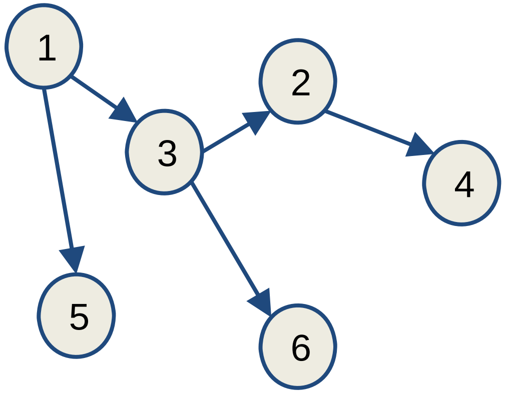
Scale-up Graph Processing: A Storage-centric View

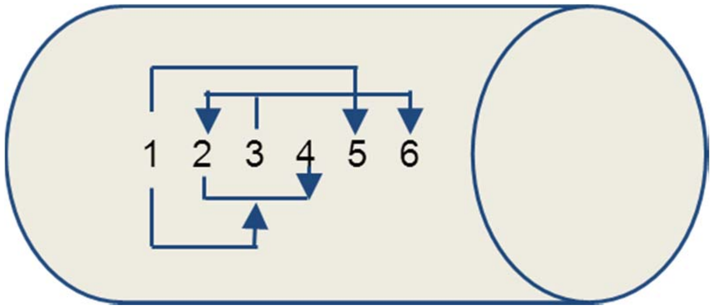
Eiko Yoneki University of Cambridge

Amitabha Roy EPFL

Graph Storage

- Storing and accessing graphs is a challenge
  - Edge traversal produces an access pattern that is
    - Random
    - Unpredictable





- For scale up or limited scale out (small clusters)
  - Storage bottlenecks (RAM, SSD, Magnetic disk) in critical path

RASP and X-Stream

- Storage-centric: two different novel ways to access graph structured data
  - Batch processing of large graphs on single machine
  - Establish useful limits for single machine processing
  - Directly address storage bottlenecks

RASP: Accelerates random access using a novel prefetcher

X-Stream: Sequentially streaming a large set of (potentially unrelated) edges

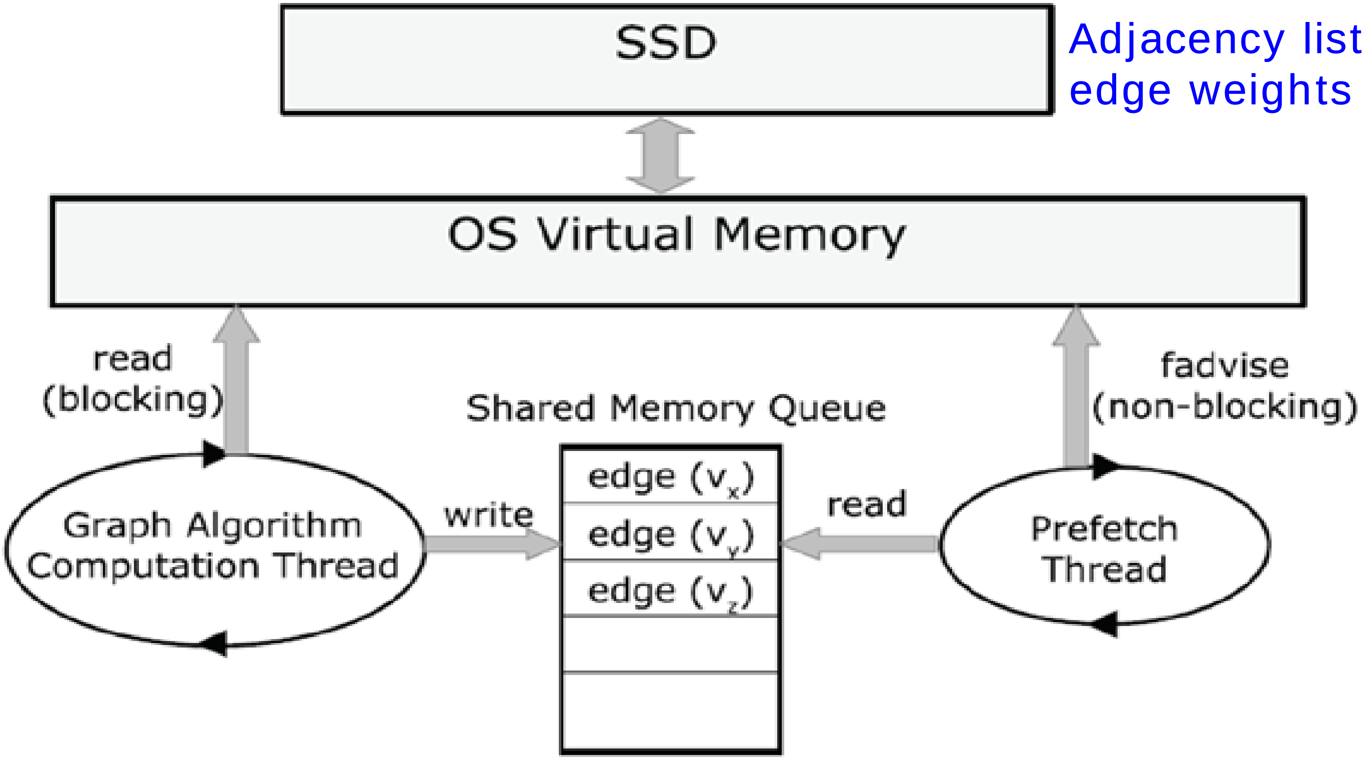
- RASP and X-stream take (diametrically opposite) storage centric view of graph processing problems

RASP: Run Ahead SSD Prefetcher

- Prefetching allows cheap hardware to compete with supercomputing for suitable graph traversal
- Prefetcher ensures that edge data to progress computation is always available in memory
- Allows graph traversal to keep queue depth high → SSD to achieve good performance
- Vertices (O(V)) size structure in memory
- Edges in SSD in CSR format
- Efficient on traversal: WCC, BFS, SSSP, A*...

Edge Queue Management

- Prefetcher invokes any registered callbacks, accessing the current state of the main program's iterator
- Asynchronous page load requests to OS via fadvise
- Repeat to ensure future data to active LRU list



Adjacency list and edge weights in CSR

RASP Speedup

- Speedups from up to 13x comparing over single and multithreaded versions
- RASP Memory usage WCC

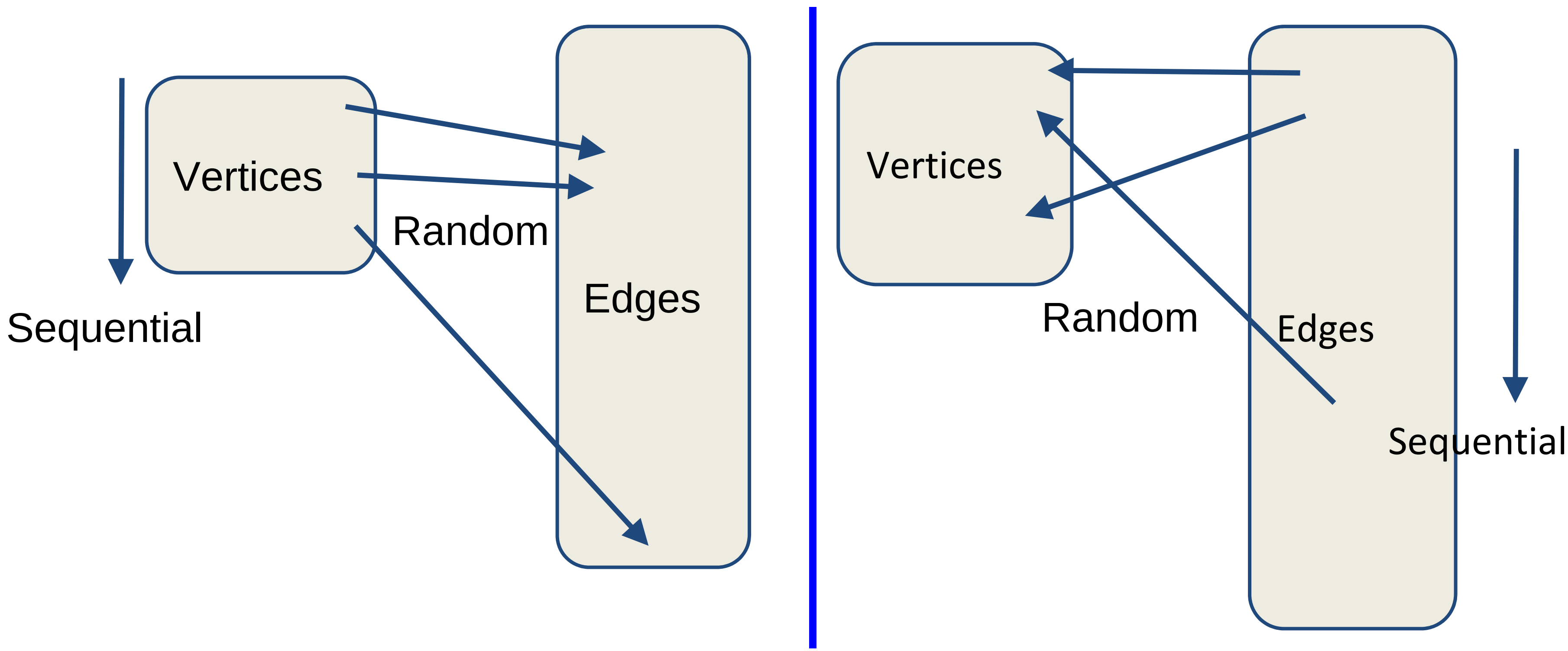
Graph	Vertices	Edges	RAM	SSD
Twitter [10]	52M	1.6B	1.18GB	8.4GB
Erdos-Reyni [11]	20M	2B	0.45GB	10.4GB
Scale-free-small [12]	32M	1.28B	1.10GB	12GB
Scale-free-large [12]	1B	40B	24GB	315GB

- RASP Runtime (mins) for WCC

Graph	Base	RASP	MT(8)	RAM
Twitter	36.08	6.36	7.17	2.31
Erdos-Reyni	80.96	6.03	11.30	4.11
Scale-free-small	88.56	10.77	15.74	3.95
Scale-free-large	>2.5 days	402.56	>2 days	cannot fit

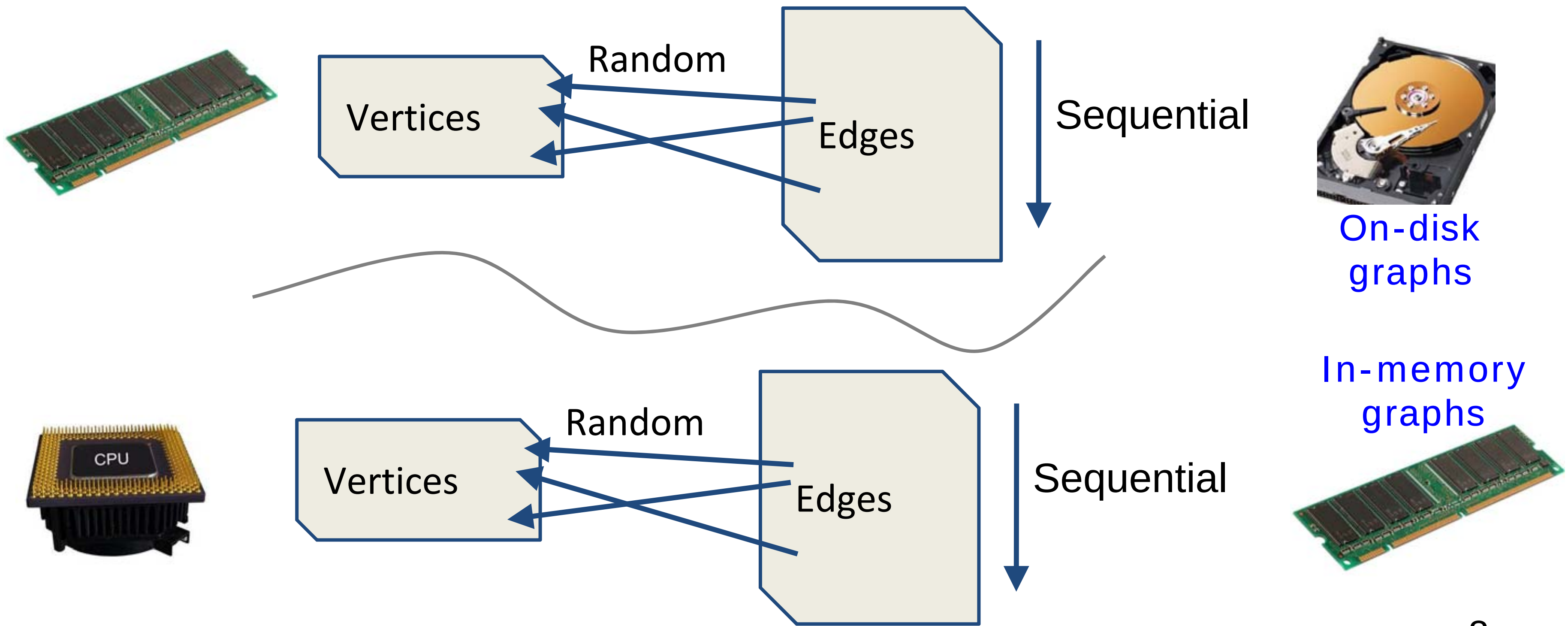
Vertex/Edge Centric Access

- Vertex centric access is random
- Edge centric access is more sequential
- Can subdivide into streaming partitions



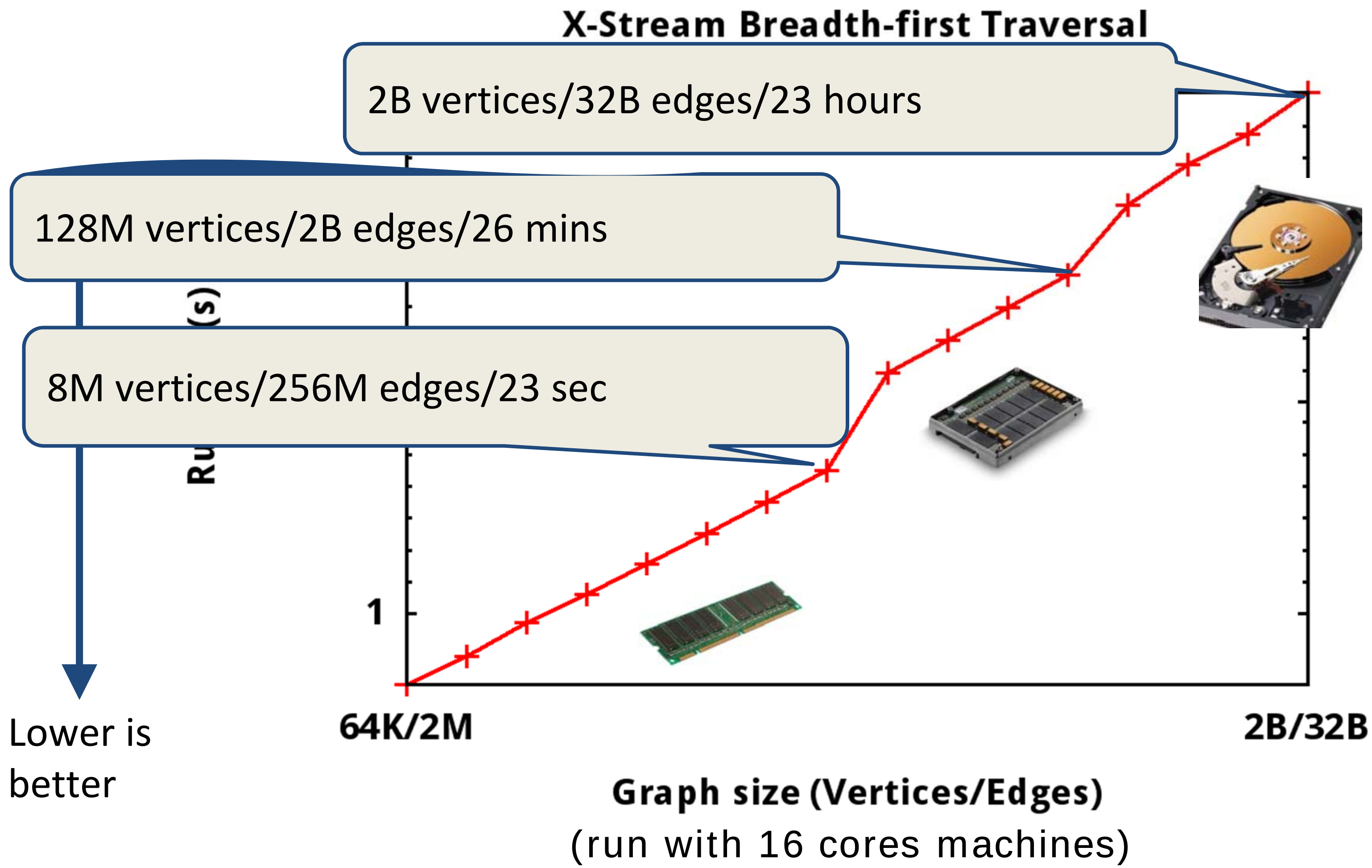
X-Stream: *Streaming Partitions*

- Sequential access to any medium
- Eliminate random access to edges
- Ensure randomly accessed vertices held in cache



Scale-up with X-stream

- Scaling up through RAM, SSD and Magnetic Disk



Pros and Cons

- RASP clearly provides impressive speedup
- Improving inefficiency of random access to SSD by prefetching
- Limitation
  - RASP requires pre-processing to CSR format
  - RASP is specific to SSD
  - Focus is traversal based graph computation (not for DFS)
- X-Stream transforms them to sequential access
- Single building block of streaming partitions
  - Works well with RAM, SSD, and Magnetic Disk
- Limitation
  - X-stream needs to trade off fewer random accesses to edge list for sequential bandwidth of streaming a large number of potentially unrelated edges

RASP+X-Stream Hybrid Approach

- Allow streaming partitions to sort their associated edges and access them randomly
  - Starting point is X-stream style streaming
  - Low utilization of edges due to few active vertices triggers index building
  - Switch to RASP style prefetching after index is available
- Streaming partition has the necessary vertex subset in memory: a requirement for RASP
- RASP mitigates limitations of X-Stream
  - Wasted edges due to inactive vertices
  - Particular problem for high diameter graphs

IVEC Programming Model

- Abstract interface for graph algorithms, we intend to support
- Iterative Vertex-Centric programming model
  - Scatter: Vertex state → updates along edges
  - Gather: Updates on incoming edges → vertex state
  - IVEC can be mapped to Pregel, Powergraph, Graphchi …
  - GreenMarl (optimised iterative operation)
- Can express variety of graph operations
  - BFS/WCC, SSSP, PageRank, MIS…
  - But not algorithms with O(E) state, such as triangle counting
- Hides complexity of algorithms and storage from each other

Conclusion

- Storing and accessing graphs is a challenge since it is determinant for performance in graph processing
- RASP and X-stream: address diametrically opposite storage centric view of graph processing problem

RASP: Accelerates random access with prefetcher
X-Stream: Sequentially streaming edges

- Towards hybrid approach of RASP + X-Stream
- Scale out for bandwidth and capacity
  - Target 1T edges
- Explore 'limited' scale out
  - Network does not become the bottleneck
  - Multiply storage capacity, bandwidth and compute
  - Tightly coupled solutions: micro servers, low power boards