

# The next frontier: GPUs

---

ANIL SHANBHAG

---

**Intelligent Machines**

# Moore's Law Is Dead. Now What?

Shrinking transistors have powered 50 years of advances in computing—but now other ways must be found to make computers more capable.

- MIT Tech Review, May 2016

# Variety of hardware

Common theme: Large number of cores with SIMD/SIMT



**Unveiling Details of Knights Landing**  
(Next Generation Intel® Xeon Phi™ Products)

**Platform Memory:** DDR4 Bandwidth and Capacity Comparable to Intel® Xeon® Processors

**Compute:** Energy-efficient IA cores<sup>2</sup>

- Microarchitecture enhanced for HPC<sup>3</sup>
- **3X Single Thread Performance** vs Knights Corner<sup>4</sup>
- Intel Xeon Processor Binary Compatible<sup>5</sup>

**On-Package Memory:**

- up to **16GB** at launch
- **1/3X the Space**<sup>6</sup>
- **5X Bandwidth** vs DDR4<sup>7</sup>
- **5X Power Efficiency**<sup>6</sup>

*Jointly Developed with Micron Technology*

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. <sup>1</sup>Over 3 Teraflops of peak theoretical double-precision performance based on 16 cores running floating point operations per cycle. <sup>2</sup>Up to 1.75 GHz clock speed, 256-bit floating-point operations per second per cycle. <sup>3</sup>Modified version of Intel® Silvermont microarchitecture currently found in Intel® Atom™ processors. Modifications include AVX512 and 4 threads/core support. <sup>4</sup>Projected peak theoretical single-thread performance relative to 1<sup>st</sup> Generation Intel® Xeon Phi™ Coprocessor 7120P (formerly codenamed Knights Corner). <sup>5</sup>Binary Compatible with Intel Xeon processors using Haswell Instruction Set (except TSX). <sup>6</sup>Projected results based on internal Intel analysis of Knights Landing memory vs Knights Corner (GDDR5). <sup>7</sup>Projected result based on internal Intel analysis of STREAM benchmark using a Knights Landing processor with 16GB of ultra high-bandwidth versus DDR4 memory only with all channels populated.

**Conceptual—Not Actual Package Layout**

A 3D diagram of the Knights Landing processor package. It shows a central blue cube representing the "Processor Package" containing "Intel® Silvermont Arch. Enhanced for HPC" cores. Above the package are "Platform Memory" blocks, and below it is the "Integrated Fabric". A green arrow points from the text "Jointly Developed with Micron Technology" to the package diagram.

# Introduction to NVIDIA GPUs in Azure

Introducing Amazon EC2 P2 Instances, the largest GPU-Powered virtual machine in the cloud

GPUs are now available for Google Compute Engine and Cloud Machine Learning

Tuesday, February 21, 2017

You can now get VMs

With multiple Tesla K80 cards

Up to 8 K80s per machine  
Total GPU memory: 192GB DDR5 !!

# Outline

---

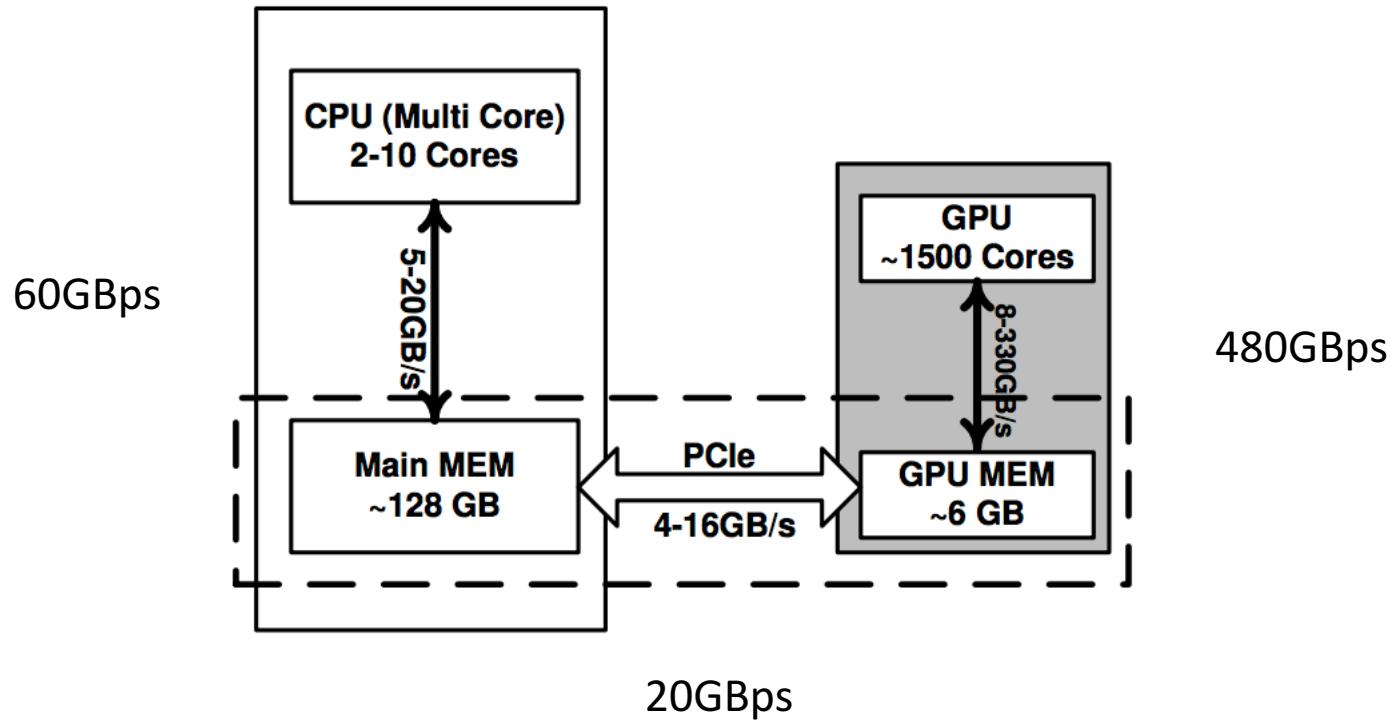
GPU Architecture

Doing Top-K on the GPU

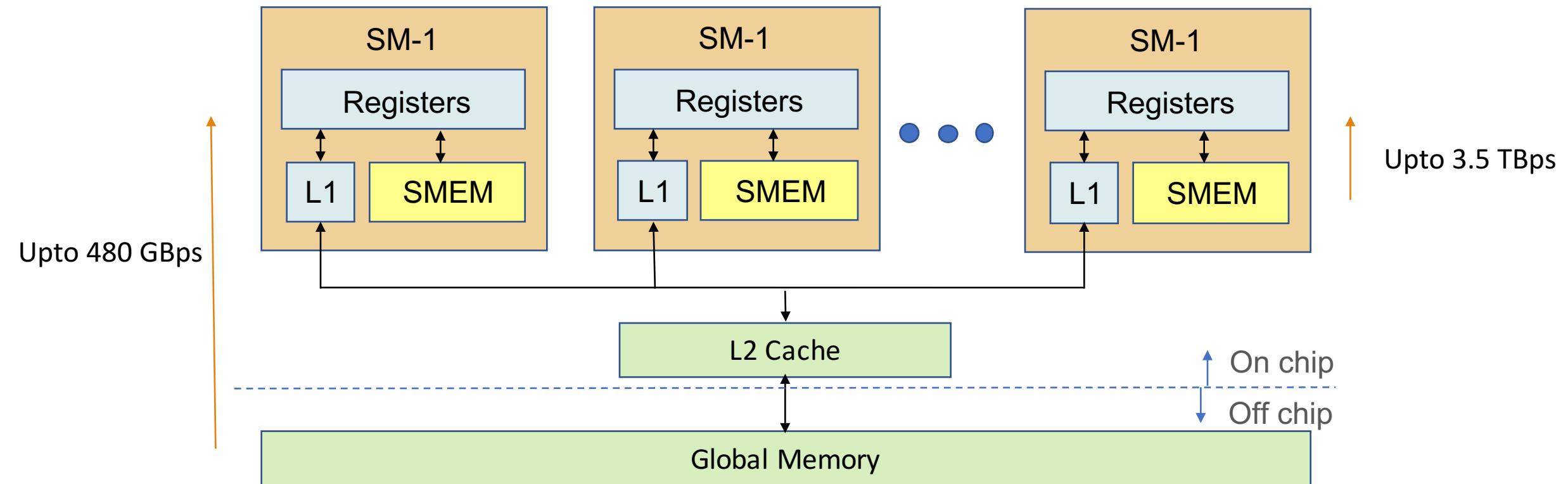
Some recent work on CPU-GPU co-processing

# CPU-GPU

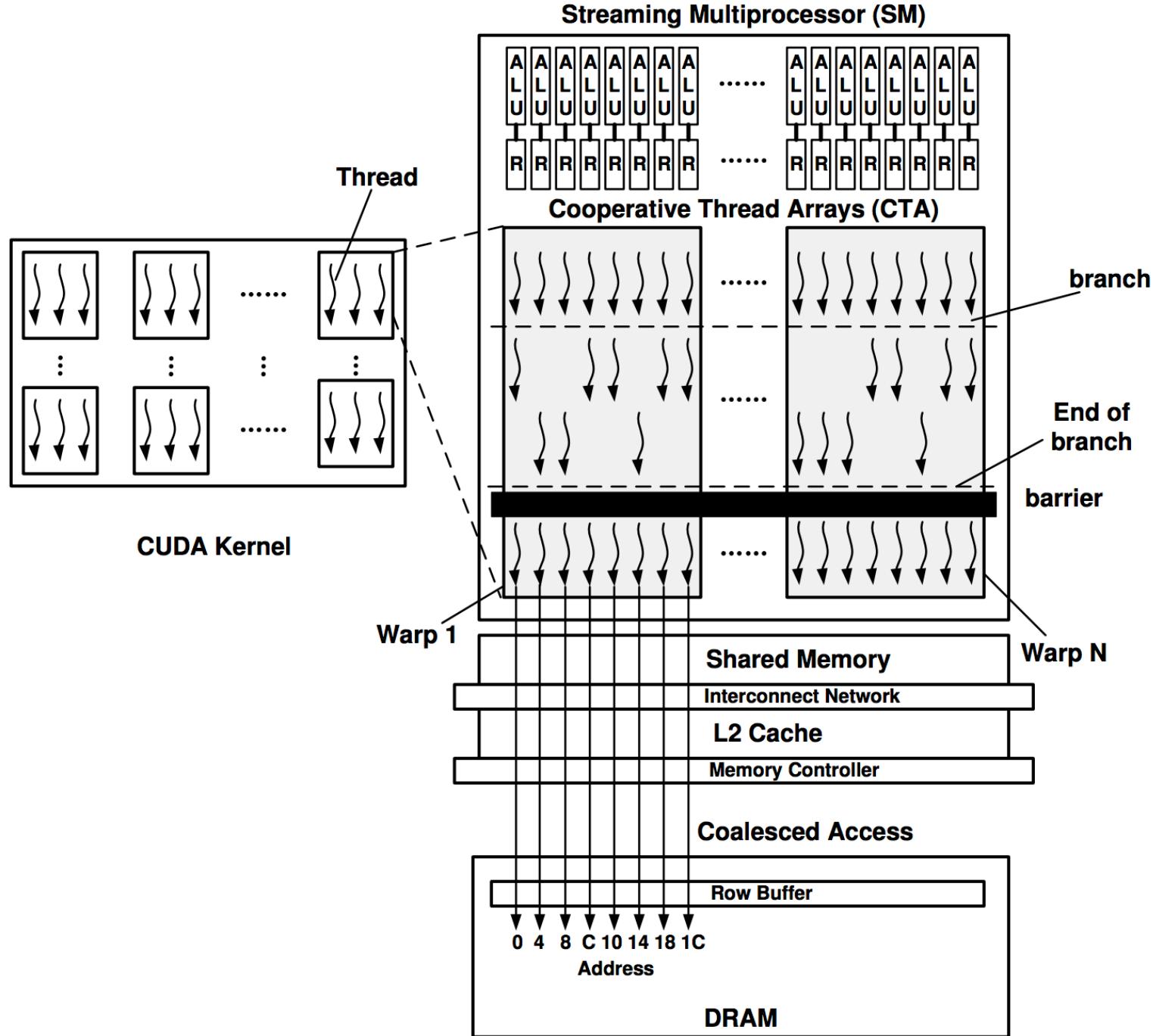
---



# Inside a GPU



# GPU Execution Model



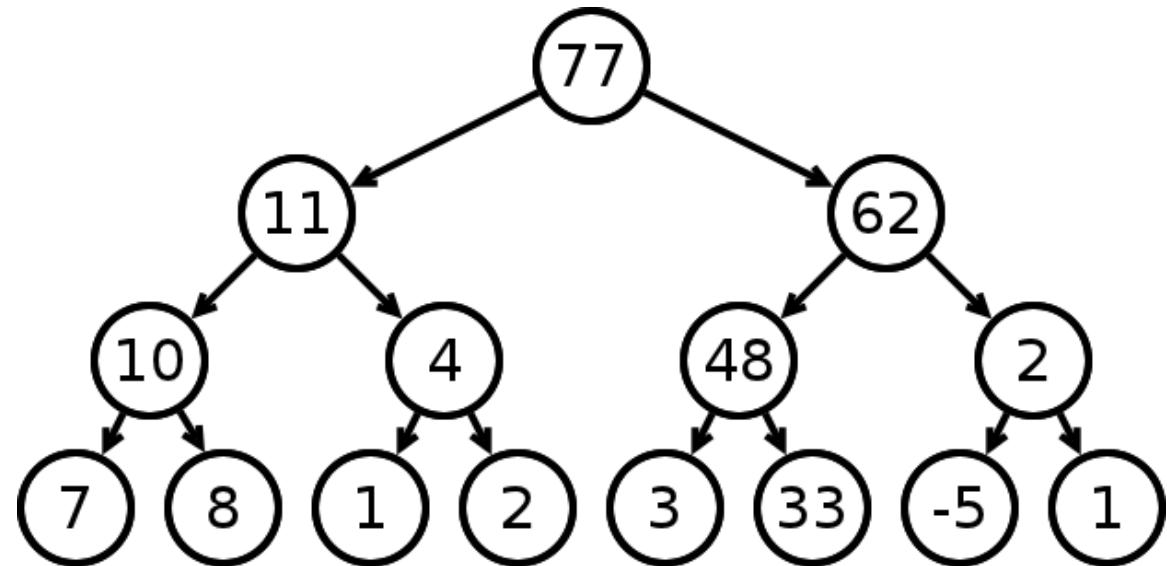
# Top-K

---

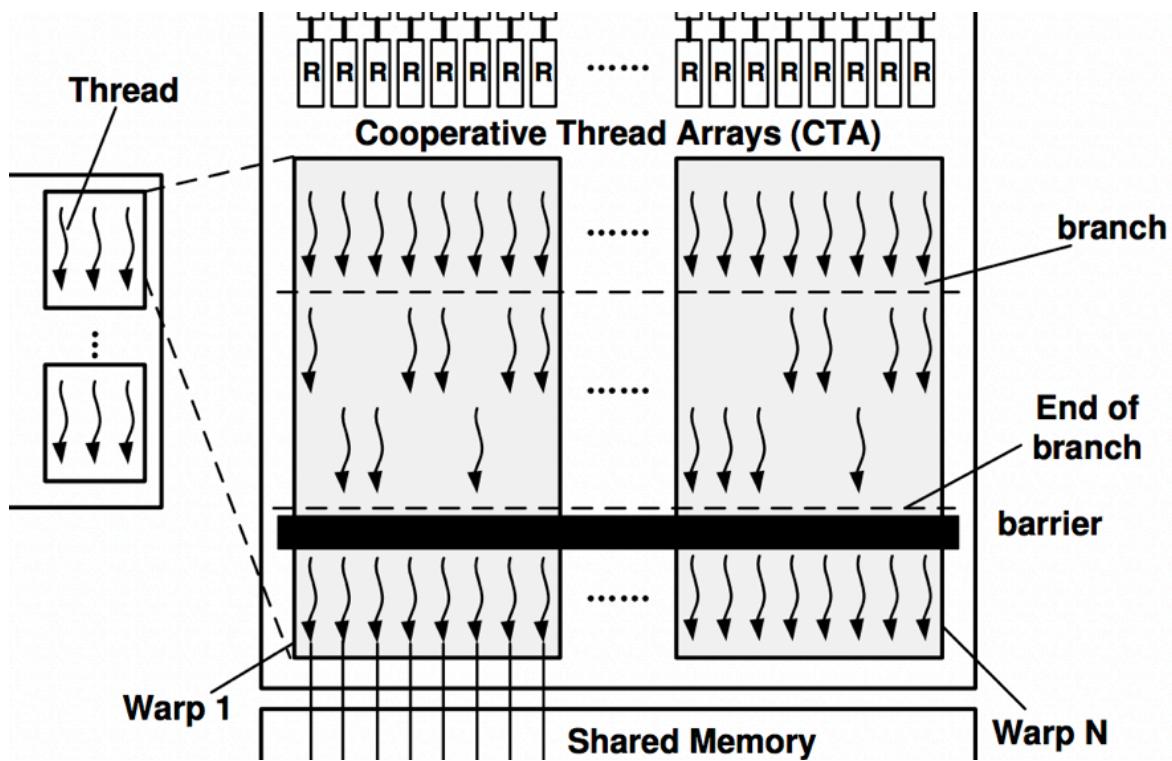
From a list on n elements – find the highest k elements

Use a min-heap to maintain the top-k ----->

On multi-core machines,  
each core can maintain its local top-k  
find the global top-k among local top-k's



# Problems !!

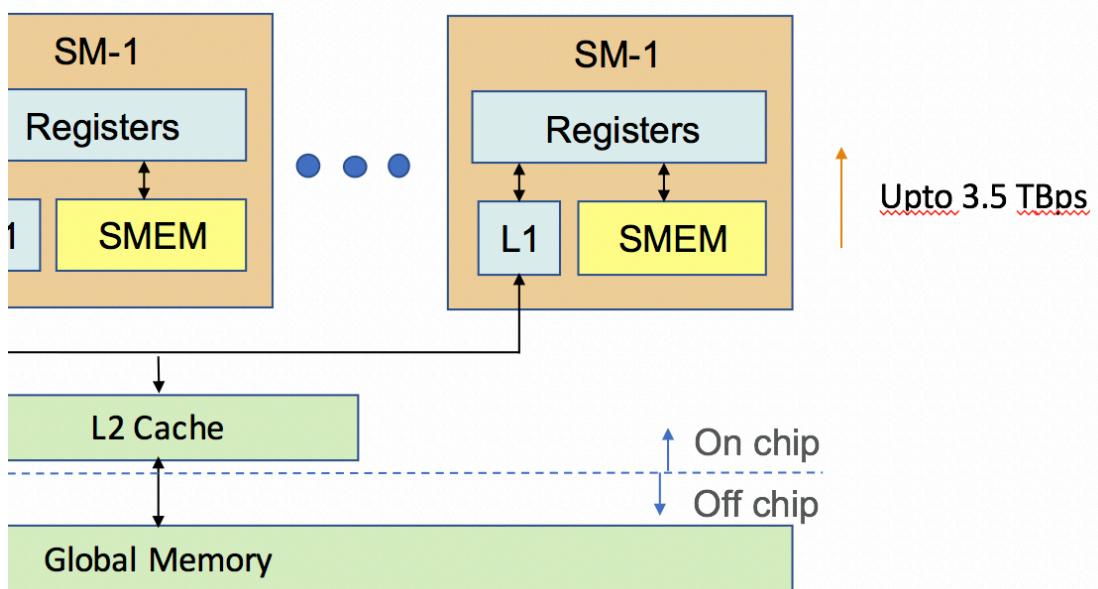


## 1) Thread Divergence

A warp (32 threads) execute in lock steps.  
So if one thread wants to update its heap,  
everyone runs the same code path 😞

# Problems !!

---



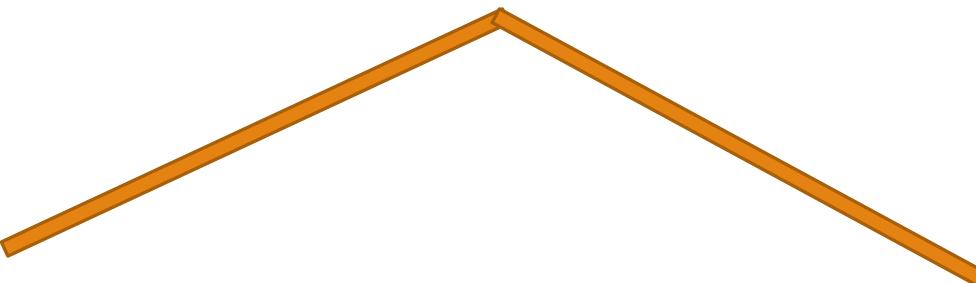
## 2) Occupancy

Shared memory is a precious resource.

Each thread needs  $k$  entries in shared memory.  
Grows linearly with  $k$ .

After a point affects occupancy.

# Bitonic Top-K



# Sorting a Bitonic Sequence

- Let  $s = \langle a_0, a_1, \dots, a_{n-1} \rangle$  be a bitonic sequence such that
  - $a_0 \leq a_1 \leq \dots \leq a_{n/2-1}$ , and
  - $a_{n/2} \geq a_{n/2+1} \geq \dots \geq a_{n-1}$



- Consider the following subsequences of  $s$

$$s_1 = \langle \min(a_0, a_{n/2}), \min(a_1, a_{n/2+1}), \dots, \min(a_{n/2-1}, a_{n-1}) \rangle$$

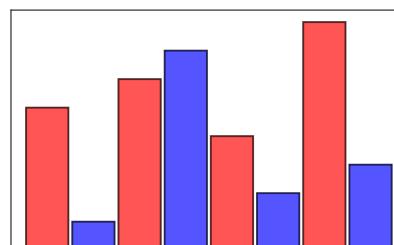
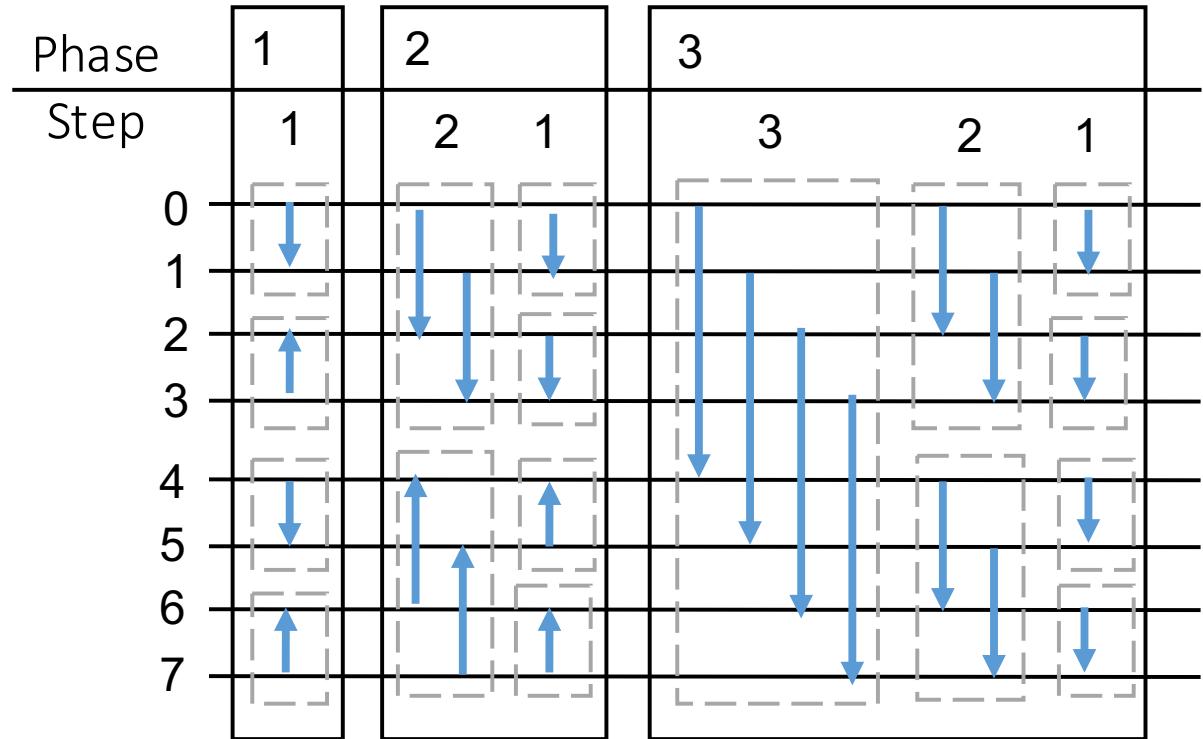
$$s_2 = \langle \max(a_0, a_{n/2}), \max(a_1, a_{n/2+1}), \dots, \max(a_{n/2-1}, a_{n-1}) \rangle$$



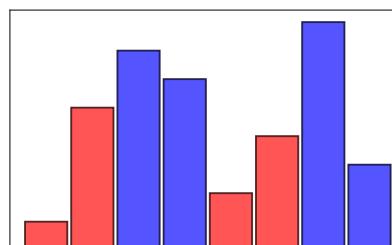
- Sequence properties
  - $s_1$  and  $s_2$  are both bitonic
  - $\forall_x \forall_y x \in s_1, y \in s_2, x < y$
- Apply recursively on  $s_1$  and  $s_2$  to produce a sorted sequence
- Works for any bitonic sequence, even if  $|s_1| \neq |s_2|$

# Bitonic Sort

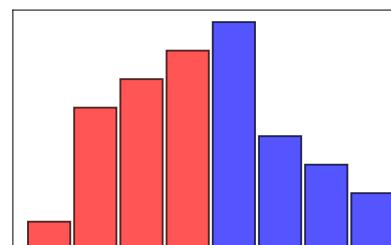
- List of length 1 sorted by definition
- List of length 2 bitonic by definition
- Sort length 2 sequences ascending /descending using previous
- Neighboring sequences form bitonic sequences of length 4
- Sort them ascending descending to create length 8 sequence
- .....
- Sort bitonic sequence of length n to form sorted sequence of length n



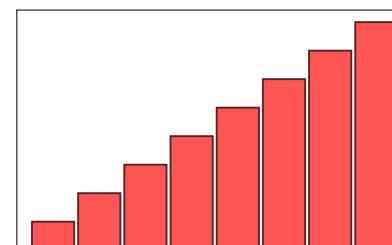
Unsorted Input



After Phase 1

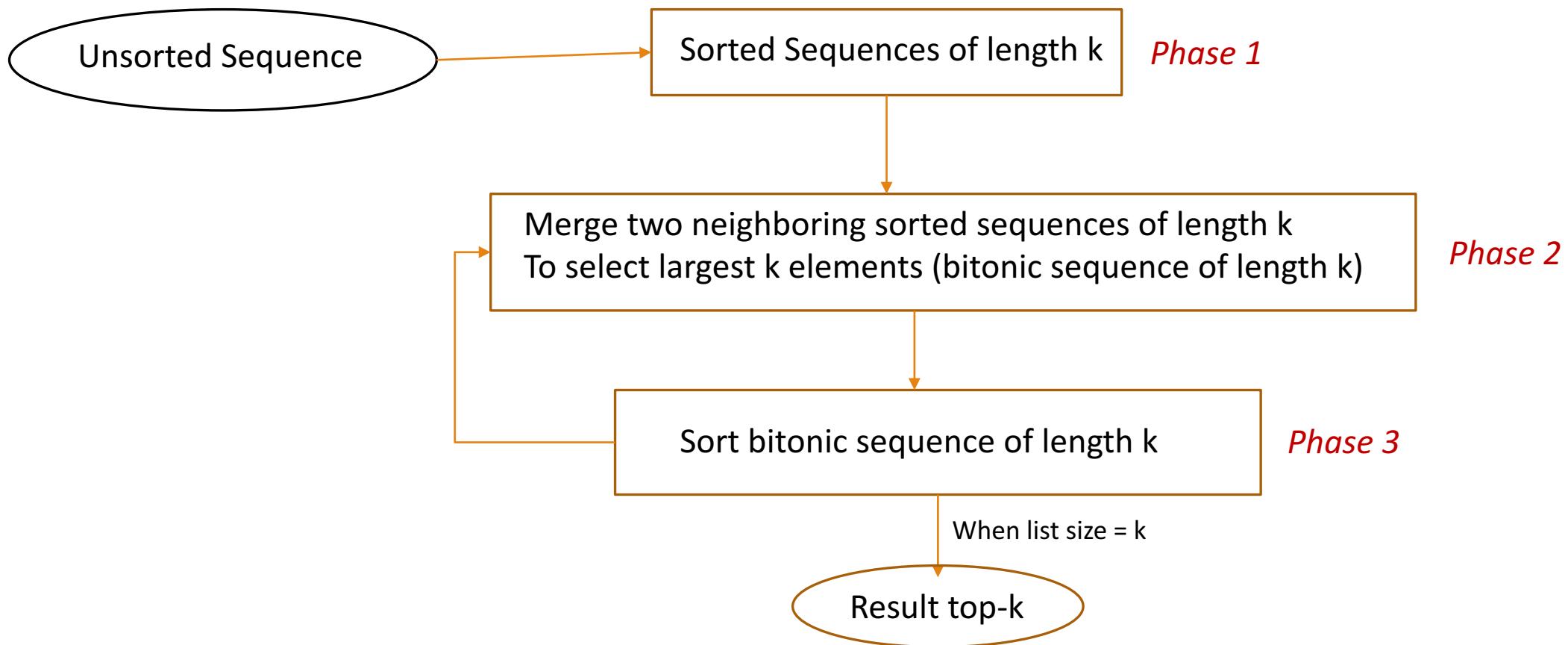


After Phase 2

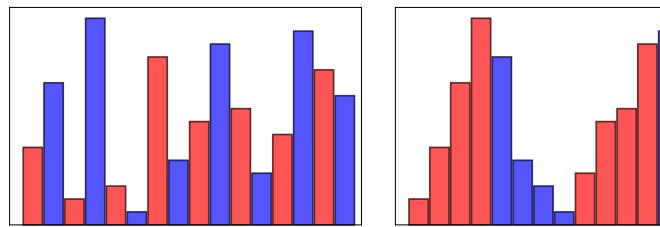


After Phase 3

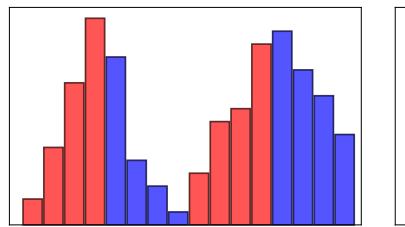
# Bitonic Top-K



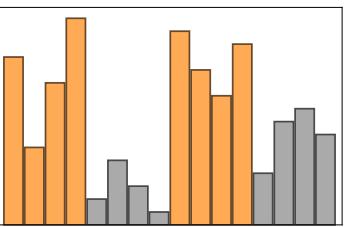
# Bitonic Top-K Visualized



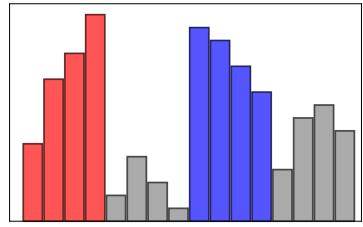
Unsorted Input



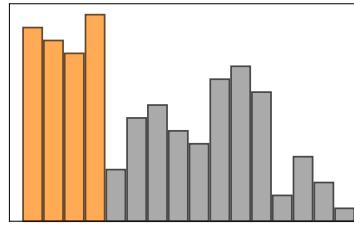
After Phase 1



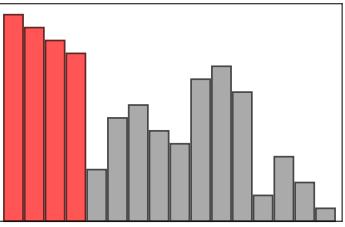
After Phase 2



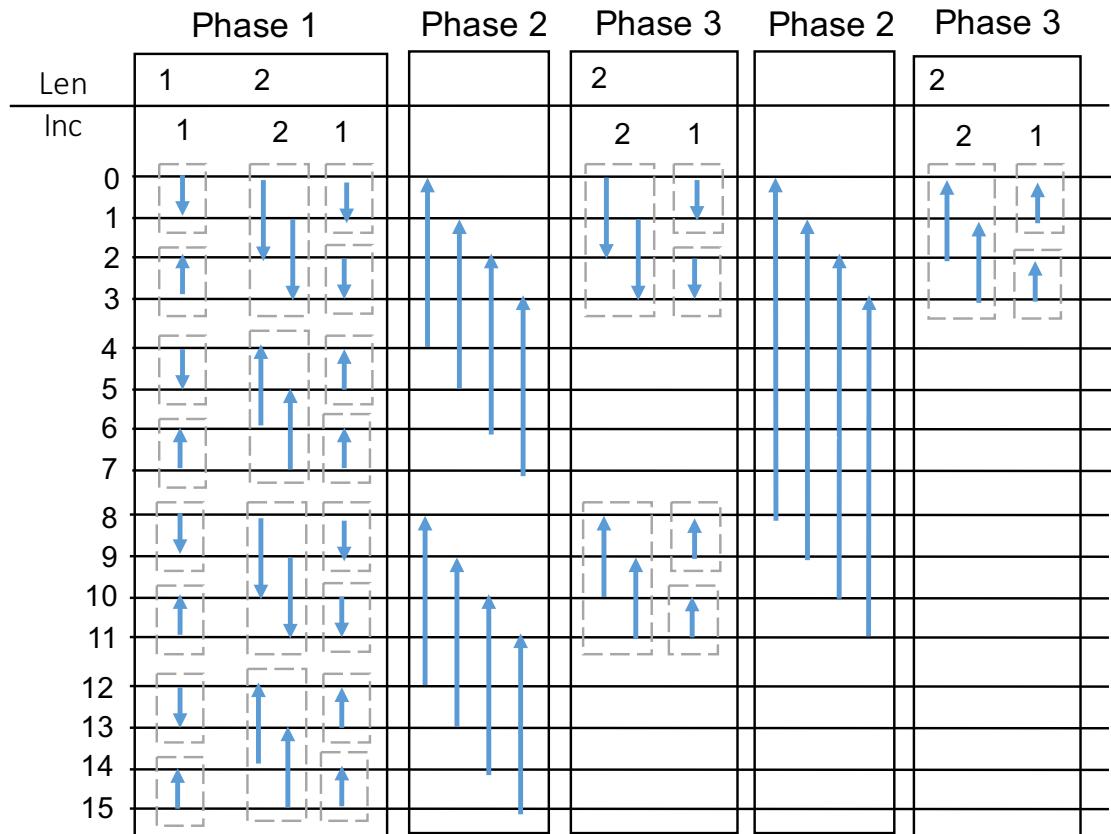
After Phase 3



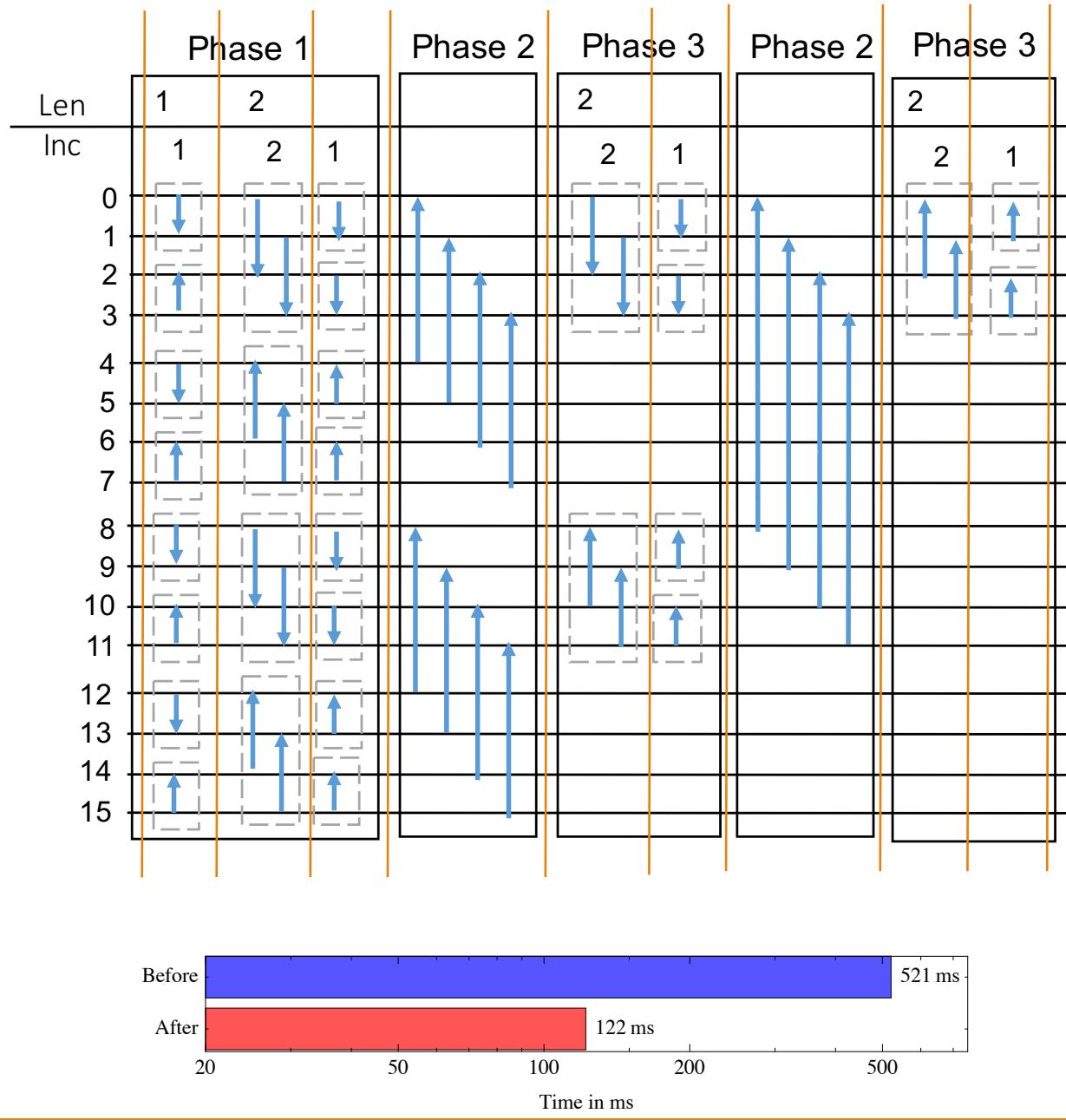
After Phase 2 (2)



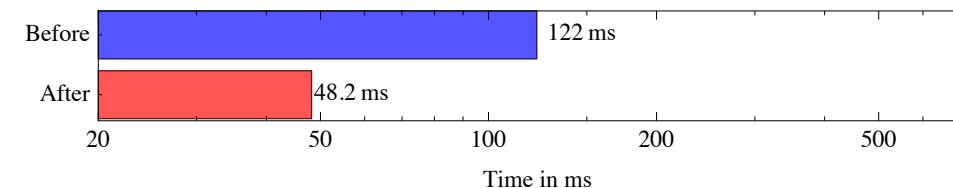
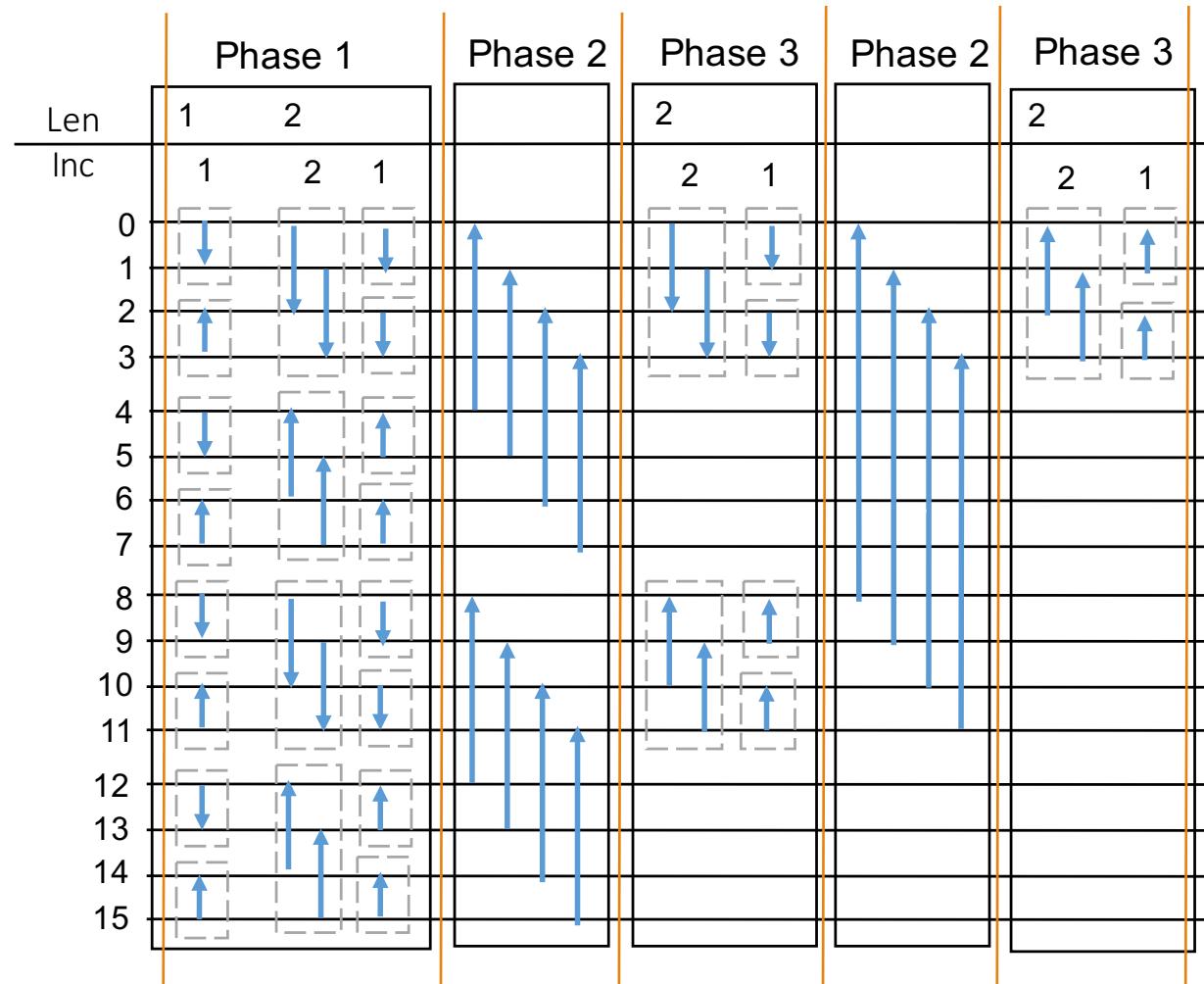
After Phase 3 (2)



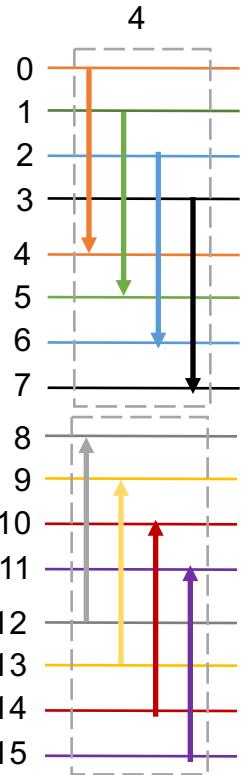
## Optimization 1: Using Shared Memory



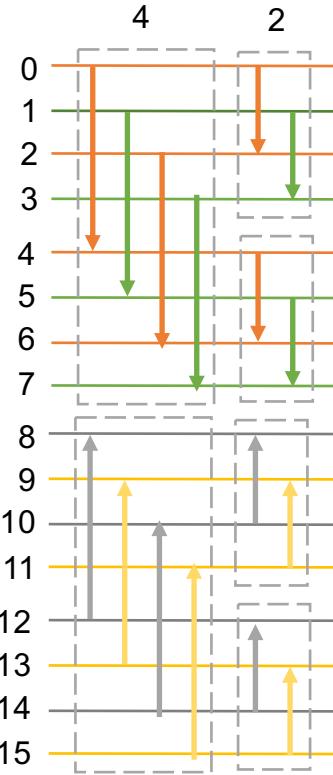
## Optimization 2: Combining Phases



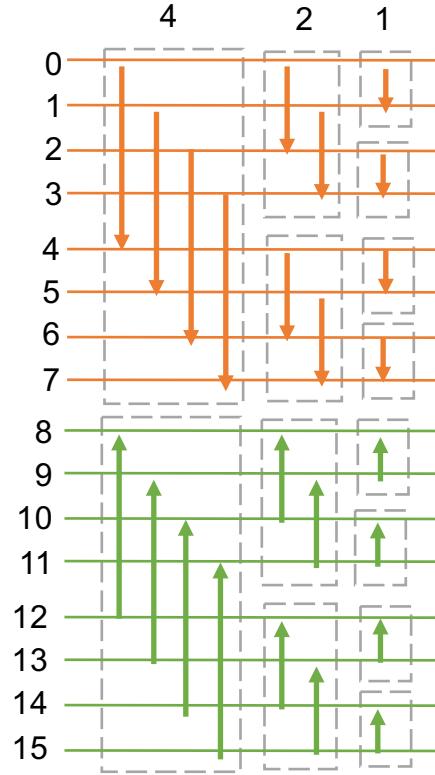
### Optimization 3: Combining Steps



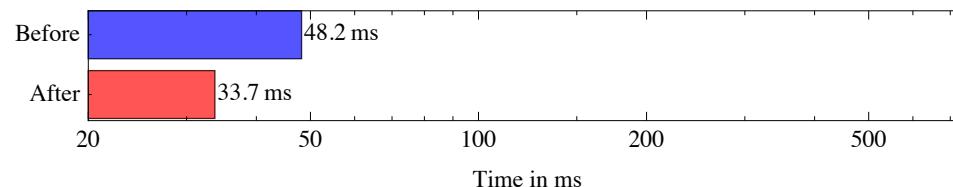
One step at a time



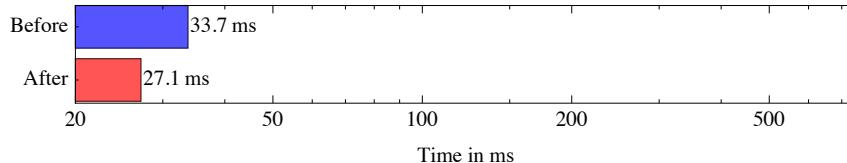
Two steps at a time



Three steps at a time



## Optimization 4: Read from Global > Process > Write to Shared

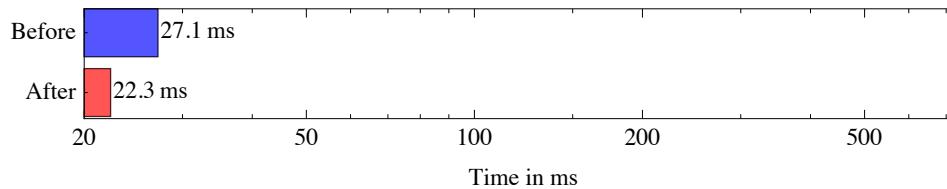


0		1		2		15	
4		5		6		16	
8		9		10		17	
12		13		14		18	
16		17		18		19	
20		21		22		23	
24		25		26		27	
28		29		30		31	

## Optimization 5: Padding

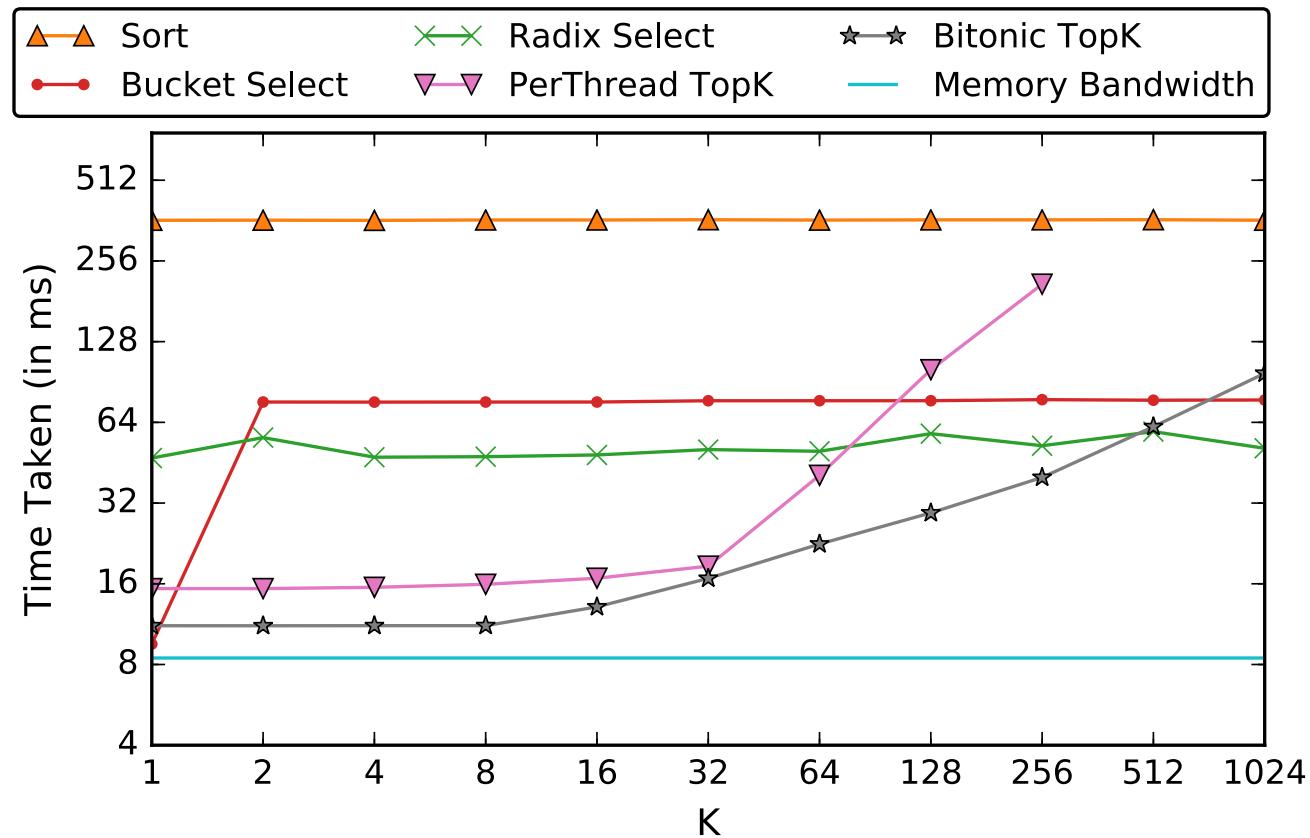
Reshape array of size n  
( $32 * n/32$ )  
To  
( $33 * n/32$ )

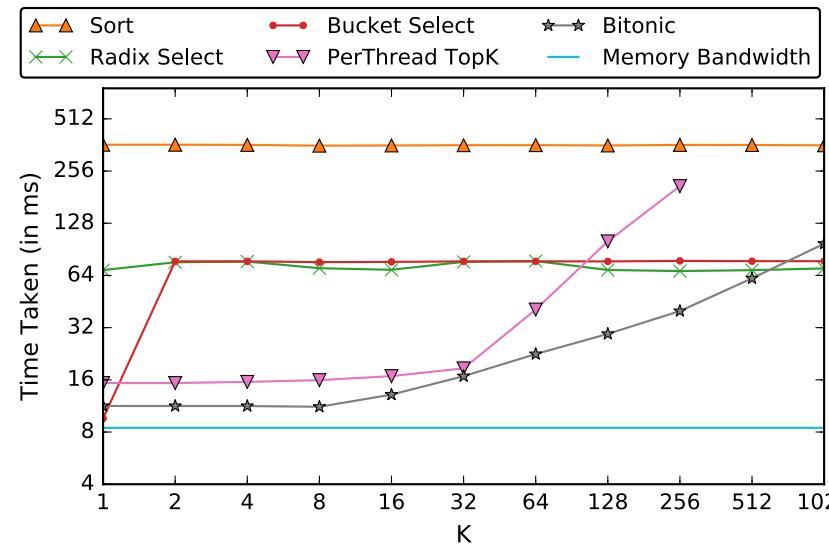
0		1		2		15	
x 4		5		6		16	
x 8		9		10		17	
x 12		13		14		18	
x 16		17		18		19	
x 20		21		22		23	
x 24		25		26		27	
x 28		29		30		31	



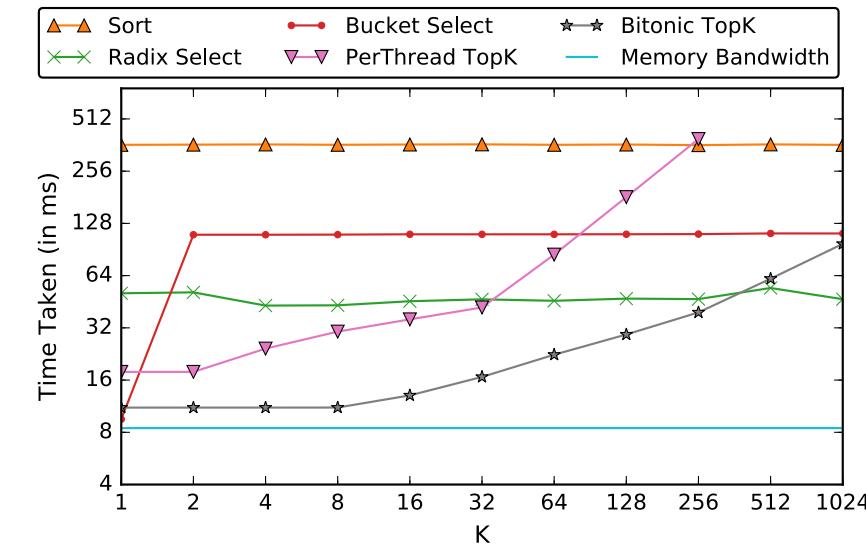
# Experiments

For  $2^{29}$  (1/2 billion) floats from  $U(0,1)$   
On Nvidia Titan X Maxwell

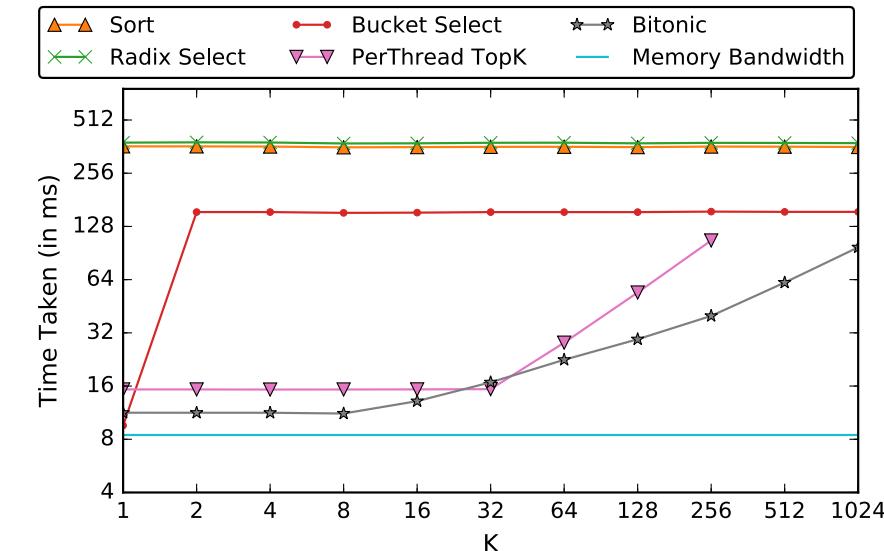




a) Large Floats U(0, 1million)

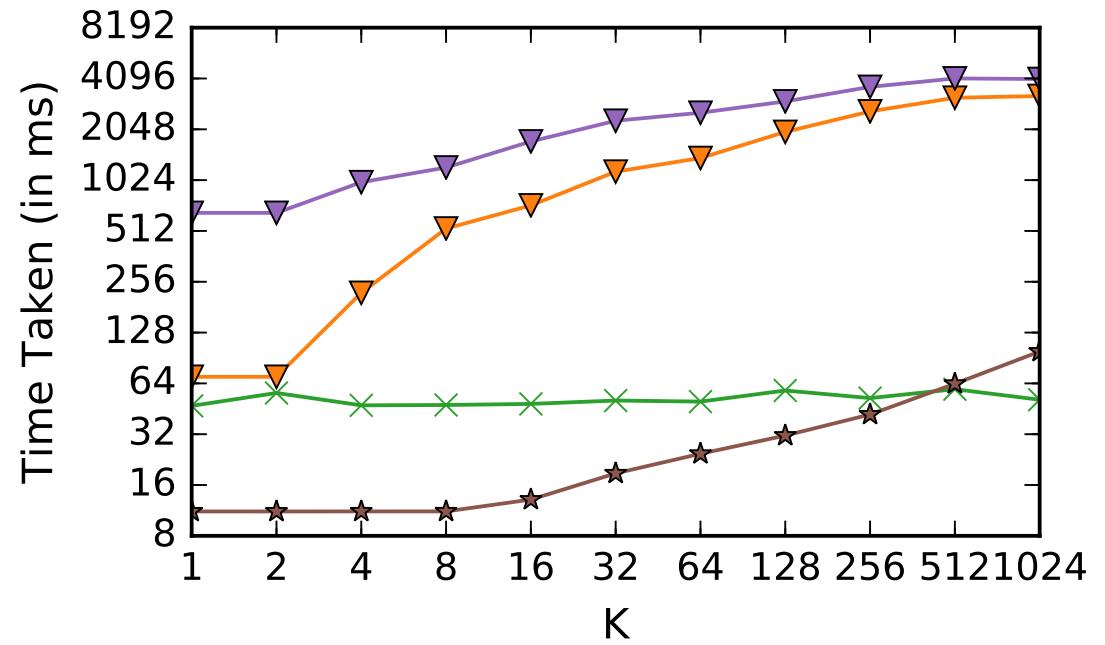
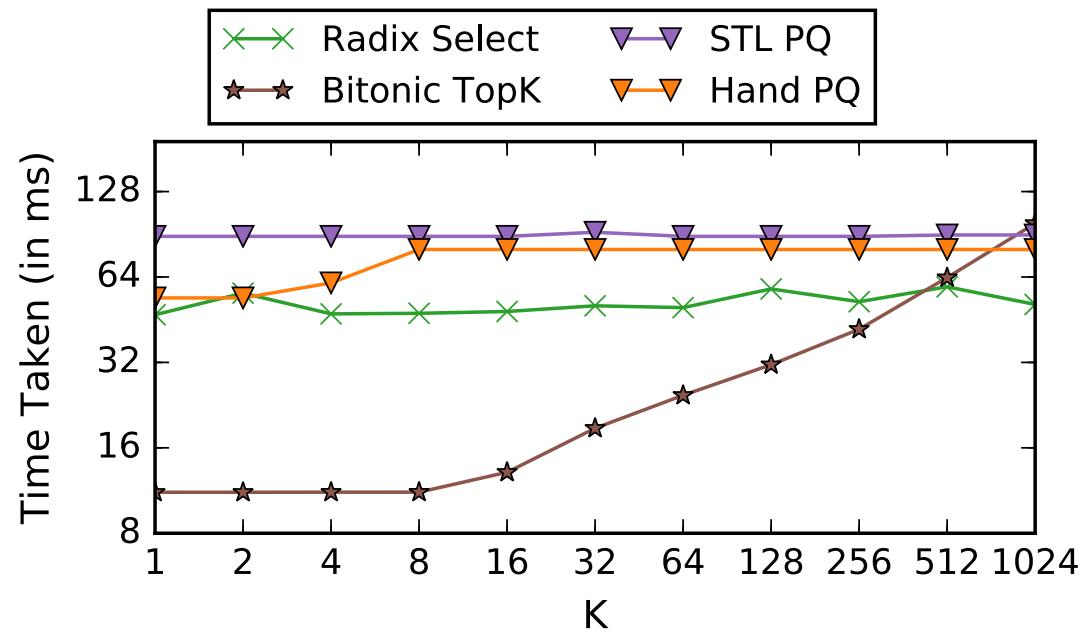


b) Increasing



c) Bucket Killer

With  
Different Distributions



Comparing to CPU

# Hybrid Databases



# So-Far

[Hardware-oblivious parallelism for in-memory column-stores \(Ocelot\)](#)

VLDB 2013

[The Yin and Yang of Processing Data Warehousing Queries on GPU](#)

VLDB 2013

[Concurrent Analytical Query Processing with GPUs](#)

VLDB 2014

[HippogriffDB: Balancing I/O and GPU Bandwidth](#)

VLDB 2016

All use old hardware: The latest of these uses K20 – with 5GB DDR5 GPU memory ☹

Single GPU – used to run everything – CPU ~ idle

All of them compare against **MonetDB** .....

# The Bottleneck: PCIe

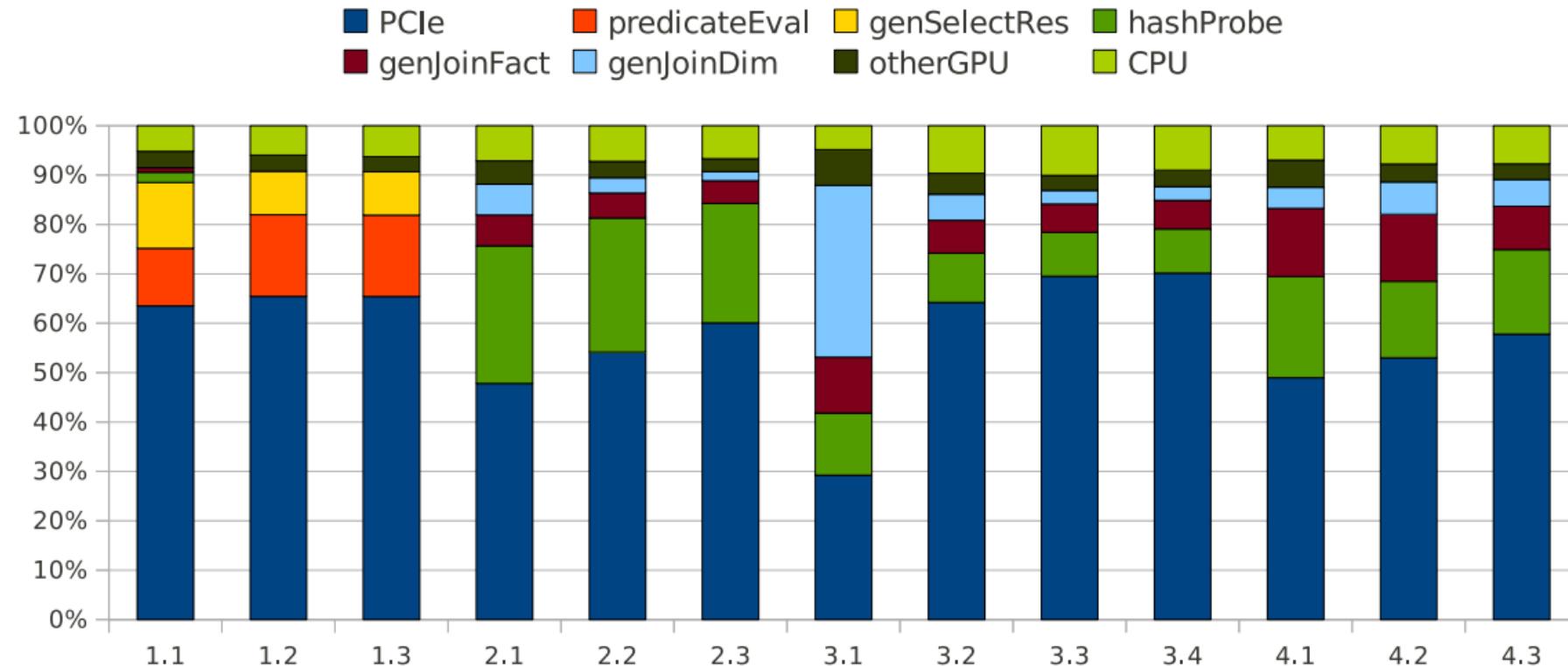
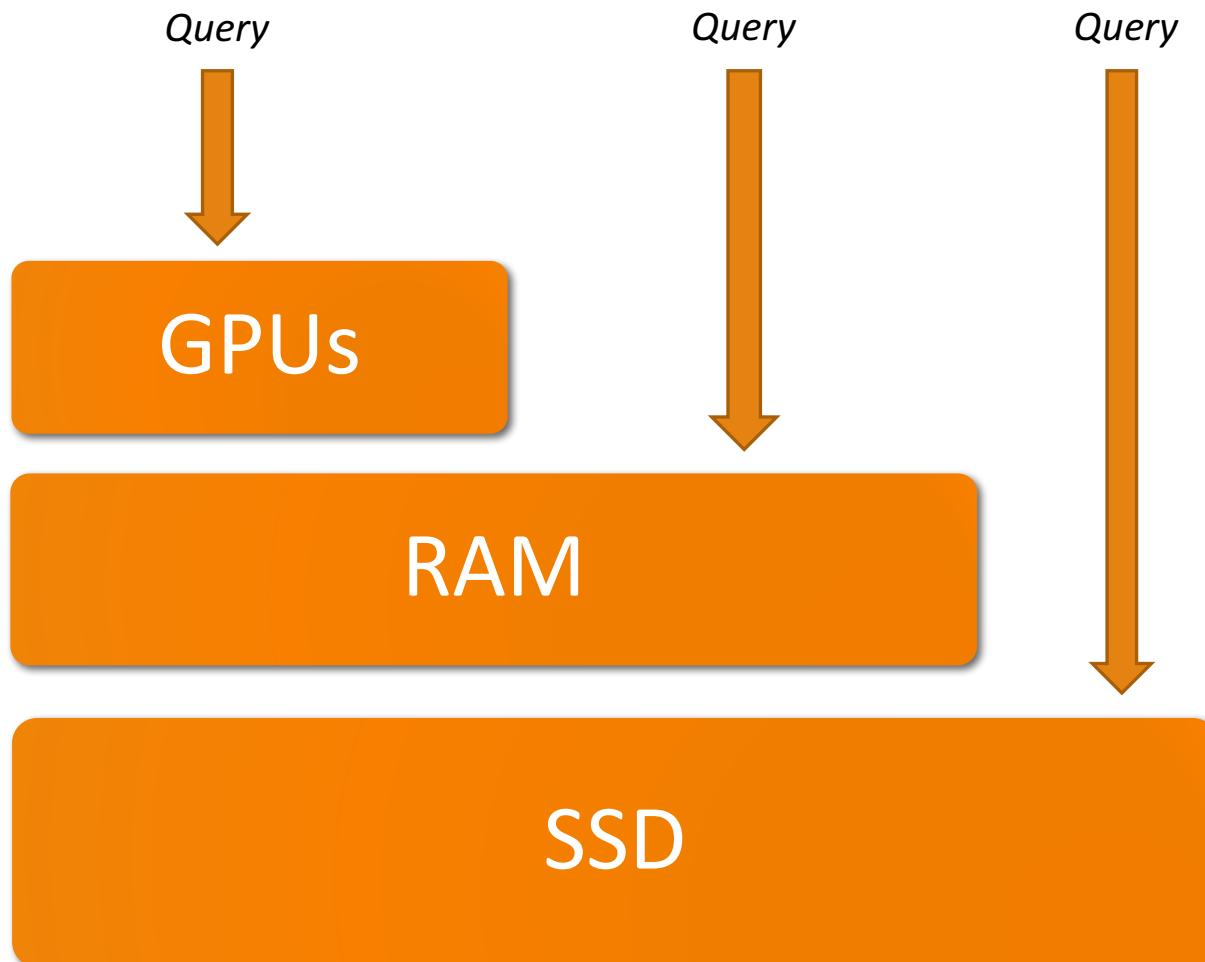


Figure 5: SSBM execution time breakdown

From the YinYang paper, VLDB 13



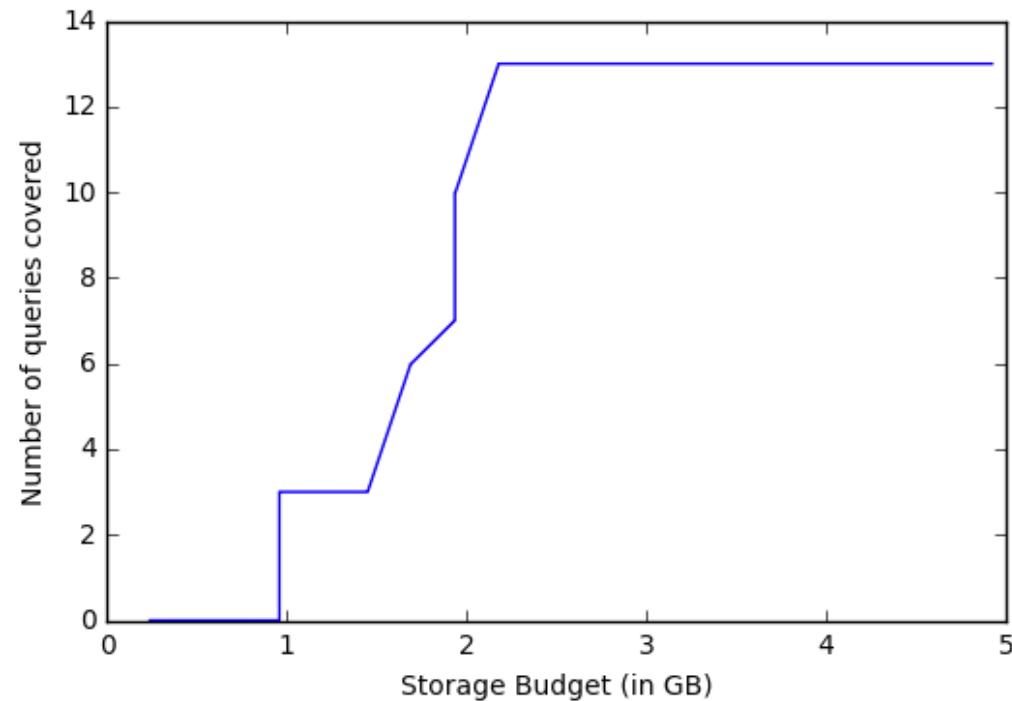
Maintain hot columns in GPU

Assuming column-orientated storage

All the data is in SSD

Most of the data is in RAM

Most frequently accessed columns are in GPU



For Star Schema Benchmark (SF=10)

# GPU / CPU Co-Processing System

Based on cost models, we classify query into 3 categories

- 1) Run Xx faster on the GPU
- 2) Run < Xx faster on the GPU
- 3) Can't run on the GPU

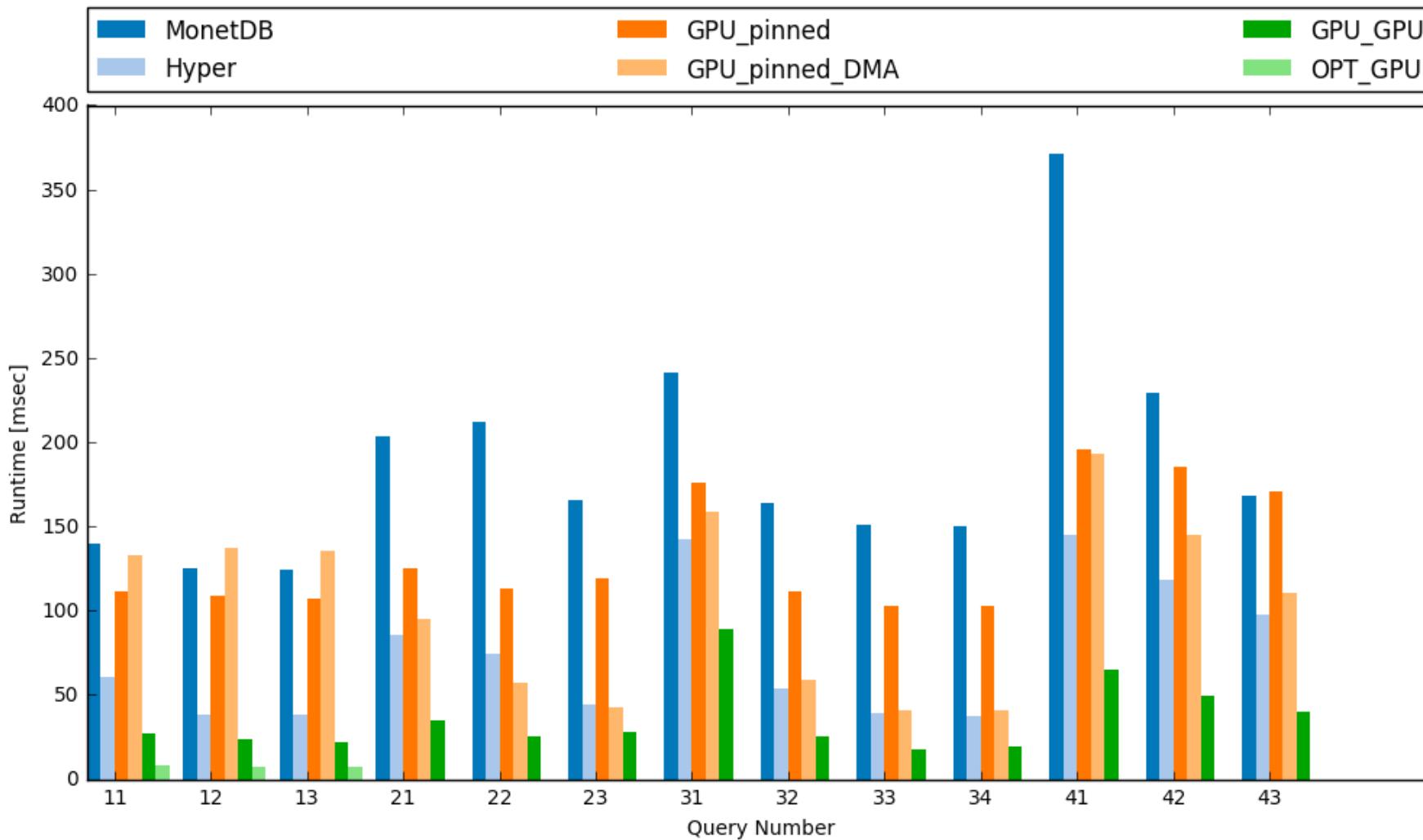
Using these, we build 3 queues:

- 1) GPU Queue
- 2) GPU or CPU Queue
- 3) CPU Queue

We can now use a work-stealing based approach to run the system.

Each processing unit (PU) runs its own queue before stealing from other queues

# Preliminary Results



# Conclusion

---

GPU memory bandwidth makes it attractive for analytics as well

Moving data across PCI is slow

Using GPUs as a column cache is probably the way to go

Allows us to use both CPU and GPU resources

Expect 3-5x performance improvement over running on the CPU