Graphene: Fine-Grained IO Management for Graph Computing

Hang Liu H. Howie Huang

The George Washington University

FAST'17
FEB 27-MAR 2, 2017
SANTA CLARA, CA

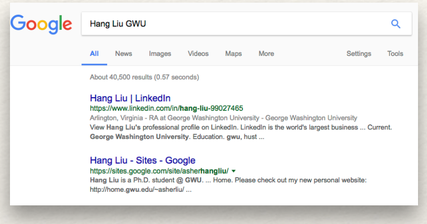
March 2, 2017

GW

Graphs are Everywhere …







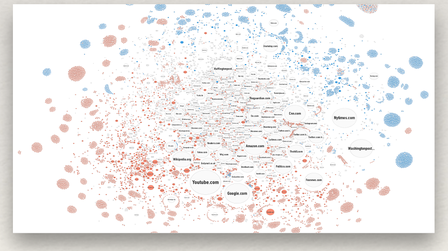


…

Our Daily Life



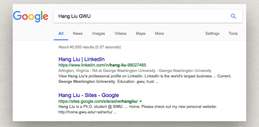




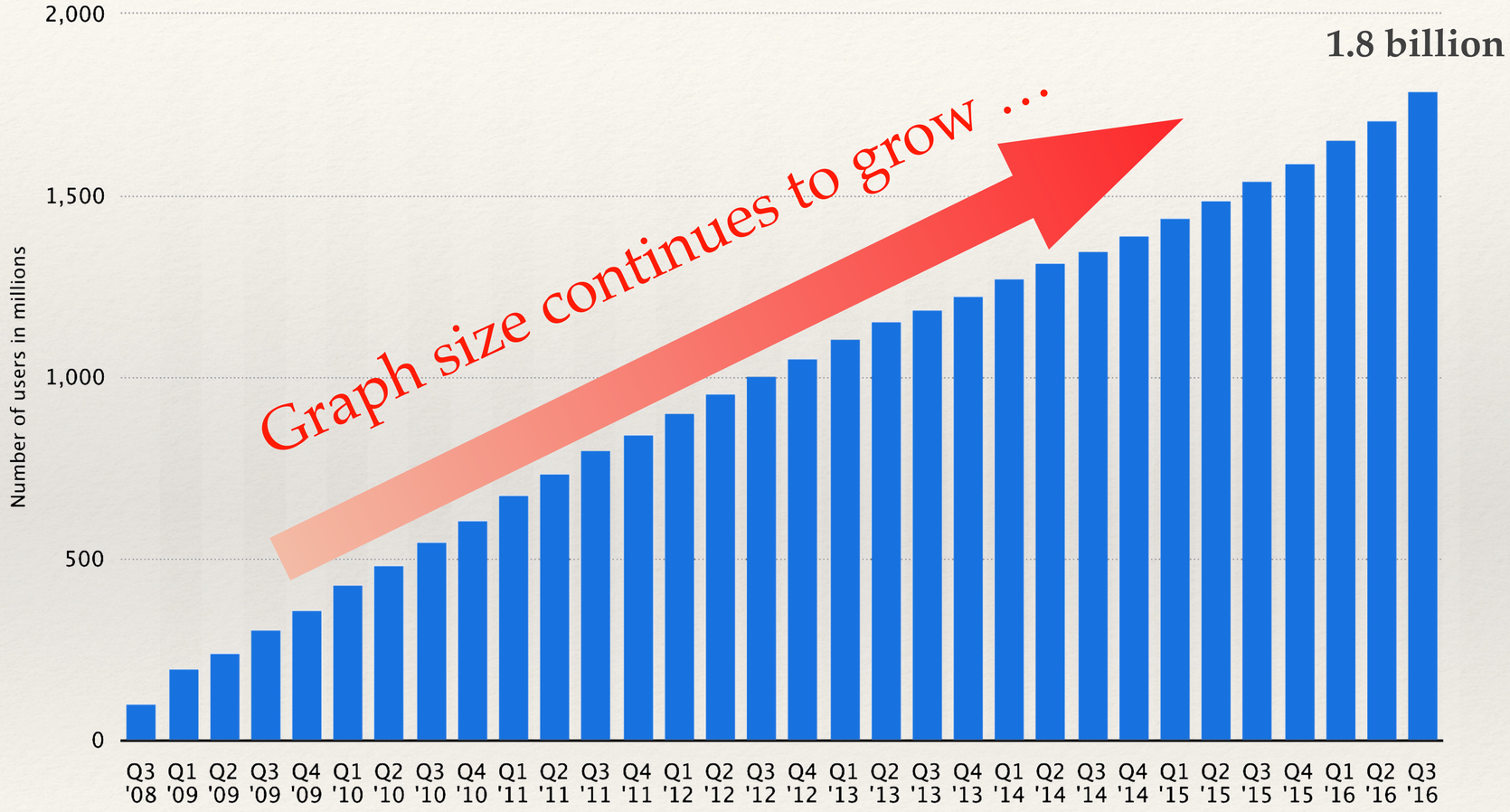


…

Graph Algorithms are Insightful

Algorithm	Functionality	Graph
Triangle completing	Friend recommendation	
Shortest path	Navigation	
PageRank	Webpage searching	
Matrix factorization	Product recommendation	

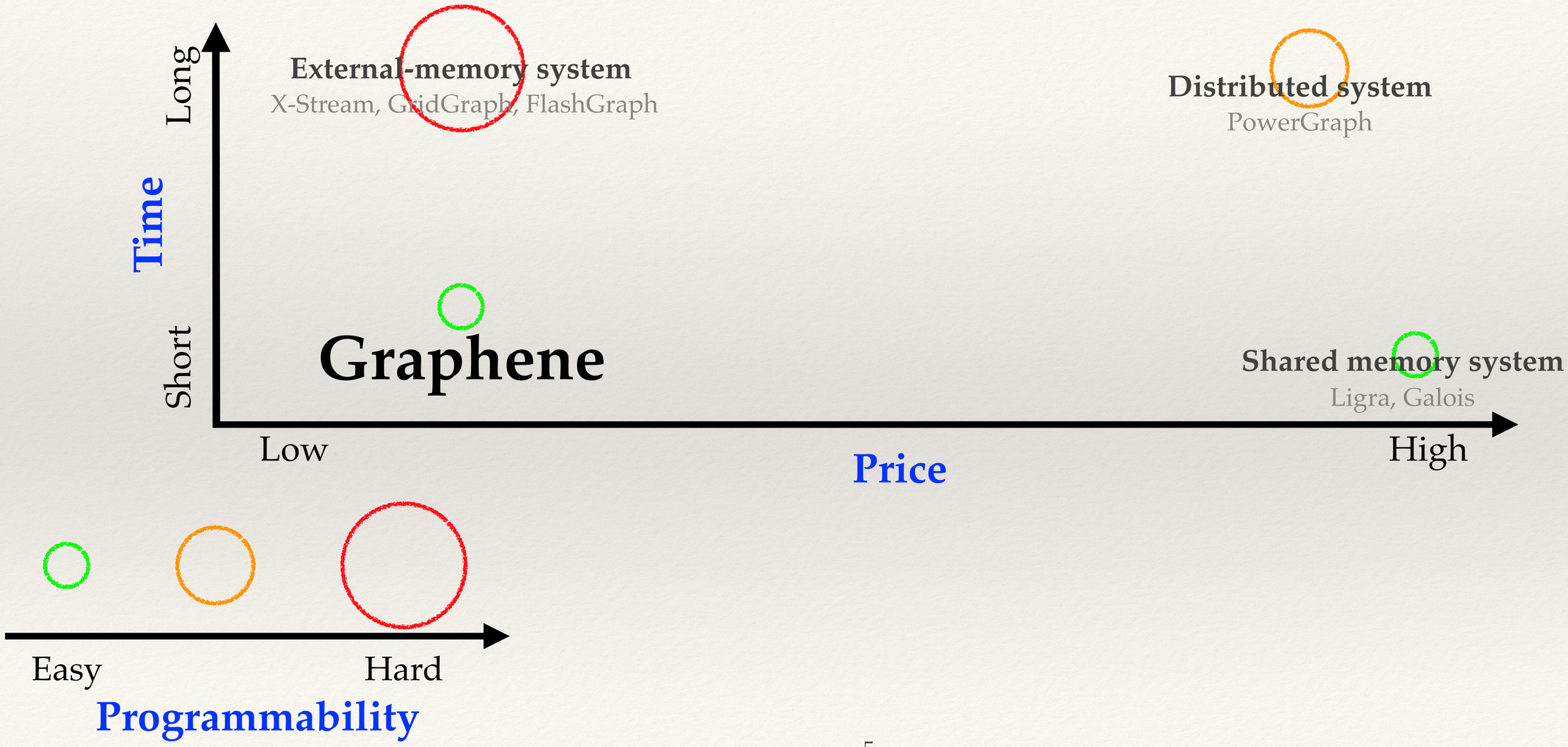
BIG Graph



Facebook, #active users

* source: statista

Related Work

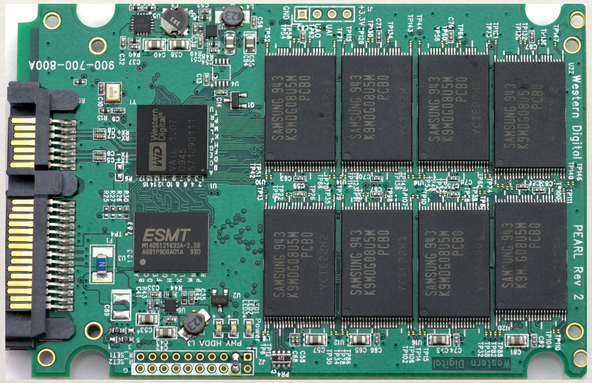


Agenda

- Background
- Graphene
  - Programming model
  - Bitmap based IO management
  - Row-column balanced partition
  - HugePage
- Evaluation
- Conclusion

Solid-State Drive

- Affordable
  - SSD ~0.3$/GB vs. DRAM ~10$/GB
- Bandwidth
  - Sequential IO 540 MB/s
  - Random IO ~300 MB/s (4KB IO block)
- Energy efficient



Difficult to Program

```
fd = open(neighborListFile, O_DIRECT);

IObuffer = posix_memalign(4KB aligned);
IOsetup(IOcontext);

while true
  foreach vertex v
    if isActive(v)
      begin = v.neighborBeginPostion();
      end = v.neighborEndPostion();
      beginAligned = IOalign(begin);
      endAligned = IOalign(end);

      IOsubmit(IOcontext, IObuffer, fd, beginAligned, endAligned);
      IOpoll(IOcontext, timeout);
      if(IOcontext == error){Errorhandler(IOcontext);}

      neighborList = neighborParser(IObuffer);
      Compute (v, neighborList)
  level++;
```

Open file descriptor

Allocate aligned IO buffer

Setup kernel IO context, lack error support

Align the IO offset and size

Submit IO request, lack error support

Poll IO request, lack error support

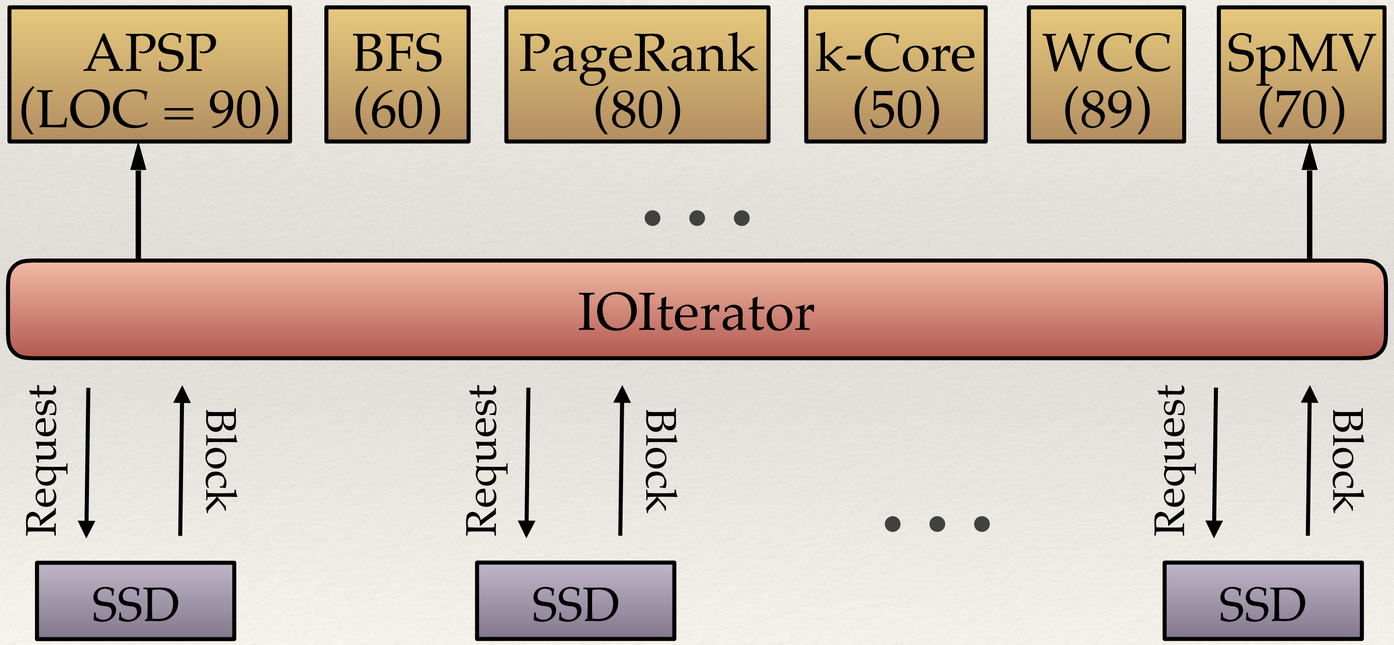
Define IO error handler

Parse IO buffer to filter out useful neighbors

IO Centric Graph Processing

Philosophy:

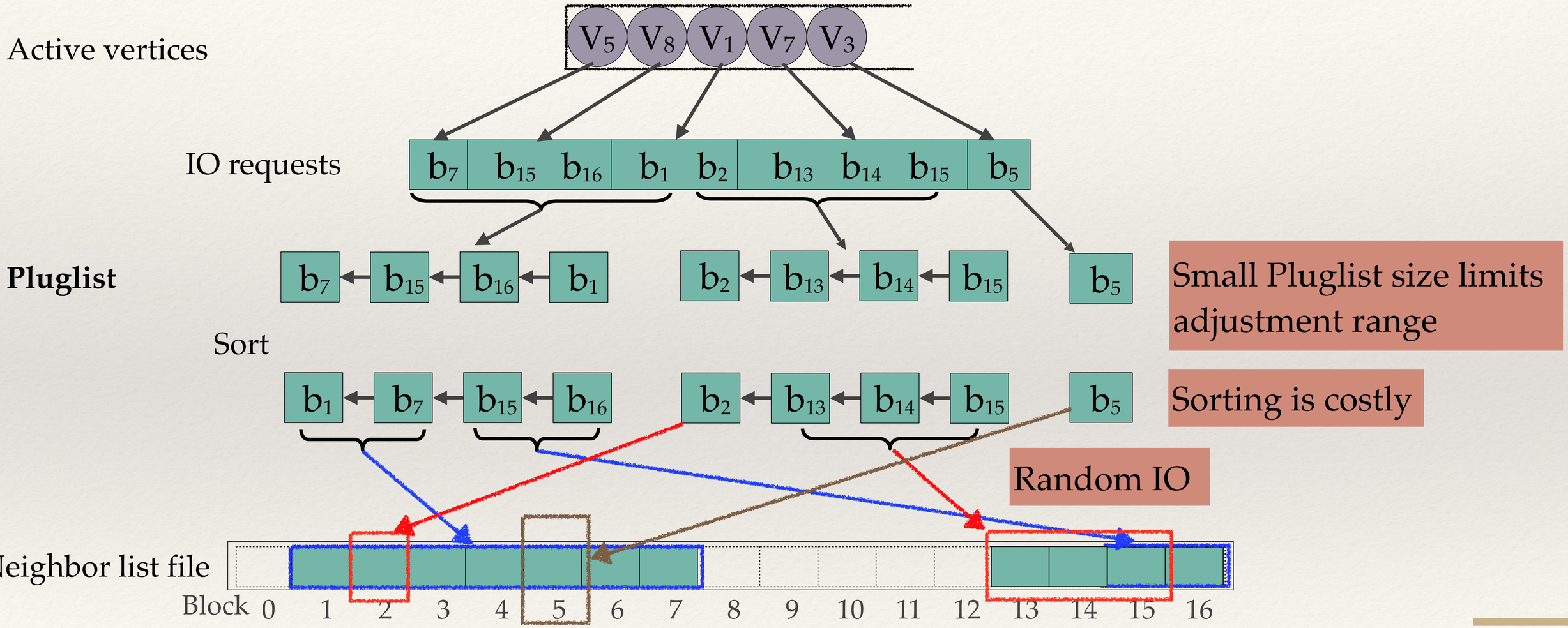
- Graphene knows which neighbors to load
- Pushes data to algorithms for computation




```
IOIterator it = IOIterator (neighborListFile, isActive);

while true
  while it -> hasMore()
      vertex v = it -> current();
      Compute (v, it -> getNeighbor(v));
  level++;
```

Linux Pluglist



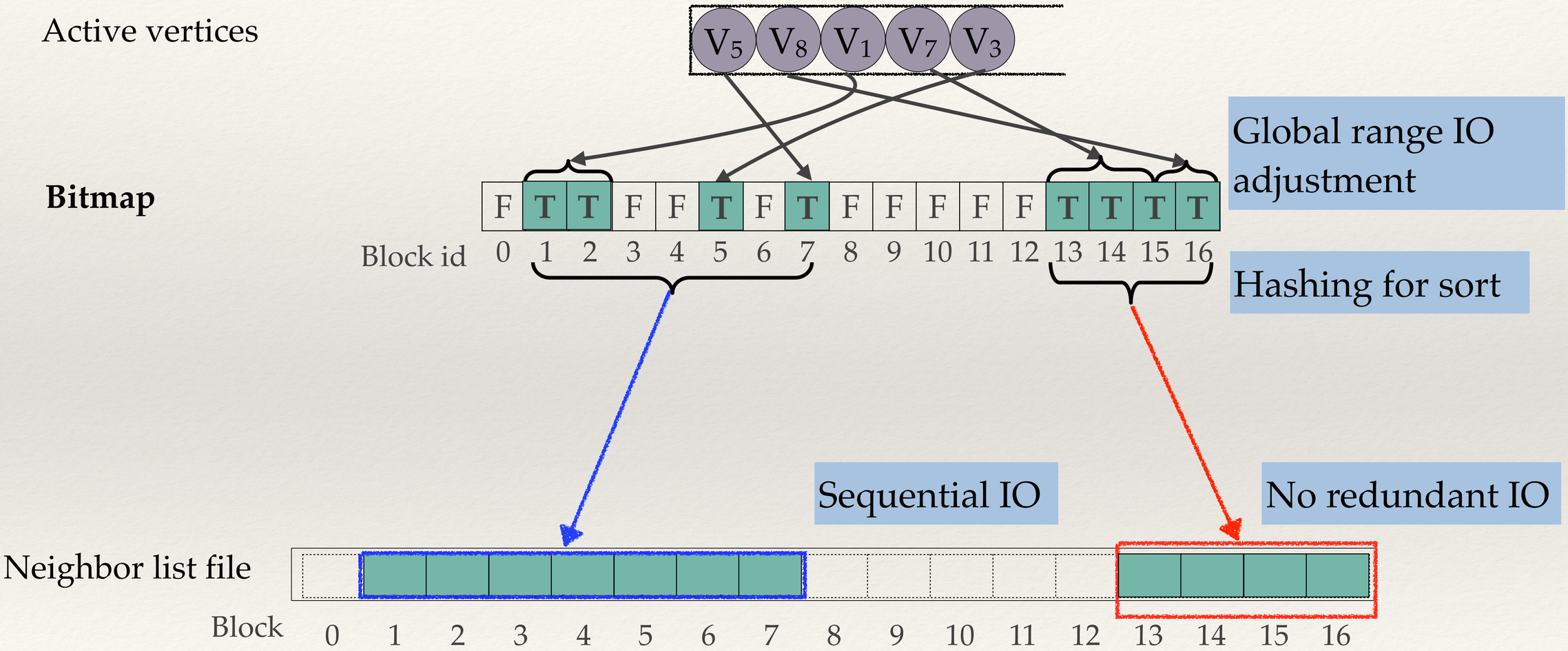
Small Pluglist size limits adjustment range

Sorting is costly

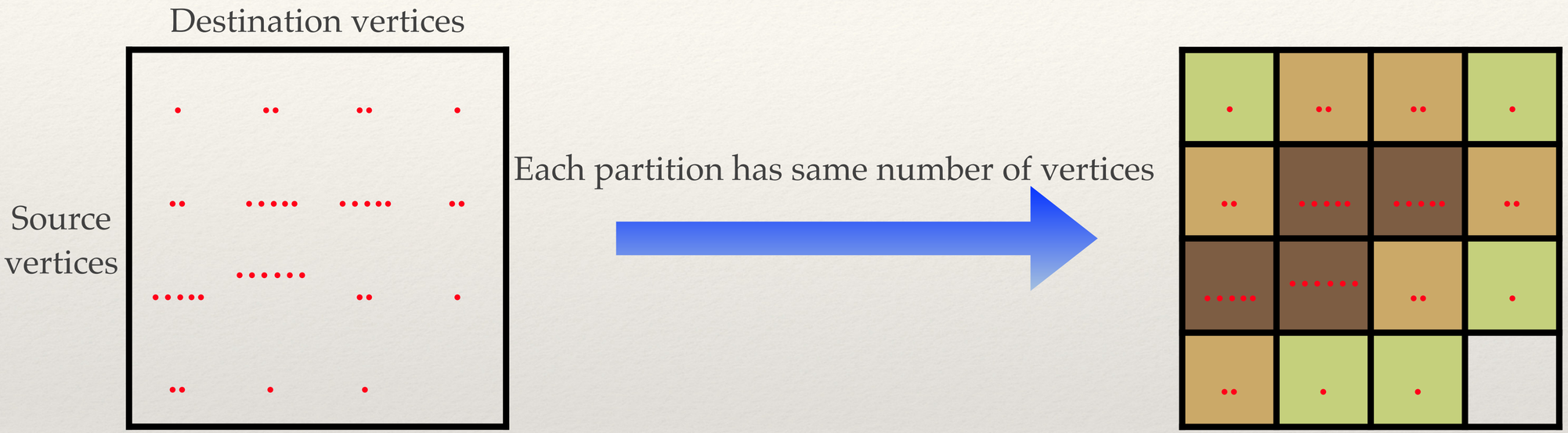
Random IO

Load redundant blocks

Novel Bitmap Solution



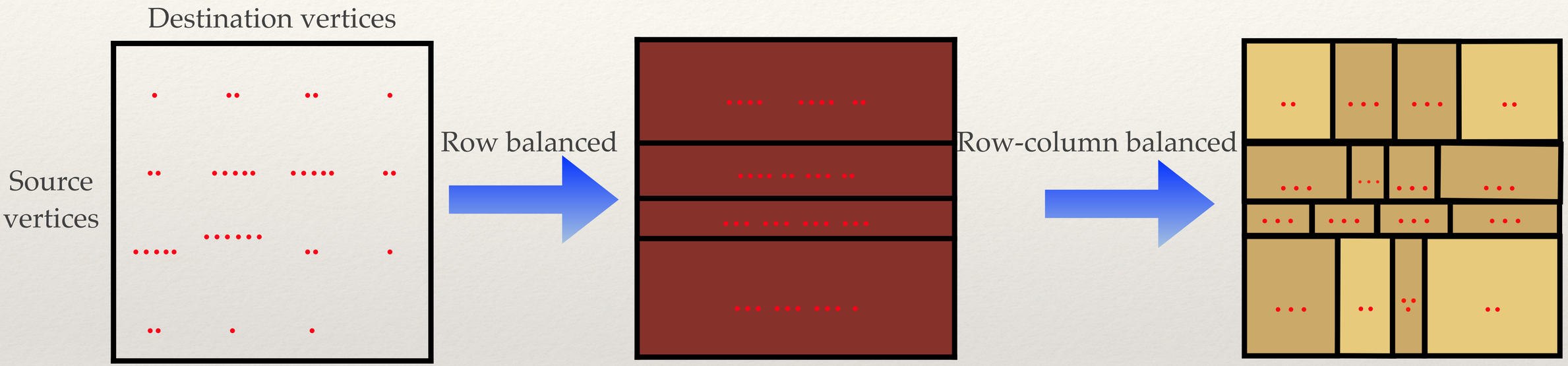
Conventional 2D Partition





Balance #edges should be the goal!

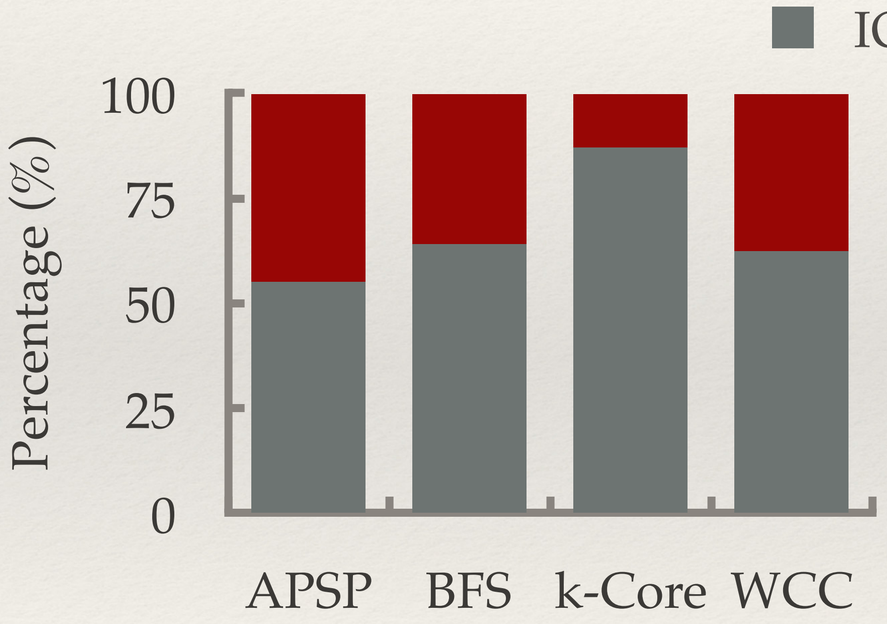
Row-Column Balanced 2D Partition



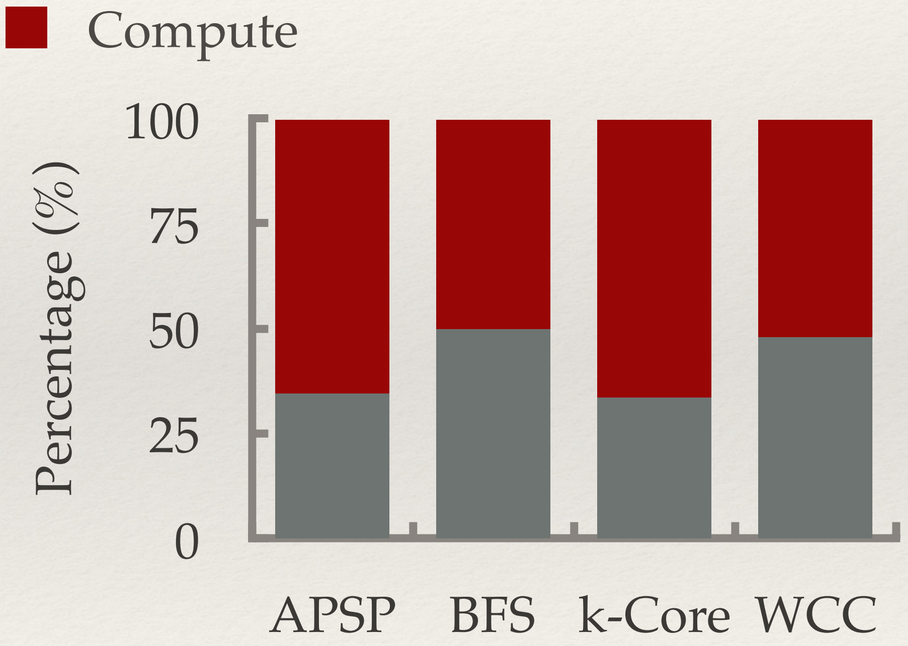
2.5x speedup over conventional 2D partition on Twitter graph



IO vs Compute



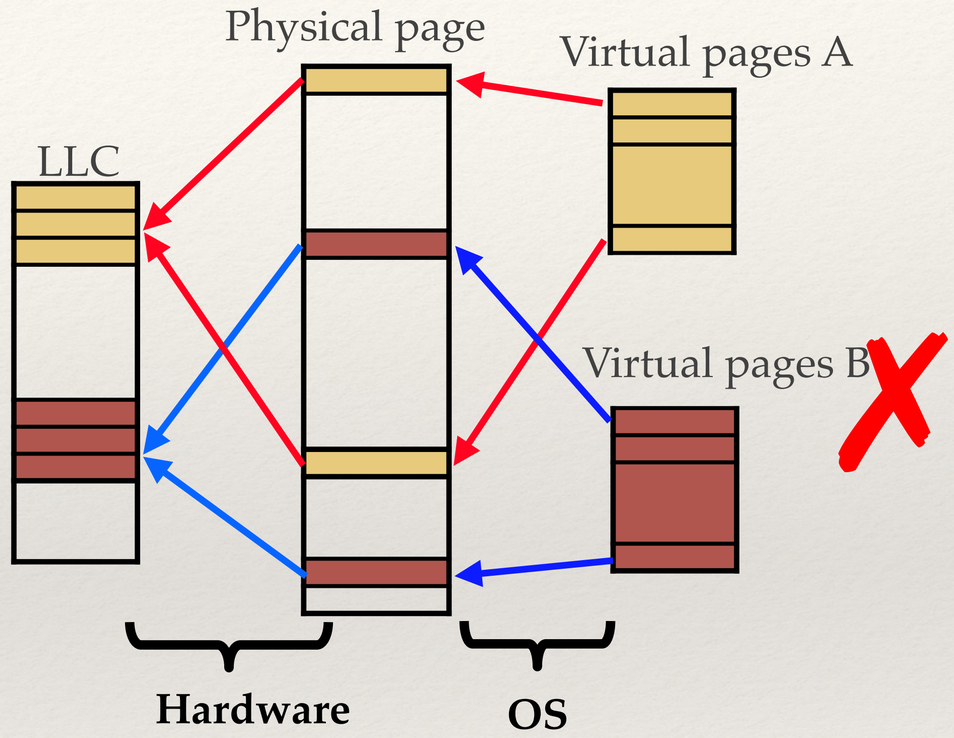
Before IO optimization



After IO optimization

Cache Optimization

- Last-level cache (LLC)
  - Page coloring has some benefits
- Translation lookaside buffer (TLB)
  - Use HugePage for buffering IO, managing metadata, etc.



I/O Buffer:



HugePage

Evaluation

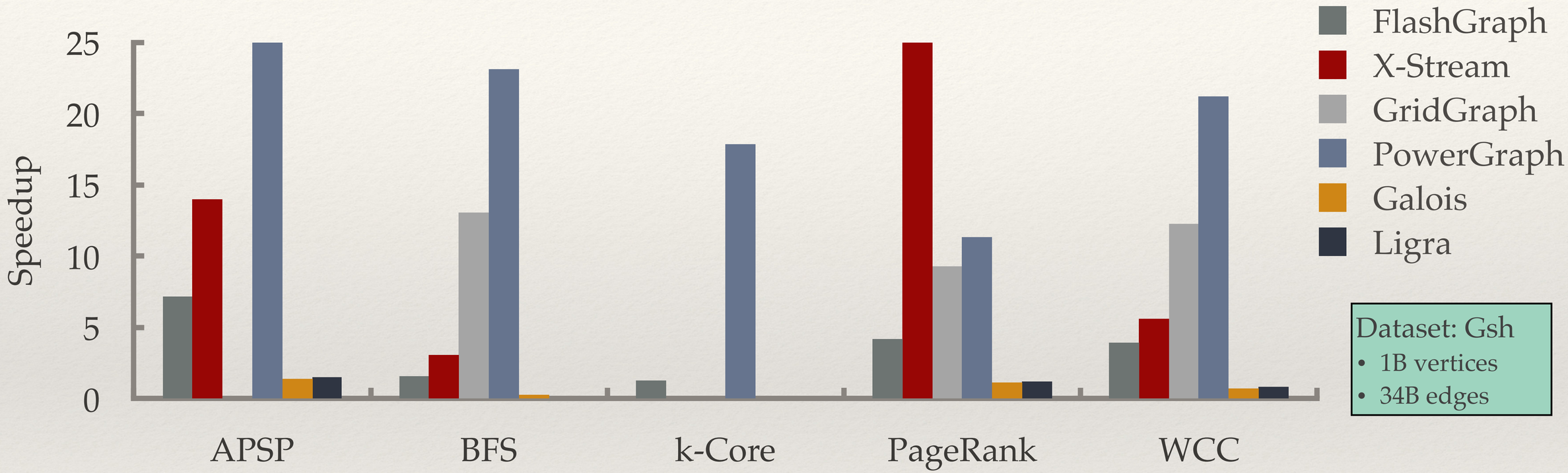
- Intel Xeon E5-2620 CPUs
  - 12 cores, supporting 24 physical threads with hyper-threading
  - 2MB, 1GB HugePage
  - 15MB LLC
- 256GB main memory
- 16 SSDs (Samsung 850 EVO SSDs with PCI-HBAs).
- Linux kernel 4.4.0, g++ 4.8.5

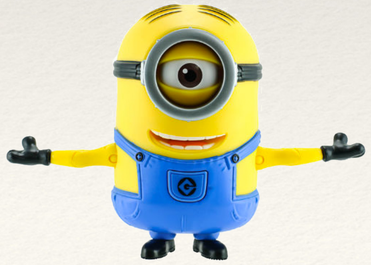
GW

Datasets Specifications

Name	Description	#Vertices	#Edges	Size
Friendster	Friendster social	68M	2.6B	20GB
Twitter	Twitter social	53M	2.0B	15GB
Clueweb	ClueWeb12	978M	42.6B	336GB
EU	EU domains	1071M	92B	683GB
Gsh	Gsh webpage	988M	33.8B	252GB
UK	UK domains	788M	48B	270GB
Kron30	Graph500 Generator	1B	32B	256GB
Kron31	Graph500 Generator	2B	1T	8TB

Graphene vs. State-of-the-art



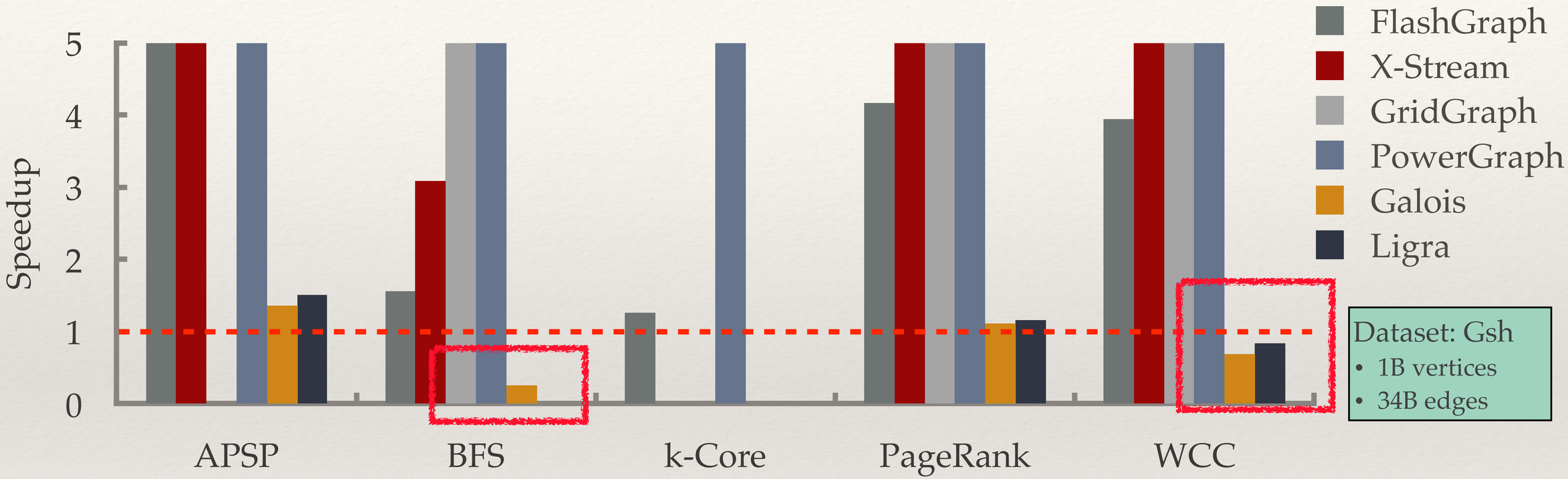


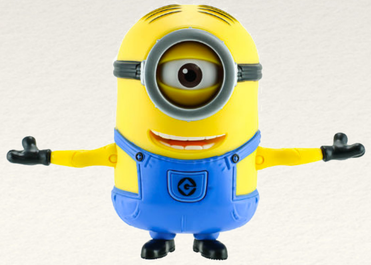
External memory system: FlashGraph 3.6X, X-Stream 16X, GridGraph 6.9X

Distributed system: PowerGraph 20X.

Shared memory system: Galois 5% and Ligra 16% (no BFS)

Graphene vs. State-of-the-art (Cont'd)



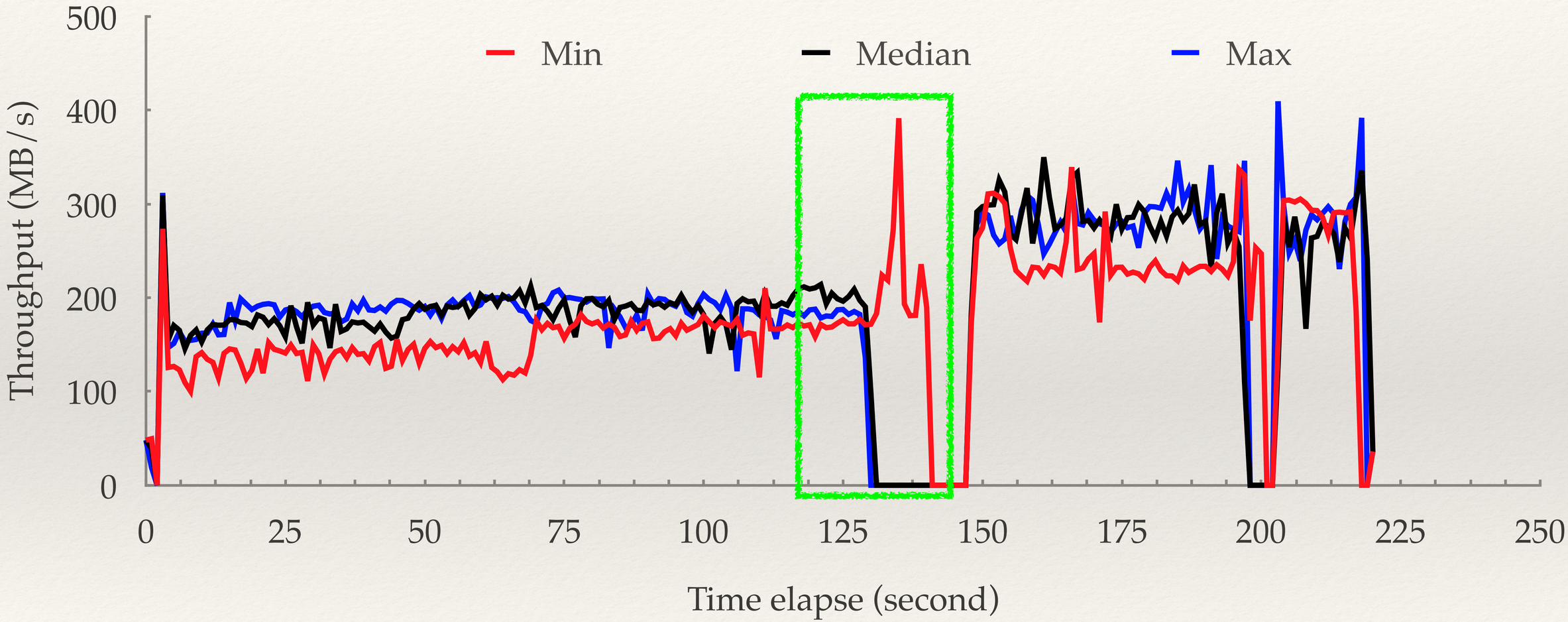


External memory system: FlashGraph 3.6X, X-Stream 16X, GridGraph 6.9X

Distributed system: PowerGraph 20X.

Shared memory system: Galois 5% and Ligra 16% (no BFS)

Disk Bandwidth Utilization



Conclusion

- Easy programming: “IO centric graph processing”
- IO optimizations:
  - Bitmap, Asynchronous direct IO
  - Fine grained IO
- Balancing:
  - Row-column based partition
  - Workload stealing
- HugePage optimization

Despite external memory, Graphene performs comparably with in-memory solutions

GW

Thank You

{asherliu, howie}@gwu.edu

https://github.com/iHeartGraph/Graphene

GW