Big Data Analytics Using Hardware-Accelerated Flash Storage

Sang-Woo Jun

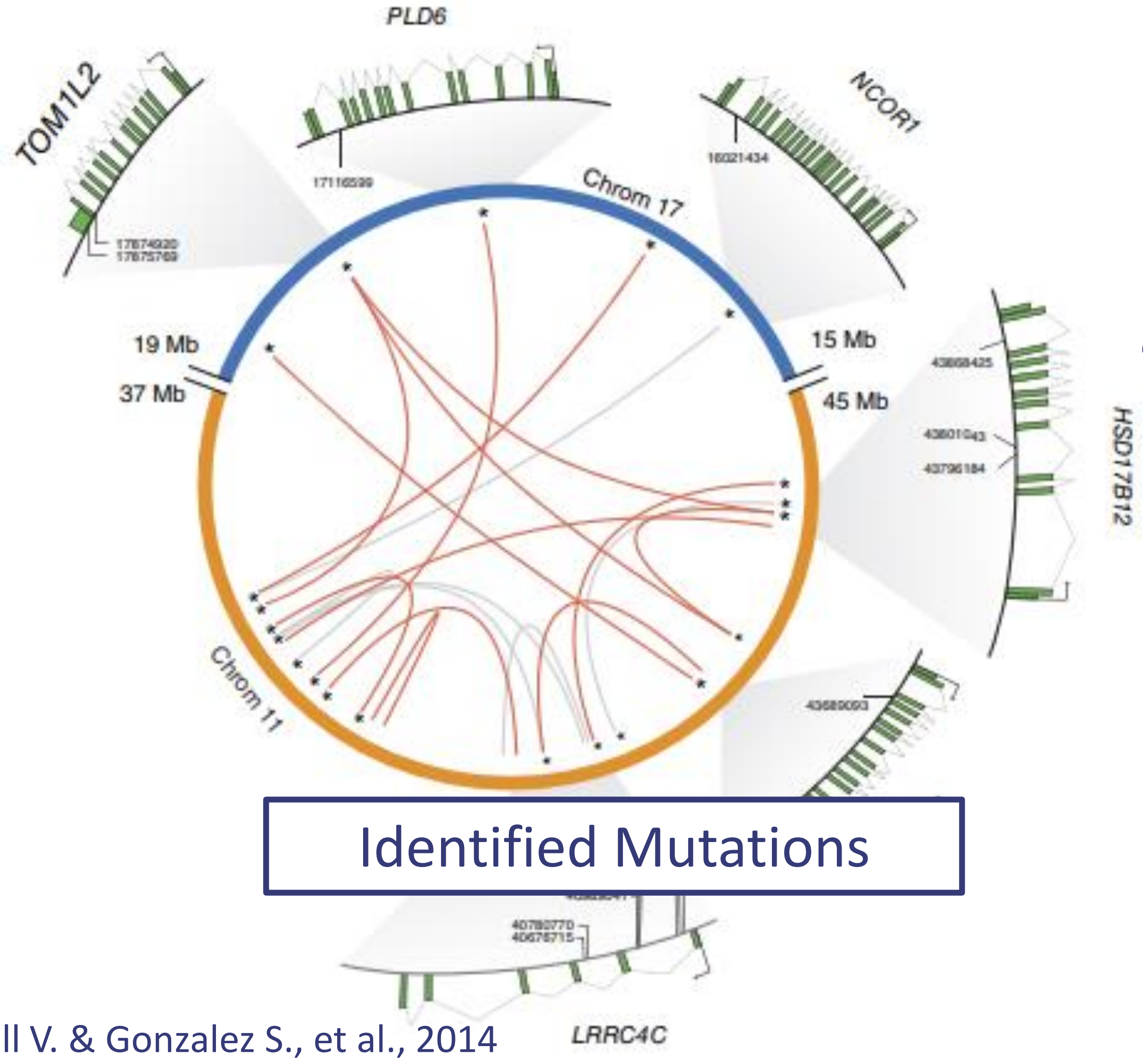
University of California, Irvine

(Work done while at MIT)

Flash Memory Summit, 2018

A Big Data Application: Personalized Genome

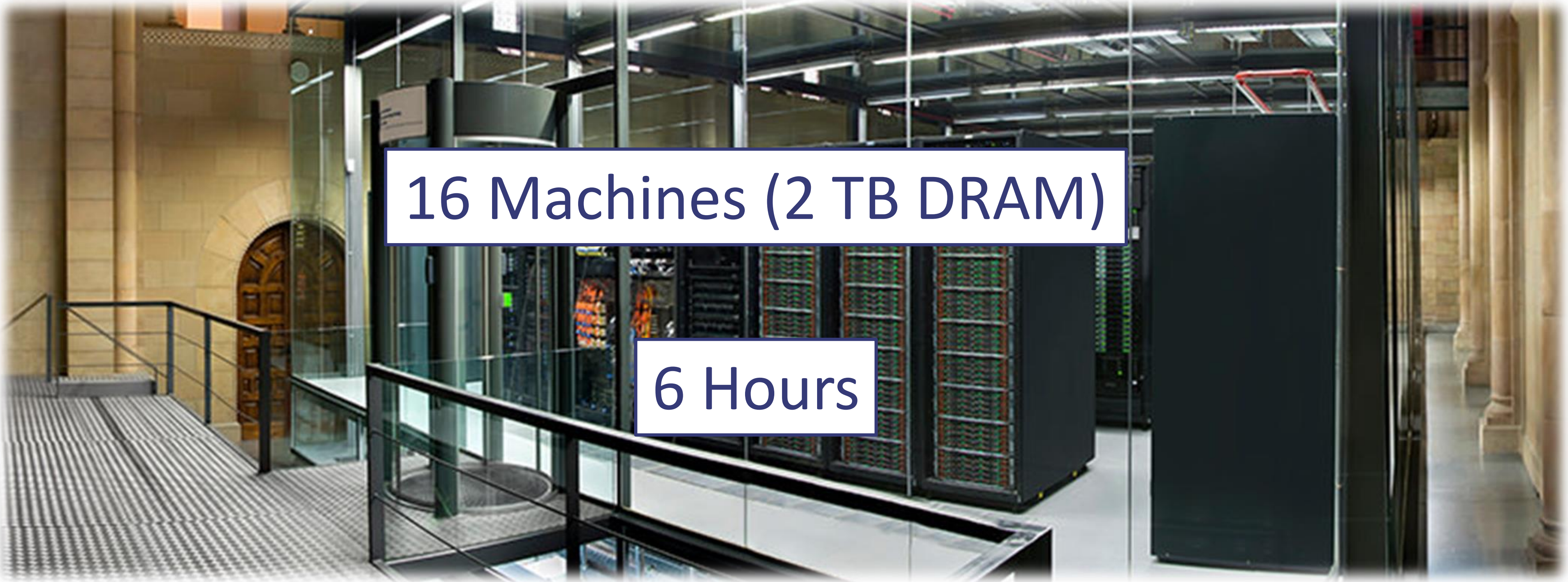
Cancer Patient



Identified Mutations

“Comprehensive characterization of complex structural variations in cancer by directly comparing genome sequence reads,” Moncunill V. & Gonzalez S., et al., 2014

Cluster System for Personalized Genome

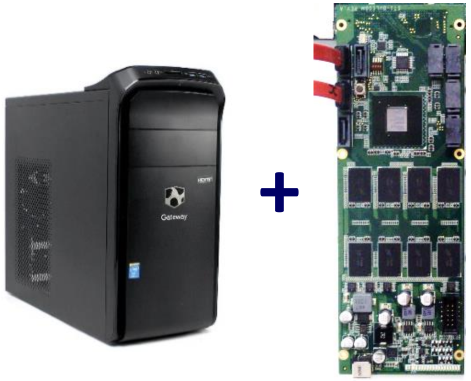


A Cheaper Alternative Using Hardware-Accelerated SSD

+

Collaborating with

Success with Important Applications



VS

Graph analytics
with billions of vertices

10x Performance

1/3 Power consumption

Key-value cache
with millions of users

Comparable performance

1/5 Power consumption

"GraFBoost: Using accelerated flash storage for external graph analytics," ISCA 2018
"BlueCache: A Distributed Flash-based Key Value Store," VLDB 2017

Contents

Introduction

Flash Storage and Hardware Acceleration

Example Applications

Graph Analytics Platform
Key-Value Cache

BlueDBM

Architecture Exploration Platform

Storage for Analytics

Fine-grained,
Irregular access

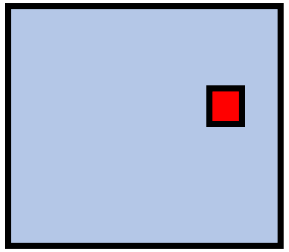
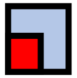
Terabytes in size

$$$ $8000/TB, 200W

Our goal:

$ $500/TB, 10W

Random Access Challenge in Flash Storage

	Flash	DRAM
Bandwidth:	~10 GB/s	~100 GB/s
Access Granularity:	 8192 Bytes	 128 Bytes

Wastes performance by not using most of fetched page

Using 8 bytes in a 8192 Byte page uses 1/1024 of bandwidth!

Reconfigurable Hardware Acceleration

Field Programmable Gate Array (FPGA)

Program application-specific hardware

High performance, Low power

Reconfigurable to fit the application

amazon web services EC2

Azure

intel CYCLONE 10 inside

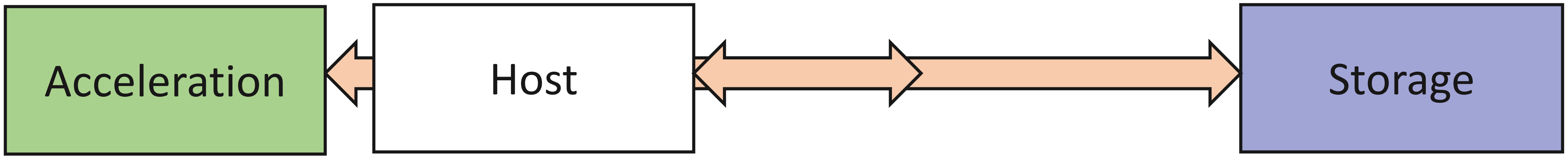
Future of Reconfigurable Hardware Acceleration

High Availability	Easy Programmability
Amazon EC2	Bluespec
Intel Accelerated Xeon	Xilinx HLS
Accelerated SSD platforms	Amazon F1 Shell
…	…

Normal to do HW/SW codesign

(Like with GPU computing)

Benefits of In-Storage Acceleration



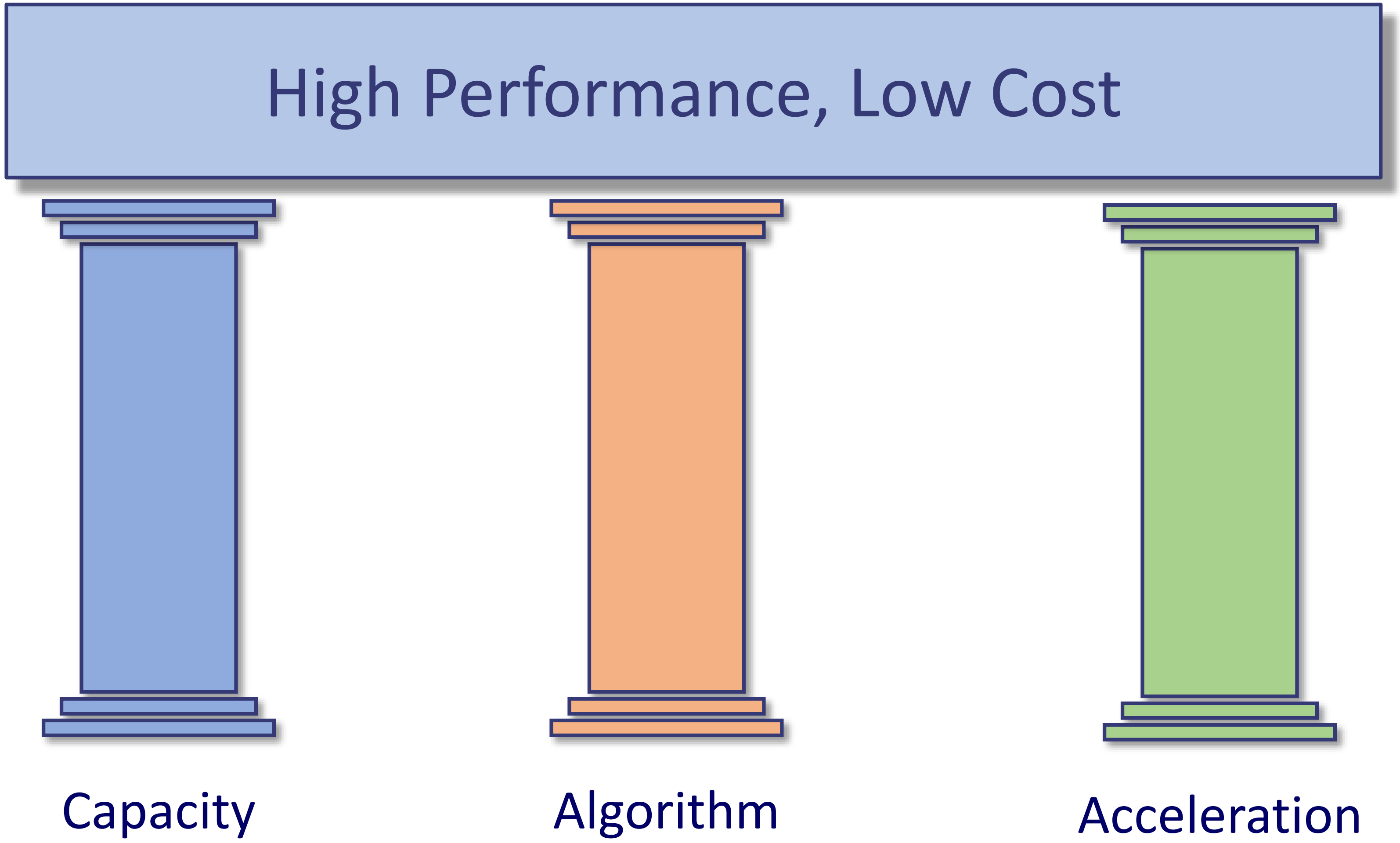
Lower latency, higher bandwidth to storage

Reduce data movement cost

Lower engineering cost

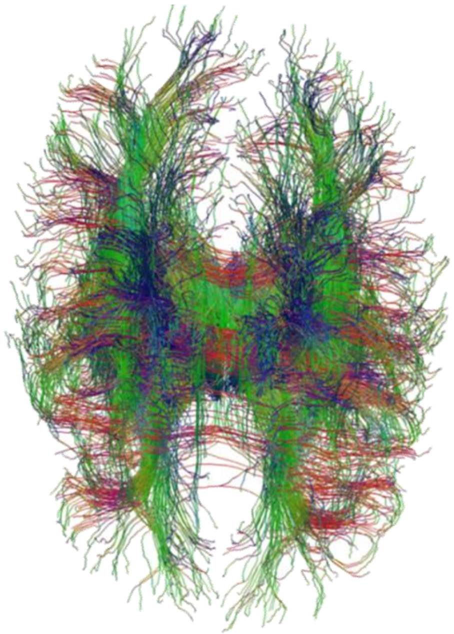
Most data comes from storage anyways

The Three Pillars of Competitive Flash-Based Analytics

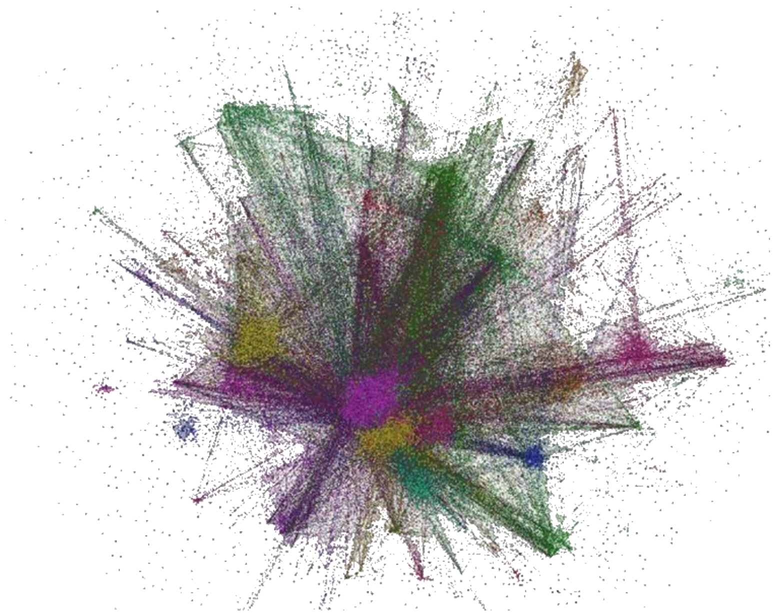


Contents

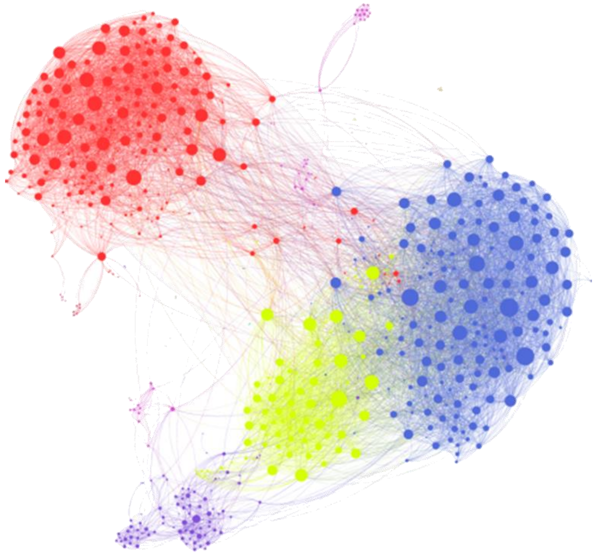
Large Graphs are Found Everywhere in Nature



Human neural network



Structure of the internet



Social networks

TB to 100s of TB in size!

1) Connectomics graph of the brain - Source: "Connectomics and Graph Theory" Jon Clayden, UCL
2) (Part of) the internet - Source: Criteo Engineering Blog
3) The Graph of a Social Network – Source: Griff's Graphs

Various Models for Graph Analytics

Vertex-Centric

Pregel,
GraphLab,
TurboGraph,
Mosaic,
FlashGraph,
GraphChi,
…

Edge-Centric

X-Stream,
…

Linear Algebraic

CombBLAS,
GraphMat,
Graphulo,
…

Graph-Centric

Giraph++,
…

Frontier-Based

Ligra,
Gemini,
Grunrock,
…

Optimized Algorithms

And more…

Galois,
…

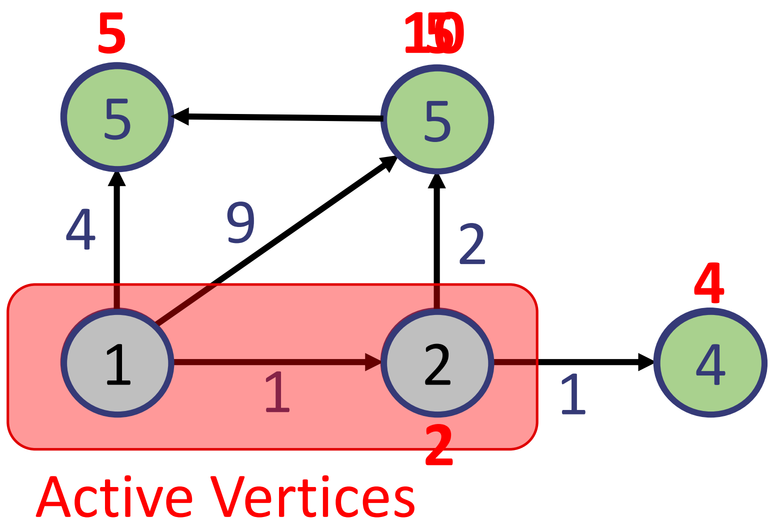
Vertex-Centric Programming Model

Popular model for efficient parallism and distribution

"Vertex program" only sees neighbors

Algorithm executed in terms of disjoint iterations

Vertex program is executed on one or more "Active Vertices"

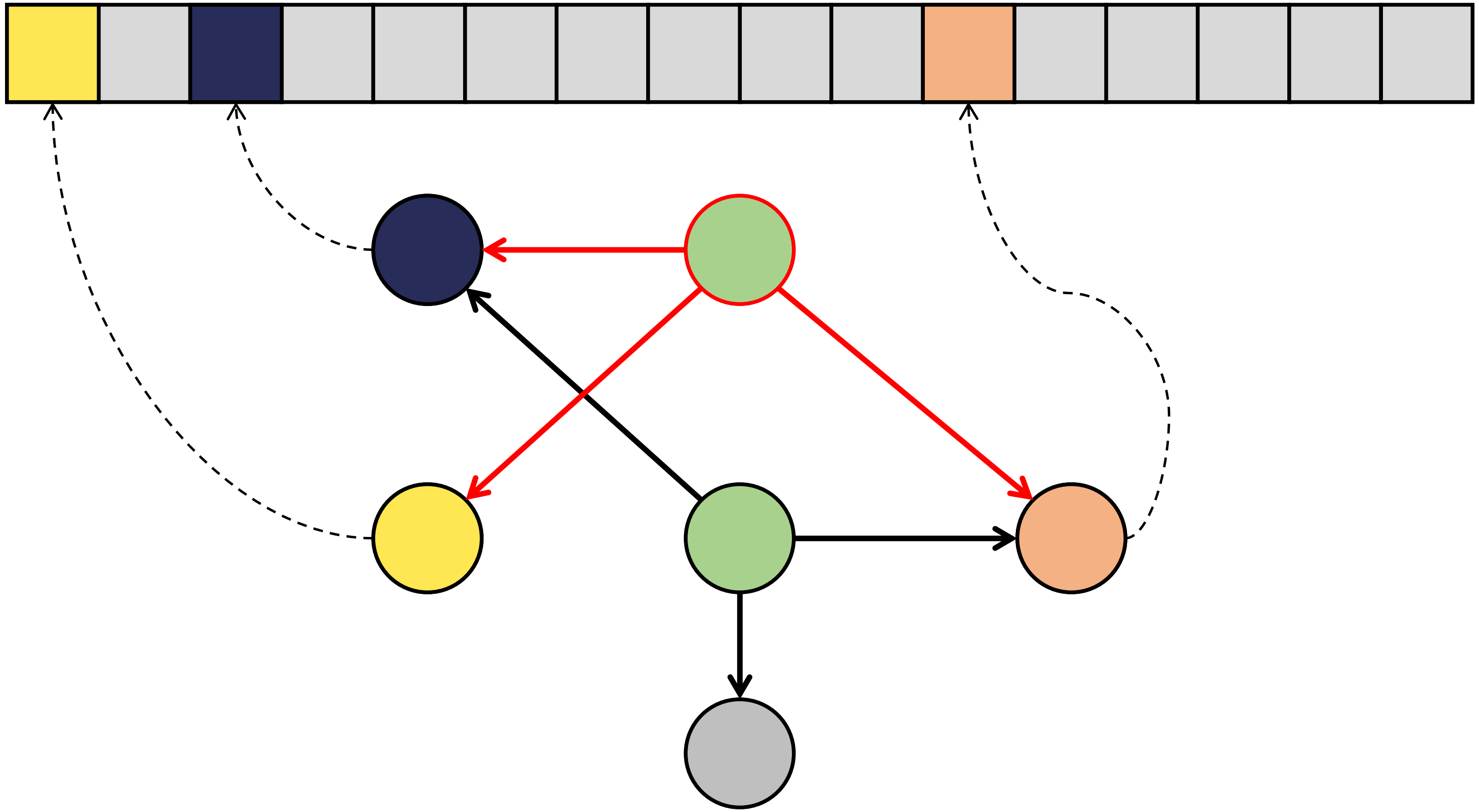


Algorithmic Representation of a Vertex Program Iteration

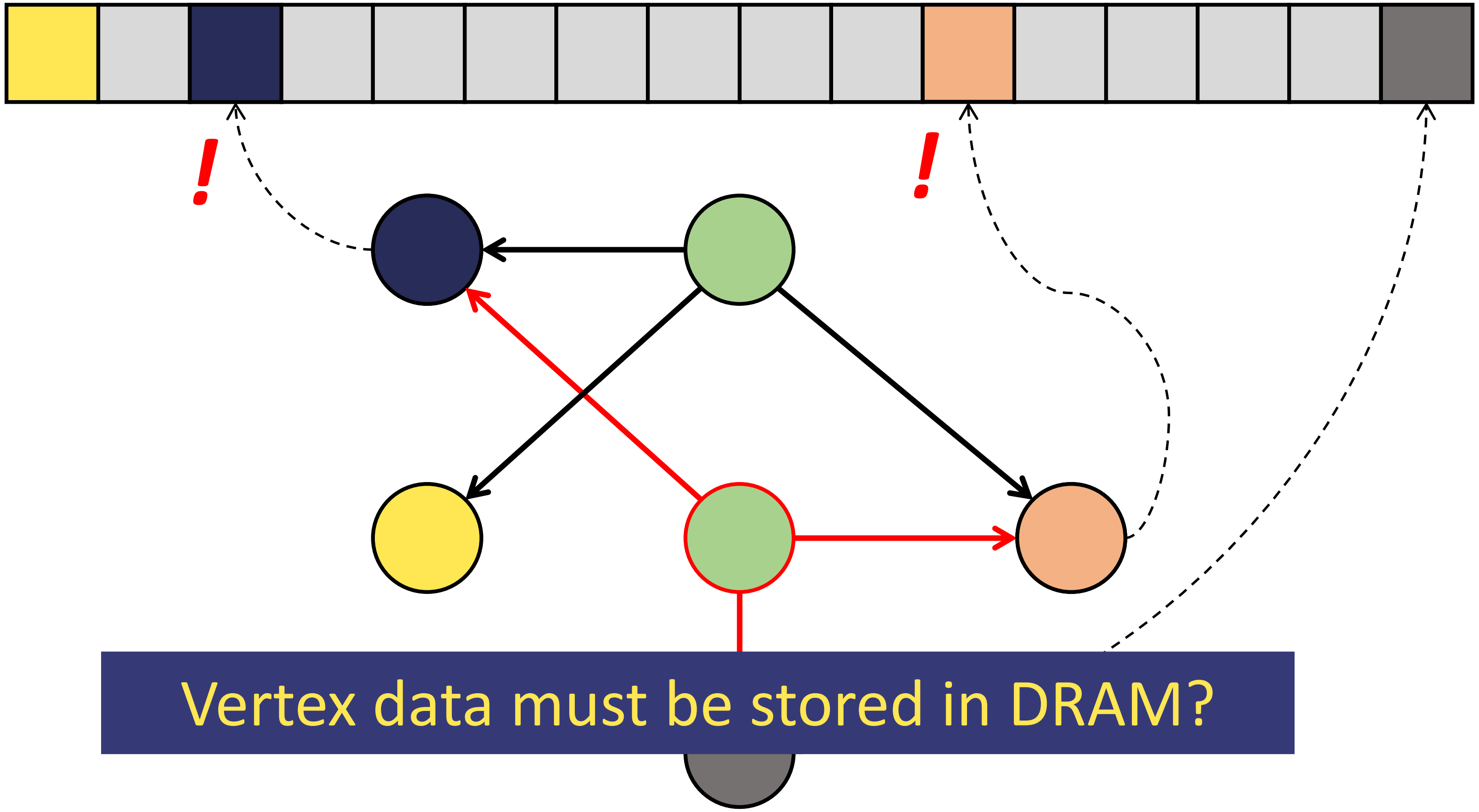
for each v_{src} **in** $ActiveList$ **do**
  for each $e(v_{src}, v_{dst})$ **in** G **do**
    ev = **edge_program** $(v_{src}.val, e.weight)$
    $\boxed{v_{dst}.next\_val}$ = **vertex_update**$(\boxed{v_{dst}.next\_val}, ev)$
  end for
end for

Random read-modify update into vertex data

Random Access Within an Active Vertex



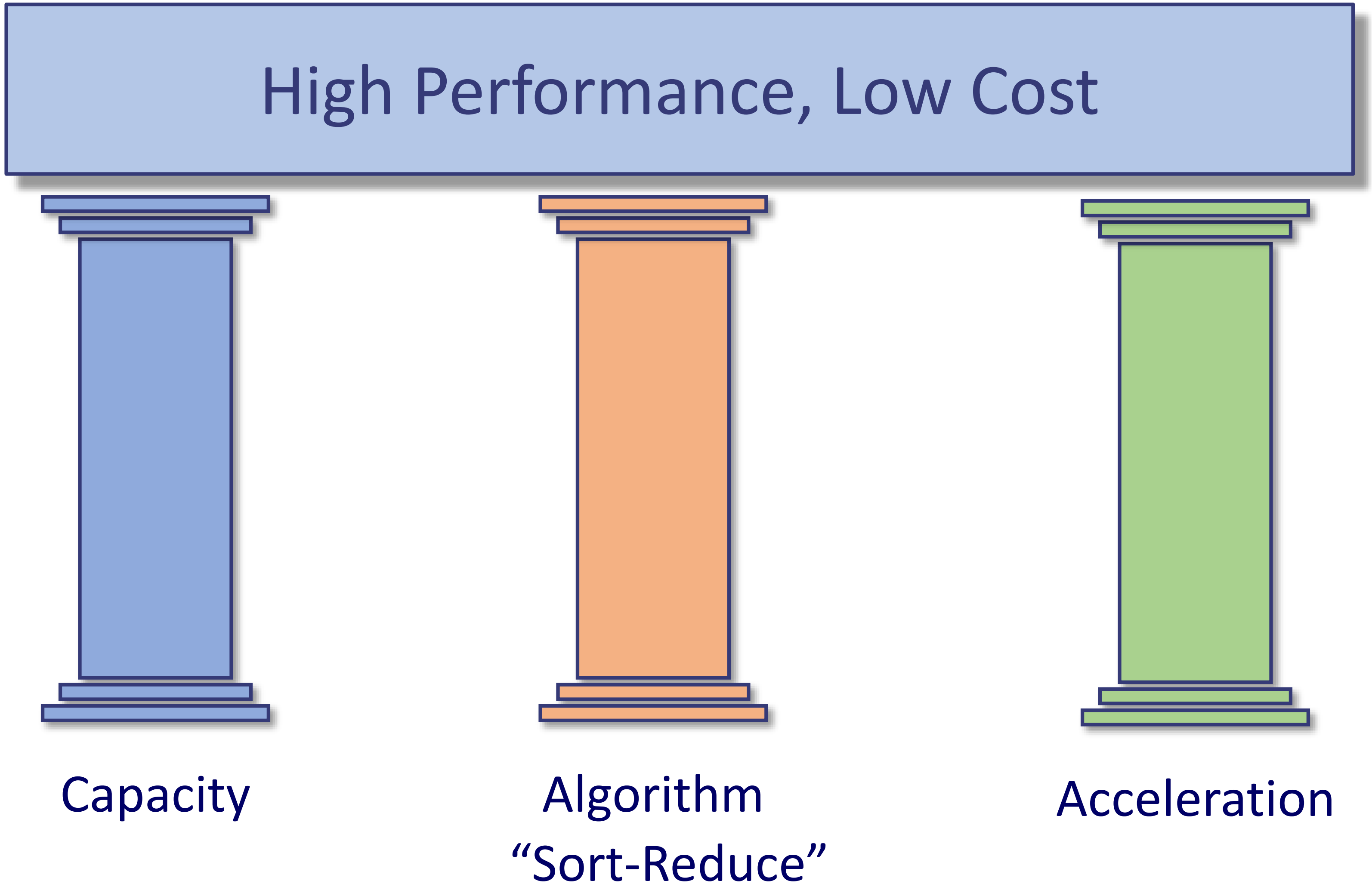
Random Access Across Active Vertices



Vertex data must be stored in DRAM?

Data size and irregularity limit caching effectiveness

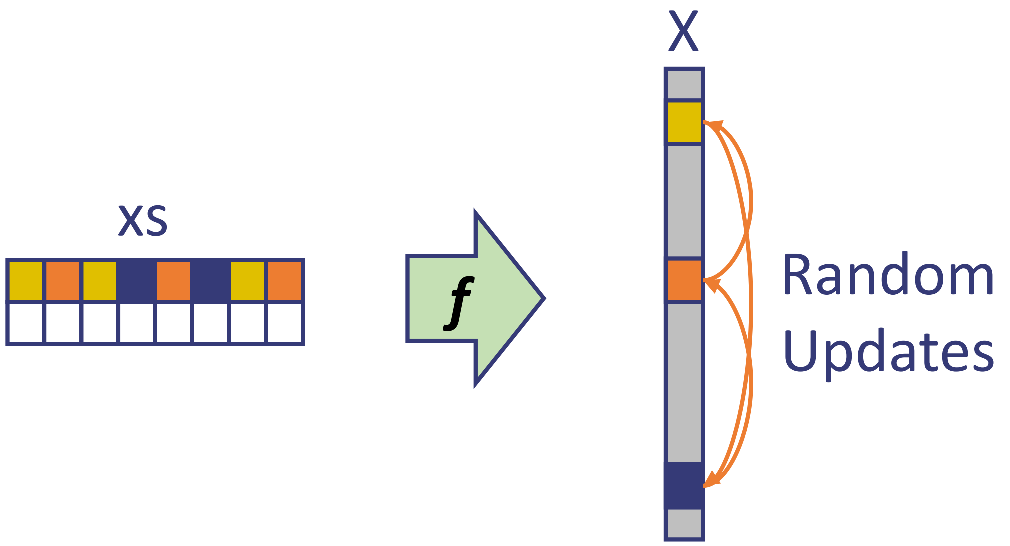
The Three Pillars of Competitive Flash-Based Analytics



General Problem of Irregular Array Updates

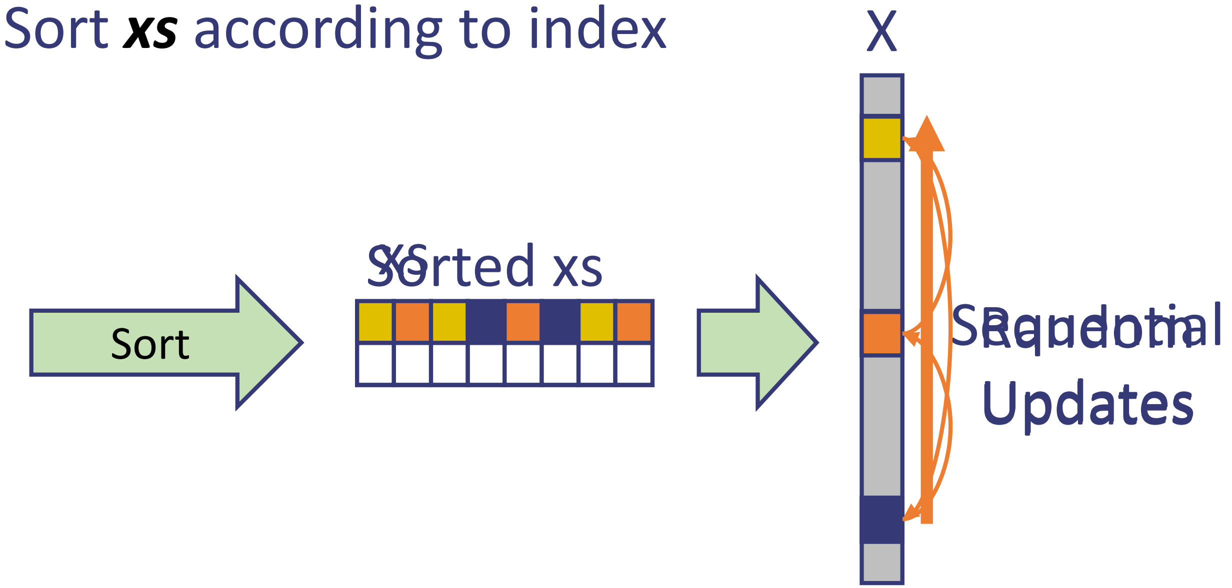
$$\textbf{For each} < idx, arg > \textbf{in } xs:$$
$$x[idx] = \boldsymbol{f}(x[idx], arg)$$

Updating an array ***x***
with a stream of update requests ***xs***
and update function ***f***



Solution Part One - Sort

Sort ***xs*** according to index



Much better than naïve random updates

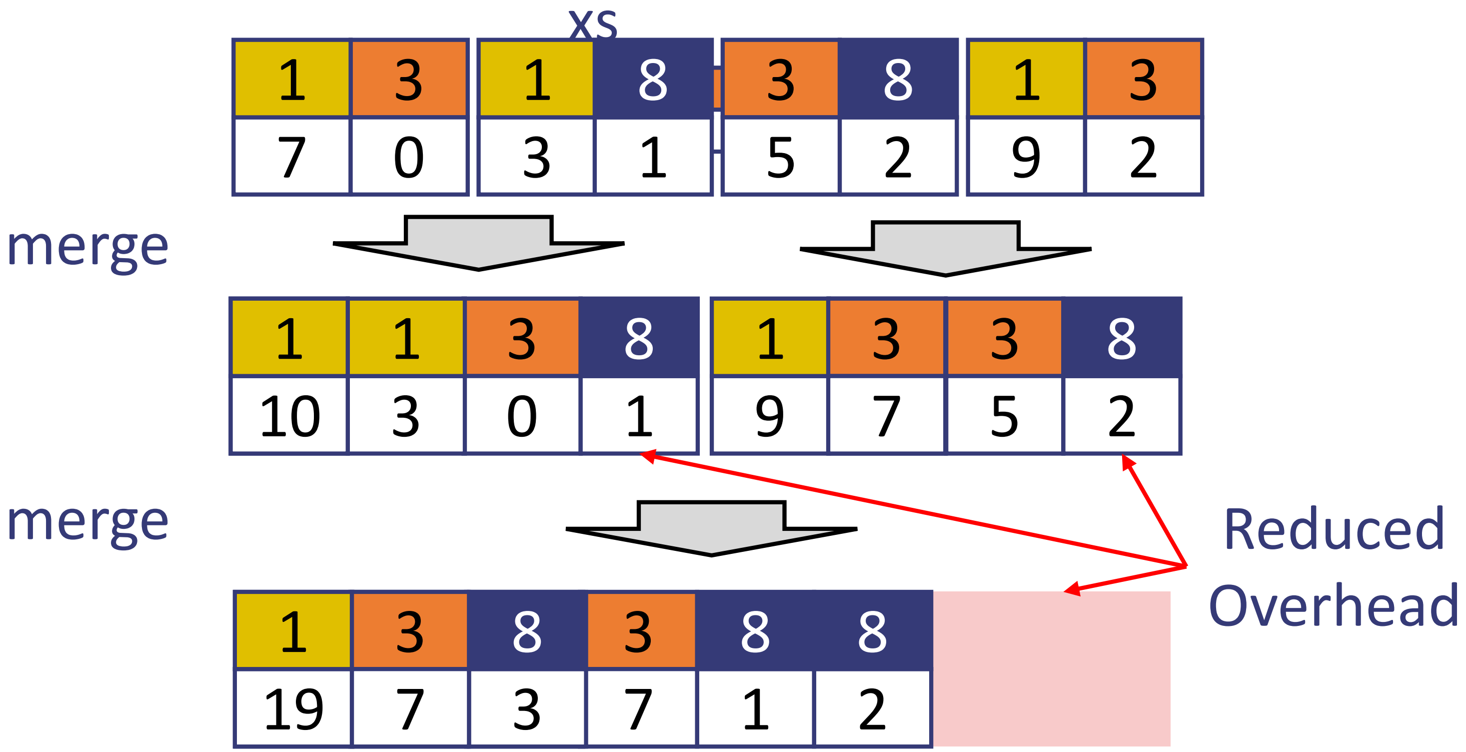
Terabyte graphs can generate terabyte logs

Significant sorting overhead

Solution Part Two - Reduce

Associative update function ***f*** can be interleaved with sort

e.g., (A + B) + C = A + (B + C)



Removing Random Access Using Sort-Reduce

for each v_{src} **in** $ActiveList$ **do**

  for each $e(v_{src}, v_{dst})$ **in** G **do**

    $ev =$ **edge_program** $(v_{src}.val, e.weight)$

    $list.append(v_{dst}, ev)$

    $v_{dst}.next\_val =$ **vertex_update**$(v_{dst}.next\_val, ev)$

  end for

end for

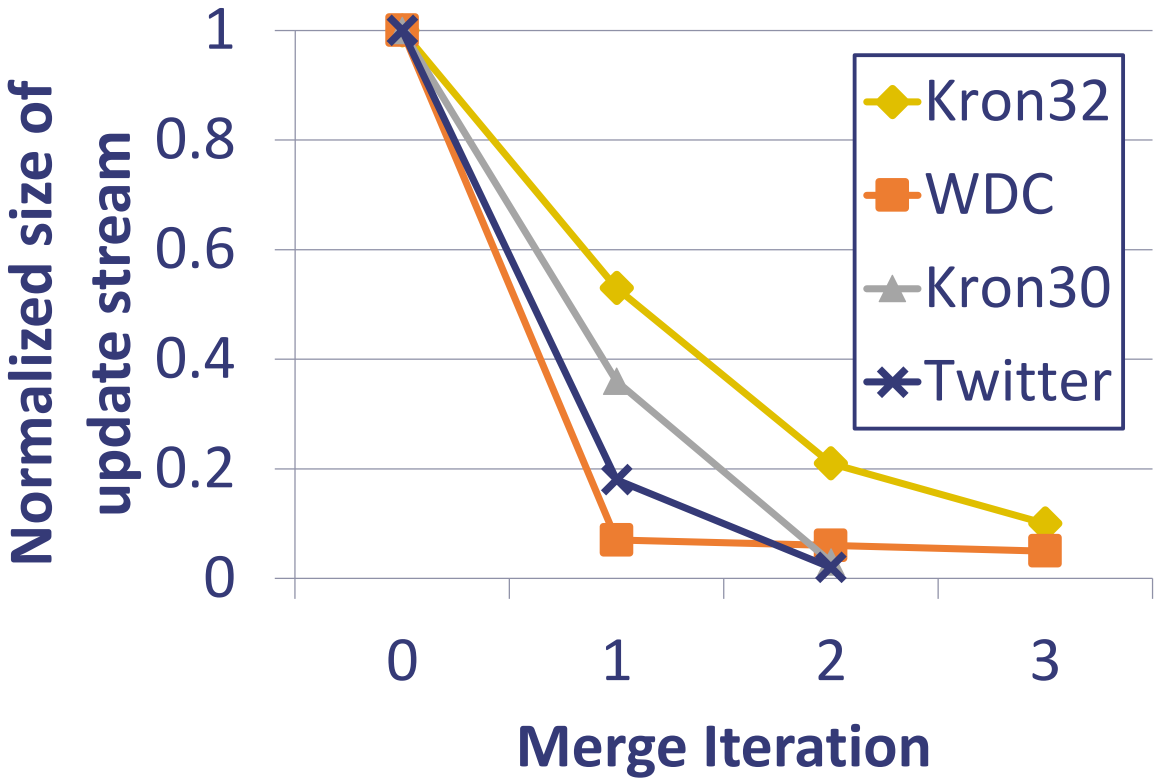
$v = SortReduce_{\mathbf{vertex\_update}}(list)$

No more random access!

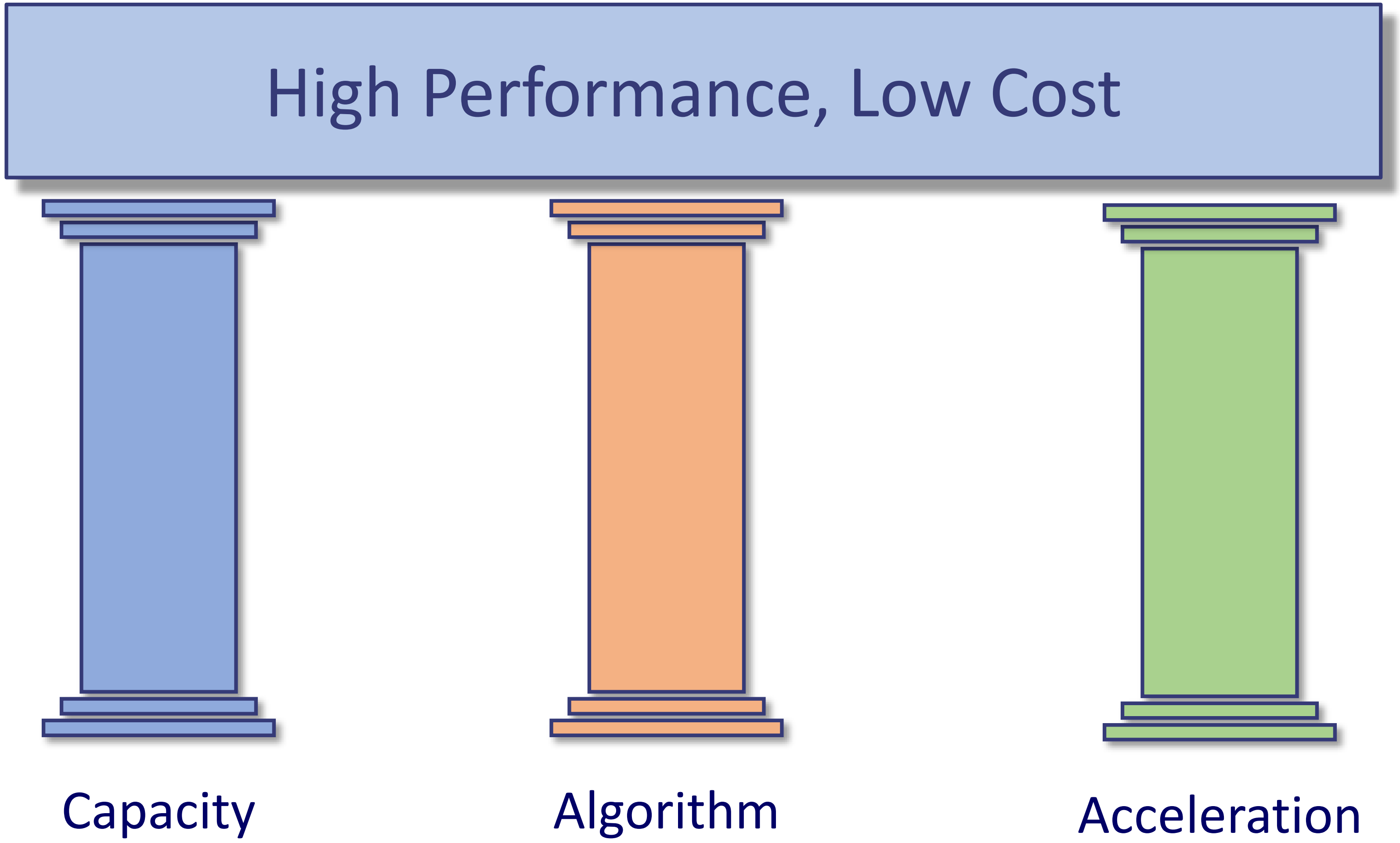
Associativity requirement is not very restrictive
(CombBLAS, PowerGraph, Mosaic, …)

Big Benefits from Interleaving Reduction

Ratio of data copied at each sort phase

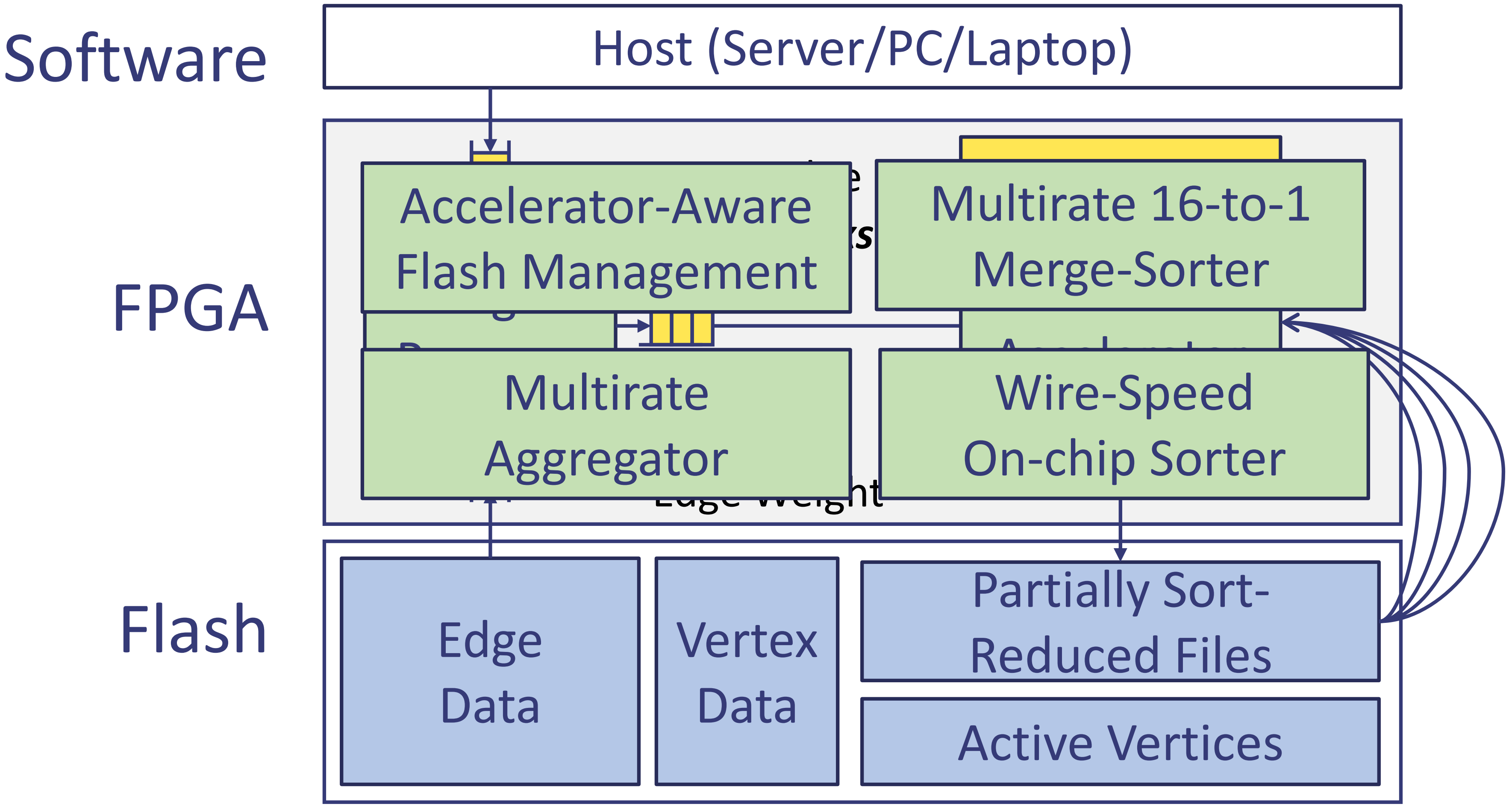


The Three Pillars of Competitive Flash-Based Analytics



Accelerated Graph Analytics Architecture

In-storage accelerator reduces data movement and cost



Evaluated Graph Analytic Systems

In-memory | Semi-External | External

GraphLab (***IN***)

FlashGraph (***SE1***)
X-Stream (***SE2***)

GraphChi (***EX***)
GraFSoft
GraFBoost

“Distributed GraphLab: a framework for machine learning and data mining in the cloud,” VLDB 2012
“FlashGraph: Processing billion-node graphs on an array of commodity SSDs,” FAST 2015
“X-Stream: edge-centric graph processing using streaming partitions,“ SOSP 2013
“GraphChi: Large-scale graph computation on just a PC,“ USENIX 2012

Evaluation Environment

32-core Xeon
128 GB RAM
5x 0.5TB PCIe Flash

$8000

All software experiments

+

4-core i5
4 GB RAM

$400

+

Virtex 7 FPGA
1TB custom flash
1GB on-board RAM

$1000

Evaluation Result Overview

	In-memory	Semi-External	External	GraFBoost
Large graphs:	**Fail**	**Fail**	**Slow**	**Fast**
Medium graphs:	**Fail**	**Fast**	**Slow**	**Fast**
Small graphs:	**Fast**	**Fast**	**Slow**	**Fast**

GraFBoost has very low resource requirements

Memory, CPU, Power

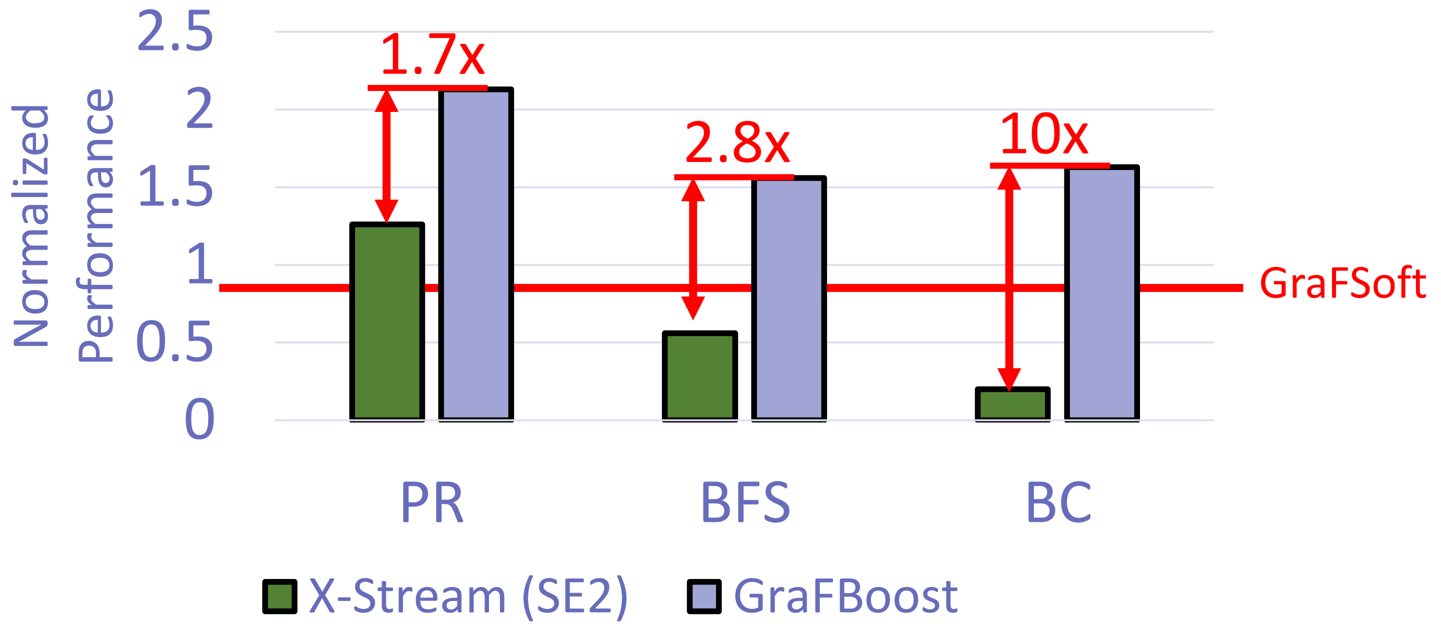
Results with a Large Graph: Synthetic Scale 32 Kronecker Graph

0.5 TB in text, 4 Billion vertices

GraphLab (***IN***) out of memory

FlashGraph (***SE1***) out of memory

GraphChi (***EX***) did not finish

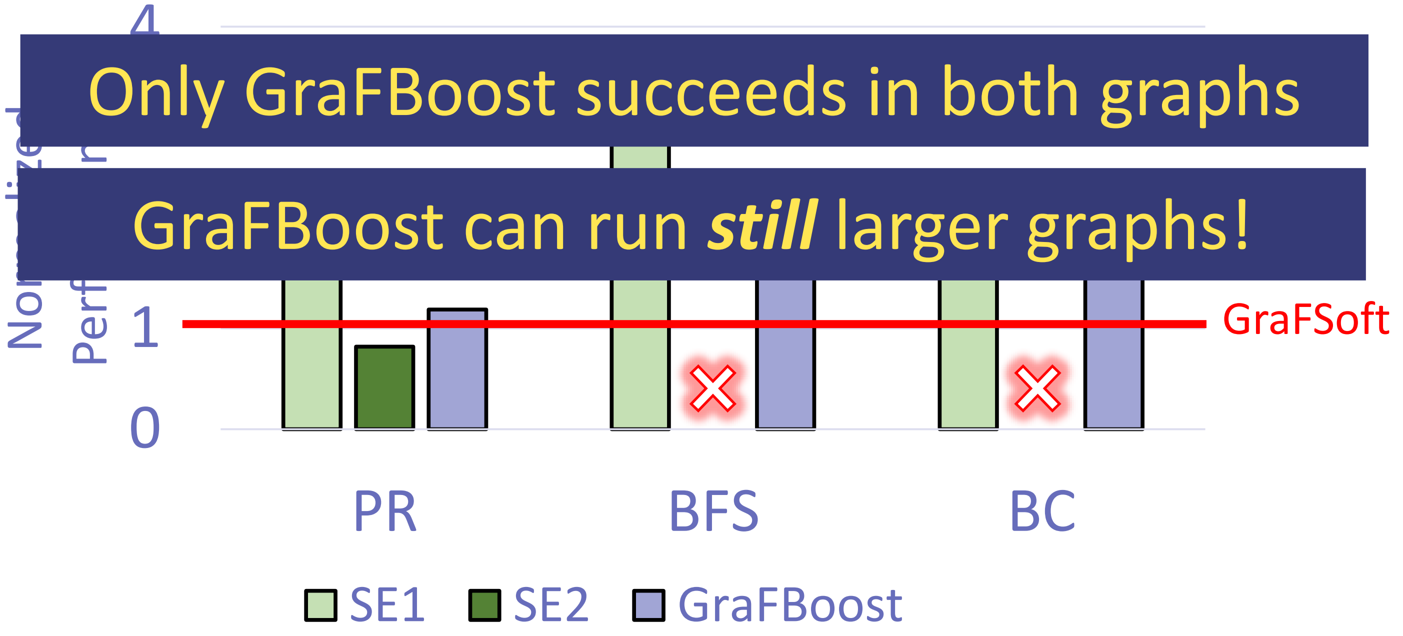


Results with a Large Graph: Web Data Commons Web Crawl

2 TB in text, 3 Billion vertices

GraphLab (***IN***) out of memory

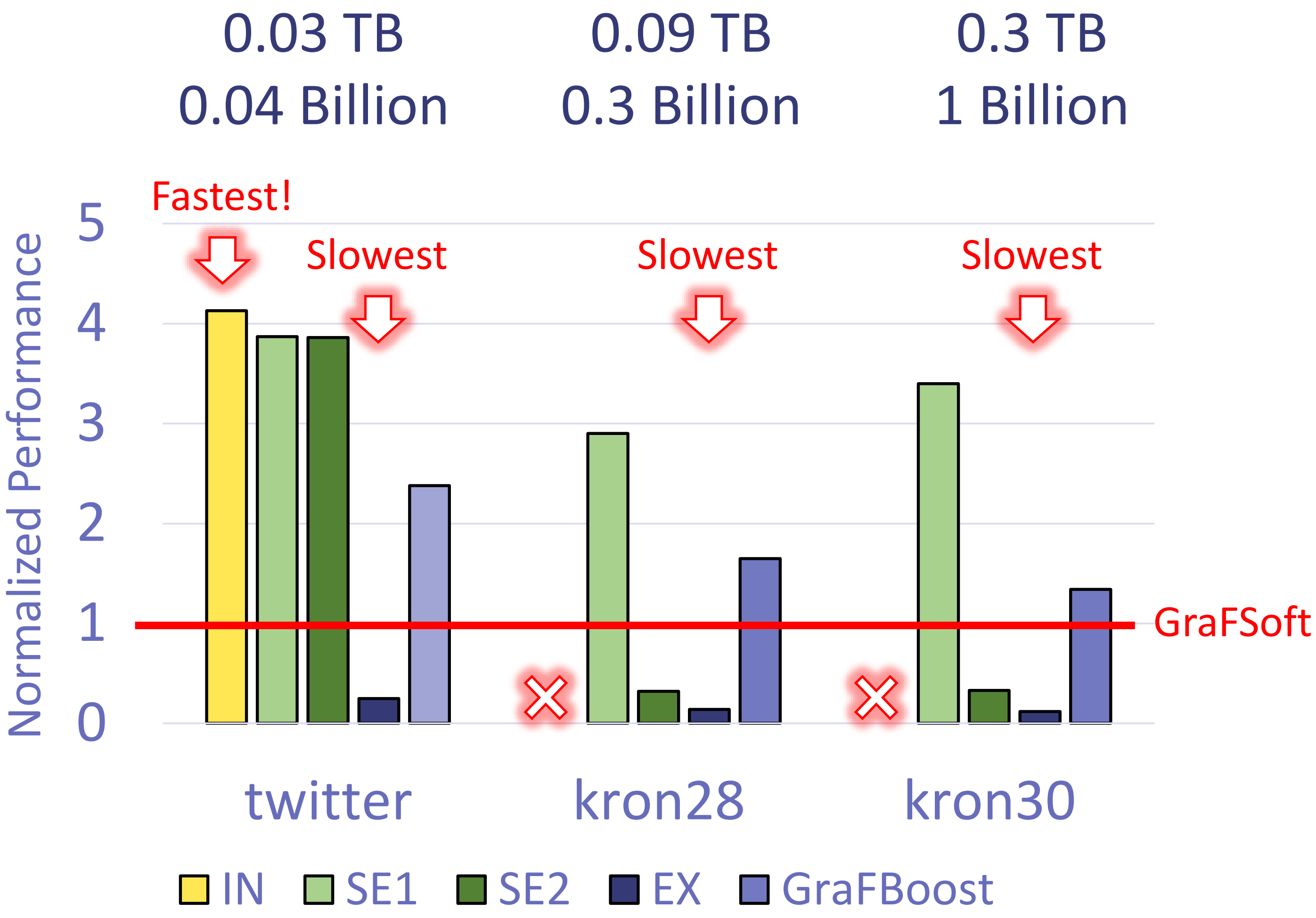
GraphChi (***EX***) did not finish



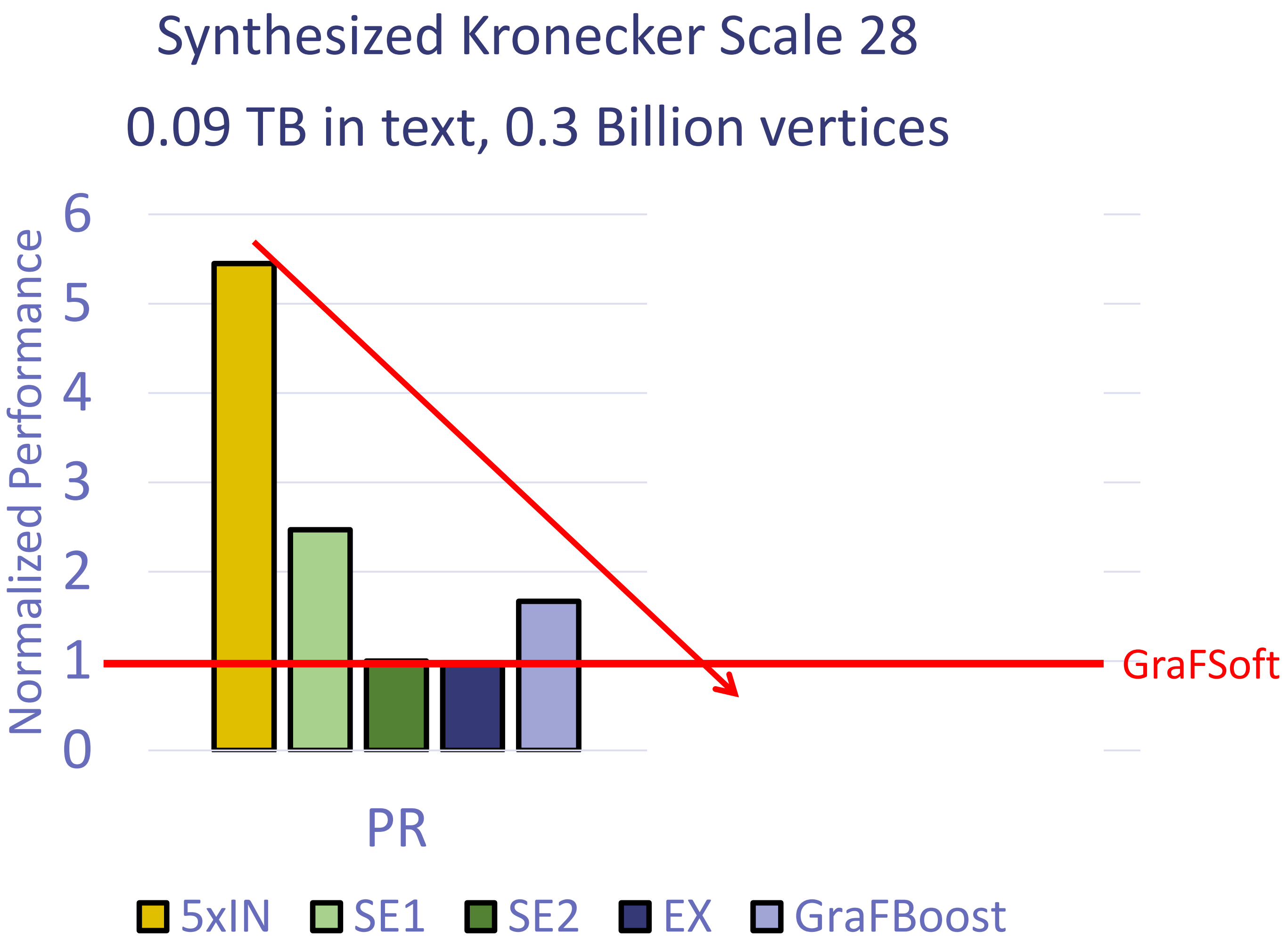
Only GraFBoost succeeds in both graphs

GraFBoost can run ***still*** larger graphs!

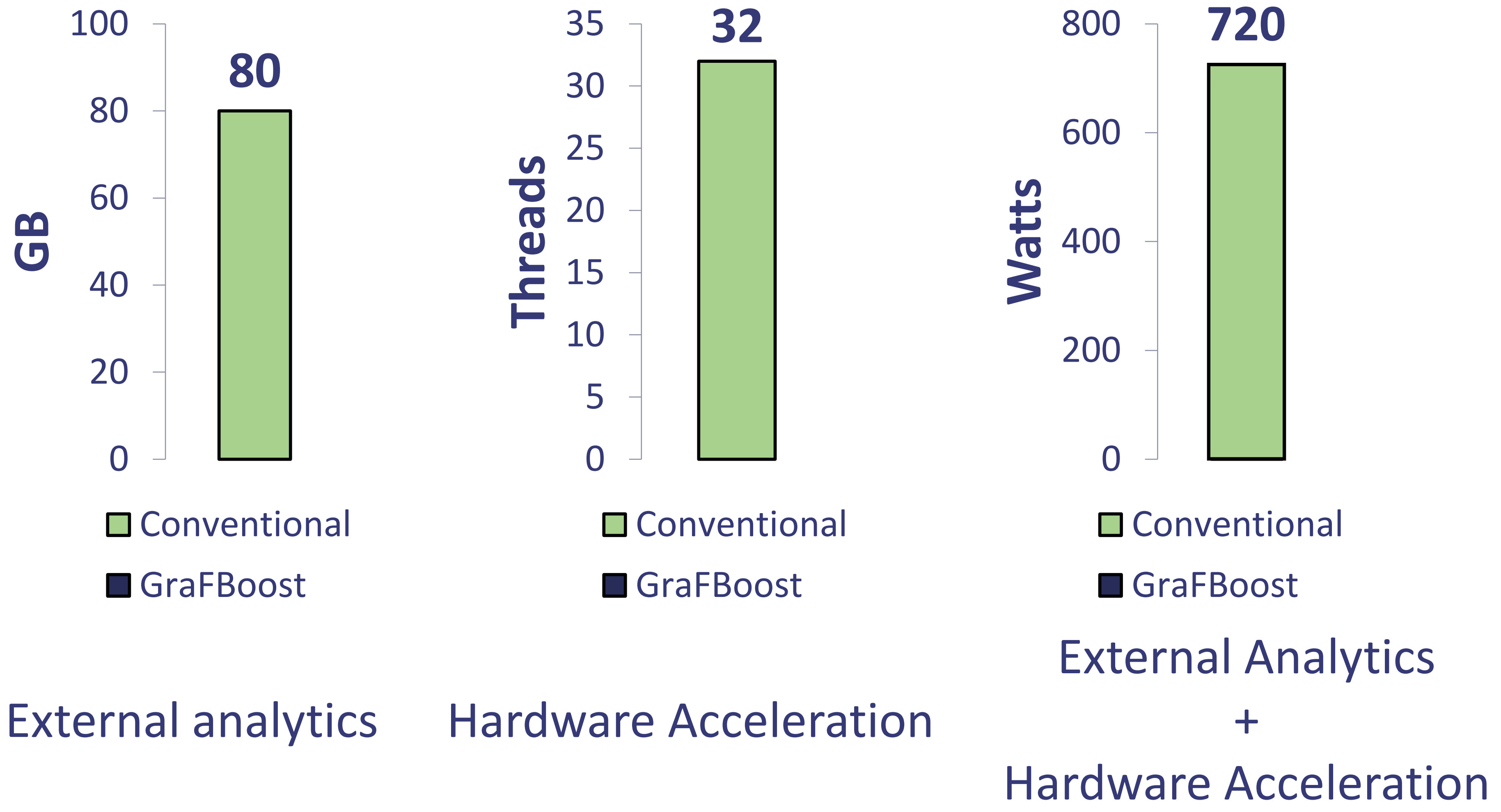
Results with Smaller Graphs: Breadth-First Search



Results with a Medium Graph: Against an In-Memory Cluster



GraFBoost Reduces Resource Requirements



Evaluation Result Recap

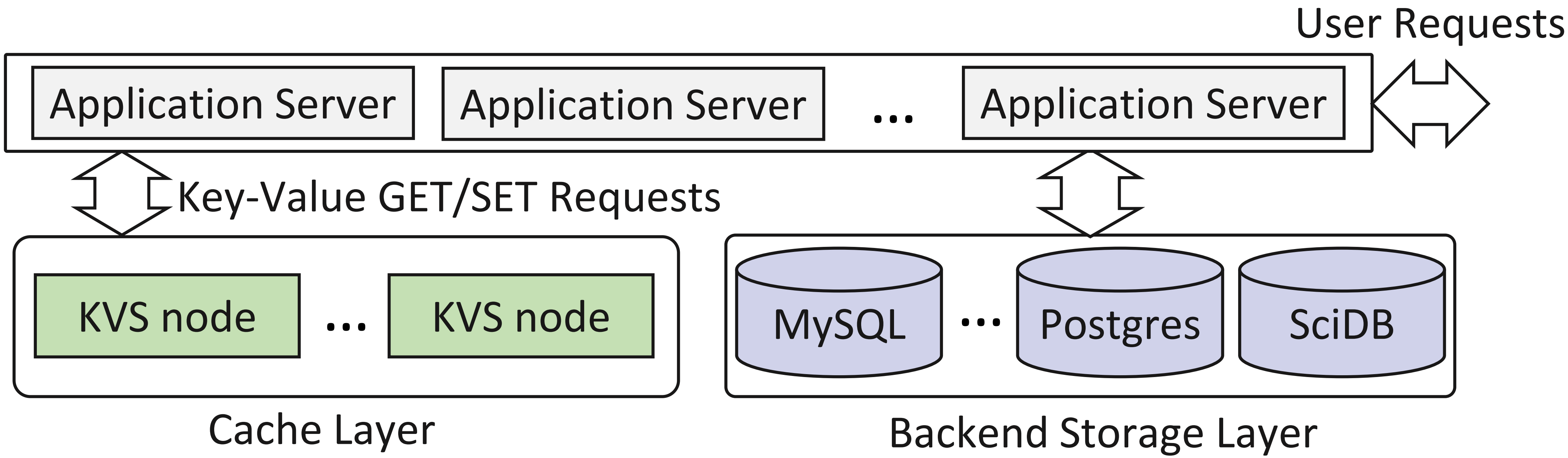
	In-memory	Semi-External	External	GraFBoost
Large graphs:	Fail	Fail	Slow	Fast
Medium graphs:	Fail	Fast	Slow	Fast
Small graphs:	Fast	Fast	Slow	Fast

GraFBoost has very low resource requirements

Memory, CPU, Power

Contents

Key-Value Cache in the Data Center



Transactional DB operations can become bottleneck

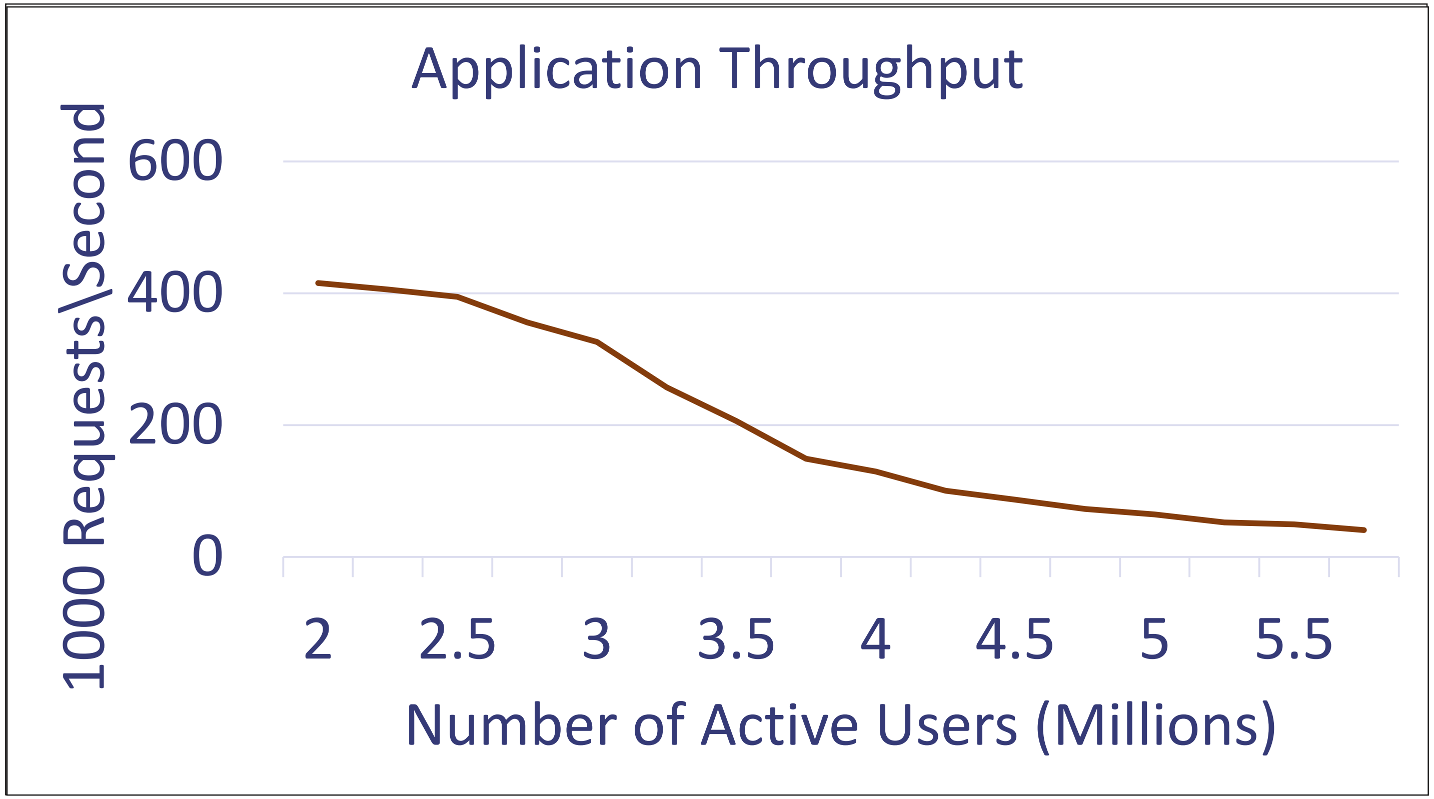
Key-Value caches like memcached cache DB query results

Facebook reported 300 TB of memcached capacity (2010)

Cached Application Performance Example

Using the BG social network benchmark*

MySQL backend and 16 GB memcached



*Open-source project by Twitter, Inc.

Software KV Caches Are Fast, but Not Power-Efficient

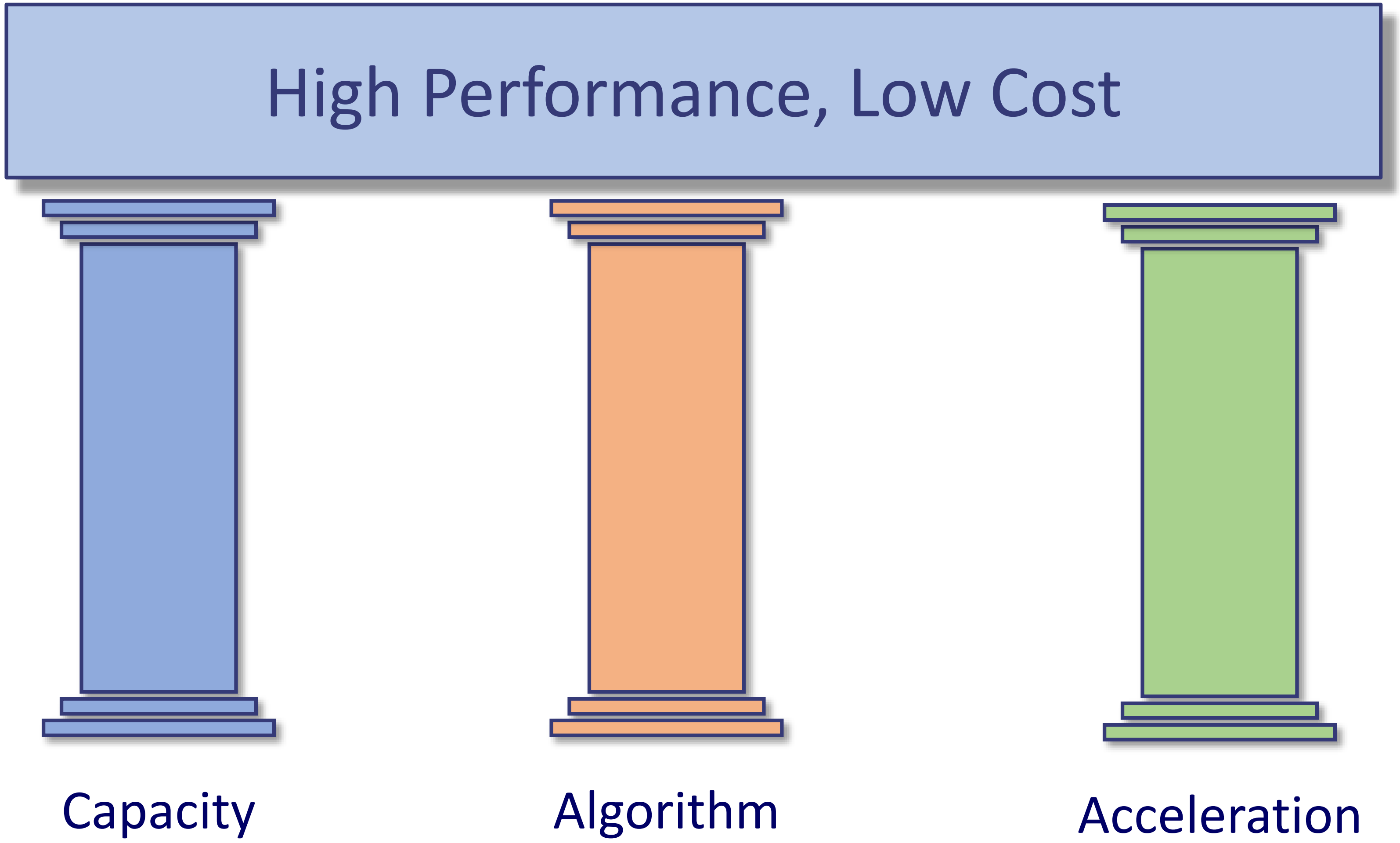
Achieving high throughput requires a lot of resources

MICA	Mega-KV
120 Million Requests Per Second	160 Million Requests Per Second
24 cores, 12 10GbE	2 GTX 780 GPUs
400 W	900 W

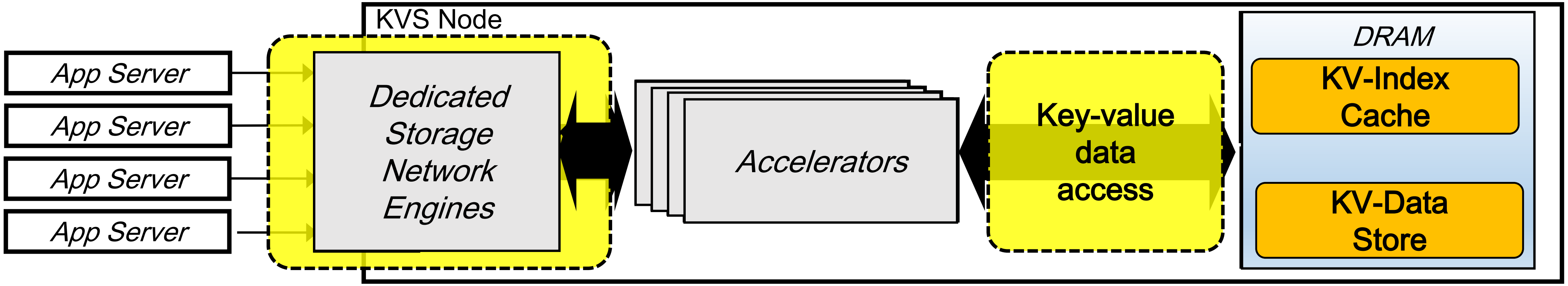
Superscalar OoO pipeline is underutilized for KVS

Only 3MB LLC is needed to sustain MICA’s throughput

The Three Pillars of Competitive Flash-Based Analytics



BlueCache: Flash-based KVS Architecture



Store KV pairs in flash storage

Pipelined KV cache accelerators

Hardware accelerated dedicated storage network engines

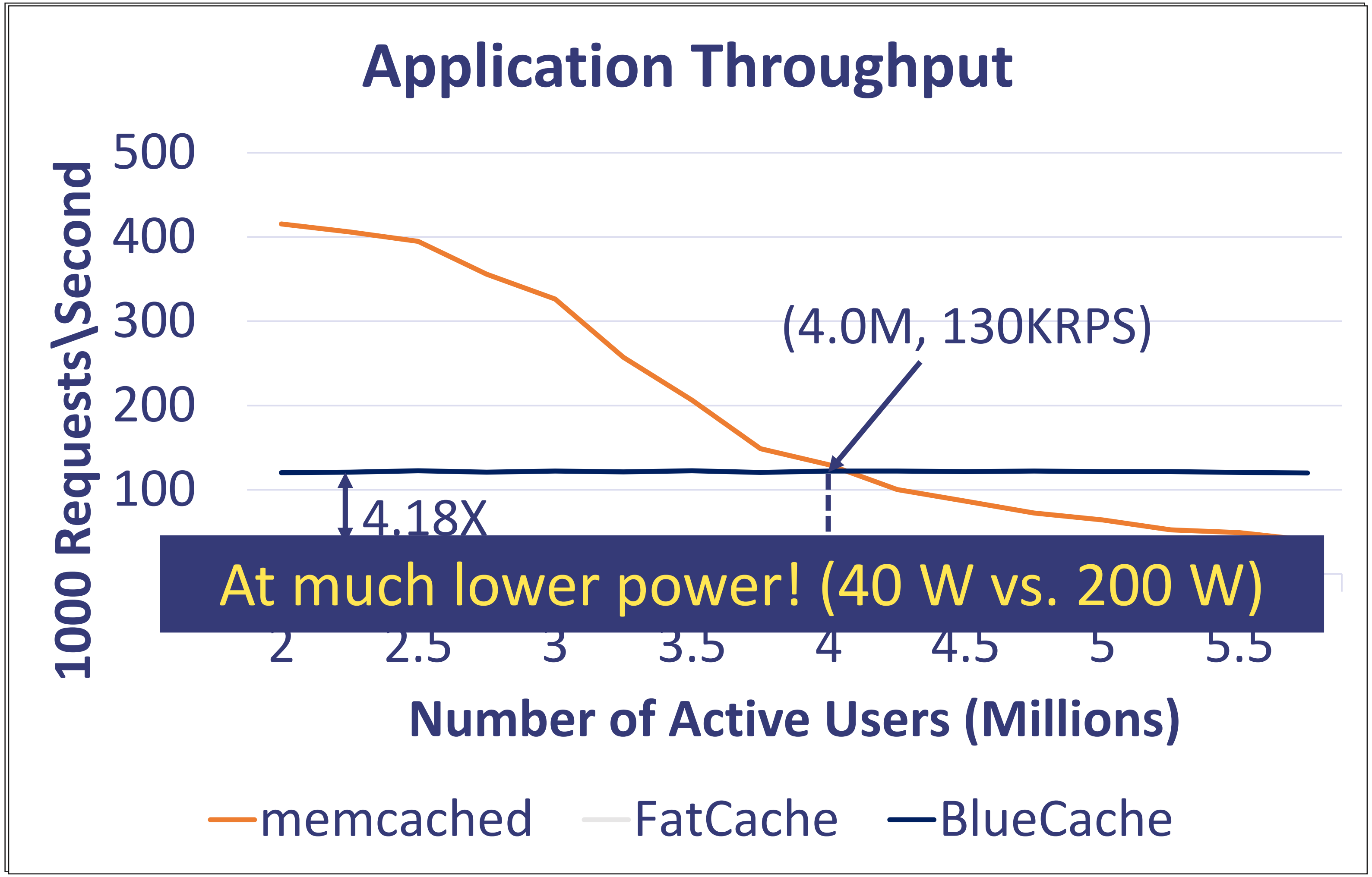
Log-structured KV data store

Evaluation Setup

Frontend	BG social network benchmark server		
Backend	MySQL server 32-core Xeon server 64 GB DRAM 1.5 TB PCIe SSDs		
KV Cache	***memcached*** In-memory 48 Cores 16 GB DRAM	***FatCache**** Flash-based 48 Cores 1 GB DRAM 0.5 TB PCIe Flash	***BlueCache*** Flash-based with acceleration 1 GB DRAM 0.5 TB PCIe Flash

*Open-source project by Twitter, Inc.

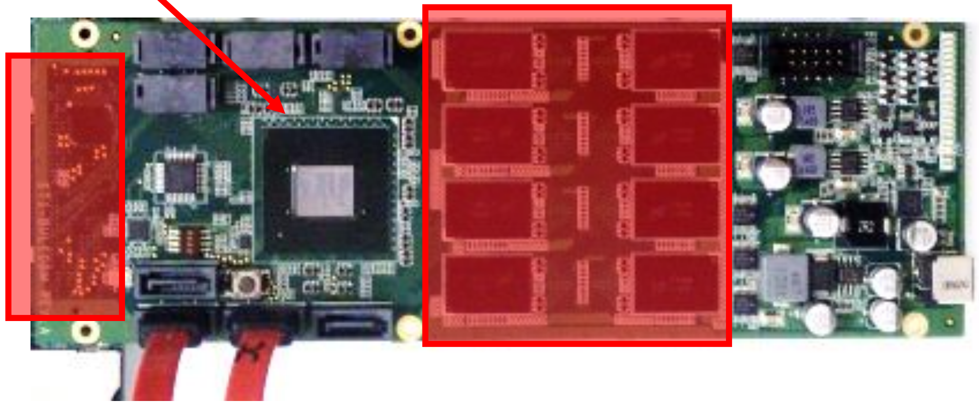
Performance Evaluation with More Users



Contents

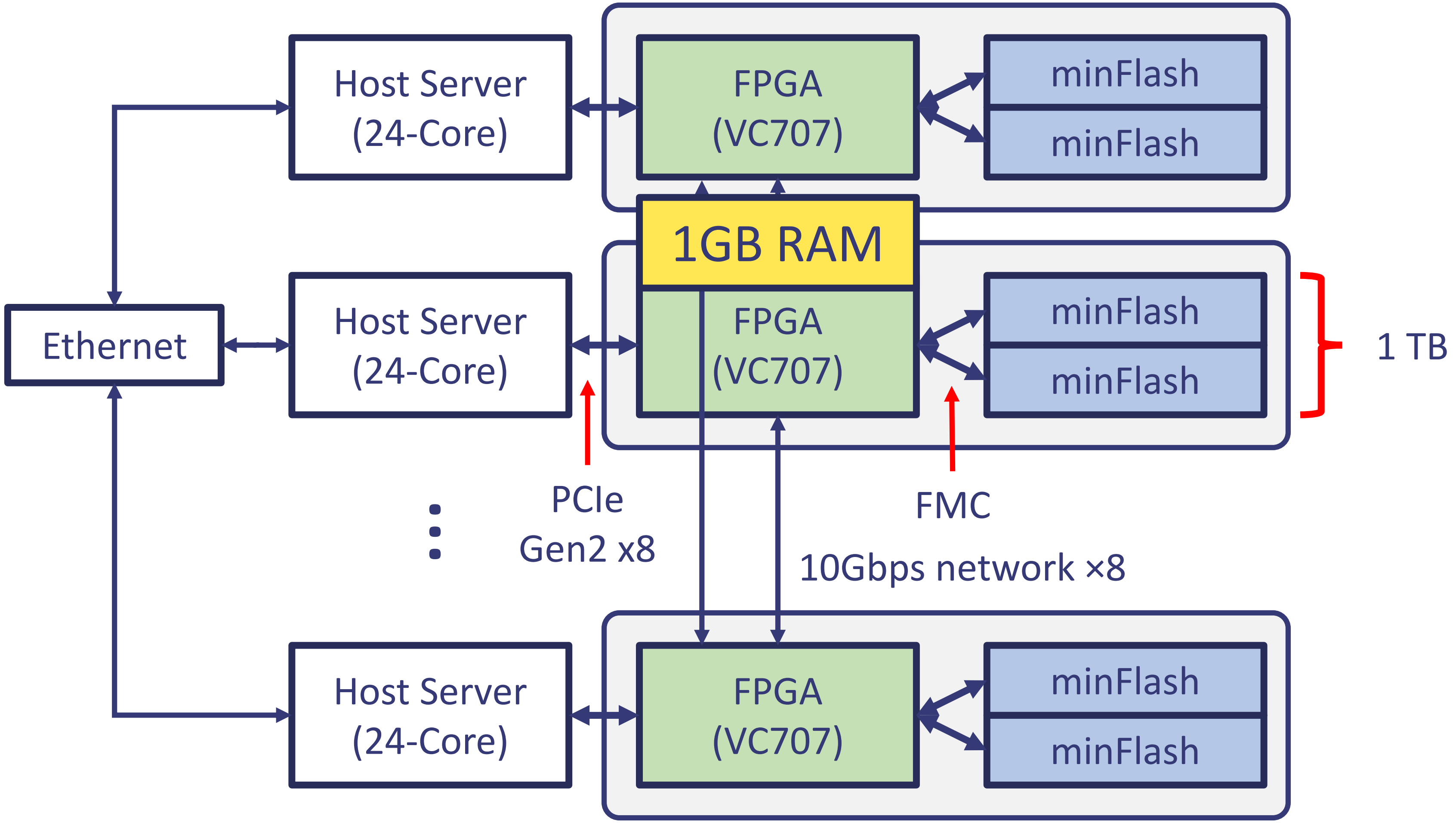
Our Custom Flash Card for Distributed Accelerated Flash Architectures

- Requirement 1: Modify flash management
- Requirement 2: Dedicated storage-area network
- Requirement 3: In-storage hardware accelerator



"minFlash: A Minimalistic Clustered Flash Array," DATE 2016

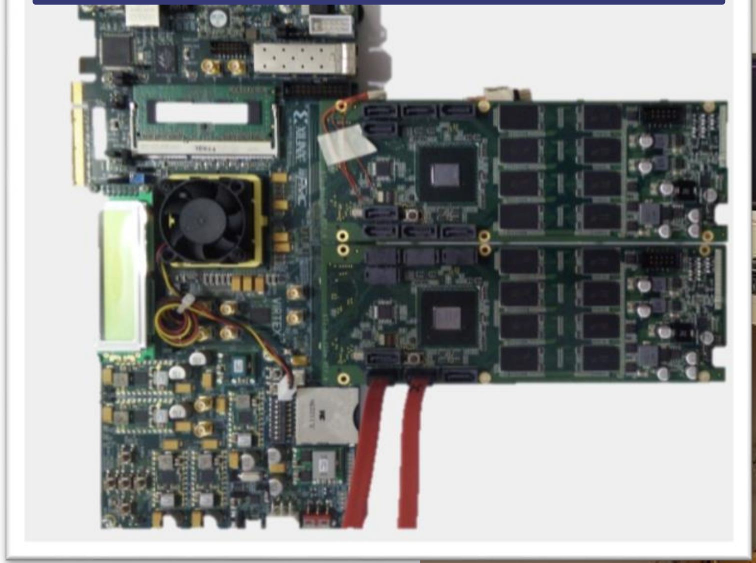
BlueDBM Cluster Architecture



Uniform latency of 100 µs!

The BlueDBM Cluster

BlueDBM Storage Device



Our Research Enabled by BlueDBM

1. “Scalable Multi-Access Flash Store for Big Data Analytics,” FPGA 2012
2. “BlueDBM: An Appliance for Big Data Analytics,” ISCA 2015
3. “A Transport-Layer Network for Distributed FPGA Platforms,” FPL 2015
4. “Large-scale high-dimensional nearest neighbor search using Flash memory with in-store processing,” ReConFig 2015
5. “minFlash: A Minimalistic Clustered Flash Array,” DATE 2016
6. “Application-managed flash,” FAST 2016
7. “In-Storage Embedded Accelerator for Sparse Pattern Processing,” HPEC 2016
8. “Terabyte Sort on FPGA-Accelerated Flash Storage,” FCCM 2017
9. “BlueCache: A Scalable Distributed Flash-based Key-value Store,” VLDB 2017
10. “GraFBoost: Using accelerated flash storage for external graph analytics,” ISCA 2018
11. “NoHost: Software-defined Network-attached KV Drives,” (Under Review)

Future Work

Next generation of BlueDBM

Newer SSDs are much faster than 2.4 GB/s

Prototype flash chips are aging

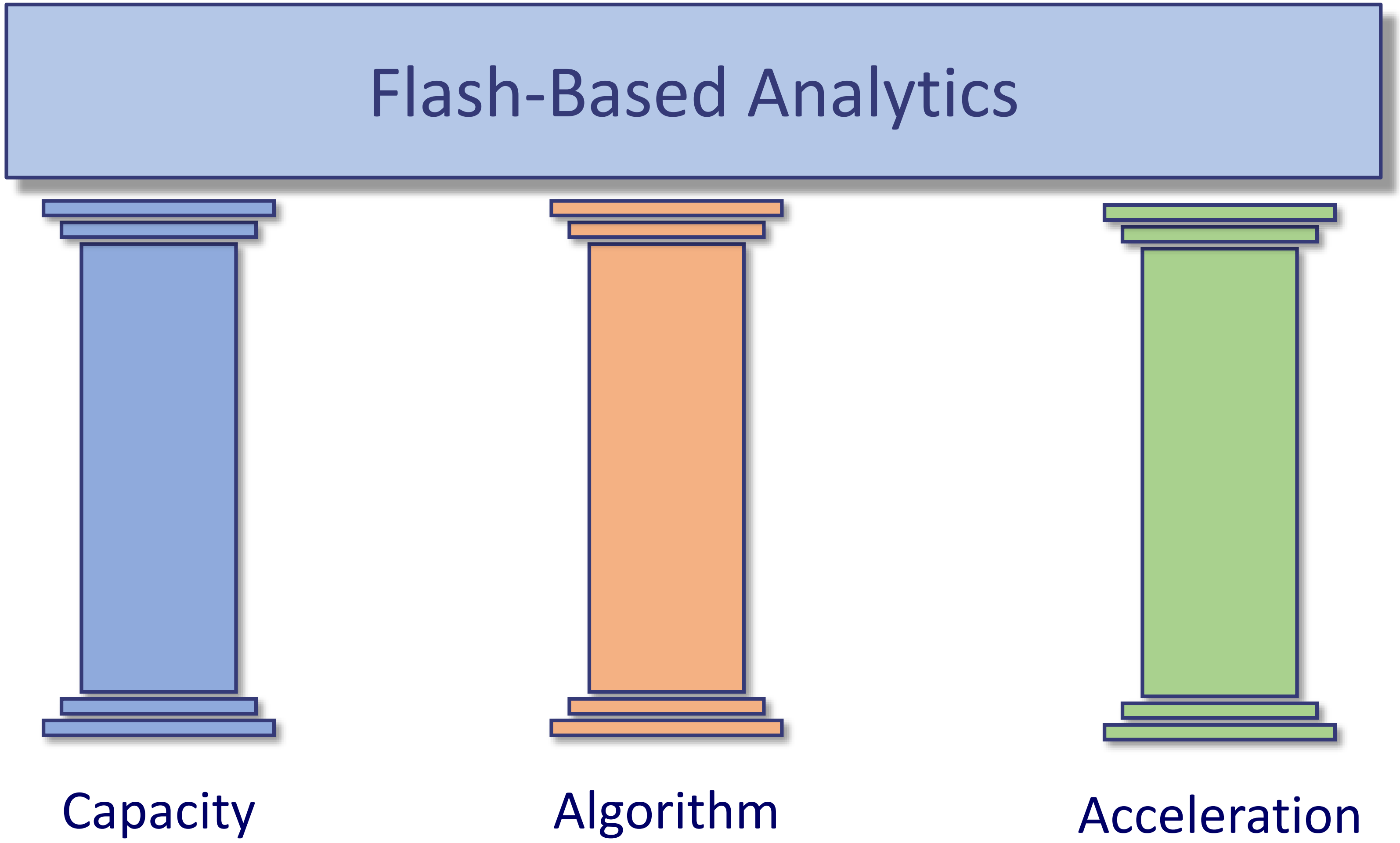
More applications using sort-reduce

Bioinformatics collaboration with Barcelona Supercomputering Center

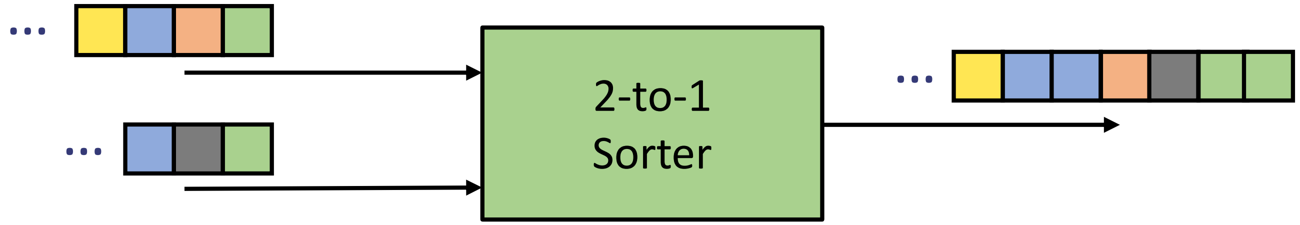
More applications using accelerated flash

SQL acceleration collaboration with Samsung

Thank you



A Baseline Hardware Merge Sorter

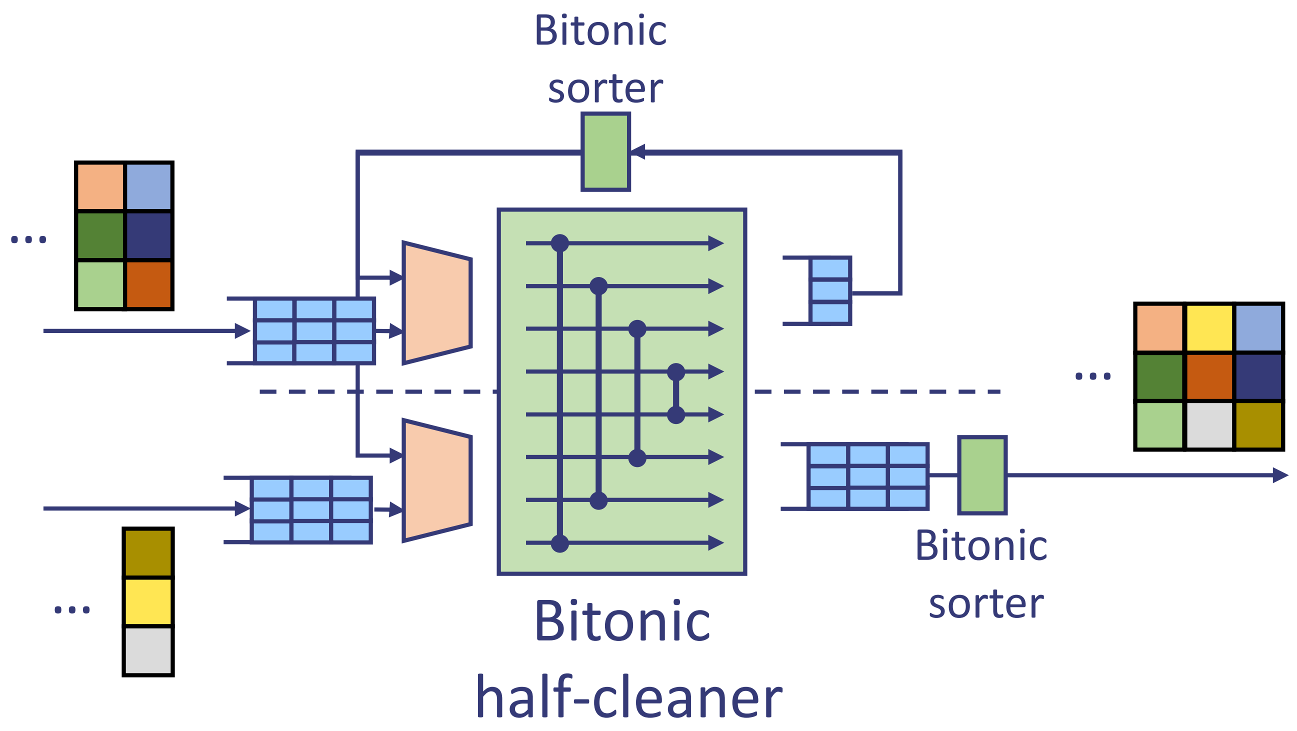


Sorter emits one item at every cycle

Easy to become bottleneck

High-Throughput Hardware Sorter using Sorting Networks

Sorter emits sorted tuple at every cycle

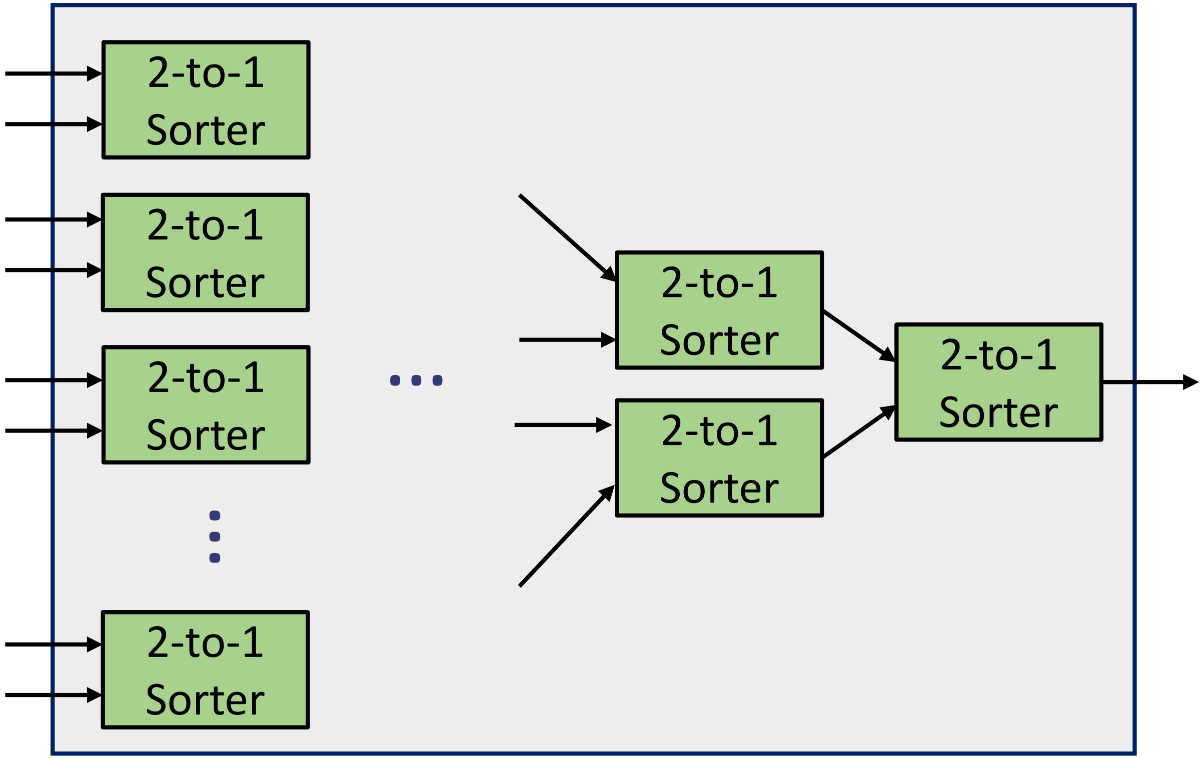


Merge-Sorts at constant 4 GB/s

"Terabyte Sort on FPGA-Accelerated Flash Storage," FCCM 2016

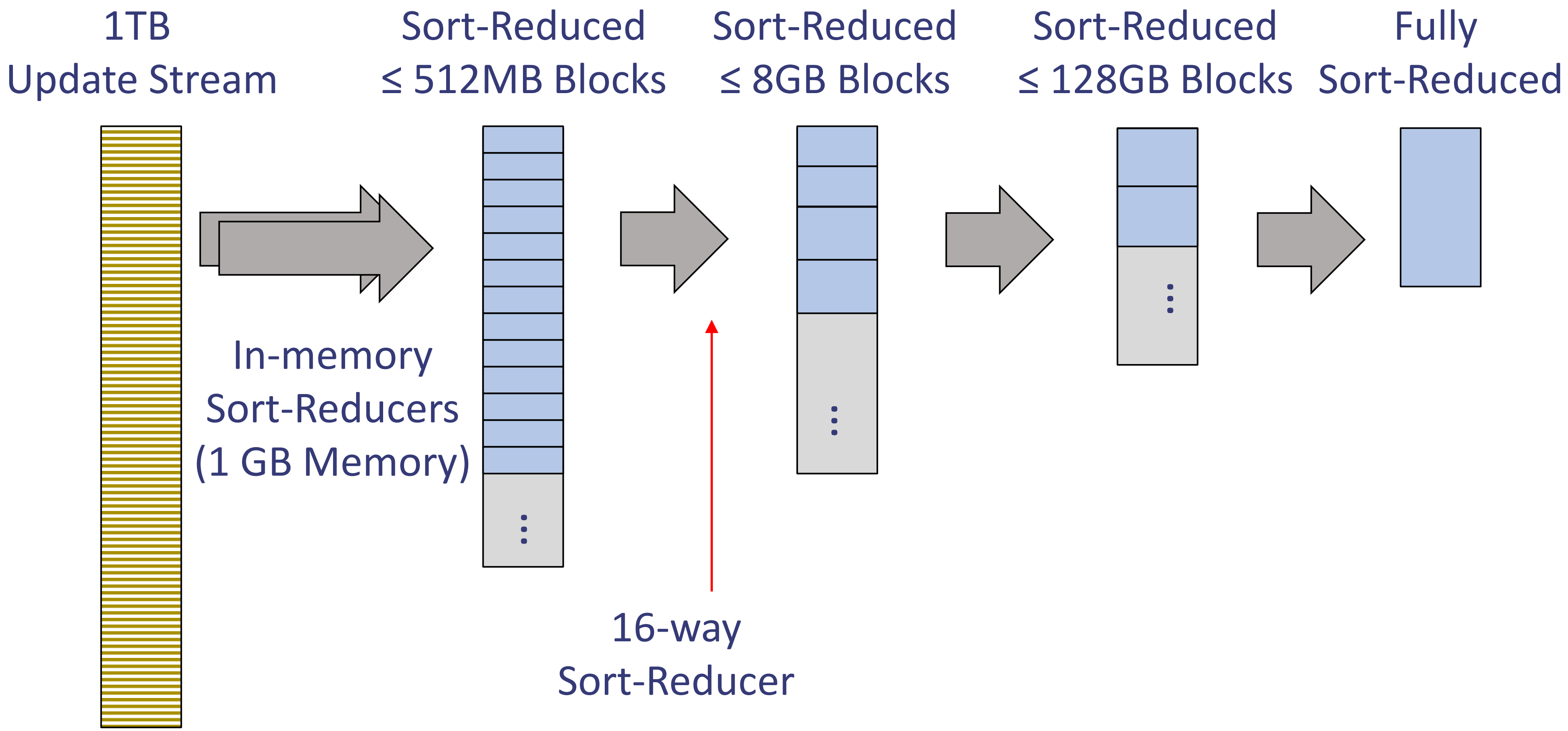
Hardware 16-to-1 Sorter Reduces Sorting Passes

Constructed as a pipelined tree of 2-to-1 Sorters

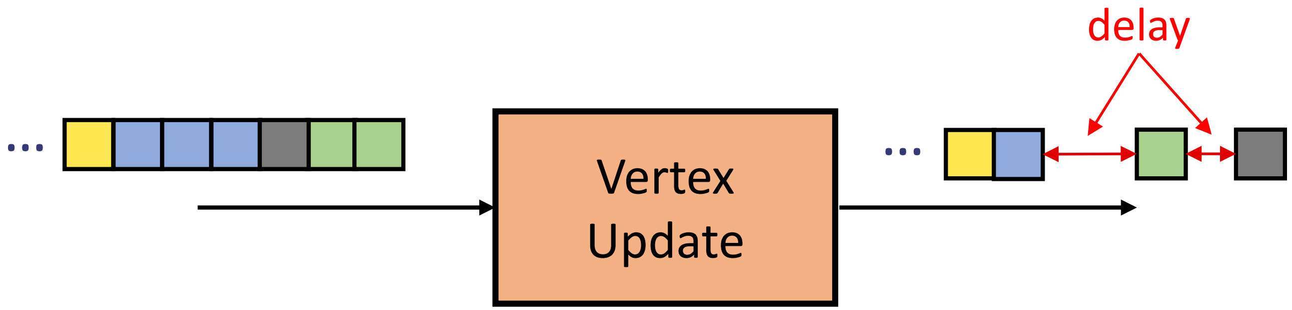


"Terabyte Sort on FPGA-Accelerated Flash Storage," FCCM 2016

Sort-Reduce Process



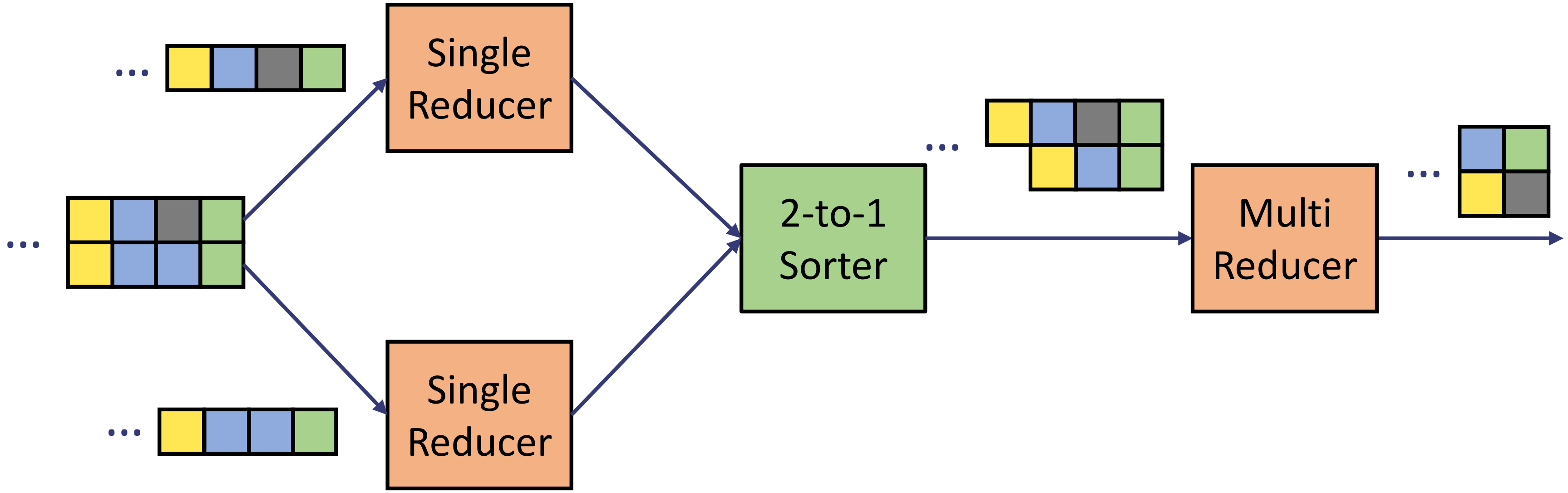
A Baseline Hardware Reducer



Reducer consumes up to one item at every cycle

Vertex update latency impacts performance

Wire-Speed Reducer Using Sorting Networks



Single Reducers achieve single element wire-speed

Multi Reducer uses sorting networks to achieve multi-rate

Reduces at constant 4 GB/s

"Wire-Speed Accelerator for Aggregation Operations,“ (Under Review)