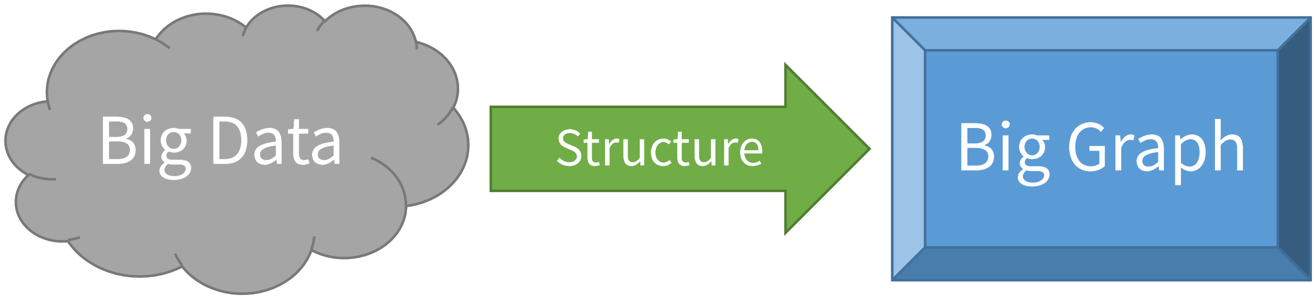
NXgraph: An Efficient Graph Processing System on a Single Machine

Yuze Chi[1], Guohao Dai[1], Yu Wang[1], Guangyu Sun[2], Guoliang Li[1] and Huazhong Yang[1]

Tsinghua National Laboratory for Information Science and Technology[1], Tsinghua University

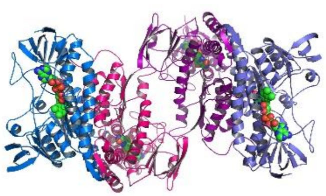
Center for Energy Efficient Computing and Applications[2], Peking University

Motivation



social media

science



advertising

web

Motivation

Can we exploit it well?

- Computation capacity of a single CPU
  - Intel i7-5820K: 12 hyper-threads, 3.3GHz
  - Assume 50 clock cycles/edge: **792MEPS** (Million Edges Per Second)

System	Throughput/MEPS (PageRank on Twitter)	Notes
Spark[HotCloud2010]	15 in total/0.15 each	100 CPUs in 50 nodes
PowerGraph[OSDI2012]	408 in total/6.4 each	64 CPUs in 64 nodes
GraphChi[OSDI2012]	50	
TurboGraph[SIGKDD2013]	108	
X-stream[SOSP2013]	20	no pre-processing
VENUS[ICDE2015]	15.4	on HDD
GridGraph[ATC2015]	61	

- Large gap!
- Our objective: **higher MEPS/CPU**

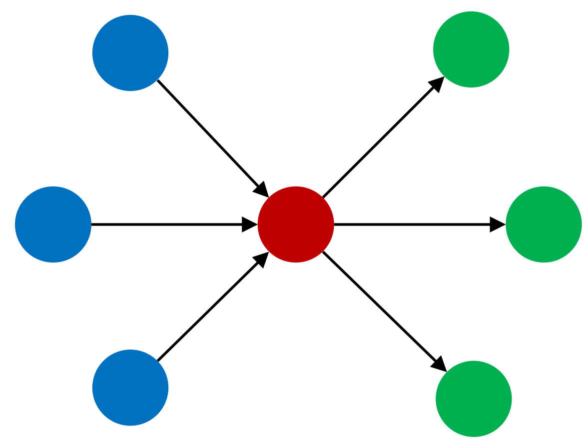
Vertex-centric Model[SIGMOD2010]

Update my neighbors...

- Graph G = (V,E)
  - vertex v = (id, at
  - edge e = (src vid, ds onal attribute)
- "Think like a vertex"

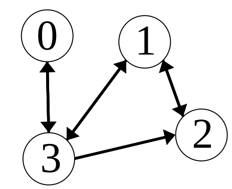
- Example: Breadth-First Search (BFS)

```
for each dst in my.out_edges
  if dst.depth > my.depth+1
  then
    dst.depth = my.depth+1
```

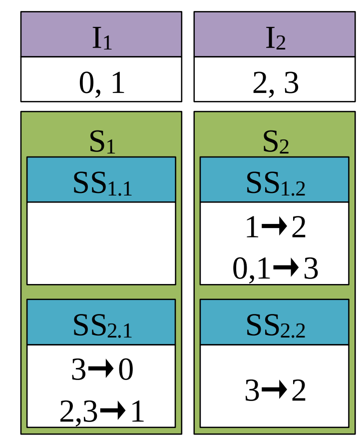


Interval-shard Partitioning[OSDI2012]

- Vertices → Intervals
- Edges → Shards
  - Edges in each shard → Sub-Shards

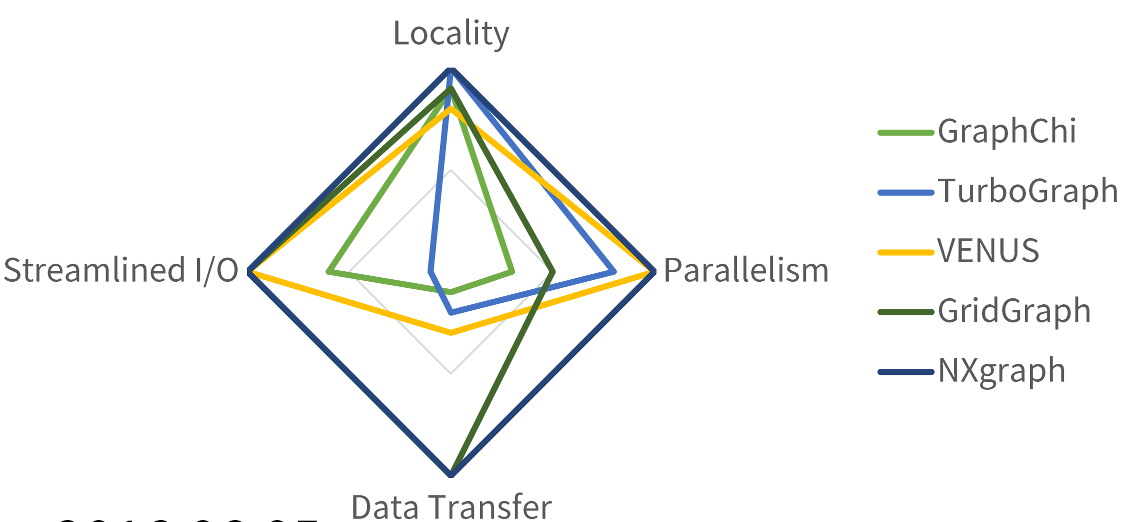


- Objective
  - Limit memory access to a small region
  - Improve locality



Four Optimizing Rules

	GraphChi OSDI2012	TurboGraph SIGKDD2013	VENUS ICDE2015	GridGraph ATC2015	NXgraph ICDE2016
1. Exploit the locality of graph data	☺	☺	☺	☺	☺
2. Utilize the parallelism of multi-thread CPU		☺	☺		☺
3. Reduce the amount of disk data transfer				☺	☺
4. Streamline the disk I/O	☺		☺	☺	☺



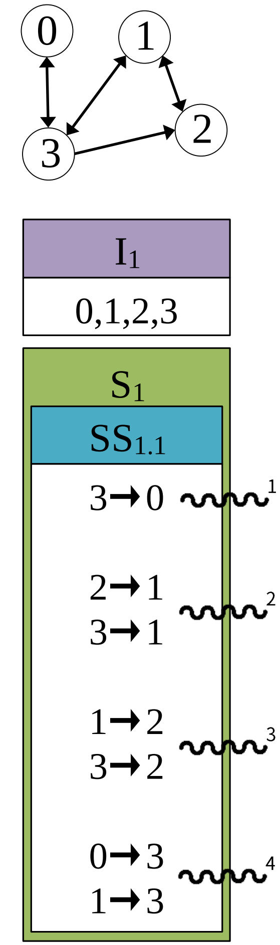
No previous system considers them all

Our Effort

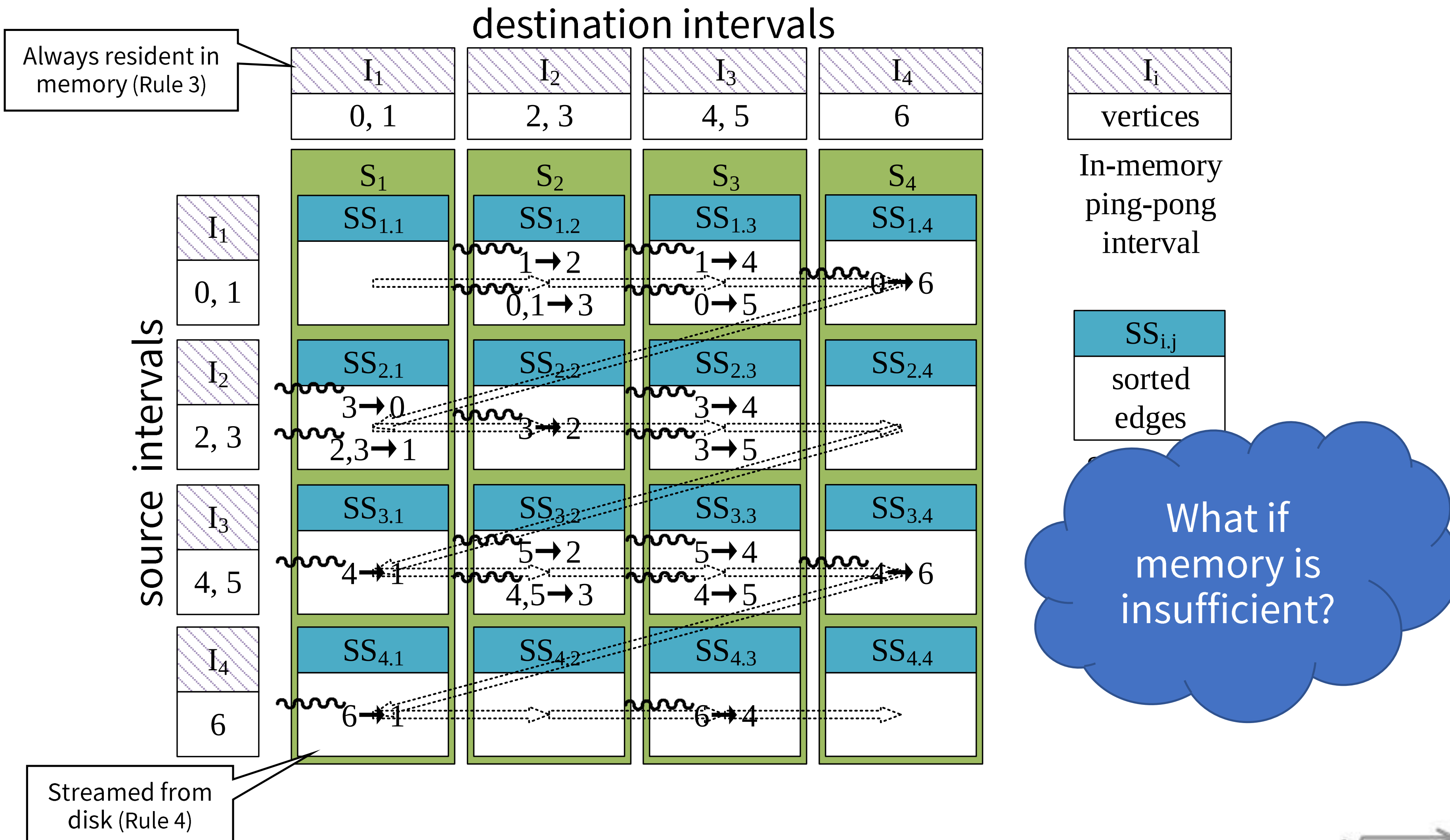
- **Follows all four optimizing rules**
- According to optimizing rule 1 & 2, we design
  - Destination-Sorted Sub-Shard (DSSS) structure
- According to optimizing rule 3 & 4, we design
  - Adaptive updating strategies
    - Single-Phase Update (SPU)
    - Double-Phase Update (DPU)
    - Mixed-Phase Update (MPU)

Destination-Sorted Sub-Shards

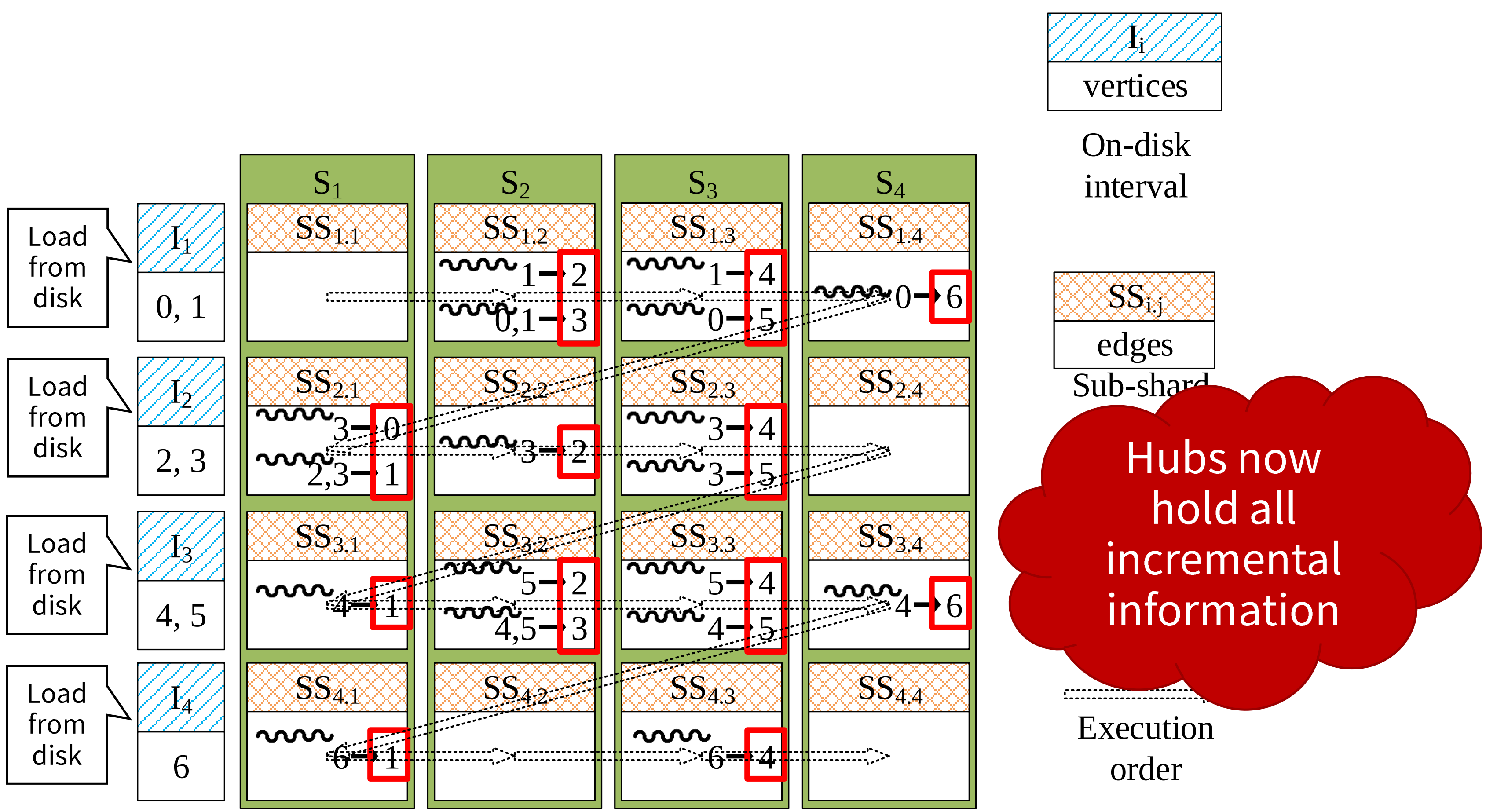
- Sort edges in each sub-shard
- Sequential reads from source interval (Rule 1)
  - Potentially improves cache hit rate
- Parallel writes to destination vertices (Rule 2)
  - No write conflict among threads
- Defines behavior **inside** each sub-shard



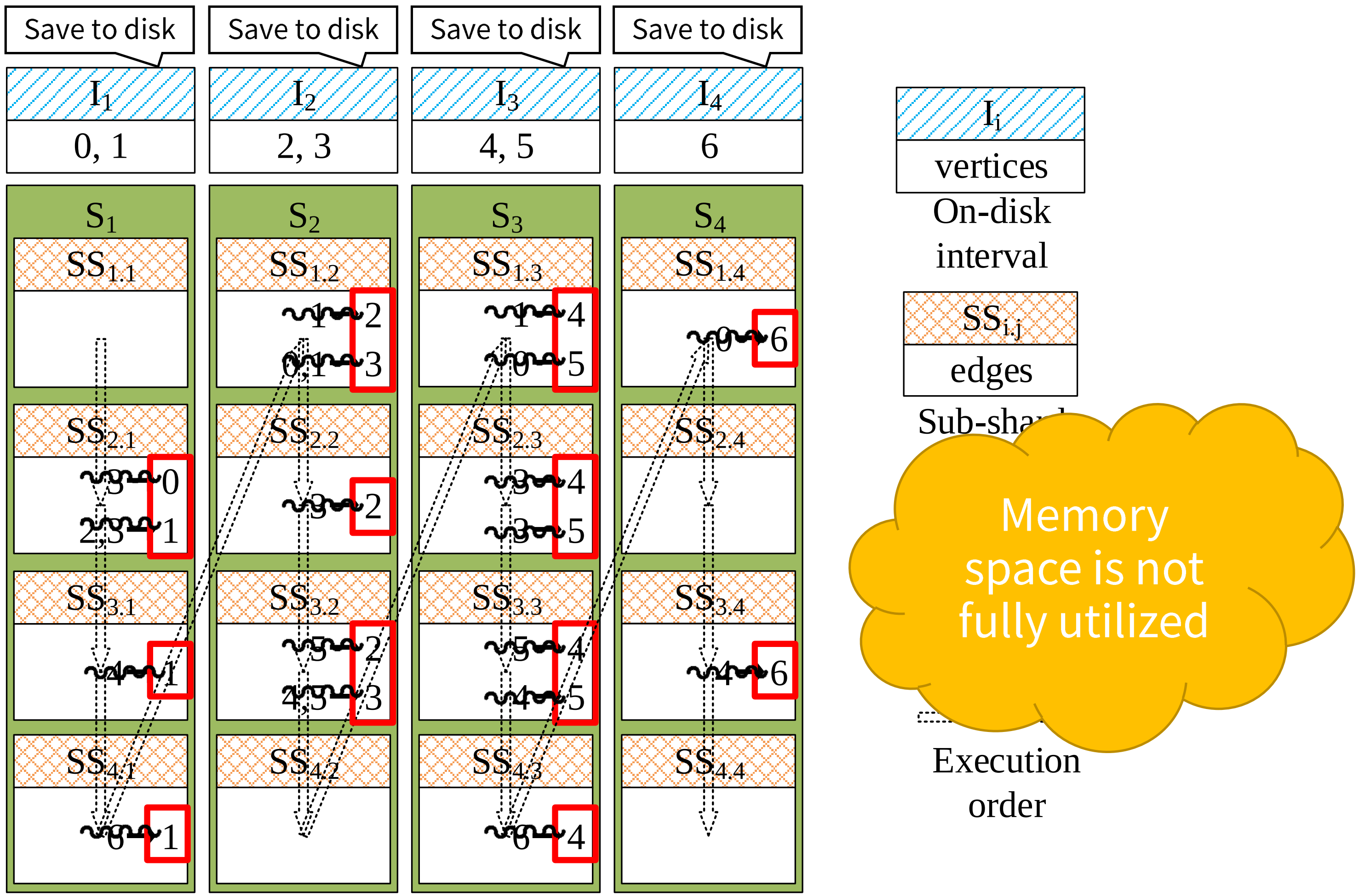
Single-Phase Update



Double-Phase Update (To-Hub Phase)

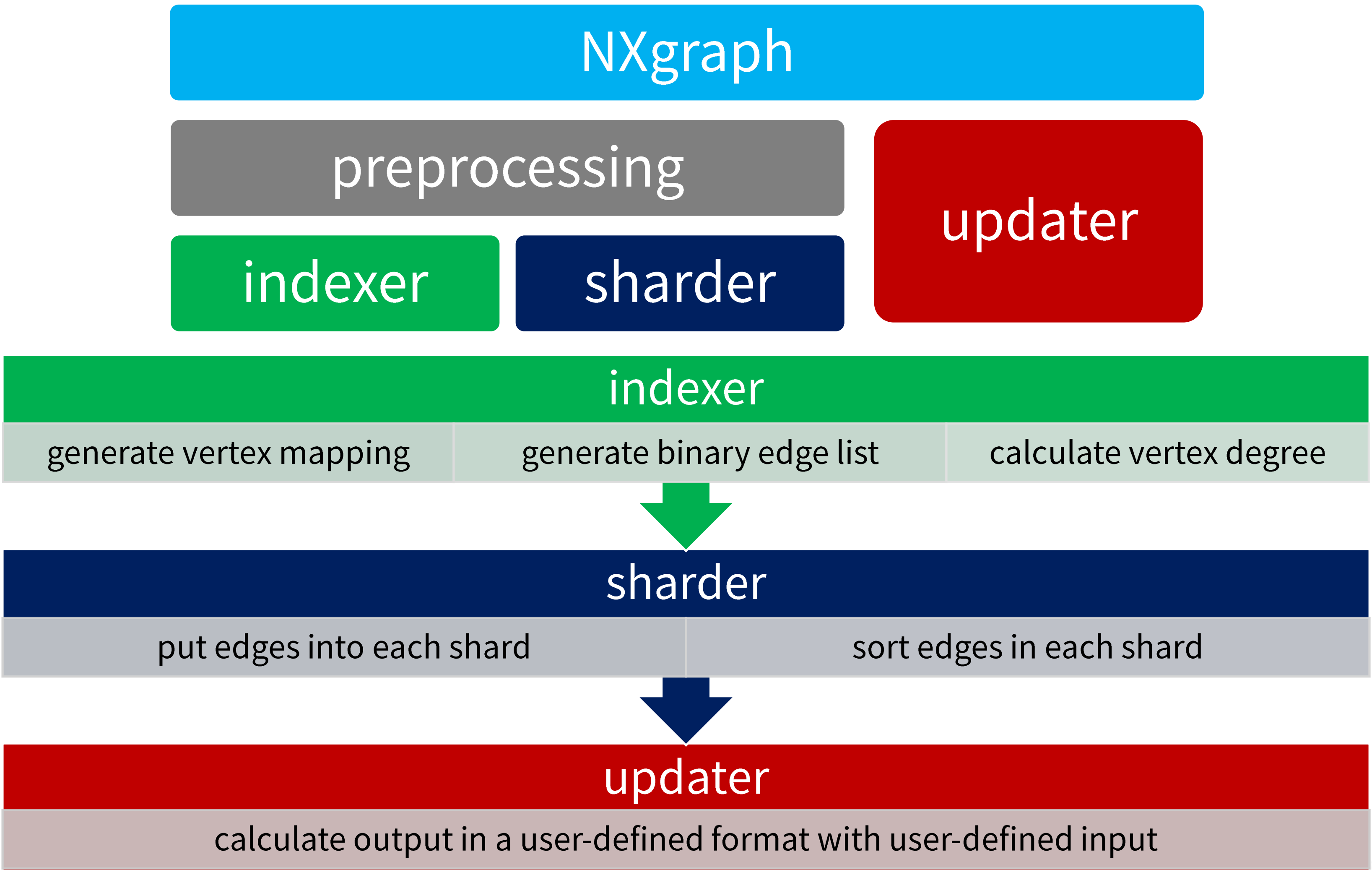


Double-Phase Update (From-Hub Phase)



Mixed-Phase Update

System Architecture



Results

- Evaluation platform
  - Hex-core hyper-threading Intel i7-5820K CPU @ 3.3GHz
  - 8x8G DDR4 RAM, 2x128G RAID0 SSD, 1T HDD
  - Ubuntu 14.04 LTS 64bit/Windows 10 Edu 64bit

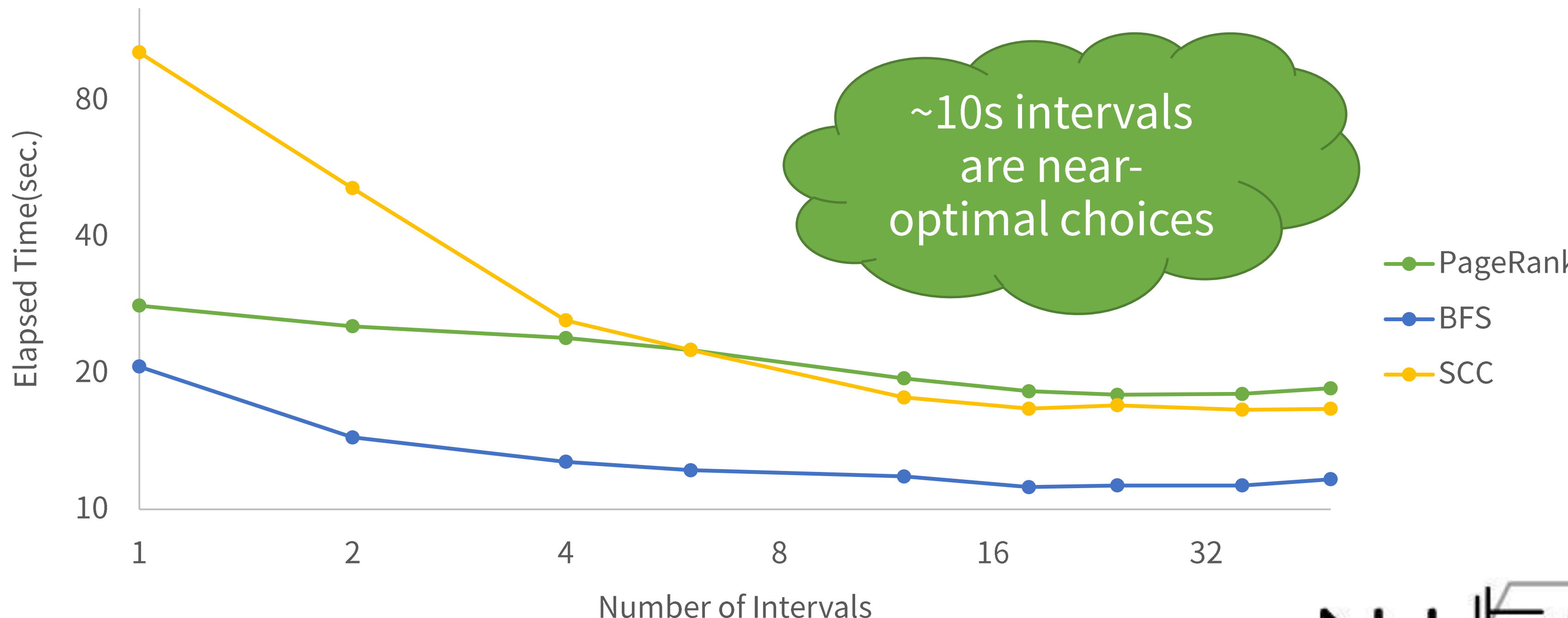
Results: Design Decisions

Performance with different sub-shard model

Model	Elapsed Ti[illegible]		
	Live-journal	Tw[illegible]	[illegible]
src-sorted, coarse-grained	1.44s	72.[illegible]	[illegible]
dst-sorted, fine-grained	1.00s	20.50s	[illegible].51s

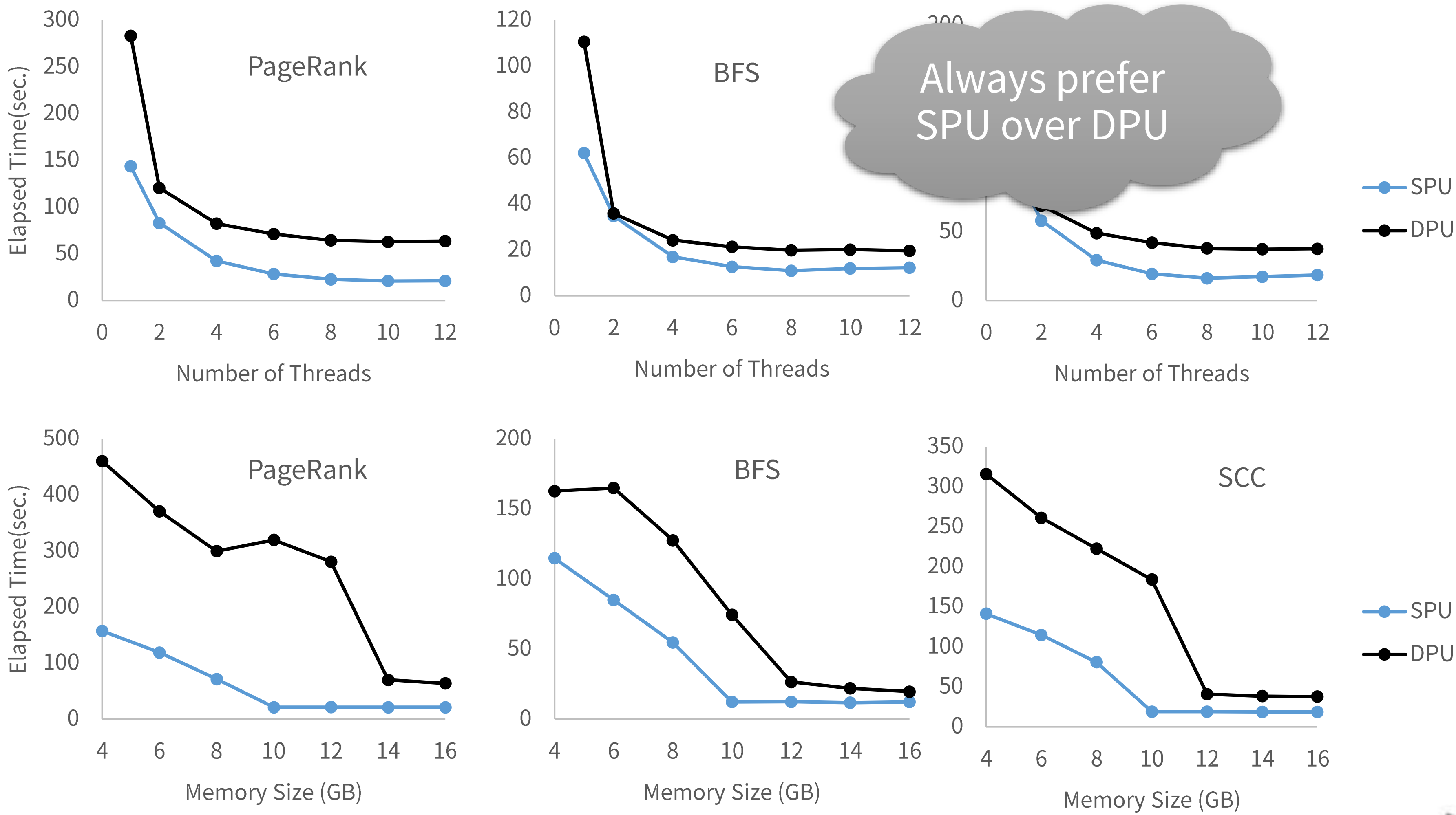
Destination-sorted is the right choice!

Performance with different numbers of interval on Twitter

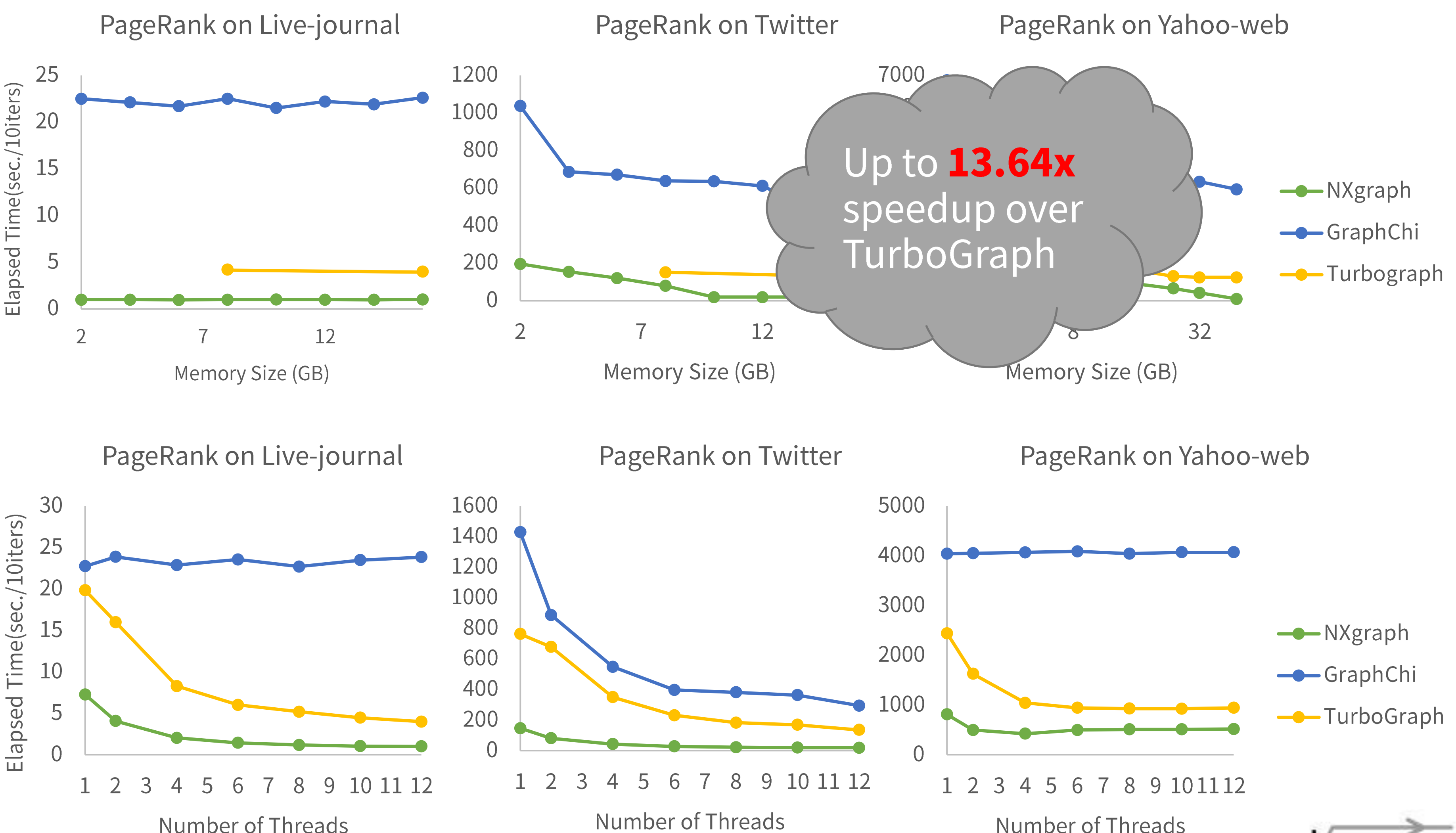


Results: Design Decisions

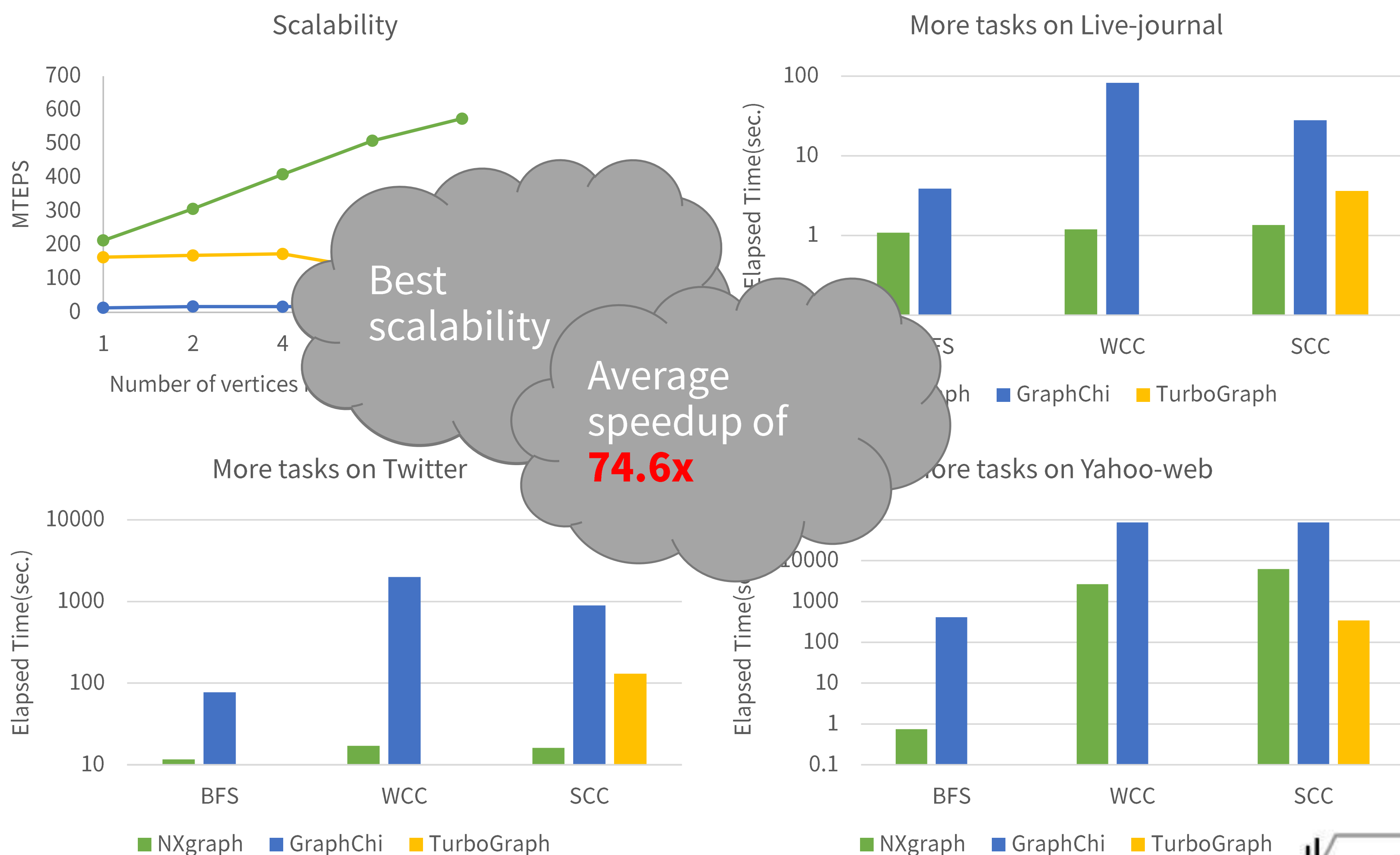
SPU vs DPU on performance



Results: Different Environments



Results: Scalability and More Tasks



Results: More Systems (Limited Resources)

System performance with limited resources

System	Time (s)	Speedup	Evaluation environment
NXgraph	7.13	1.00	Intel i7 3.3GHz, 8t, 8G, SSD
GridGraph	26.91	3.77	AWS EC2 8t, 8G/30.5G, SSD
X-stream	88.95	12.48	[illegible]
NXgraph	12.55	1.00	[illegible]
VENUS	95.48	7.60	[illegible] HDD
GridGraph	24.11	1.92	[illegible] /30.5G, HDD
X-stream	81.70	6.51	AWS EC2, 8t, 8G/30.5G, HDD

Task: 1 iteration of PageRank on Twitter graph

Average speedup of **6.6x** over various state-of-the-art single-machine systems with limited resources

Results: More Systems (Best Case)

System performance in the best case

System	Time (s)	Speedup	Evaluation environment
NXgraph	2.05	1.00	Intel i7 3.3GHz, 8t, 16G, SSD
X-stream	23.25	11.57	A… SSD
GridGraph	24.11	11.99	[illegible]
MMAP	13.10	6.52	…D
PowerGraph	3.60	1.79	…6t, 23G)

Task: 1 iteration of PageRank on Twitter graph

717MEPS actual throughput vs **792MEPS** hypothetical limit

Future Work

- NXgraph is still under development and subject to changes in data structures and APIs

v0.2 (current)

v0.3 (scheduled)

timeline

- ✓ Faster sharding (done)
- ✓ More flexible partitioning (done)
- ➢ Faster indexing (active)
- ➢ More complicated algorithms (active)
- Dynamic graph support (scheduled)
- Open source (scheduled: **mid August**)

- Asynchronous updating (scheduled)
- More intelligent partitioning (pending)
- Search for practical applications (pending)

Contact us if you would like to see a pre-release!

Reference

- H. Kwak, C. Lee, H. Park, and S. Moon, “What is twitter, a social network or a news media?” in *WWW*. ACM, 2010, pp. 591–600.
- J. Gonzalez, Y. Low, and H. Gu, “Powergraph: Distributed graphparallel computation on natural graphs,” in *OSDI*, 2012, pp. 17–30.
- J. Cheng, Q. Liu, Z. Li, W. Fan, J. C. S. Lui, and C. He, “VENUS: Vertex-Centric Streamlined Graph Computation on a Single PC,” in *ICDE*, 2015, pp. 1131–1142.
- A. Kyrola, G. Blelloch, and C. Guestrin, “GraphChi: Large-Scale Graph Computation on Just a PC,” in *OSDI*, 2012, pp. 31–46.
- W.-S. Han, S. Lee, K. Park, J.-H. Lee, M.-S. Kim, J. Kim, and H. Yu, “TurboGraph: A Fast Parallel Graph Engine Handling Billion-Scale Graphs in a Single PC,” in *SIGKDD*, 2013, pp. 77–85.
- M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: cluster computing with working sets,” in *HotCloud*, vol. 10, 2010, p. 10.
- L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: bringing order to the web.” 1999.
- X. Zhu, W. Han, and W. Chen, “GridGraph : Large-Scale Graph Processing on a Single Machine Using 2-Level Hierarchical Partitioning,” in *ATC*, 2015, pp. 375–386.
- A. Roy, I. Mihailovic, and W. Zwaenepoel, “X-Stream: Edge-centric Graph Processing using Streaming Partitions,” in *SOSP*, 2013, pp. 472–488.

Reference

- Yahoo! altavisata web page hyperlink connectivity graph, circa 2002,”
- http://webscope.sandbox.yahoo.com/.
- “Livejournal social network,” http://snap.stanford.edu/data/ soc-LiveJournal1.html.
- Z. Lin, M. Kahng, K. Sabrin, D. Horng, and P. Chau, “MMap : Fast Billion-Scale Graph Computation on a PC via Memory Mapping,” in *ICBD*. IEEE, 2014.
- G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, “Pregel : A System for Large-Scale Graph Processing,” in SIGMOD, 2010, pp. 135–145.

Thank you!

Q&A