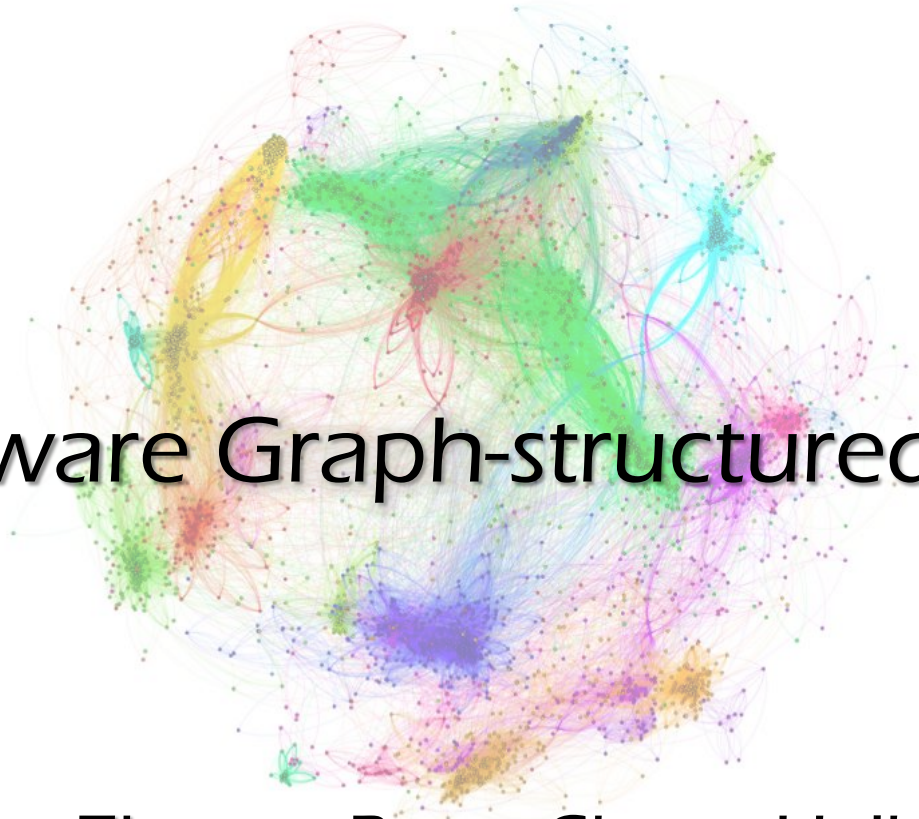


NUMA-aware Graph-structured Analytics

Kaiyuan Zhang, Rong Chen, Haibo Chen

Institute of Parallel and Distributed Systems
Shanghai Jiao Tong University, China

Big Data Everywhere

400 Million Tweets/day

1.11 Billion Users

6 Billion Photos

100 Hrs of Video every minute

How do we understand and use Big Data?

Data Analytics

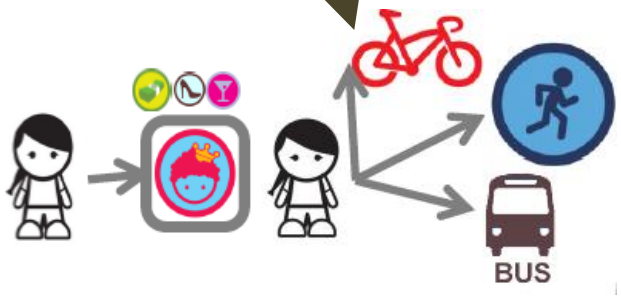
400 Million Tweets/day

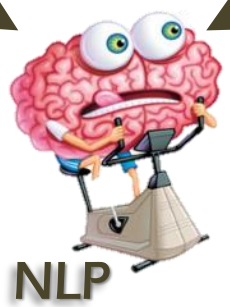
1.11 Billion Users

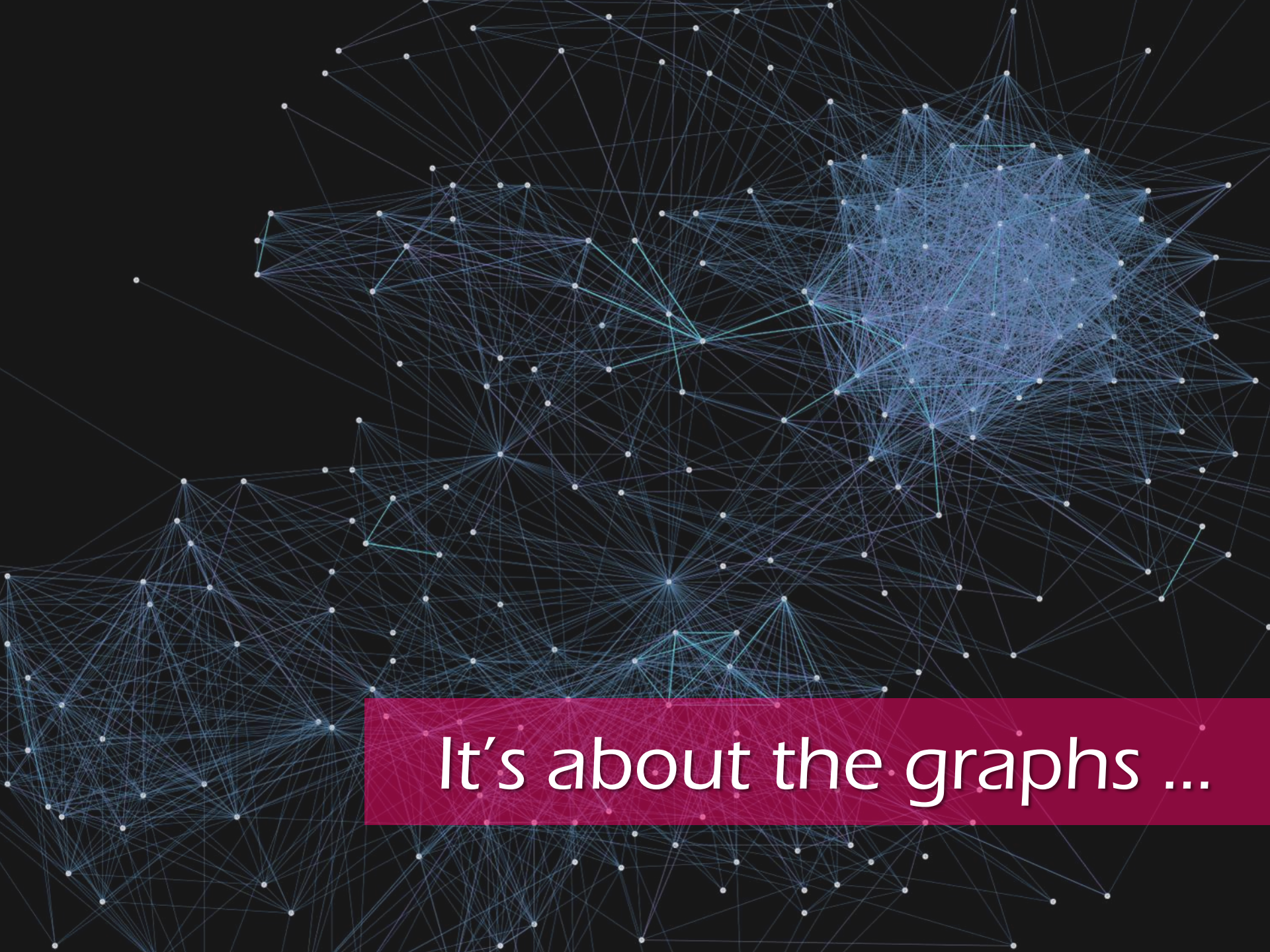
6 Billion Photos

100 Hrs of Video every minute

Machine Learning and Data Mining



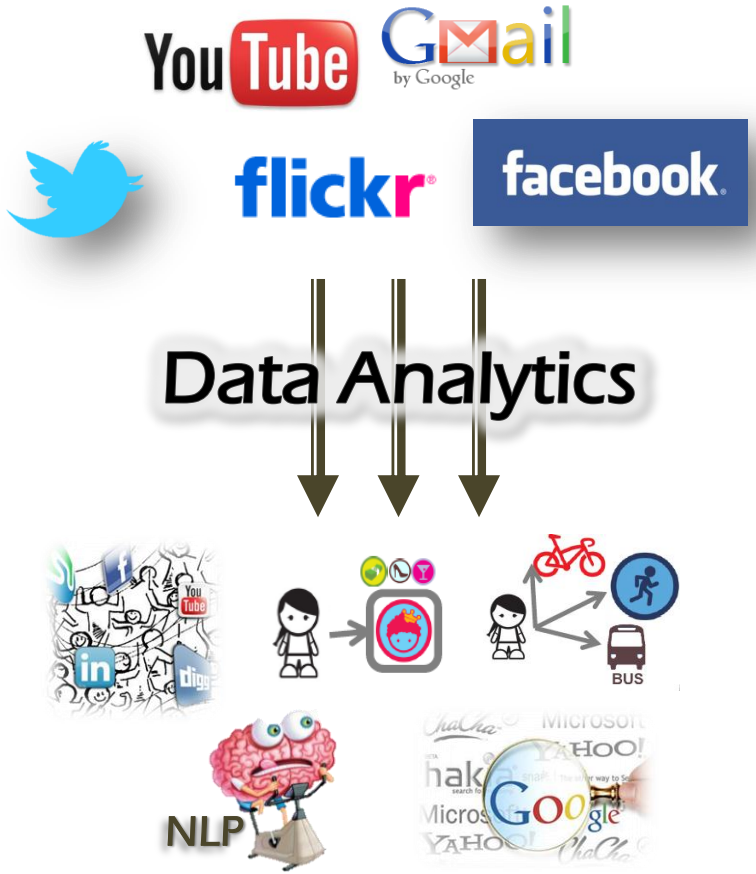




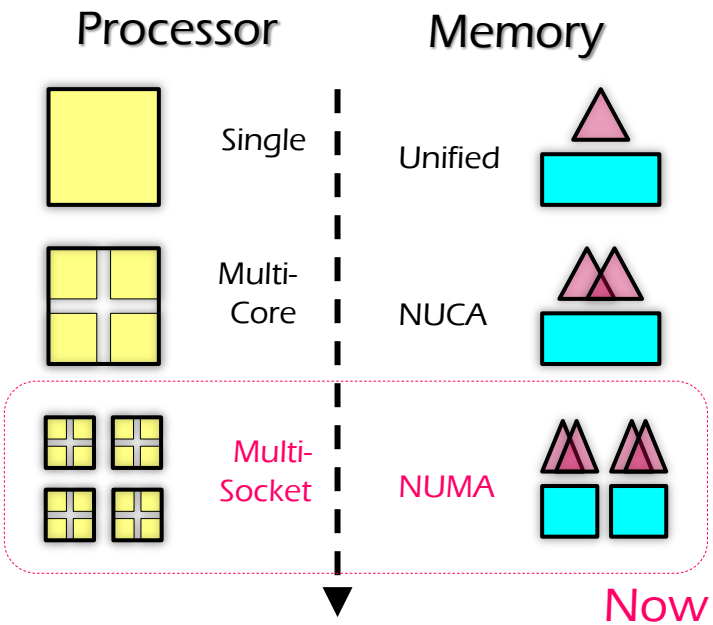
It's about the graphs ...

NUMA & Graph-analytics

Application



Hardware



e.g. 80 Cores with 1TB RAM

8 Sockets X (10 Cores with128GB local RAM)

How about ?

NUMA systems + Graph-analytics

Contribution

Polymer: NUMA-aware Graph-structured Analytics

- A comprehensive analysis that uncovers issues for running graph analytics system on NUMA platform
- A new system that exploits both NUMA-aware data layout and memory access strategies
- Three optimizations for global synchronization efficiency, load balance and data structure flexibility
- A detailed evaluation that demonstrates the performance and scalability benefits

Outline

Background & Issues

Design of Polymer

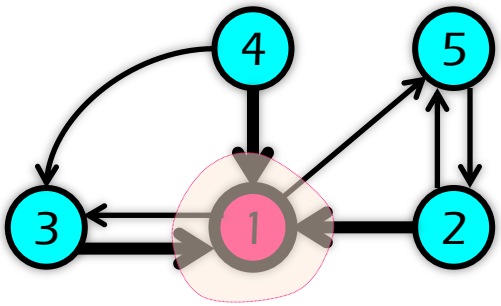
Evaluation

Outline

- Background & Issues
- Design of Polymer
- Evaluation

Example: PageRank

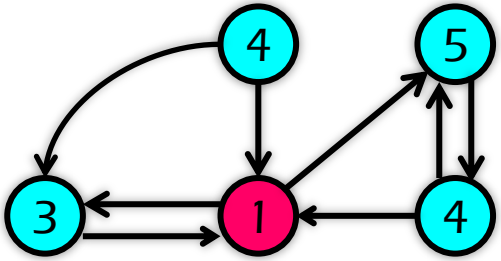
A centrality analysis algorithm
to measure the **relative rank**
for **each element** of a **linked** set

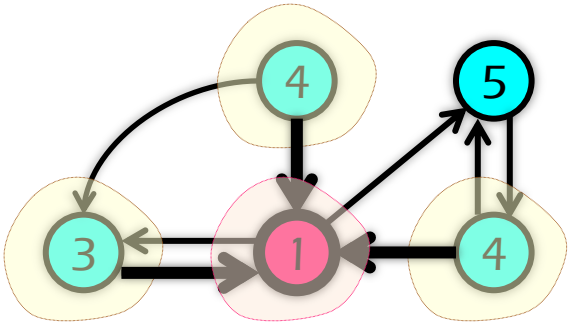


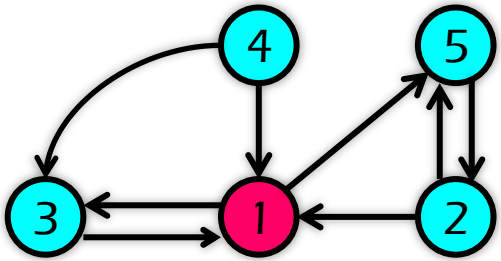
$$R_i = \alpha + (1-\alpha) \sum_{(j,i) \in E} \omega_{ij} R_j$$

Characteristics

- Linked set → data dependence
- Rank of who links it → predictable accesses
- Convergence → iterative computation







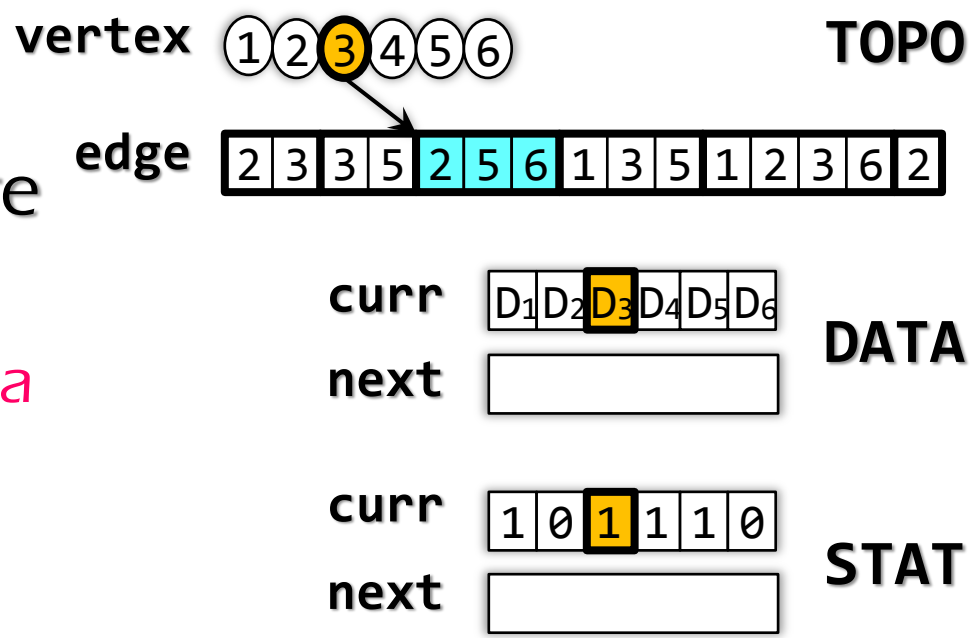
Graph-analytics

The scatter-gather model

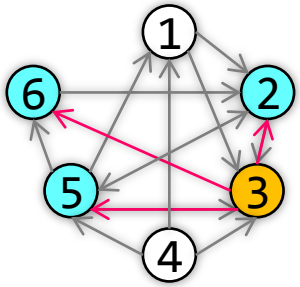
- "scatter": propagate the current value of a vertex to its neighbors along edges
- "gather": accumulate values from neighbors to compute the next value of a vertex

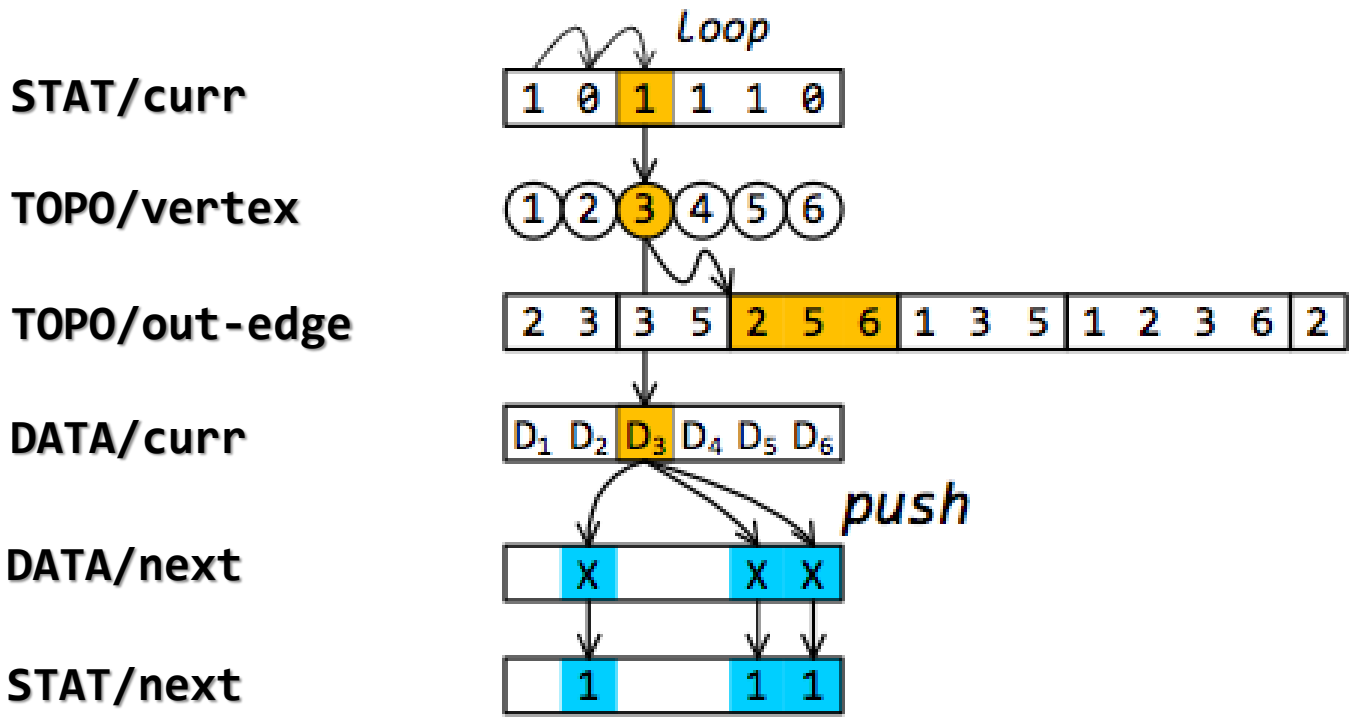
In-memory data structure

- Graph Topology
- Application-specific Data
- Runtime State

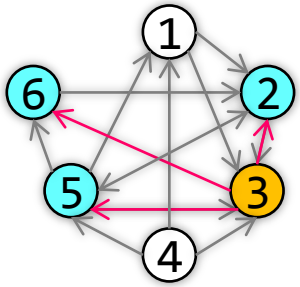


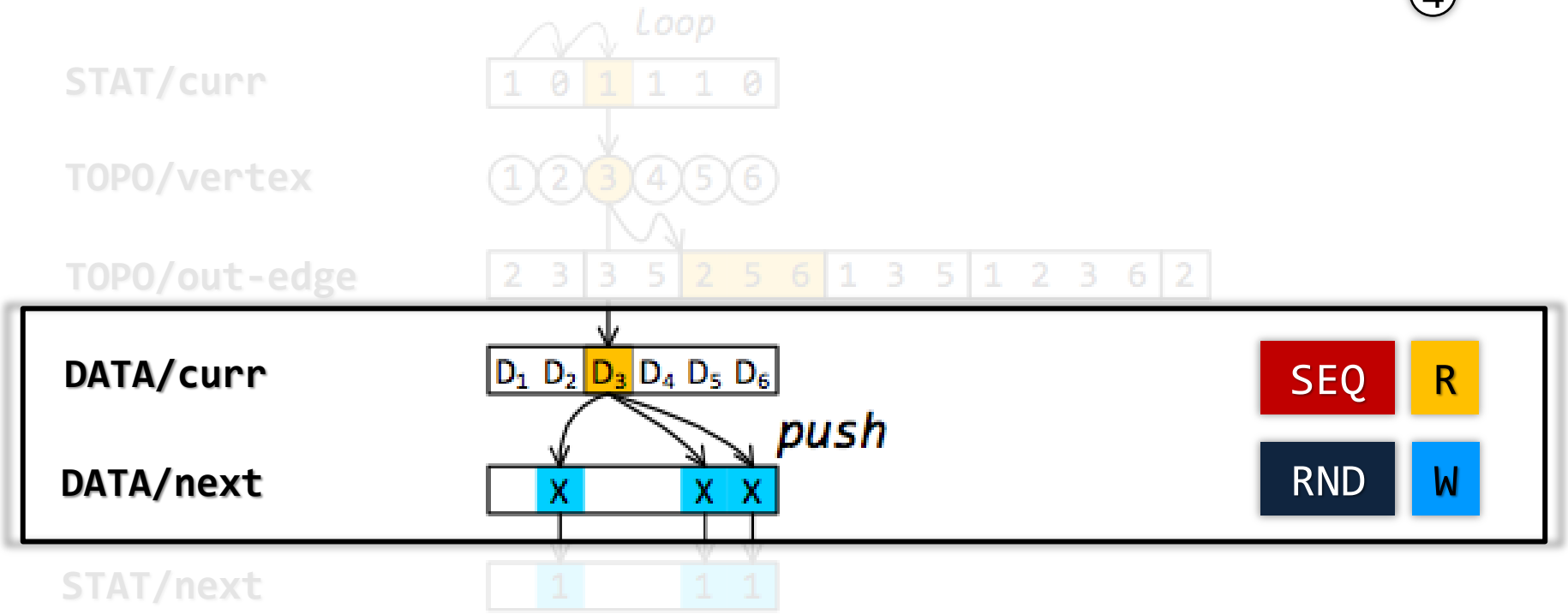
Vertex-centric (e.g. Ligra)



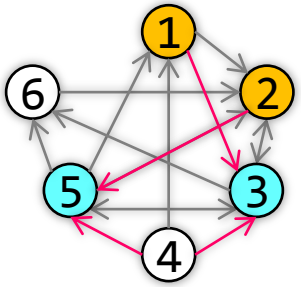


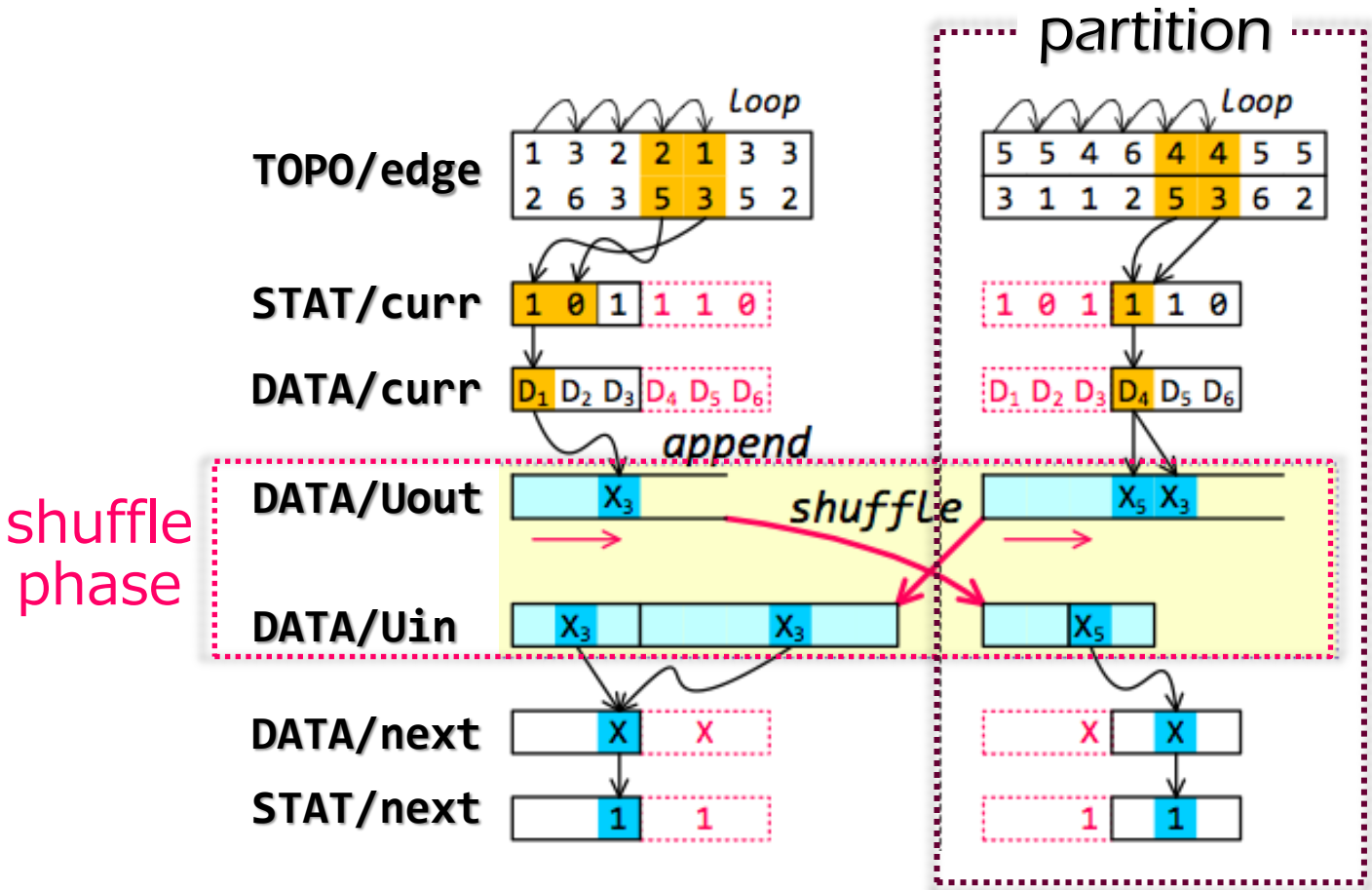
Vertex-centric (e.g. Ligra)



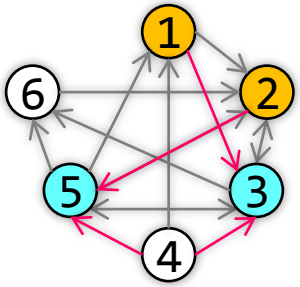


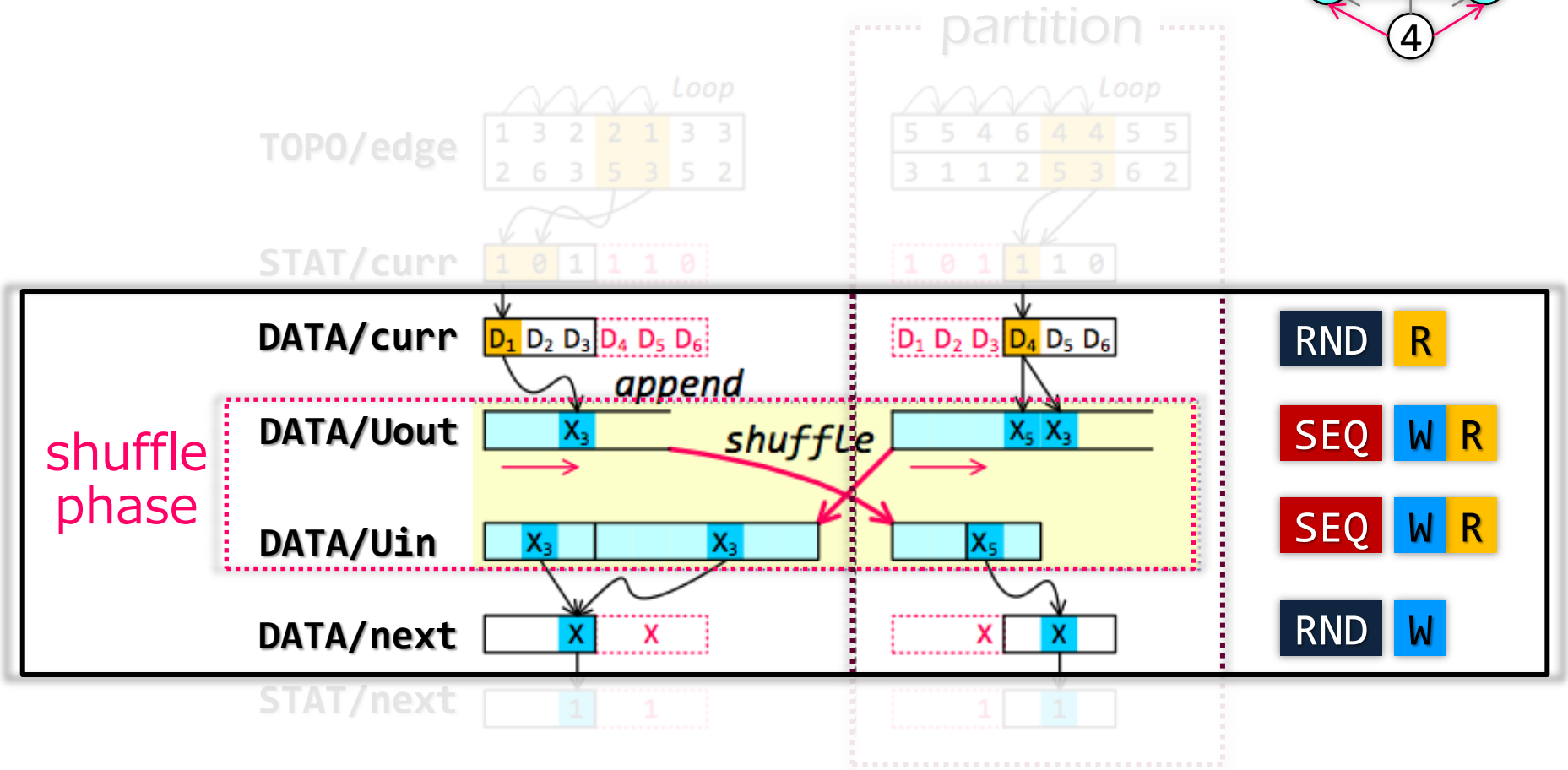
Edge-centric (e.g. X-Stream)



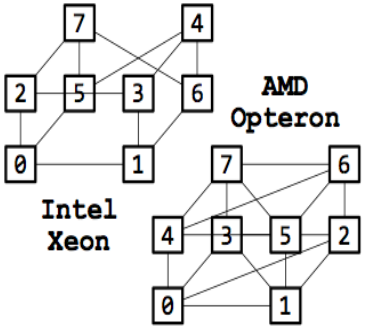


Edge-centric (e.g. X-Stream)





NUMA Characteristics



A commodity NUMA machine

- Multiple processor nodes (i.e., socket)
- Processor = multiple cores + a local DRAM
- A globally shared memory abstract (cache-coherence)
- Hallmark: Non-uniform memory access

Latency (Cycle)

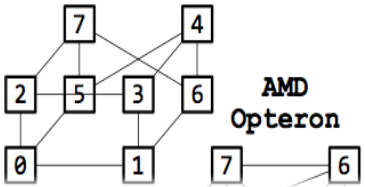
Inst.	0-hop	1-hop	2-hop
80-core Intel Xeon machine			
Load	**117**	271	**372**
Store	**108**	304	**409**

Bandwidth (MB/s)

Access	0-hop	1-hop	2-hop	IL
80-core Intel Xeon machine				
SEQ	3207	2455	**2101**	2333
RAND	**720**	348	**307**	344

NUMA Characteristics

A commodity NUMA machine



- Multiple processor nodes (i.e., socket)
- Processor = multiple cores + a local DRAM
- A globally shared memory abstract (cache-coherence)
- Hallmark: Non-uniform memory access

Sequential remote access is faster than **random local** access

&

Random remote access is **awesome!**

Latency (Cycle)

Inst.	0-hop	1-hop	2-hop
80-core Intel Xeon machine			
Load	**117**	271	**372**
Store	**108**	304	**409**

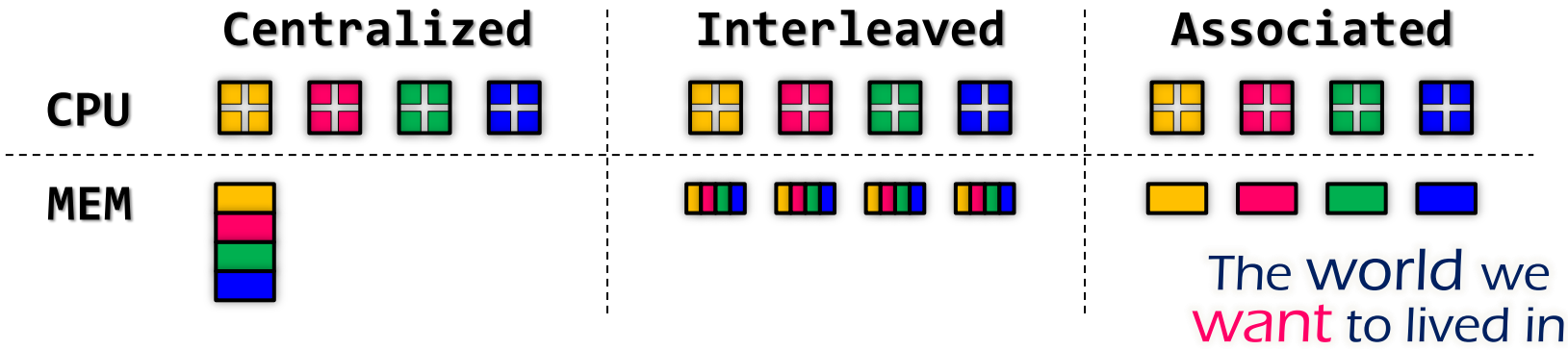
Bandwidth (MB/s)

Access	0-hop	1-hop	2-hop	IL
80-core Intel Xeon machine				
SEQ	3207	2455	**2101**	2333
RAND	**720**	348	**307**	344

NUMA Characteristics

The world we lived in: "first-touch" policy

> "binding virtual pages to physical frames locating on a memory node where a thread first touches the pages"



NUMA Characteristics

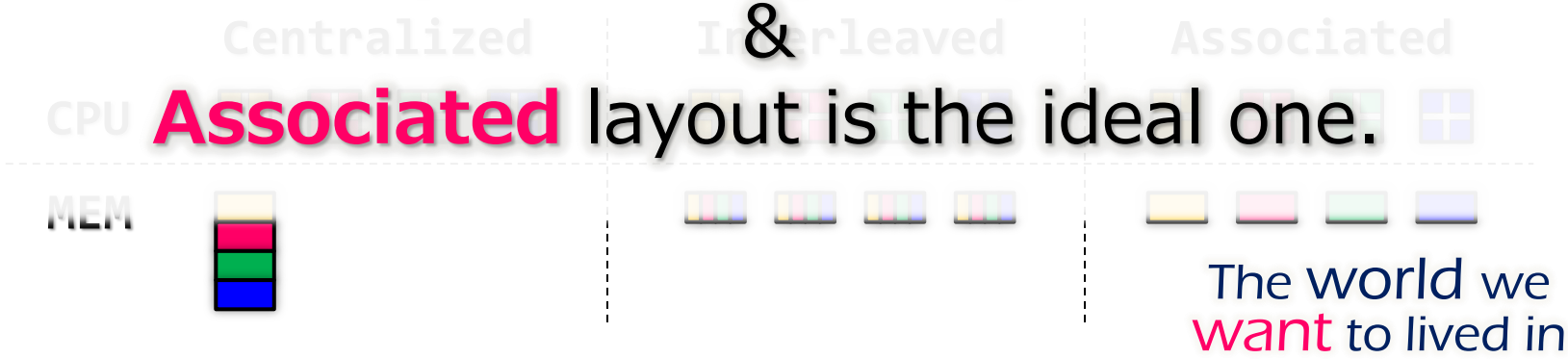
The world we lived in: "first-touch" policy

> "binding virtual pages to physical frames locating on a memory node where a thread first touches the pages"

Both **centralized** and **interleaved** data layout will hamper **locality** and **parallelism**

&

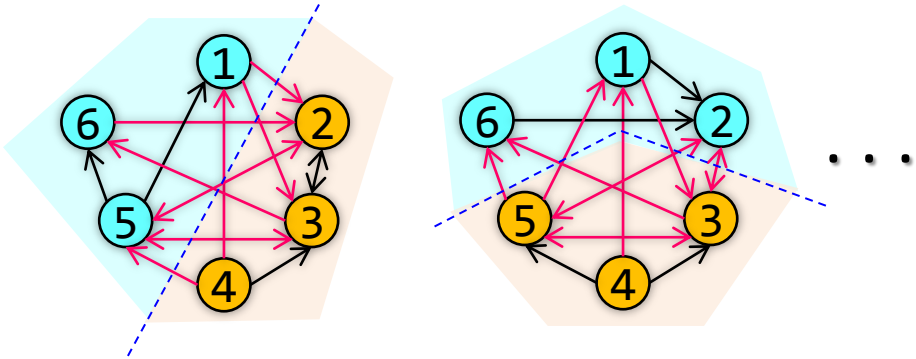
Associated layout is the ideal one.



NUMA Characteristics

Lack of locality (access neighboring vertices)

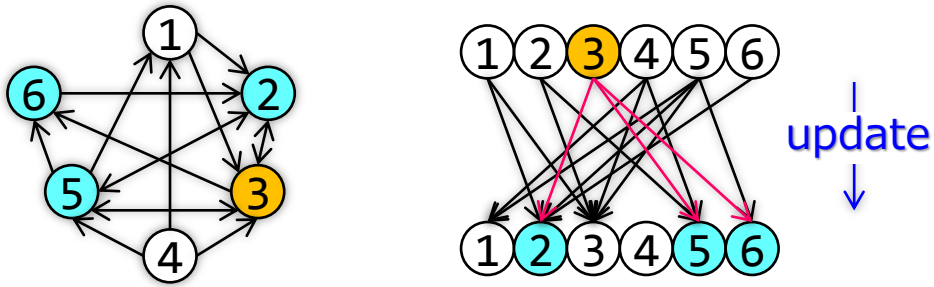
- It is inevitable to access remote memory



How to mix ??

	SEQ	RND
Local		
Global		

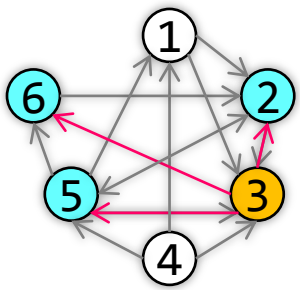
- Random access is always there

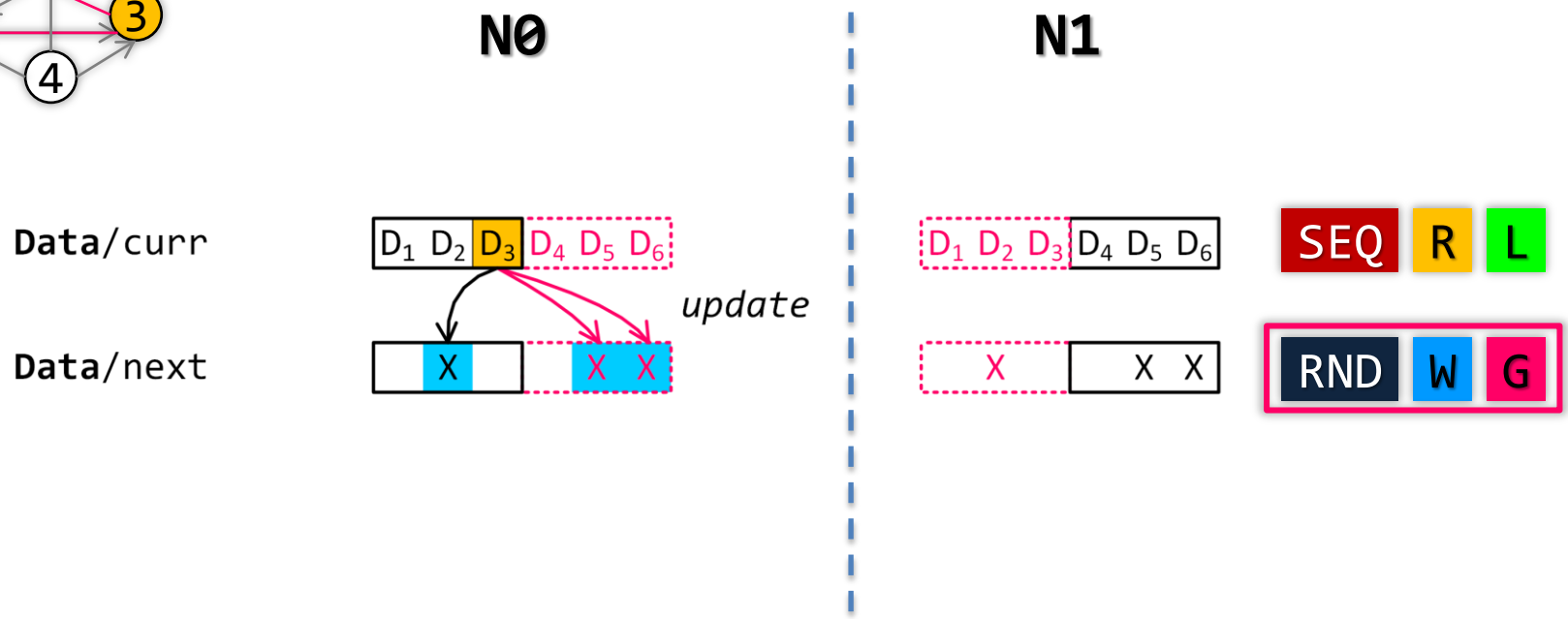


Access Strategy on NUMA

Vertex-centric Model

- Completely overlooked (e.g. Ligra)

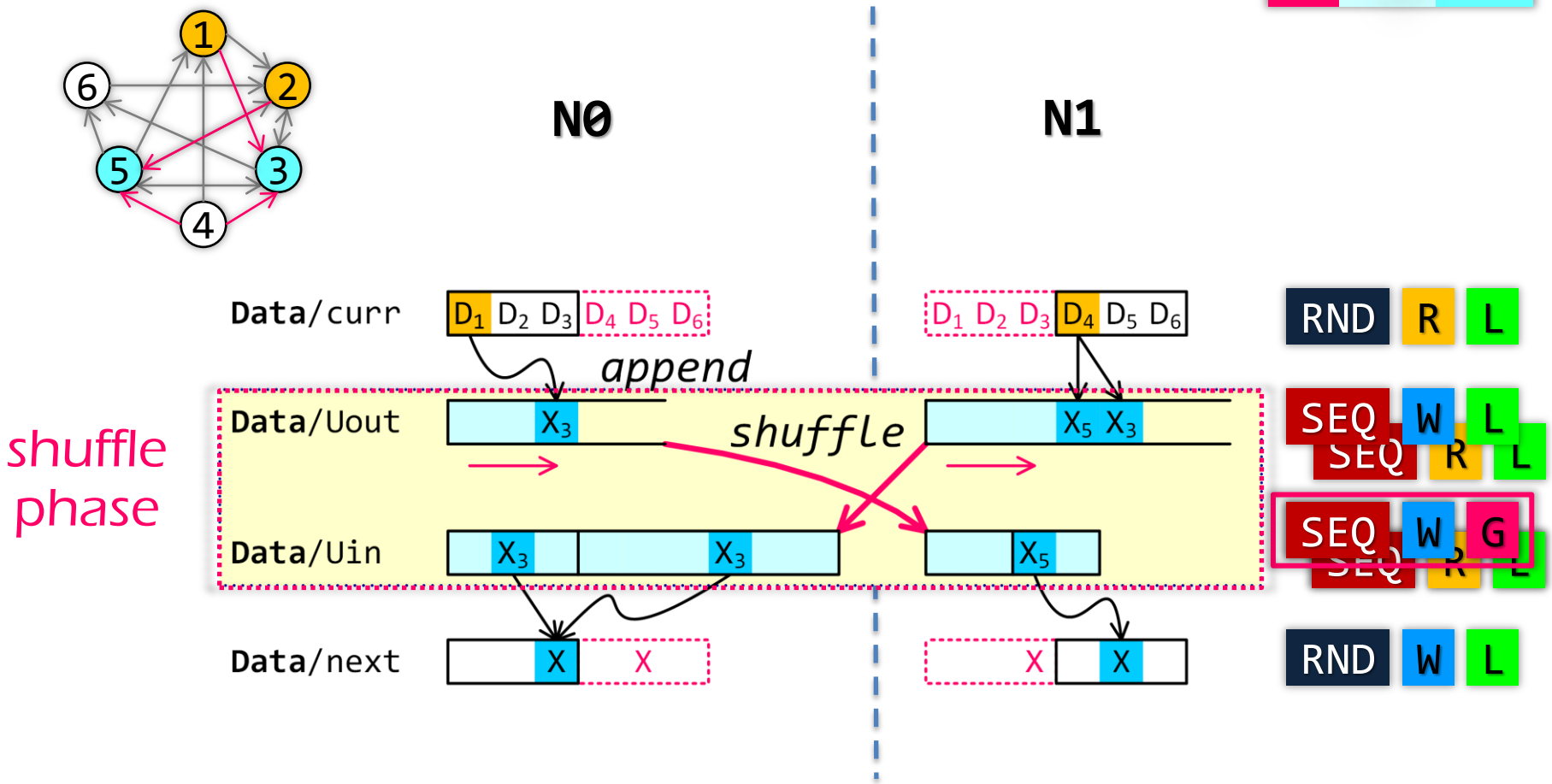




Access Strategy on NUMA

Edge-centric Model

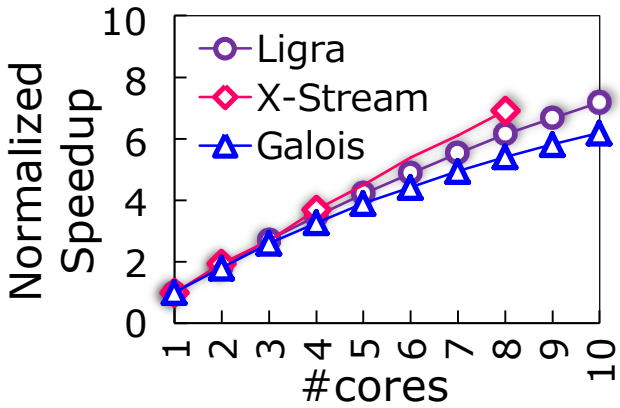
- Inefficient way (e.g. X-Stream)

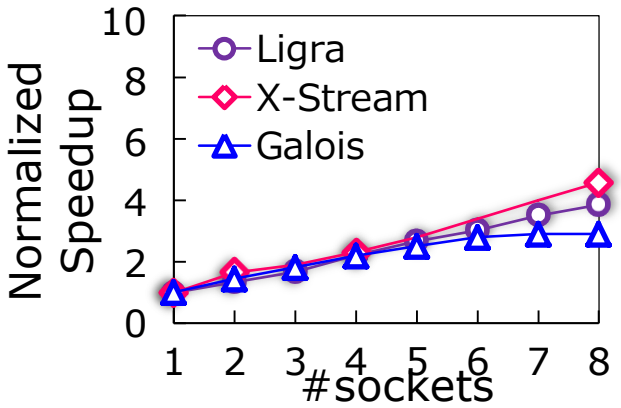


Scalability & Performance on NUMA

Scalability: #Cores vs. #sockets

Intel 80-cores (8Sx10C)





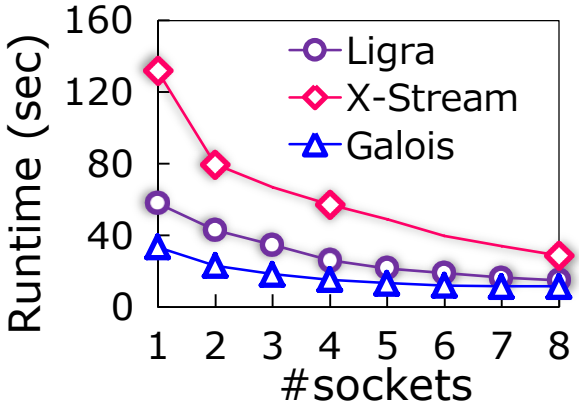
X-Stream
8C: 6.92X
8S: 4.58X

Galois
10C: 6.19X
8S: 2.90X

Performance (sec)

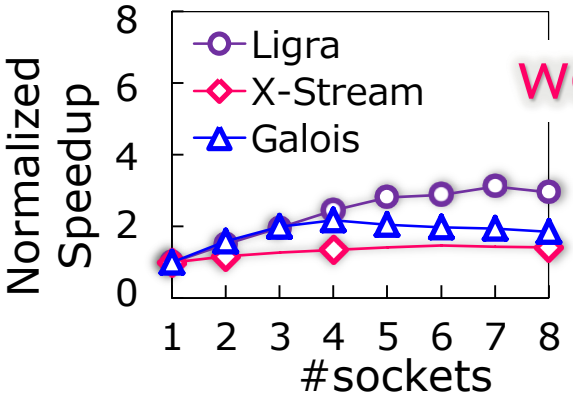
X-Stream
1S: 132s
8S: 29s

Galois
1S: 33s
8S: 12s



Intel 80-cores (8Sx10C)

Scalability: #sockets



worse !

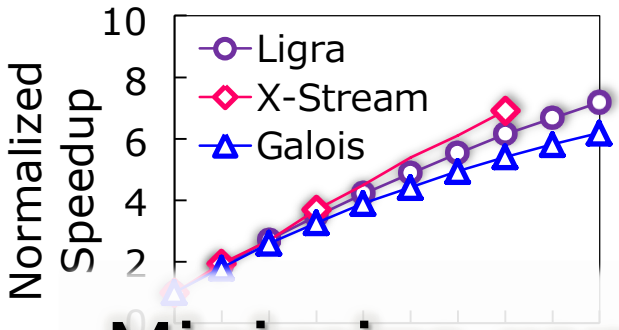
8 Socket
LG: 2.9X
XS: 1.4X

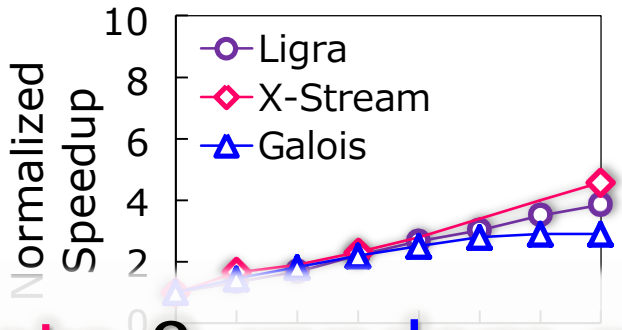
AMD 64-core (8Sx8C)

Scalability & Performance on NUMA

Scalability: #Cores vs. #sockets

Intel 80-cores (8Sx10C)





X-Stream
8C: 6.92X
8S: 4.58X

Galois
10C: 6.19X

Minimize remote & random accesses

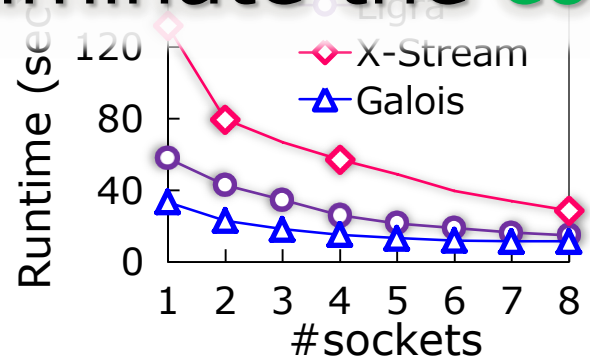
+

Eliminate the combination of them

Performance (sec)

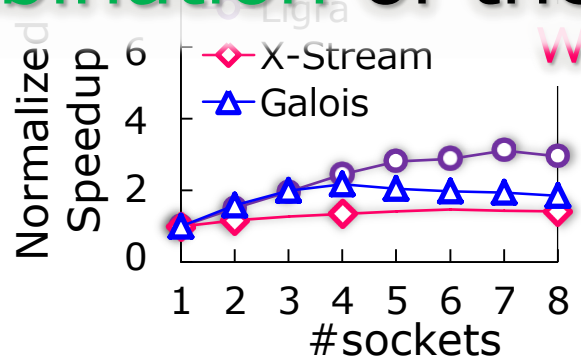
X-Stream
1S: 132s
8S: 29s

Galois
1S: 33s
8S: 12s



Intel 80-cores (8Sx10C)

Scalability: #sockets



worse!

8 Socket
LG: 2.9X
XS: 1.4X

AMD 64-core (8Sx8C)

Outline

Background & Issues

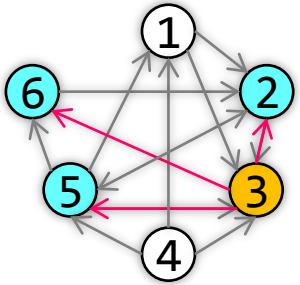
Design of Polymer

Evaluation

Goal#1: Reduce remote accesses

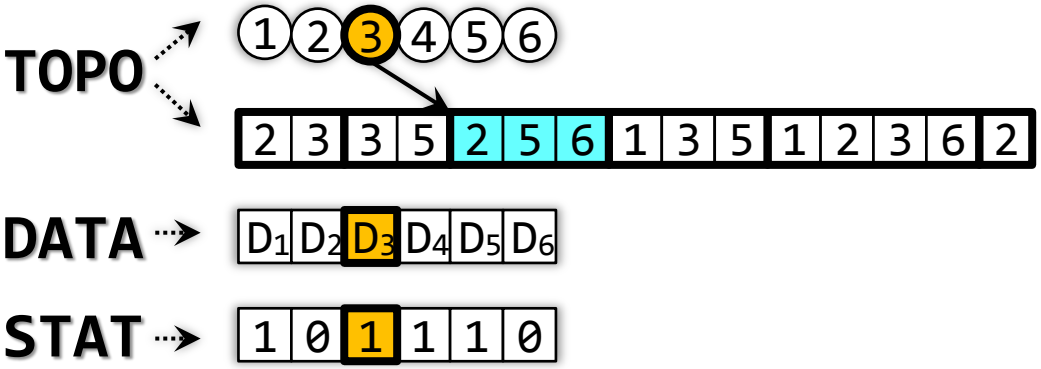
Co-locating data and computation within the same NUMA node

Graph-aware Data Layout

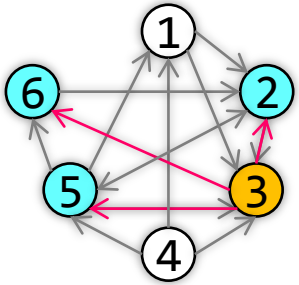


Co-locating data and computation within the same NUMA node

1. Graph-aware **partitioning**

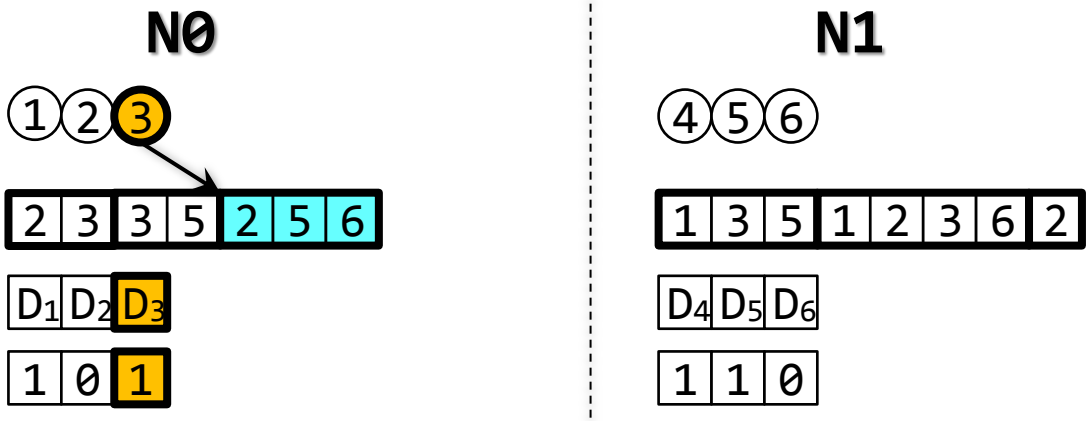


Graph-aware Data Layout



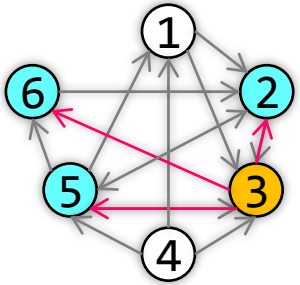
Co-locating data and computation within the same NUMA node

1. Graph-aware **partitioning**



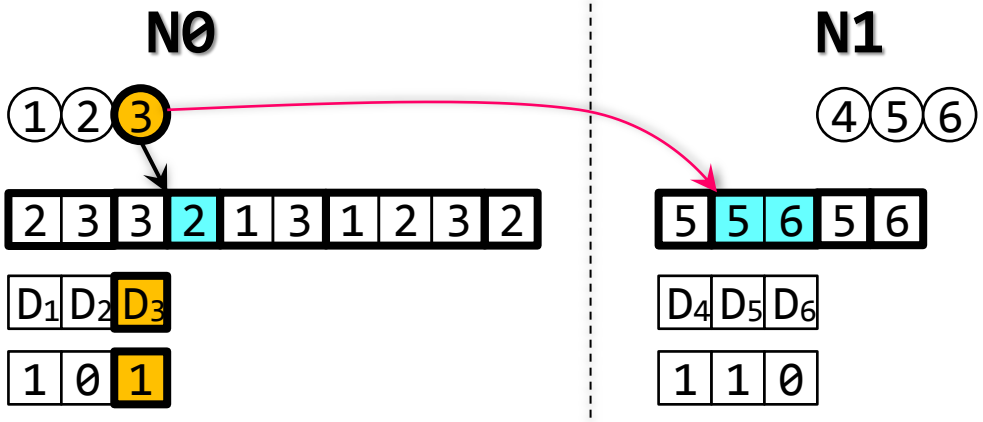
Intuitive

Graph-aware Data Layout



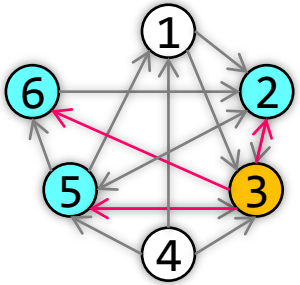
Co-locating data and computation within the same NUMA node

1. Graph-aware **partitioning**



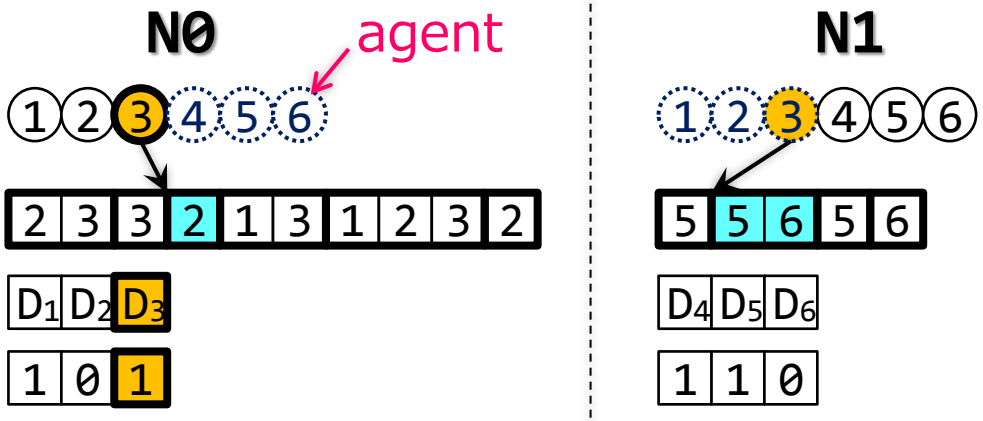
sophisticated

Graph-aware Data Layout



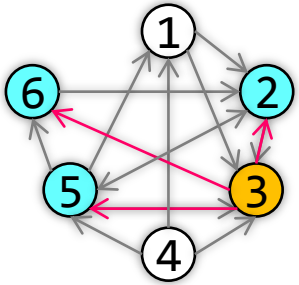
Co-locating data and computation within the same NUMA node

1. Graph-aware **partitioning**



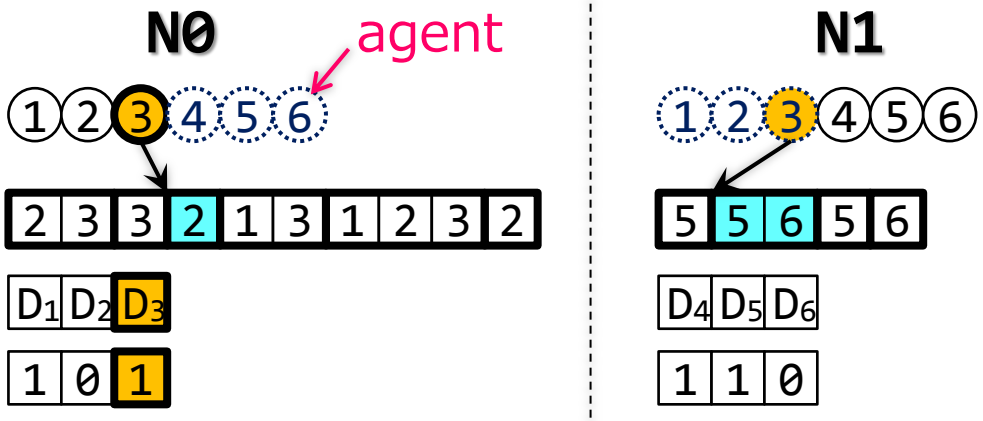
sophisticated

Graph-aware Data Layout

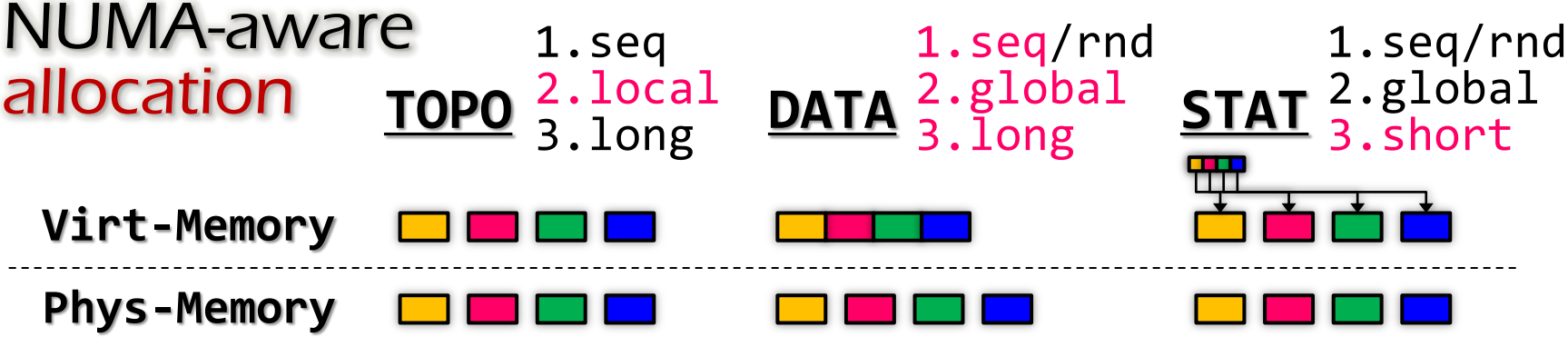


Co-locating data and computation within the same NUMA node

1. Graph-aware partitioning



2. NUMA-aware allocation



Goal#2: Eliminate "random + remote"

Random remote access

→ ~~access neighboring vertices on other nodes~~

distribute the computations on
a singe vertex over multiple nodes

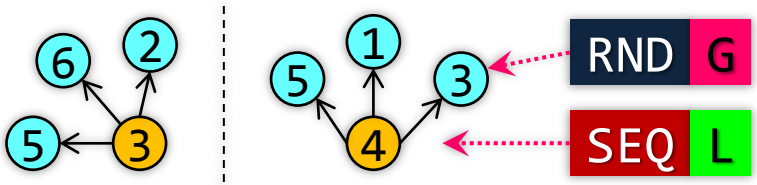
Goal#2: Eliminate "random + remote"

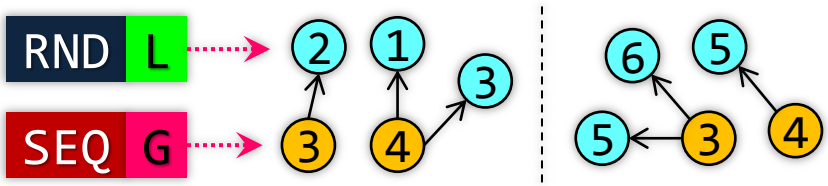
Random remote access

→ ~~access neighboring vertices on other nodes~~

distribute the computations on
a singe vertex over multiple nodes

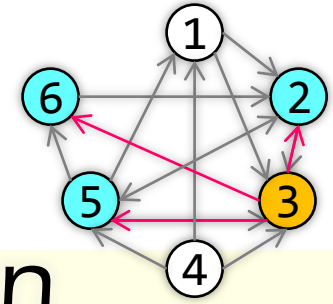
Each node handles
all edges of partial vertices



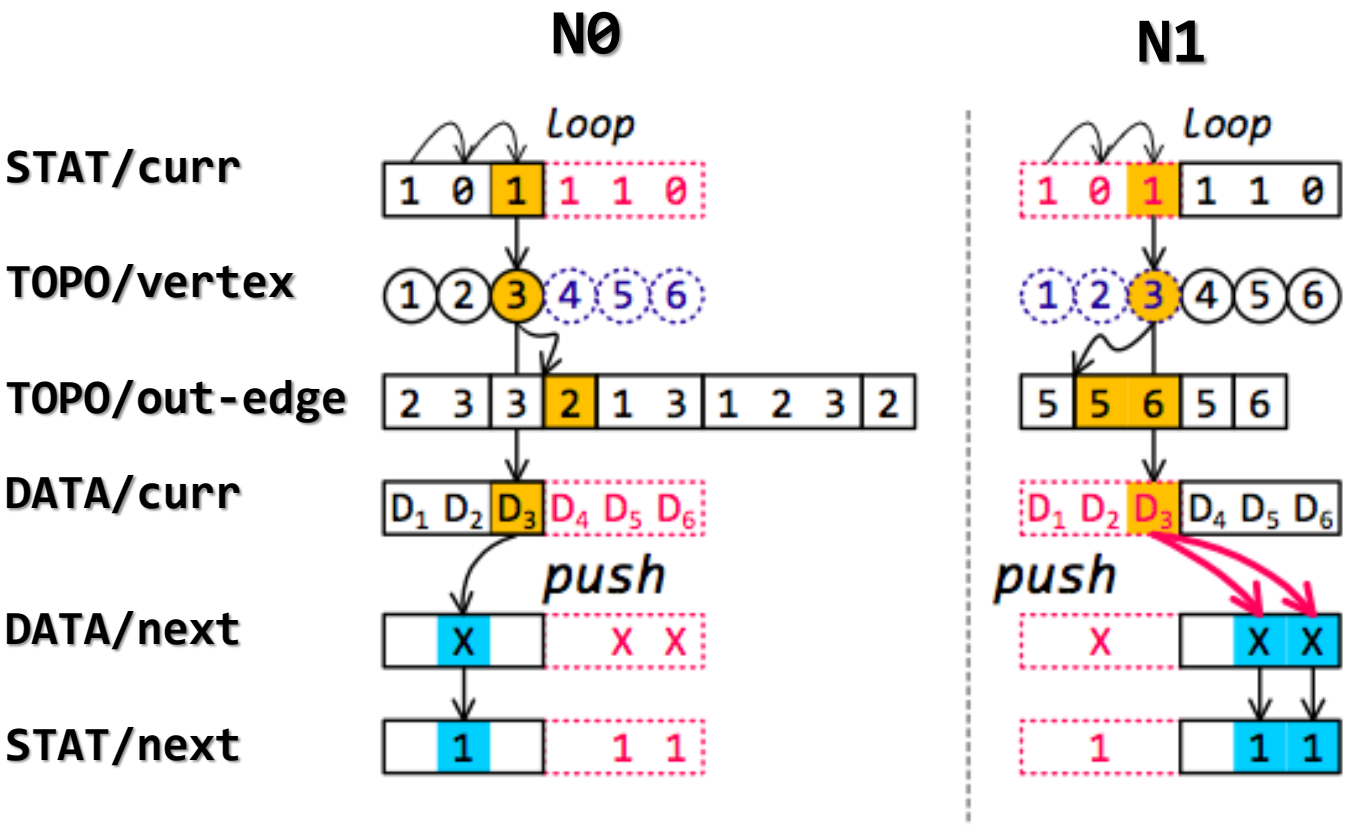


Each node handles
partial edges of all vertices

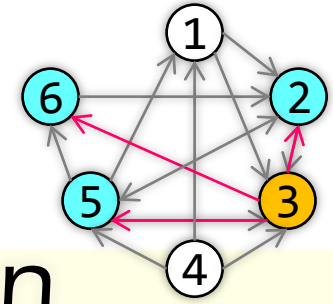
NUMA-aware Access Strategy



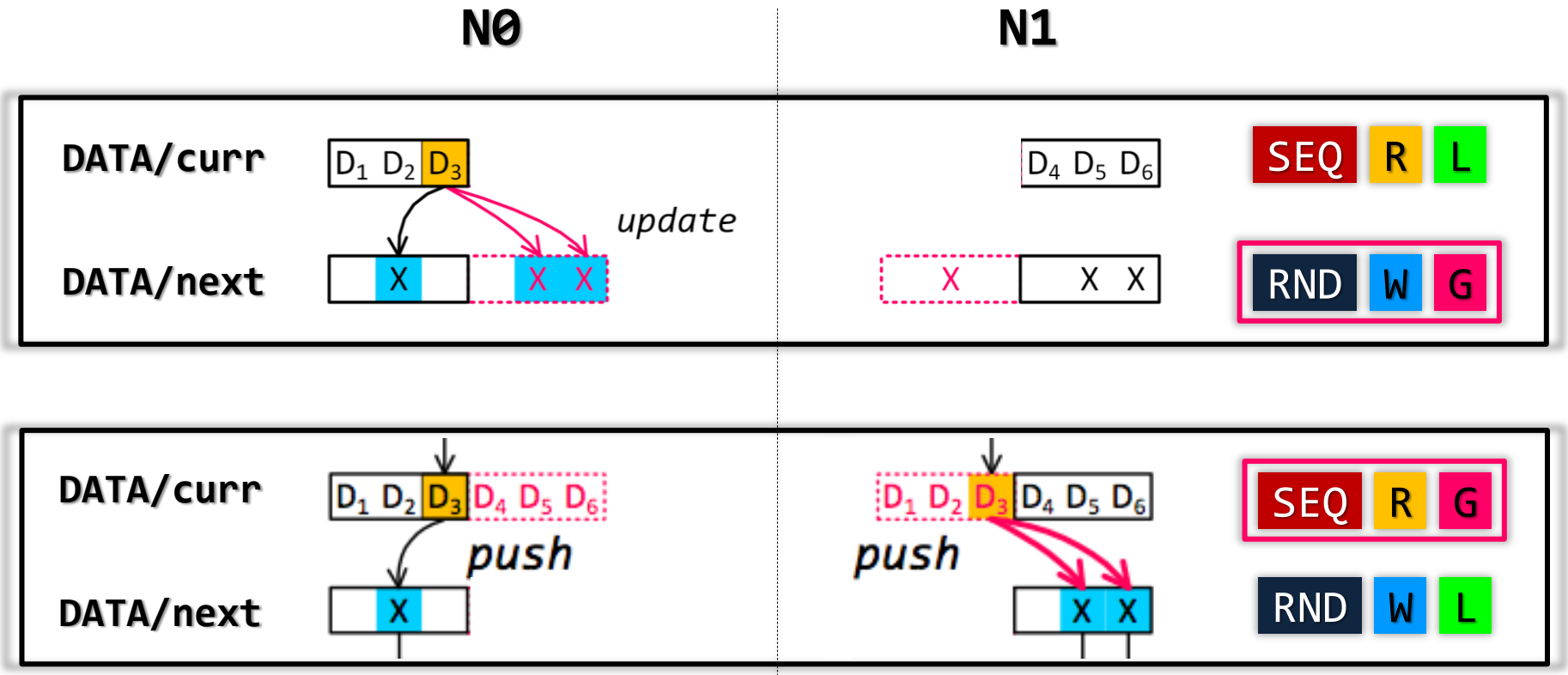
distribute the computations on singe vertex over multiple NUMA-nodes



NUMA-aware Access Strategy

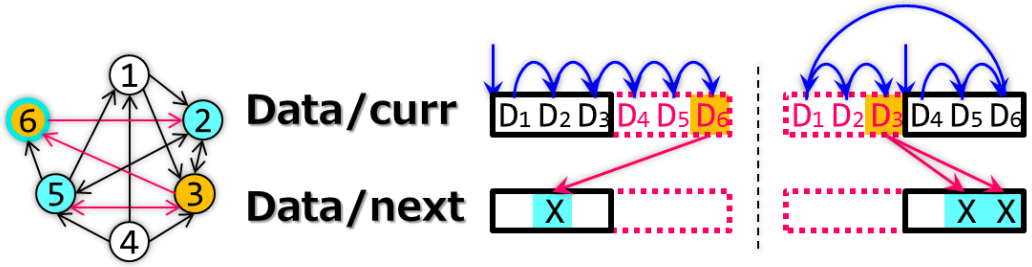


distribute the computations on singe vertex over multiple NUMA-nodes

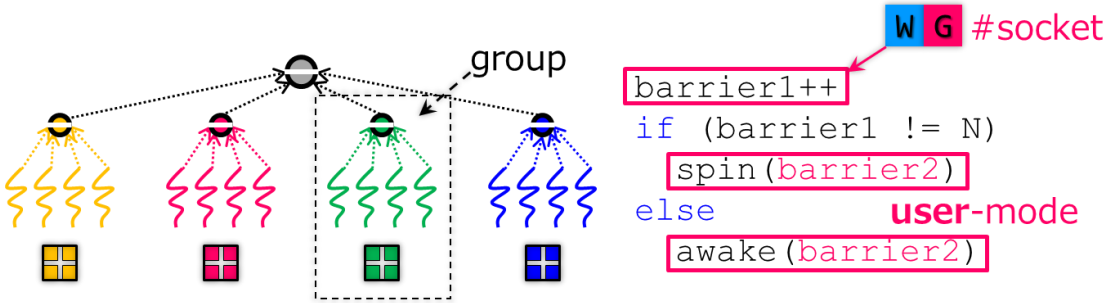


Optimizations

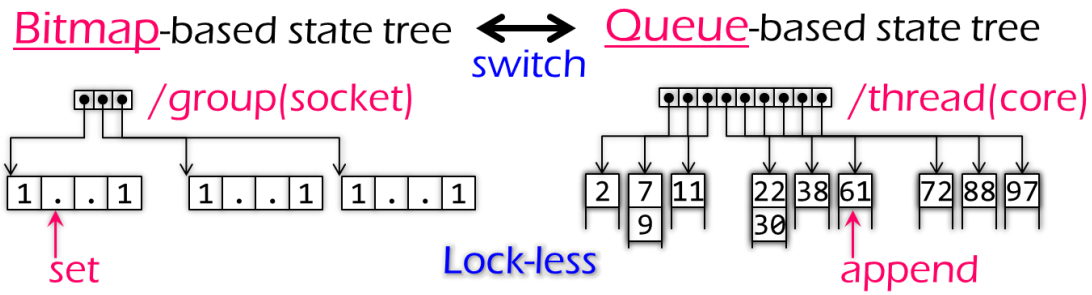
1. Rolling update



2. Hierarchical and efficient barrier



3. Adaptive data structure



Outline

Background & Issues

Design of Polymer

Evaluation

Implementation

Polymer

- ~5,300 SLOCs of C++ code
- Based on scatter-gather iterative model
  - → Support both push and pull mode
  - → Use a synchronous scheduler
- Several typical graph applications
  - → Sparse MM: PageRank, SpMV and BP
  - → Traversal: BFS, CC and SSSP

Open source:

http://ipads.se.sjtu.edu.cn/projects/polymer.html

Experiment Settings

Baseline: Ligra v2014.3, X-Stream v0.9, and Galois v2.2

Platform Intel/8X10 AMD/8X8

- 80-core Intel machine (w/o Hyper-Threading)
  8 sockets (E7-8850: 10 cores and 128 GB local RAM)
- 64-core AMD machine (also 8 sockets)

Algorithms (6)

- 3 Spars MM algorithms
- 3 Traversal algorithms

Graphs (4)

- 2 real-world and
  2 synthetic graphs

Graph	|V|	|E|
Twitter	41.7 M	1.47B
rMat27	134.2M	2.14B
Powerlaw	10.0M	105M
roadUS	23.9M	58M

Overall Performance

Algo.	|V|	Polymer	Ligra	X-Stream	Galois
PR	Twitter	5.3	15.0	28.9	11.6
	rMat27	9.6	28.0	18.2	19.6
	Power	1.6	30.5	6.1	6.6
	roadUS	1.2	2.3	2.8	1.4
SpMV	Twitter	7.6	29.0	59.6	11.7
	rMat27	19.2	54.3	52.5	41.9
	Power	1.8	31.0	5.5	6.2
	roadUS	1.3	2.8	3.0	3.6
BP	Twitter	38.0	63.1	2017	57.1
	rMat27	58.3	92.8	737	75.0
	Power	8.0	30.7	38.3	8.6
	roadUS	5.2	2.6	20.0	7.1

Overall Performance

Algo.	|V|	Polymer	Ligra	X-Stream	Galois
PR	Twitter	5.3	2.85X	5.48X	2.19X
	rMat27	9.6	2.91X	1.89X	2.04X
	Power	1.6	18.9X	3.76X	4.11X
	roadUS	1.2	1.92X	2.31X	1.14X
SpMV	Twitter	7.6	3.84X	7.89X	1.55X
	rMat27	19.2	2.83X	2.74X	2.19X
	Power	1.8	17.3X	3.08X	3.46X
	roadUS	1.3	2.18X	2.30X	2.74X
BP	Twitter	38.0	1.66X	53.1X	1.50X
	rMat27	58.3	1.59X	12.6X	1.29X
	Power	8.0	3.80X	4.74X	1.06X
	roadUS	5.2	2.01X	3.84X	1.36X

Overall Performance

Algo.	|V|	Polymer	Ligra	X-Stream	Galois
PR	Twitter	5.3	2.85X ↑	5.48X ↑	2.19X ↑
	rMat27	9.6	2.91X ↑	1.89X ↑	2.04X ↑
	Power	1.6	18.9X ↑	3.76X ↑	4.11X ↑
	roadUS	1.2	1.92X ↑	2.31X ↑	1.14X ↑
SpMV	Twitter	7.6	3.84X ↑	3.80X ↑	1.55X ↑
	rMat27	15.2	2.83X ↑	2.74X ↑	2.19X ↑
	Power	[illegible]	[illegible]	[illegible]	3.46X ↑
	roadUS	1.3	2.18X ↑	2.30X ↑	2.74X ↑
BP	Twitter	38.0	1.66X ↑	53.1X ↑	1.50X ↑
	rMat27	58.3	1.59X ↑	12.6X ↑	1.29X ↑
	Power	8.0	3.80X ↑	4.74X ↑	1.06X ↑
	roadUS	5.2	2.01X ↓	3.84X ↑	1.36X ↑

1.06X ~ 3.08X

Compared with best cases

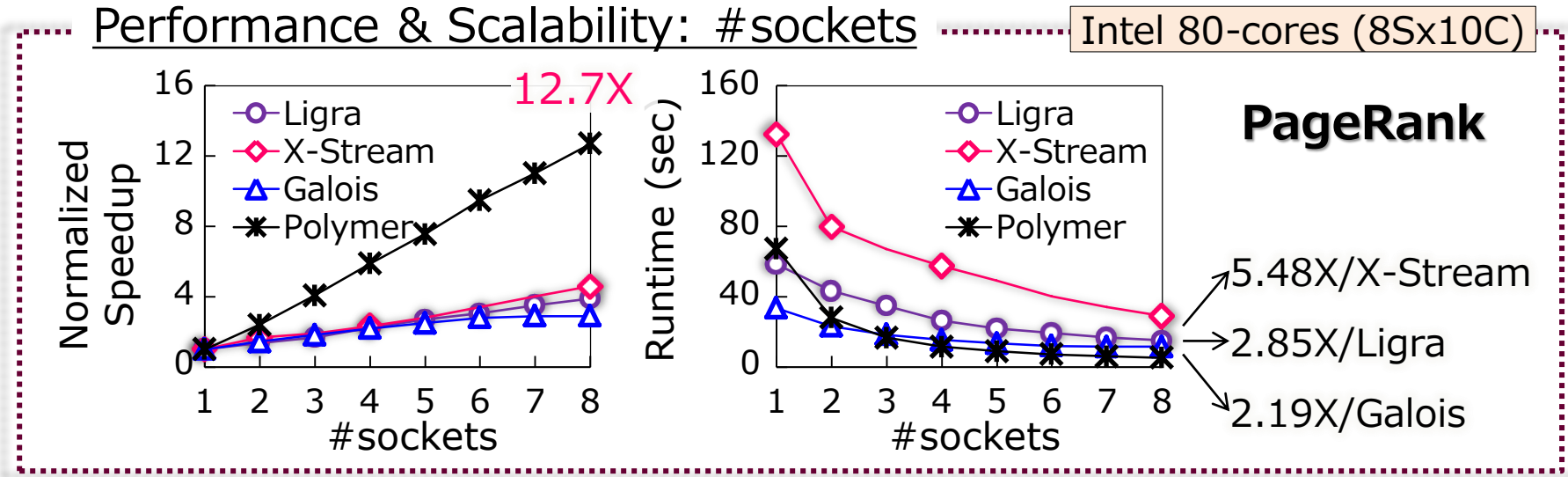
Overall Performance

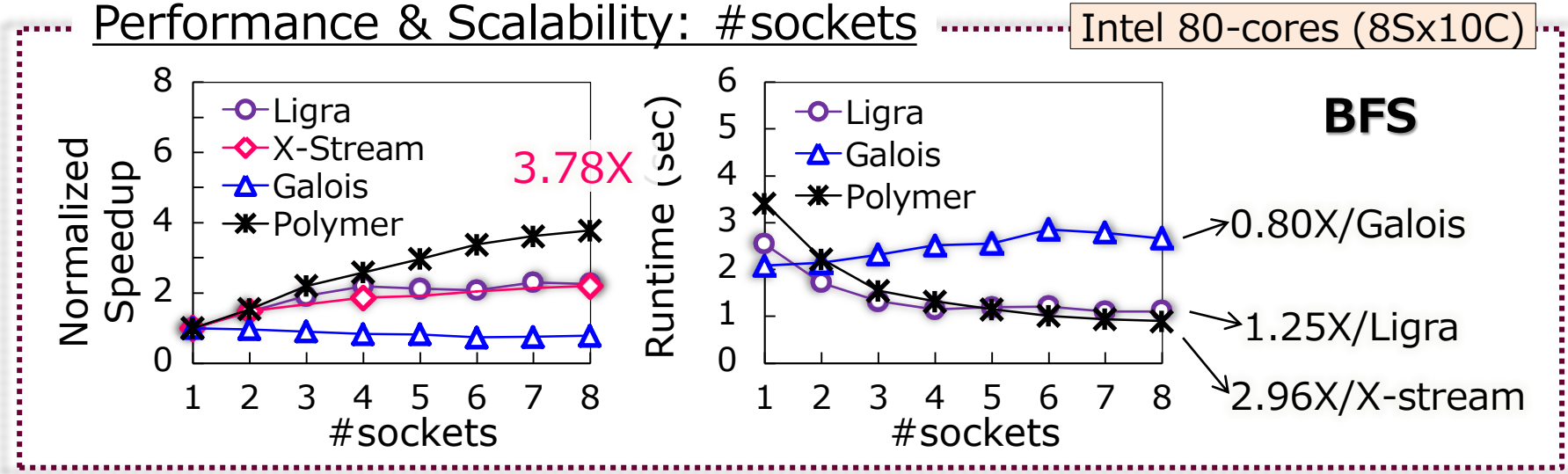
Algo.	|V|	Polymer	Ligra	X-Stream	Galois
BFS	Twitter	0.98	1.02	34.39	2.45
	rMat27	1.56	1.86	30.18	2.54
	Power	0.36	0.39	2.58	0.36
	roadUS	1.16	6.93	557.7	5.01
CC	Twitter	4.60	5.51	54.8	31.9
	rMat27	8.72	7.74	40.0	33.9
	Power	1.23	2.56	5.13	3.51
	roadUS	57.5	63.2	985	1.18*
SSSP	Twitter	2.26	3.17	165	26.3
	rMat27	5.78	5.26	17.9	28.5
	Power	0.85	1.12	126	26.6
	roadUS	341	338	1225	0.33*

Overall Performance

Algo.	|V|	Polymer	Ligra	X-Stream	Galois
BFS	Twitter	0.98	1.07X ↑	35.9X ↑	2.56X ↑
	rMat27	1.56	1.19X ↑	19.4X ↑	1.63X ↑
	Power	0.36	1.08X ↑	7.16X ↑	1.02X ↑
	roadUS	1.16	5.97X ↑	481.X ↑	4.32X ↑
CC	Twitter	4.60	1.20X ↑	11.9X ↑	6.94X ↑
	rMat27	8.72	1.12X ↓	4.58X ↑	3.88X ↑
	Power	1.23	2.08X ↑	4.17X ↑	2.85X ↑
	roadUS	57.5	1.10X ↑	17.1X ↑	1.18*
SSSP	Twitter	2.26	1.40X ↑	73.1X ↑	11.5X ↑
	rMat27	5.78	1.09X ↓	21.9X ↑	4.93X ↑
	Power	0.85	1.31X ↑	14.5X ↑	31.1X ↑
	roadUS	341	1.01X ↓	3.59X ↑	0.33*

Scalability on NUMA





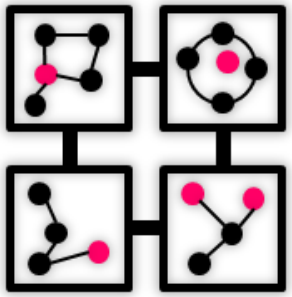
Conclusion

Polymer: NUMA-aware Graph-structured Analytics

- A comprehensive analysis that uncovers several NUMA characteristics and issues with existing NUMA-oblivious graph analytics systems
- A new system that exploits both NUMA- and graph-aware data layout and memory access strategies
  - → minimize remote access
  - → eliminate remote random access
- Three optimizations for global synchronization efficiency, load balance and data structure flexibility

Thanks

Polymer



http://ipads.se.sjtu.edu.cn/projects/polymer.html

Institute of Parallel and Distributed Systems

Questions