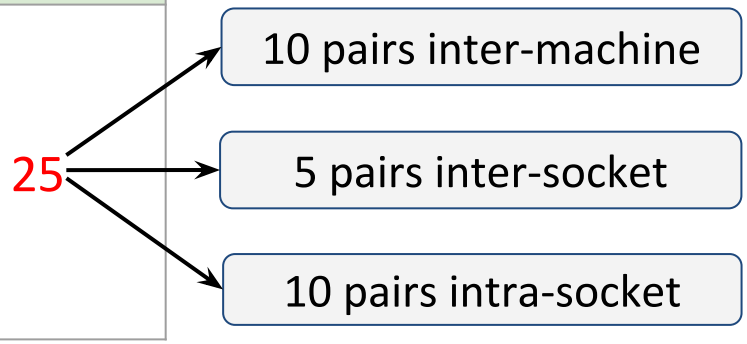
Planar+: Optimized Network Comm Cost Measurement



Planar+: Optimized Network Comm Cost Measurement

- five 20-core machines
- one process per core
- two CPU sockets per machine

	Naive Solution (Measuring all pair-wise cost)	Optimized Solution
# of Pairs Measured	(100 * 99) / 2 = 4950	25



Planar+: Optimized Vertex Gain Computation

$$g^{i,j}(v) = \alpha * (comm(v, P_i) - comm(v, P_j)) - mig(v, P_i, P_j)$$

Comm cost v would cause if it is assigned to partition i

Cost of migrating v from partition i to partition j

$$comm(v, P_i) = \sum_{k=1 \text{ and } k \neq i}^{n} d_{ext}(v, P_k) * c(P_i, P_k)$$

Amount of data v communicates with partition k

Network comm cost between partition i and partition k

O(n), where n is # of partitions.

Planar+: Optimized Vertex Gain Computation

$$comm(v, P_i) = \sum_{k=1 \text{ and } k \neq i}^{n} d_{ext}(v, P_k) * c(P_i, P_k)$$

Amount of data v communicates with **partition k**

Network comm cost between **partition i and partition k**

Optimized Solution:

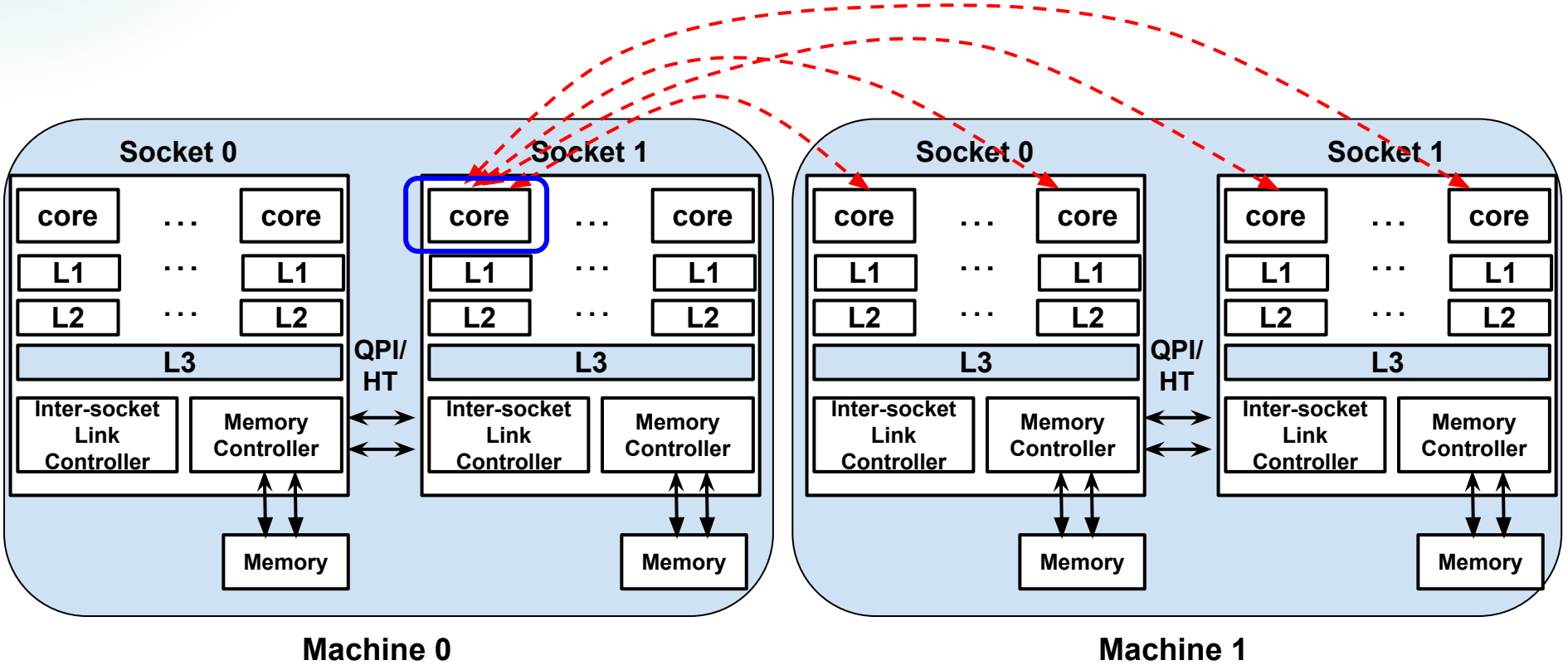
$$\sum_{k=1 \text{ and } k \neq i}^{m} d_{mach}(v, M_k) * c_{mach}(M_i, M_k)$$

Amount of data v communicates with **machine k**

Network comm cost between **machine i and machine k**

O(n)--->O(m), where m is # of machines used.

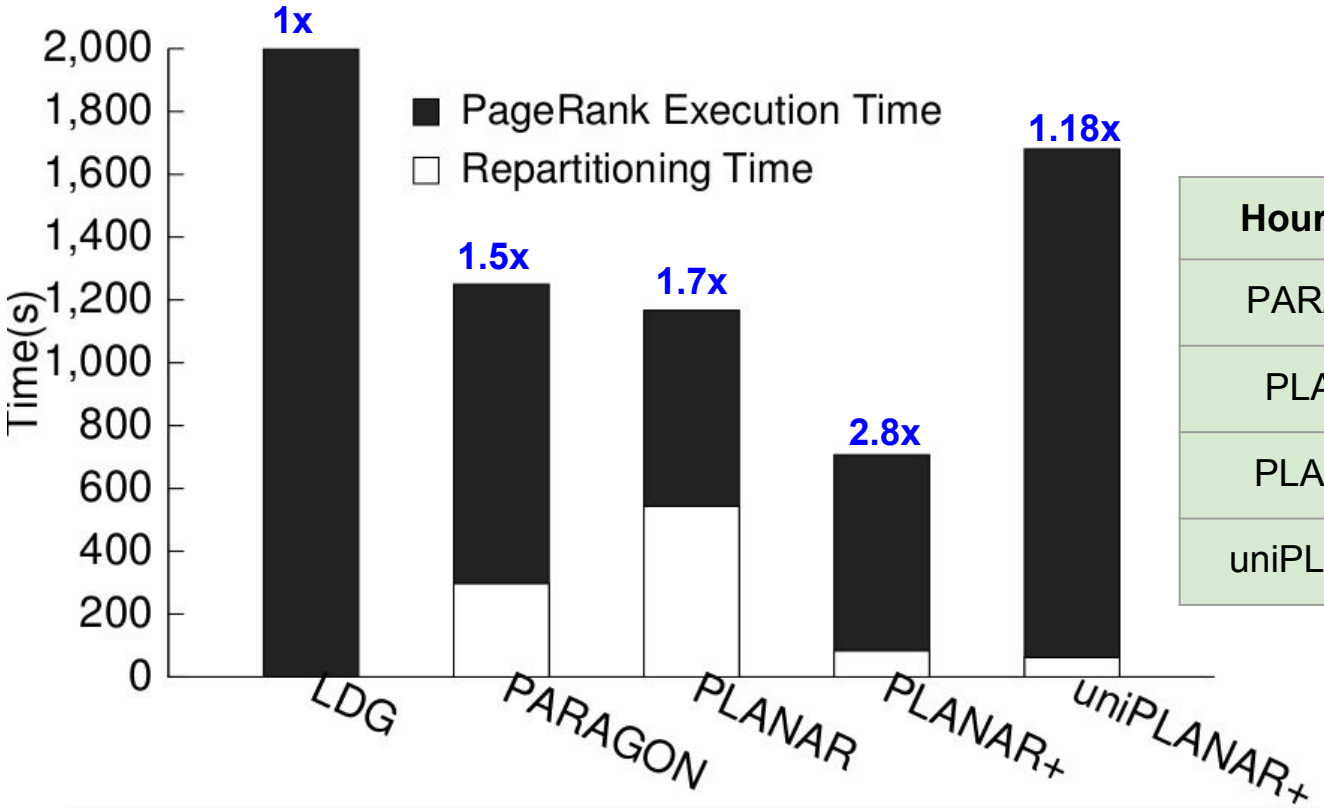
Planar+: Optimized Vertex Gain Computation



$$\sum_{k=1 \text{ and } k \neq i}^{m} d_{mach}(v, M_k) * c_{mach}(M_i, M_k)$$

Planar+: PageRank Execution Time (λ=1)

- ★ **PageRank:** 20 iterations
- ★ **Friendster:** |V|=124M, |E|=3.6B
- ★ **120 Partitions:** six 20-core machines
- ★ **Initial Partitioner:** LDG

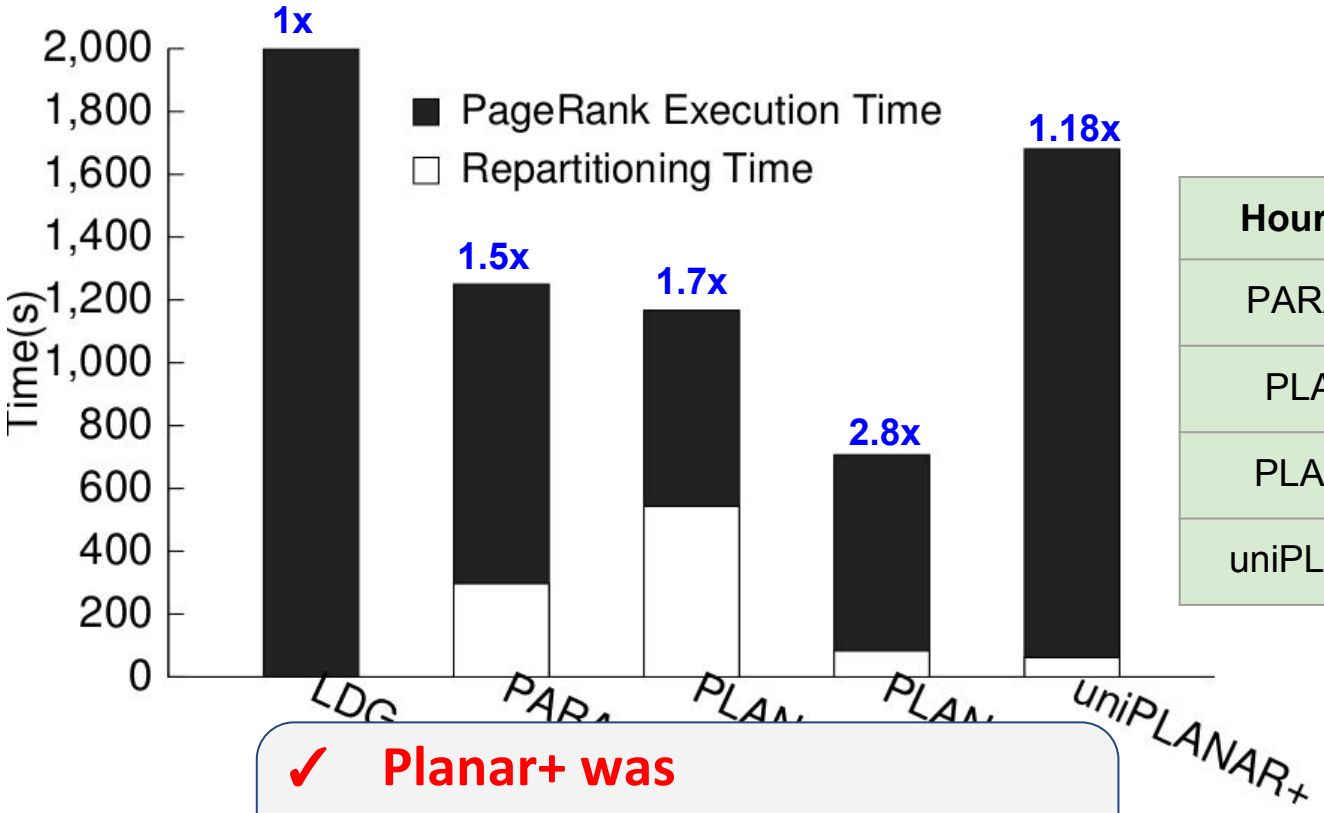


Hours CPU Time Saving	
PARAGON	25h
PLANAR	27h
PLANAR+	43h
uniPLANAR+	10h

✓ **Architecture-aware solutions performed better.**

Planar+: PageRank Execution Time (λ=1)

- ★ **PageRank:** 20 iterations
- ★ **Friendster:** |V|=124M, |E|=3.6B
- ★ **120 Partitions:** six 20-core machines
- ★ **Initial Partitioner:** LDG



Hours CPU Time Saving	
PARAGON	25h
PLANAR	27h
PLANAR+	43h
uniPLANAR+	10h

✓ **Planar+ was**
- **6.5x faster than Planar**
- **3.5x faster than Paragon**

Planar+: Summary

★ **Planar+**

- Lightweight Architecture-Aware Graph Repartitioner
  - **Communication Heterogeneity**
  - **Shared Resource Contention**
- Achieved up to **hours** CPU time saving.
- Scaled well in terms of
  - Graph size (up to **two billion-edge** graphs).
  - Number of partitions (up to **240** partitions).

ARGO: Architecture-Aware Graph Partitioning

Angen Zheng, Alexandros Labrinidis, Panos K. Chrysanthis, Jack Lange
Department of Computer Science, University of Pittsburgh
{anz28, labrinid, panos, jacklange}@cs.pitt.edu

IEEE International Conference on Big Data (IEEE BigData), 2016

Architecture- and Workload-Aware Graph (Re)Partitioning

Aragon [BigGraphs'14] (small dynamic graphs)	**Planar+ [To submit'17]** (large dynamic graphs)
Paragon [EDBT'16] (median-size dynamic graphs)	**Argo [BigData'16]** (static graphs)
Planar [ICDE'16] (large dynamic graphs)	**Sargon [ICDE'17]** (skew-resistant)

Argo: Motivation

- **Highly nonuniform network comm costs**
- **Multicore clusters with high-speed networks**
  - Contention on the memory subsystems
    - **Intra-node** data comm **increases** contention
    - **Inter-node** data comm **alleviates** contention

Argo: Problem Statement

Given G=(V, E)

$$P = \{P_i : \cup_{i=1}^{n} P_i = V \text{ and } P_i \cap P_j = \phi \text{ for any } i \neq j\}$$

Balancing Load:

$$w(P_i) \approx \frac{1}{n} \sum_{v \in V} w(v)$$

Vertex Weight

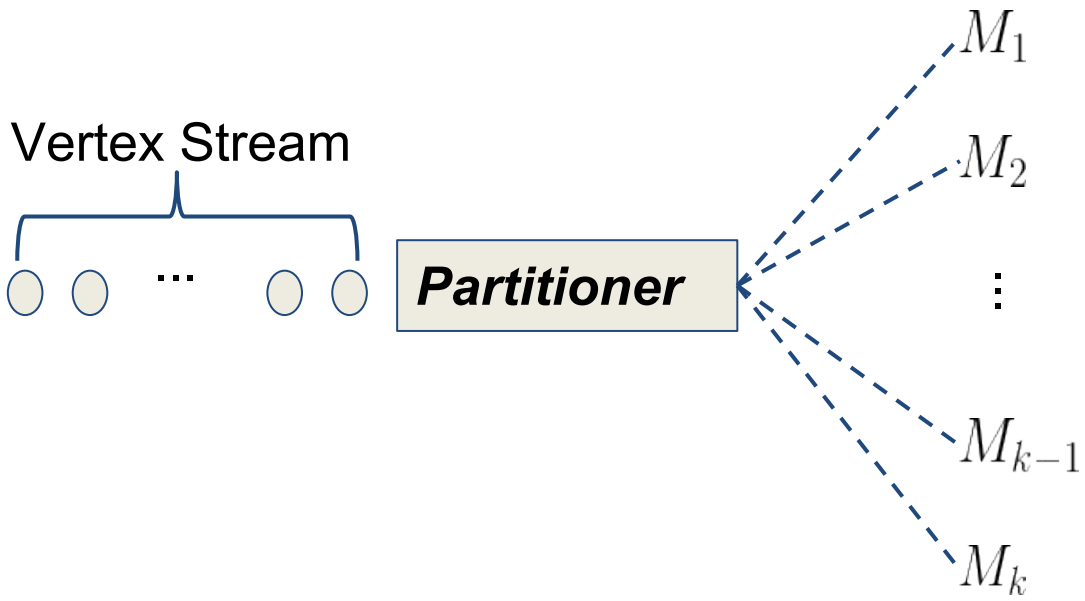
Minimizing Communication:

$$\sum_{e=(u,v) \in E \wedge u \in P_i \wedge v \in P_j \wedge i \neq j} w(e) * c[i][j]$$

Edge Weight

Network comm cost
or
Degree of contention

Argo: Graph Partitioning Model



Streaming Graph Partitioning Model (Stanton et. al, KDD'12)

Argo: Incorporating Heterogeneity

Place vertex, *v*, to a partition, *Pi*, that maximize:

$$\frac{1}{comm(v, P_i) + 1} * (1 - \frac{w(P_i)}{C(P_i)})$$

Weighted Edge-Cut

Penalize the placement based on the load of Pi

$$\sum_{e=(u,v)\in E \text{ and } u\in P_j \text{ and } i\neq j} w(e) * c(P_i, P_j)$$

- ✓ Weighted by **the relative network comm cost**, Argo will avoid
  - edge-cut across nodes
  - inter-node data comm

Argo: Incorporating Contentiousness

Place vertex, *v*, to a partition, *Pi*, that maximize:

$$\frac{1}{comm(v, P_i) + 1} * \left(1 - \frac{w(P_i)}{C(P_i)}\right)$$

Weighted Edge-Cut

Penalize the placement based on the load of Pi

$$\sum_{e=(u,v)\in E \text{ and } u\in P_j \text{ and } i\neq j} w(e) * c(P_i, P_j)$$

- ✓ Weighted by **the degree of contention**, Argo will avoid
  - edge-cut across cores of the same node
  - excess intra-node data comm

Argo: Incorporating Contentiousness

Degree of Contention
($\lambda \in [0, 1]$)

Bottleneck	
Network	**Memory**
λ=0	**λ=1**

$$c(P_i, P_j) = c(P_i, P_j) + \lambda * s$$

Refined Intra-Node Network Comm Cost

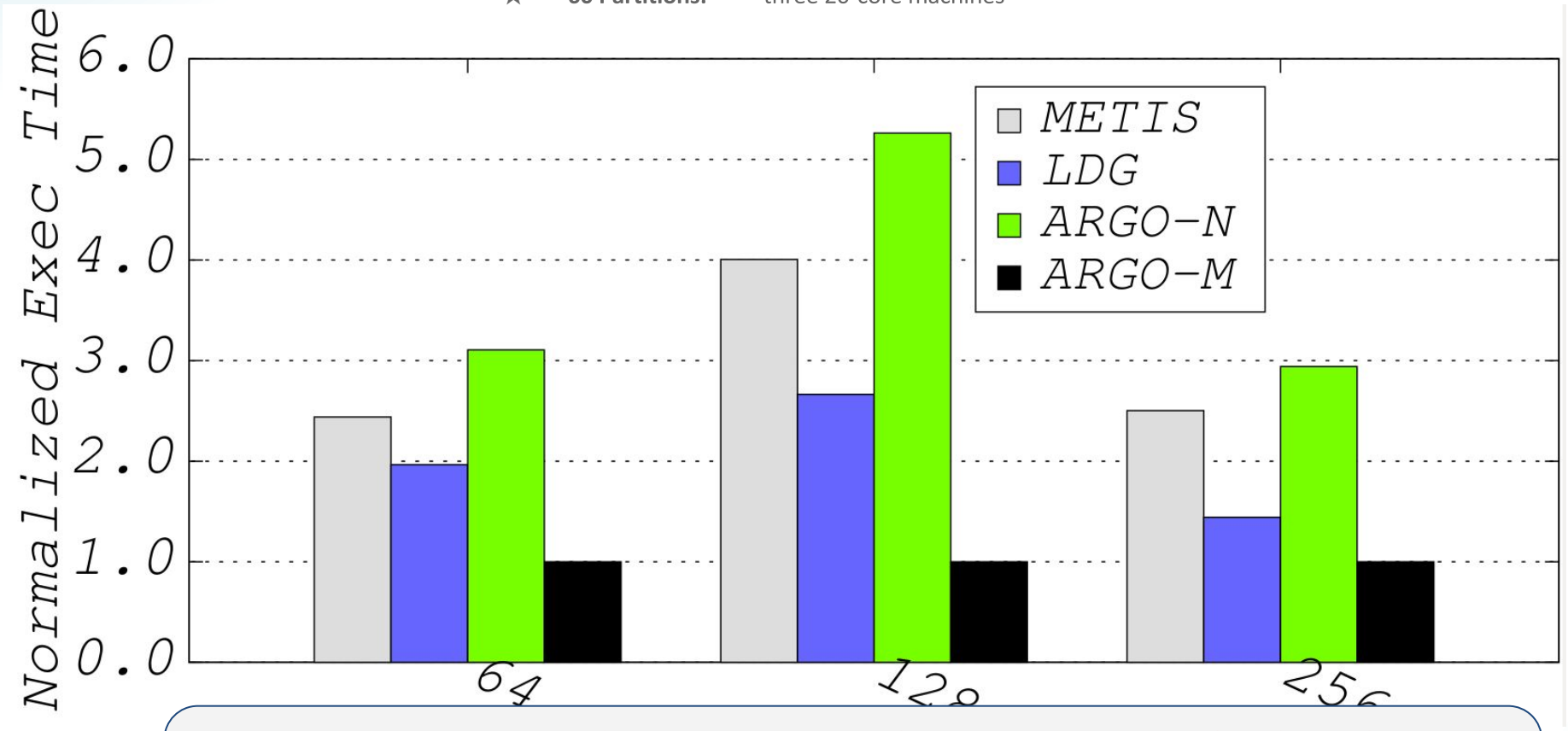
Original Intra-Node Network Comm Cost

Maximal Inter-Node Network Comm Cost

- ✓ Weighted by **the refined network comm cost**, Argo will avoid
  - edge-cut across cores of the same node
  - excess intra-node data comm

Argo: SSSP Exec. Time on Orkut dataset

- ★ **SSSP:** 10 source vertices
- ★ **Orkut:** |V| = 3M, |E| = 234M
- ★ **60 Partitions:** three 20-core machines

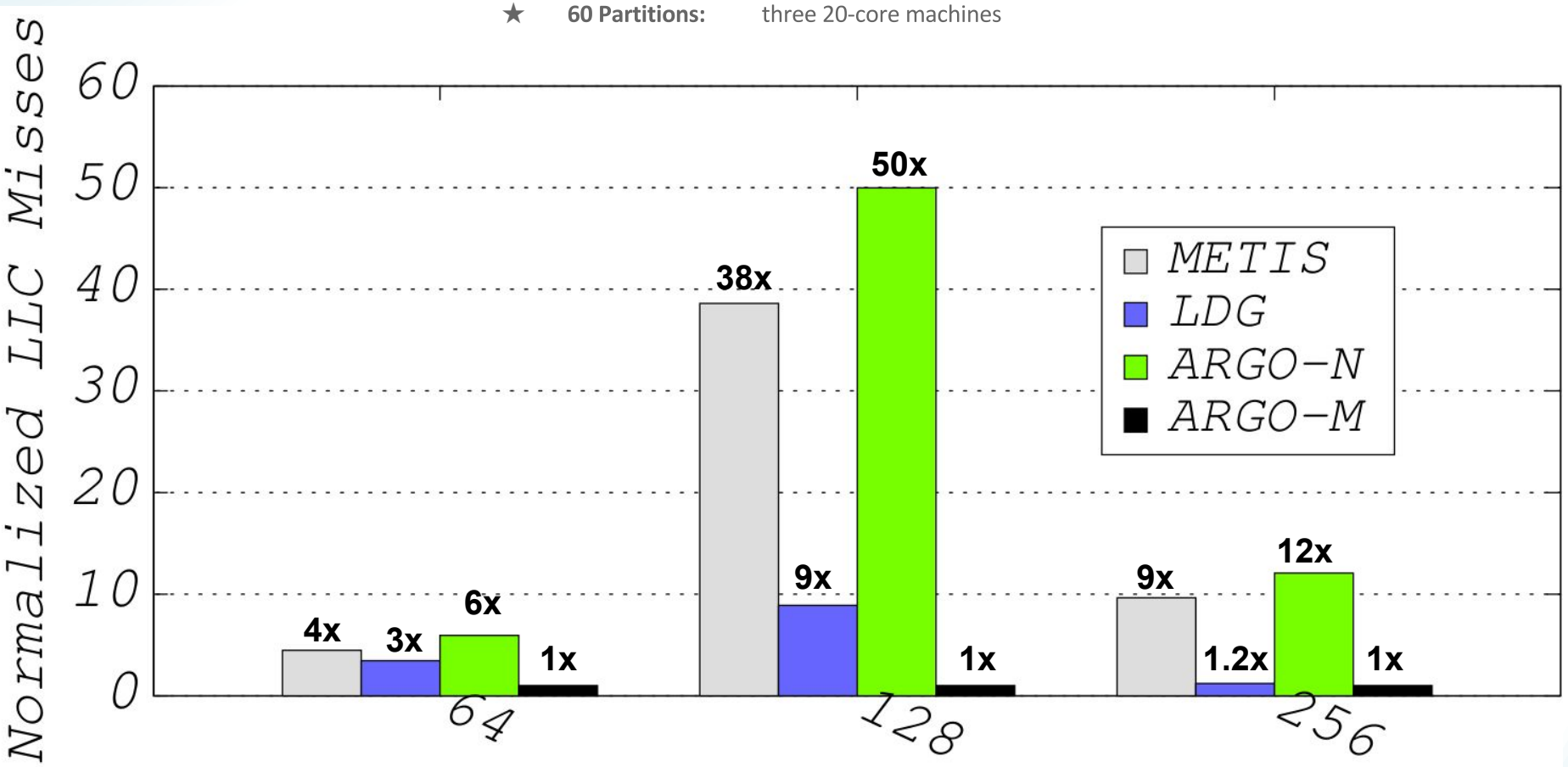


✓ **Contention on the memory subsystems was the bottleneck.**

- **ARGO-M had the lowest SSSP execution time.**
- **ARGO-N had the longest SSSP execution time.**

Argo: SSSP LLC Misses on Orkut dataset

- ★ **SSSP:** 10 source vertices
- ★ **Orkut:** |V| = 3M, |E| = 234M
- ★ **60 Partitions:** three 20-core machines



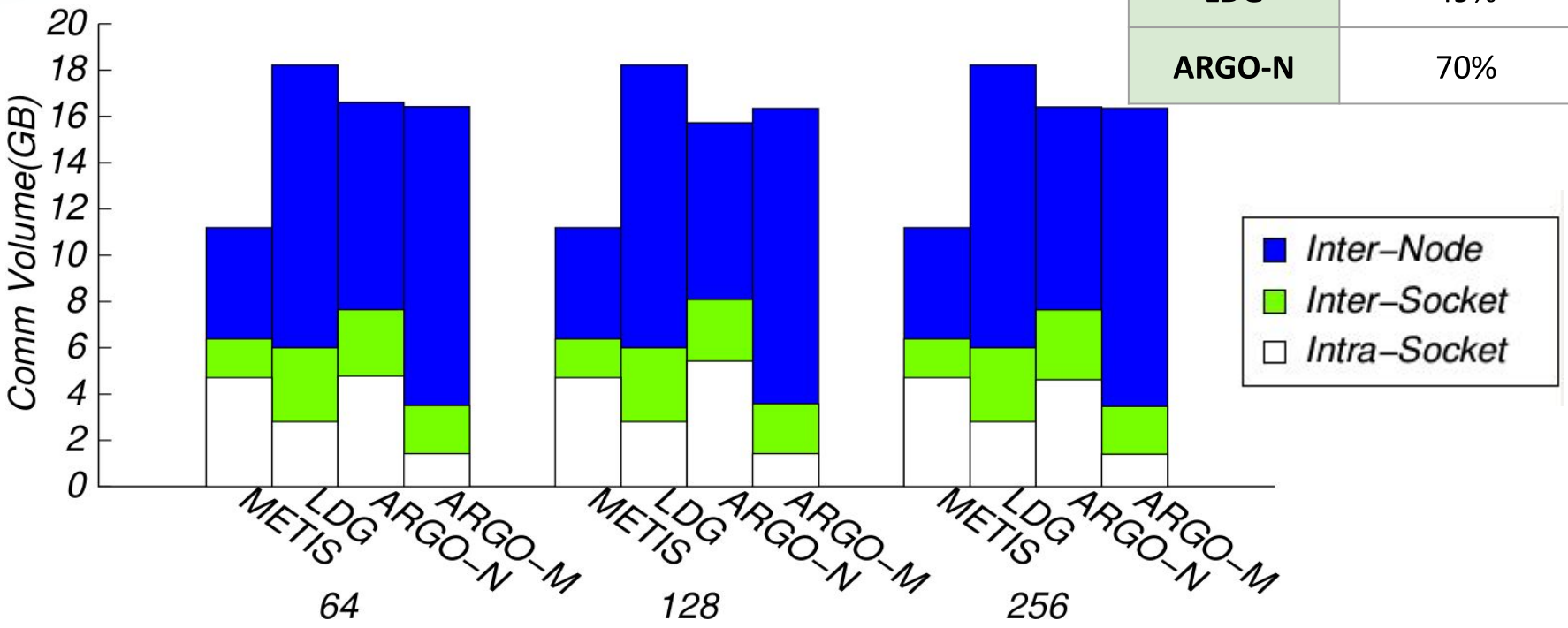
✓ ARGO-M had the lowest LLC Misses.

✓ ARGO-N had the highest LLC Misses.

Argo: SSSP Comm Vol. on Orkut dataset

- ★ **SSSP:** 10 source vertices
- ★ **Orkut:** |V| = 3M, |E| = 234M
- ★ **60 Partitions:** three 20-core machines

64	Intra-Socket
METIS	69%
LDG	49%
ARGO-N	70%



✓ ARGO-M had the lowest intra-node communication volume.
✓ ARGO-N had the highest inter-node communication volume.

Understanding the Performance Impact of Memory Subsystem Contention

Graph Partitioners	METIS and LDG
Graph Workloads	BFS, **SSSP**, and PageRank
Graph Dataset	Orkut (|V|=3M, |E|=234M)
Number of Partitions	16 (one partition per core)

Understanding the Performance Impact of Memory Subsystem Contention

m:s:c	SSSP Execution Time (s)	
	METIS	LDG
1:2:8	633	2,632
2:2:4	654	2,565
4:2:2	521	631
8:2:1	222	280

9x

m: # of machines used
s: # of sockets used per machine
c: # of cores used per socket

✓ **Denser configurations had longer execution time.**
- **Contention on the memory subsystems impacted performance.**
- Network may not always be the bottleneck.

Understanding the Performance Impact of Memory Subsystem Contention

m:s:c	SSSP Execution Time (s)	
	METIS	LDG
1:2:8	633	2,632
2:2:4	654	2,565
4:2:2	521	631
8:2:1	222	280

9x

m:s:c	SSSP LLC Misses (in Millions)	
	METIS	LDG
1:2:8	10,292	44,117
2:2:4	10,626	44,689
4:2:2	2,541	1,061
8:2:1	96	187

235x

- ✓ **Denser configurations had longer execution time.**
  - ○ **Contention on the memory subsystems impacted performance.**
  - ○ Network may not always be the bottleneck.

Understanding the Performance Impact of Memory Subsystem Contention

m:s:c	SSSP Execution Time (s)	
	METIS	LDG
1:2:8	633	2,632
2:2:4	654	2,565
4:2:2	521	631
8:2:1	222	280

9x

m:s:c	SSSP LLC Misses (in Millions)	
	METIS	LDG
1:2:8	10,292	44,117
2:2:4	10,626	44,689
4:2:2	2,541	1,061
8:2:1	96	187

235x

- ✓ **Denser configurations had longer execution time.**
  - Contention on the memory subsystems impacted performance.
  - **Network may not always be the bottleneck.**

Understanding the Performance Impact of Memory Subsystem Contention

m:s:c	SSSP Execution Time (s)	
	METIS	LDG
1:2:8	633	2,632
2:2:4	654	2,565
4:2:2	521	631
8:2:1	222	280

9x

m:s:c	SSSP LLC Misses (in Millions)	
	METIS	LDG
1:2:8	10,292	44,117
2:2:4	10,626	44,689
4:2:2	2,541	1,061
8:2:1	96	187

235x

✓ **METIS had lower execution time and LLC misses than LDG.**
- **Edge-cut matters.**
- **Higher edge-cut-->higher comm-->higher contention**

Argo: Summary

- ✓ **Findings**
  - Network is not always the bottleneck.
  - Contention on the memory subsystems may impact the performance a lot.
    - due to excess intra-node data communication
- ✓ **ARGO**
  - Avoids contention by offloading excess intra-node data comm across nodes.
  - Achieved up to **11x** improvement on real-world workloads.
  - Scaled well in terms of
    - Graph size (up to **2 billion-edge** graphs)
    - # of partitions (up to **200** partitions)

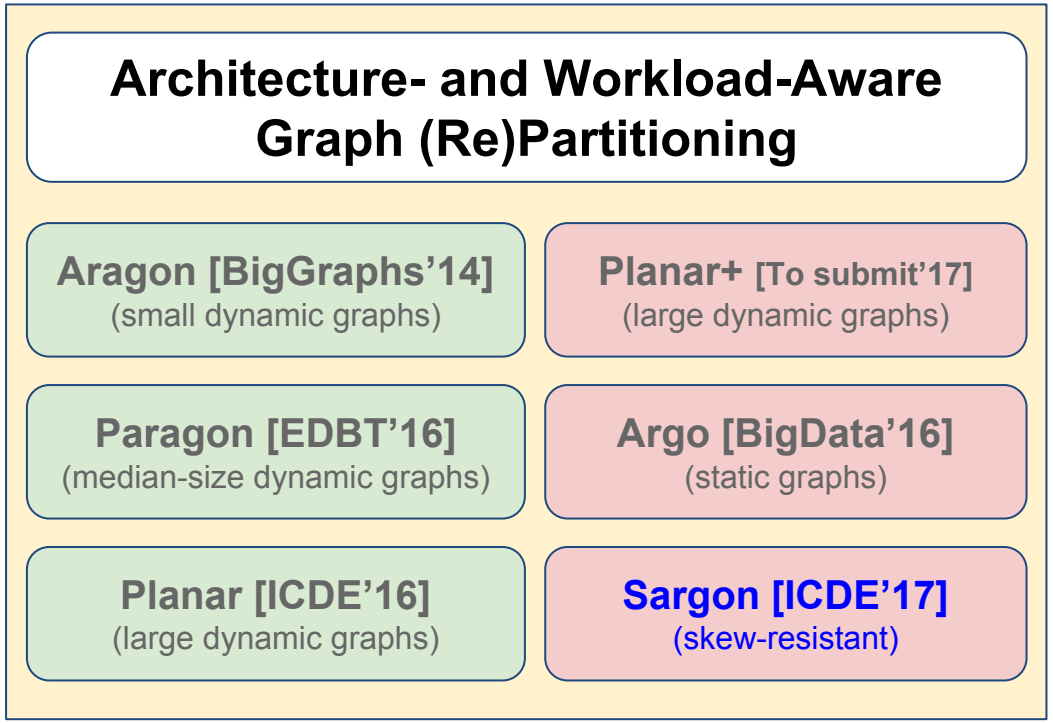
Skew-Resistant Graph Partitioning

Angen Zheng
University of Pittsburgh
anz28@cs.pitt.edu

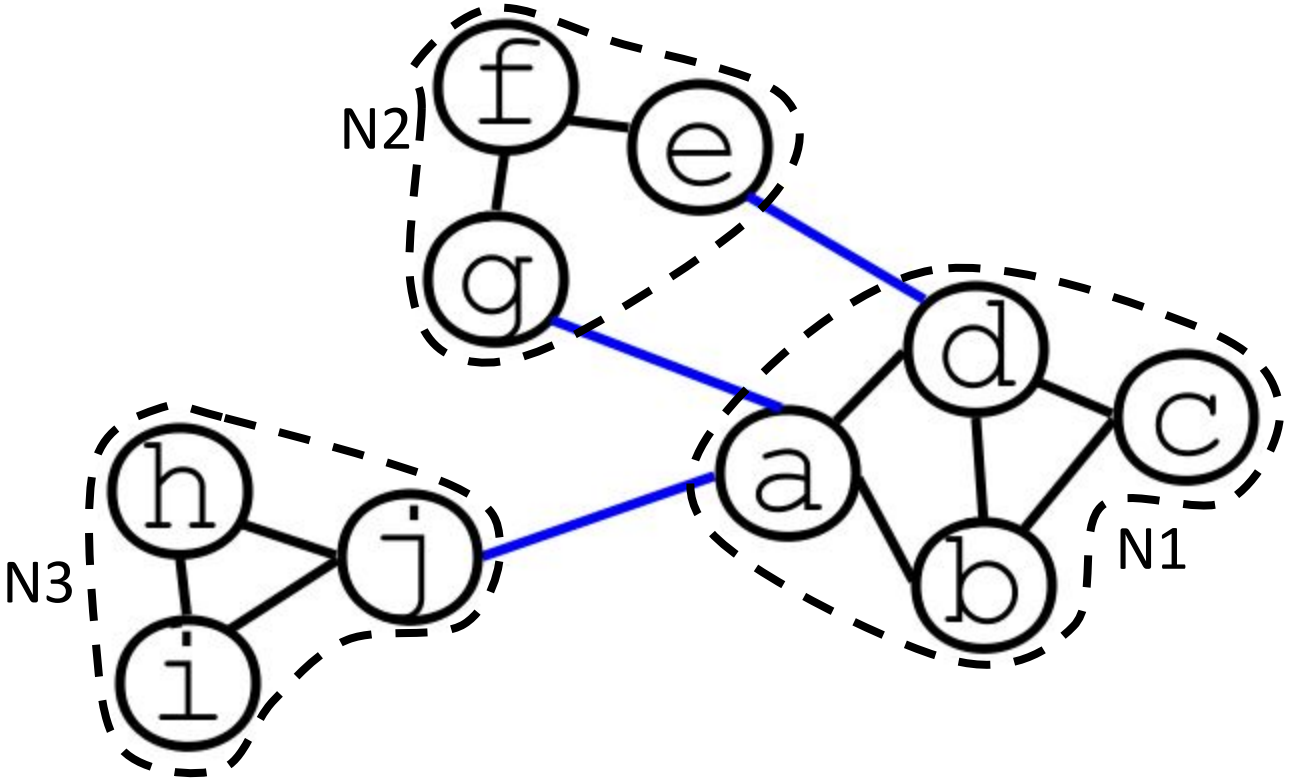
Alexandros Labrinidis
University of Pittsburgh
labrinid@cs.pitt.edu

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

33nd IEEE International Conference on Data Engineering (ICDE), 2017



A Balanced Partitioning = Even Load Distribution
Minimal Edge-Cut = Minimal Data Comm



Assumption: Vertices of the graph are always active.

A Balanced Partitioning = Even Load Distribution
Minimal Edge-Cut = Minimal Data Comm

Assumption broken by:

- Traversal-Style Graph Workloads
- Multiphase Graph Workloads
- Graph Workloads with Seasonal Access Pattern
- Graphs with Varying Hotness on the Vertices

Assumption: **Vertices of the graph are always active.**

Traversal-Style Graph Workloads:

Active Vertex Distribution Across Supersteps

Workloads	BFS & SSSP

com-orkut	
|V|	3M
|E|	234M
Max Degree	33,313
Avg. Degree	76.281
Max Diameter	10
Avg. Diameter	5.4489

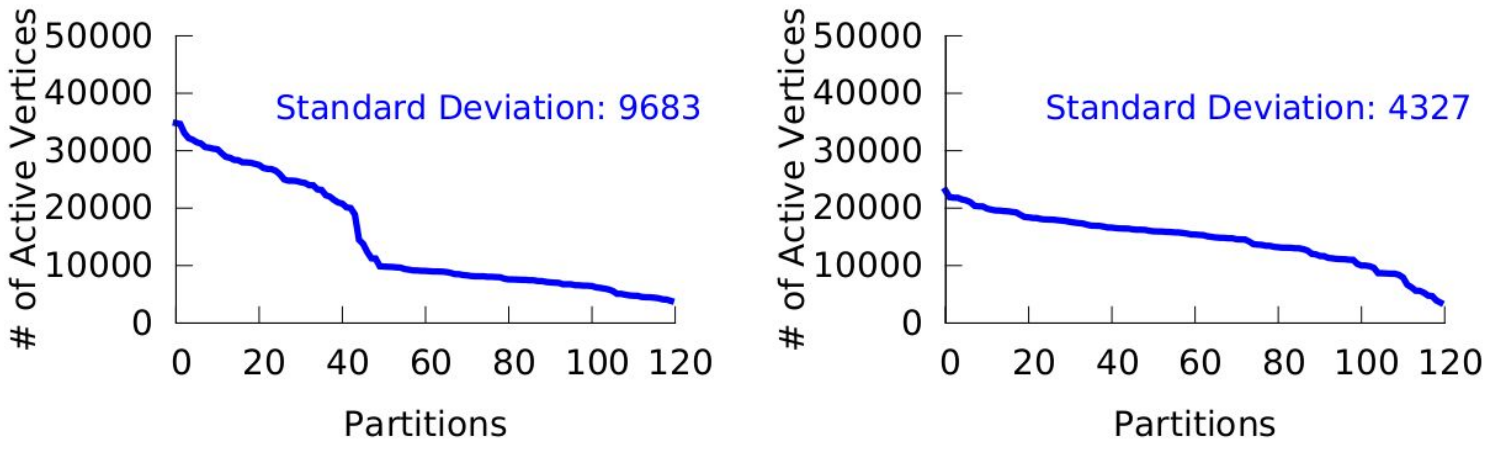
com-orkut	# of Active Vertices	
Supersteps	BFS	SSSP
0	1	1
1	72	45
2	5,871	4,663
3	215,425	297,943
4	1,753,891	1,421,993
5	1,088,870	1,229,917
6	8,242	117,496
7	69	383
8	0	0

Not all vertices are always active.

Highly skewed active vertex distribution across supersteps.

Traversal-Style Graph Workloads:

Active Vertex Distribution Across Partitions

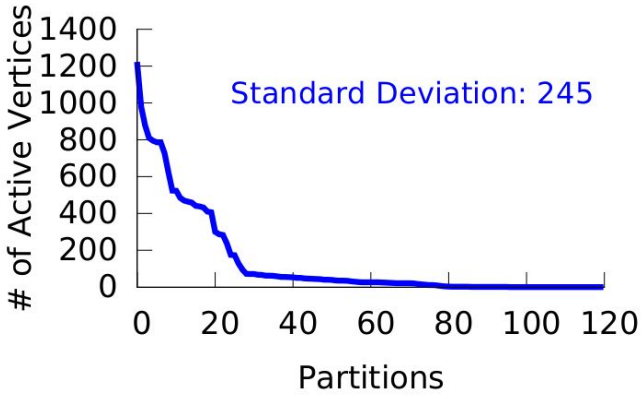


Highly skewed active vertex distribution across partitions

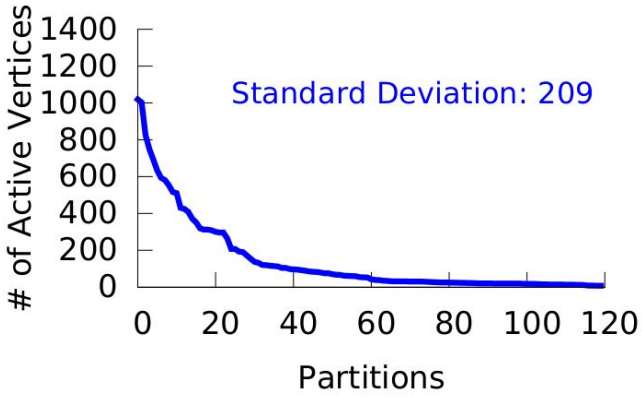
Distribute vertices active in the same time step evenly across partitions.

Traversal-Style Graph Workloads:

High-Degree Vertex Distribution Across Partitions



(a) METIS



(b) LDG

Highly Skewed high-degree vertex distribution across partitions.

Distribute high-degree vertices evenly across partitions.

Sargon: Skew-Resistant Graph Partitioning

Given G=(V, E)

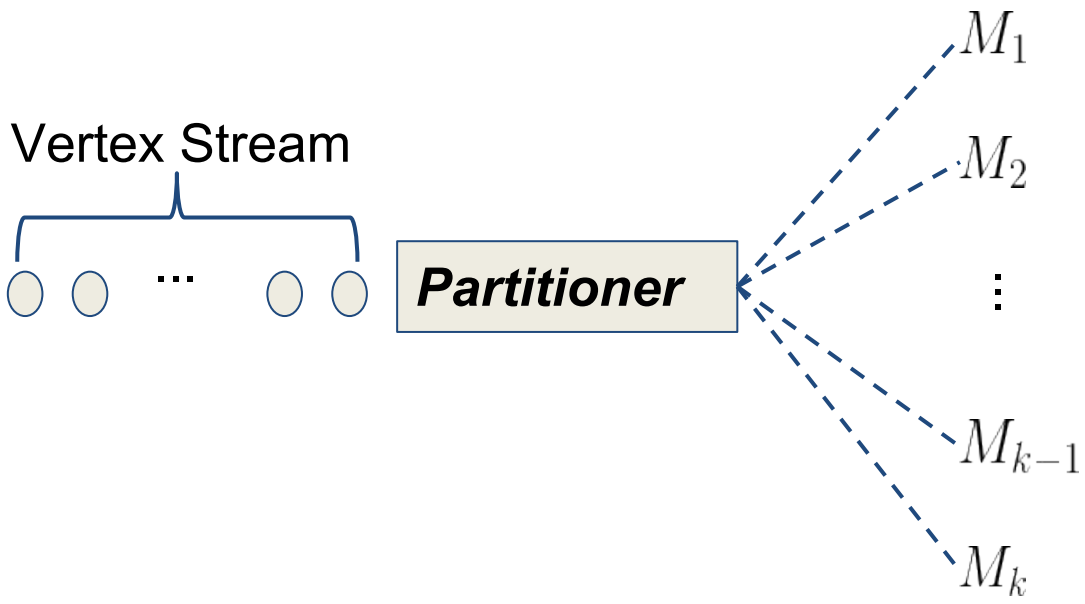
- **Assign a label vector to each vertex to indicate:**
  - the time periods the vertex is active in
  - whether it is a high- or low-degree vertex
  - the hotness of the vertex

$$P = \{P_i : \cup_{i=1}^{n} P_i = V \text{ and } P_i \cap P_j = \phi \text{ for any } i \neq j\}$$

Balancing Load: $$w(P_i, l) \approx \frac{1}{n} \sum_{v^l \in V} w(v^l), \text{ for any label } l$$

Minimizing Communication: $$\sum_{e=(u,v) \in E \wedge u \in P_i \wedge v \in P_j \wedge i \neq j} w(e)$$

Sargon: Graph Partitioning Model



Streaming Graph Partitioning Model (Stanton et. al, KDD'12)

Sargon: Vertex Placement Heuristic

Place vertex, *v*, to a partition, *Pi*, that maximize:

$$\sum_{e=(u,v)\in E \wedge u\in P_i} w(e) * loadFactor(P_i, v)$$

$$\sum_{l\in\{v\text{'s labels}\}} (1 - \frac{w(P_i, l)}{C(P_i, l)})$$

Load of l-labeled vertices

Capacity of l-labeled vertices

Sargon will try its best to:

- Distribute vertices of the same labels evenly across partitions.
- Minimize the edgecut.

Extract the labels from the execution trace of the workload.

Sargon: Assumption Validation

Workloads	BFS and SSSP (one randomly selected source vertex)
Dataset	Orkut (|V|=3M, |E|=234M)
# of Traces Collected	5
Similarity	Percentage of the vertices overlapped in the peak superstep

$$\frac{|S_1 \cap S_2|}{|S_1|}$$

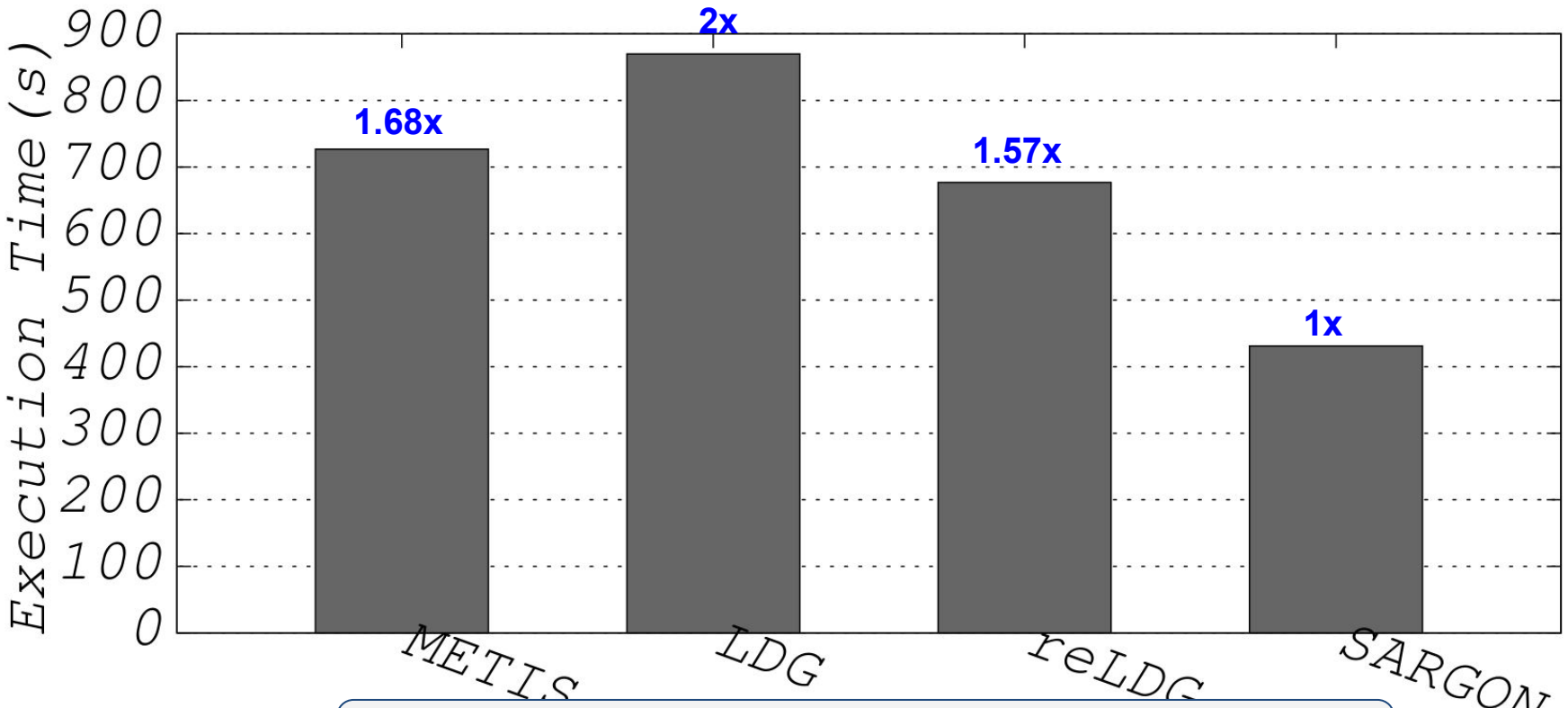
Workloads	Avg. Similarity	Std. Deviation
BFS	60.80%	8.43%
SSSP	64.73%	10.63%

Workload execution traces are quite similar.

Quite predictable runtime characteristics.

Sargon: SSSP Execution Time

- ★ **SSSP:** 100 source vertices
- ★ **Orkut:** |V|=3M, |E|=234M
- ★ **60 Partitions:** three 20-core machines



✓ Up to 2x speedups (hours CPU time saving).

Sargon: Summary

- ✓ **A study of the traversal-style graph workloads**
  - Not all vertices are always active.
  - The distribution of active vertices across supersteps may be highly skewed.
  - The execution traces of the workloads are quite similar in many cases.
- ✓ **SARGON**
  - Avoids time-varying skewness of the target workload.
  - Achieved up to **2x** speedups on real-world workloads.

Conclusions

Thanks!

Architecture- and Workload-Aware Graph (Re)Partitioning

Aragon [BigGraphs'14] (small dynamic graphs)	**Planar+ [To submit'17]** (large dynamic graphs)
Paragon [EDBT'16] (median-size dynamic graphs)	**Argo [BigData'16]** (static graphs)
Planar [ICDE'16] (large dynamic graphs)	**Sargon [ICDE'17]** (skew-resistant)

Acknowledgments:

- Thesis Committee
- Christos Faloutsos

Funding:

- NSF OIA-1028162
- NSF CBET-1250171
- NSF CBET-1609120

(P)aragon: https://github.com/admtlab/paragon
Planar+ : https://github.com/admtlab/planar

Publications

[1]. Planar+: Parallel Lightweight Architecture-Aware Graph Repartitioning,
Angen Zheng, Patrick Pisciuneri, Alexandros Labrinidis, Panos K. Chrysanthis, Jack Lange, Peyman Givi
To submit, 2017

[2]. Skew-Resistant Graph Partitioning (Short Paper)
Angen Zheng, Alexandros Labrinidis, Christos Faloutsos
33nd IEEE International Conference on Data Engineering (ICDE), 2017

[3]. Argo: Architecture-Aware Graph Partitioning
Angen Zheng, Alexandros Labrinidis, Panos K. Chrysanthis, Jack Lange
IEEE International Conference on Big Data (IEEE BigData), 2016

[4]. Planar: Parallel Lightweight Architecture-Aware Adaptive Graph Repartitioning
Angen Zheng, Alexandros Labrinidis, Panos K. Chrysanthis
32nd IEEE International Conference on Data Engineering (ICDE), 2016

[5]. Paragon: Parallel Architecture-Aware Graph Partition Refinement Algorithm
Angen Zheng, Alexandros Labrinidis, Patrick Pisciuneri, Panos K. Chrysanthis, Peyman Givi
19th International Conference on Extending Database Technology (EDBT), 2016

[6]. Architecture-Aware Graph Repartitioning for Data-Intensive Scientific Computing
Angen Zheng, Alexandros Labrinidis, Panos K. Chrysanthis
1st International Workshop on High Performance Big Graph Data Management, Analysis, and Mining, 2014