TRITONSORT

A Balanced Large-Scale Sorting System

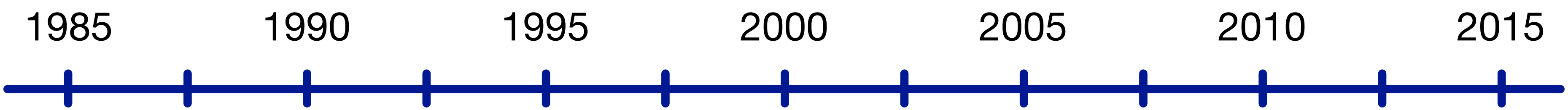
CSE 224 – Jan 20, 2022

Alex Rasmussen, George Porter, Michael Conley,
Radhika Niranjan Mysore, Amin Vahdat (UCSD)
Harsha V. Madhyastha (UC Riverside)
Alexander Pucher (Vienna University of Technology)

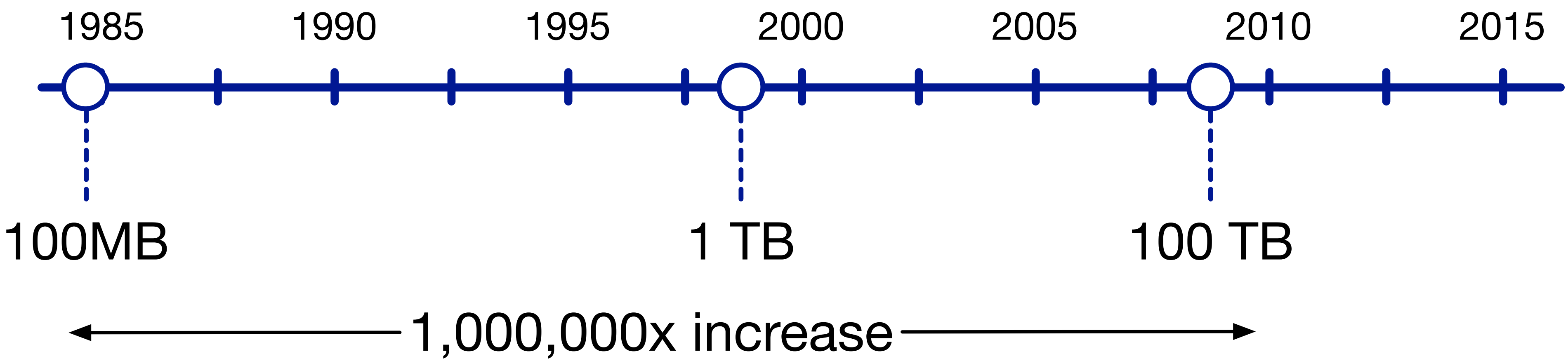
The Rise of Big Data Workloads

- Very high I/O and storage requirements
  - Large-scale web and social graph mining
  - Business analytics – “you may also like …”
  - Large-scale “data science”
- Recent new approaches to “data deluge”: data intensive scalable computing (DISC) systems
  - MapReduce, Hadoop, Dryad, …

Size of “data-intensive” has grown



Size of “data-intensive” has grown

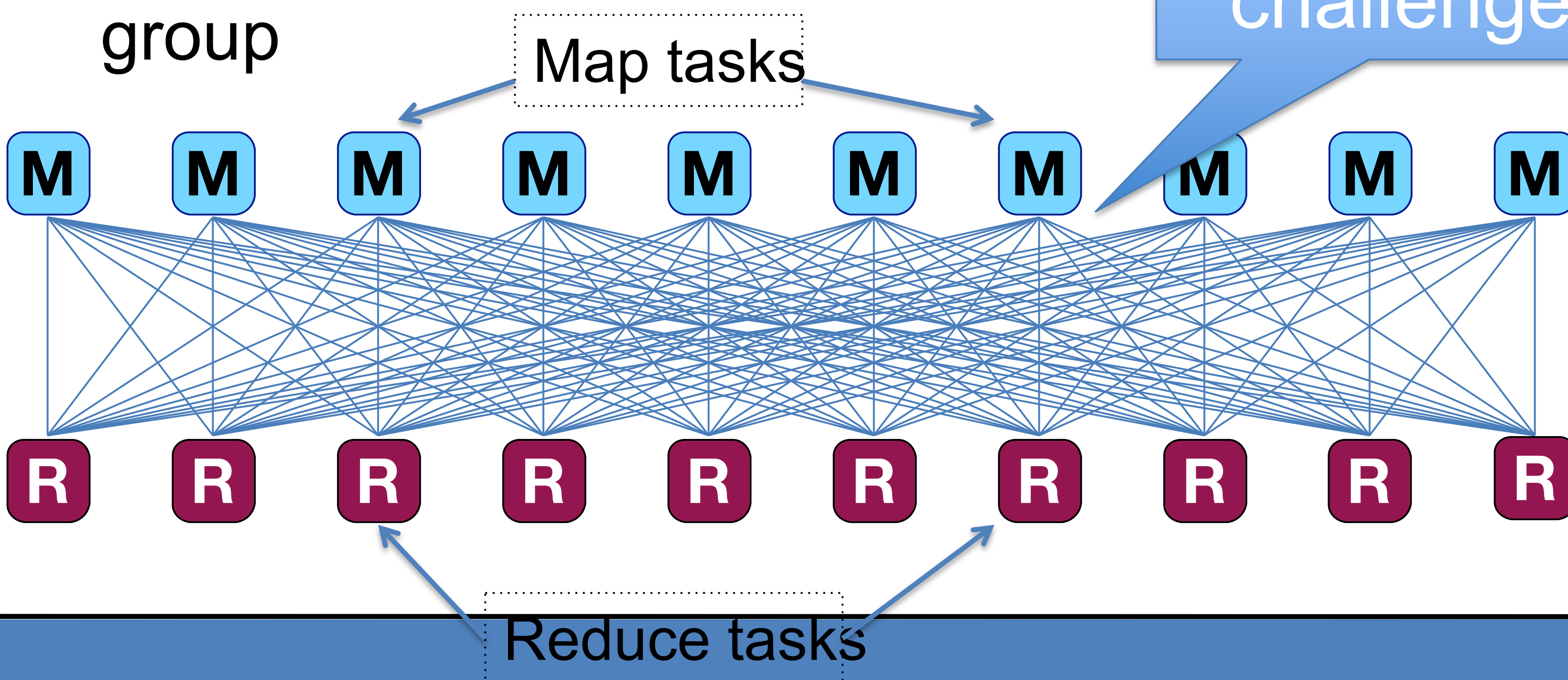


“Data-intensive” commonly means MapReduce

Input: Set<key-value pairs>

1. Apply *map()* to each pair
2. Group by key; sort each group
3. Apply *reduce()* to each sorted group

Sorting is the challenge!



Performance via scalability

- 10,000+ node MapReduce clusters deployed
  - With impressive performance
- Example: Yahoo! Hadoop Cluster Sort
  - 3,452 nodes sorting 100TB in less than 3 hours
- But…
  - Less Than 3 MB/sec per node
  - Single disk: ~100 MB/sec
- Not an isolated case
  - See “Efficiency Matters!”, SIGOPS 2010

Overcoming Inefficiency With Brute Force

- Just add more machines!
  - But expensive, power-hungry mega-datacenters!
- What if we could go from 3 MBps per node to 30?
  - 10x fewer machines accomplishing the same task
  - or 10x higher throughput

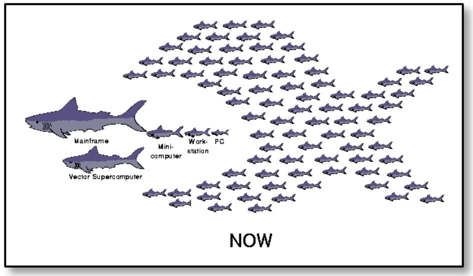

TDK
Bolt
NEW WR
SEIKO
9.58

GUINNESS WORLD RECORDS

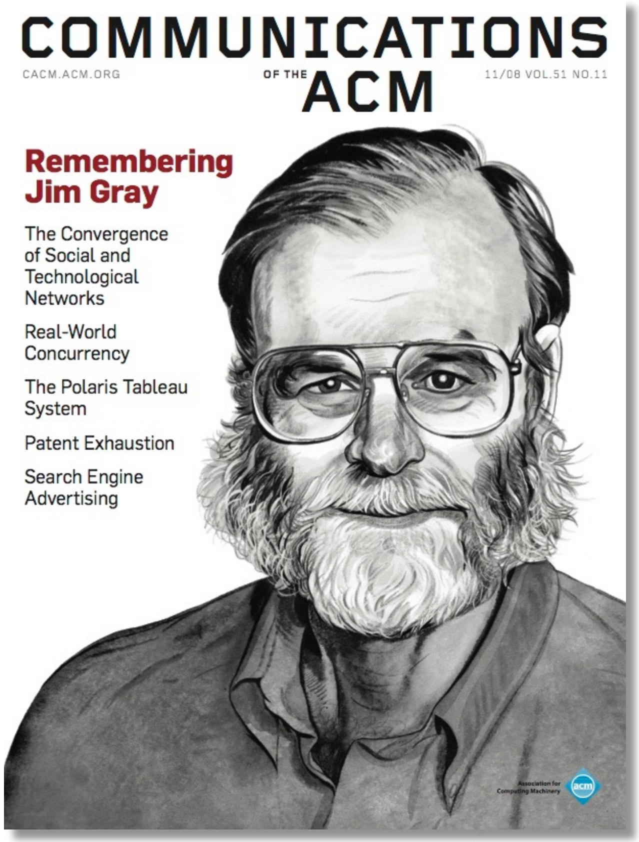
Sorting!

Evaluating IO efficiency: GraySort

- Sorting contest [Jim Gray et al.,1985]
  - Importance of the IO subsystem
  - 1985: Sort 100MB
  - 1999: Sort 1TB



  - 2009: Sort 100 TB



Anon et al, "A measure of transaction processing power," Datamation 31, 7 (April 1985), 112-118.

Sorting records

- GraySort:
  - Time to sort 100TB
- MinuteSort:
  - How much in 60s
- JouleSort:
  - How much in 1 Joule
- PennySort:
  - How much for 1¢
- CloudSort:
  - How much $$ to sort 100TB?

Top Results

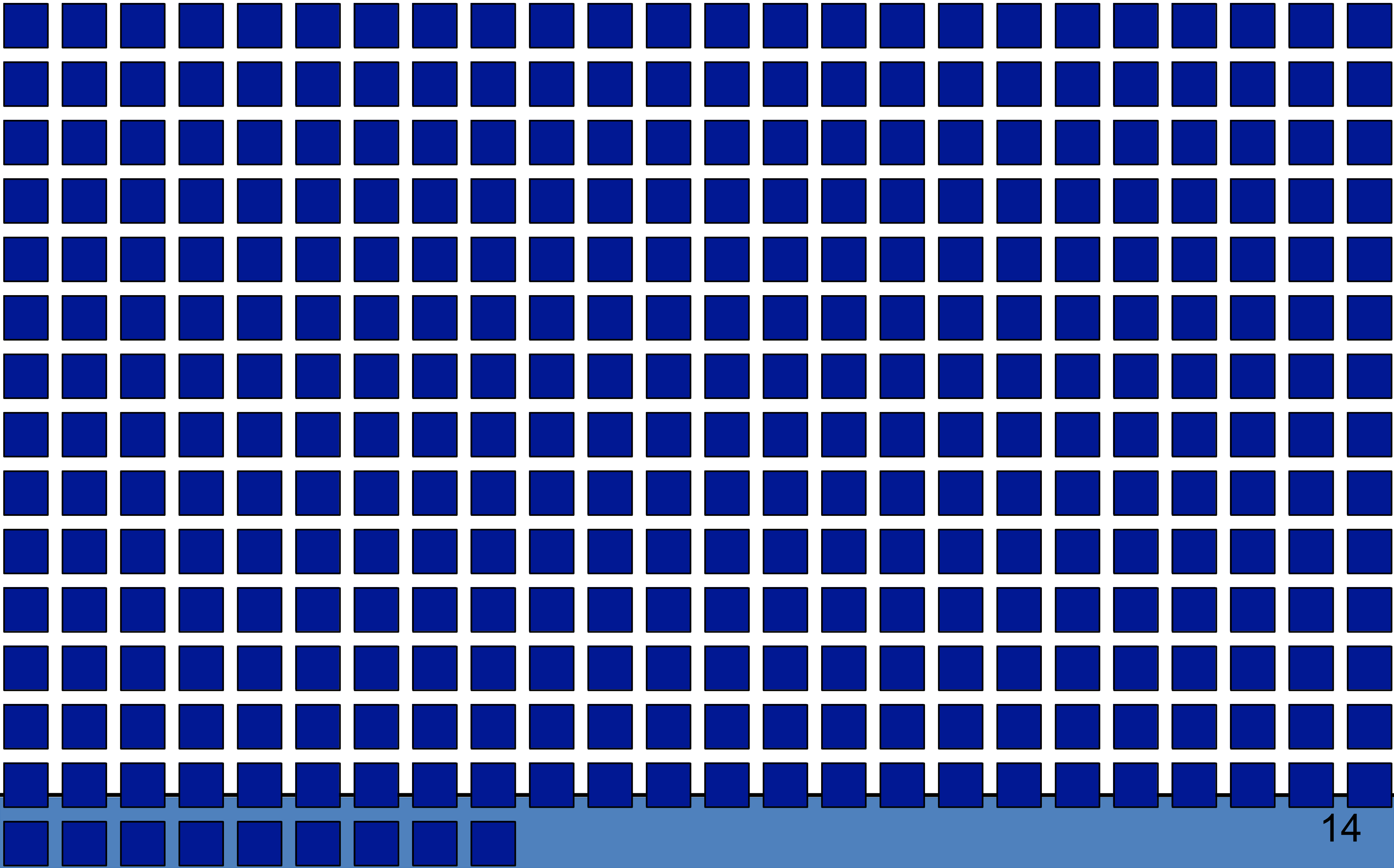
	Daytona	Indy
Gray	2011, 0.725 TB/min **TritonSort** 1,000,000,000,000 records in 8,274 seconds 52 nodes x (2 Quadcore processors, 24 GB memory, 16x500GB disks) Cisco Nexus 5096 switch Alex Rasmussen, Michael Conley, George Porter, Amin Vahdat, University of California, San Diego	2011, 0.938 TB/min **TritonSort** 1,000,000,000,000 records in 6,395 seconds 52 nodes x (2 Quadcore processors, 24 GB memory, 16x500GB disks) Cisco Nexus 5096 switch Alex Rasmussen, Michael Conley, George Porter, Amin Vahdat, University of California, San Diego
Penny	2011, 286 GB **psort** 2.7 Ghz AMD Sempron, 4 GB RAM, 5x320 GB 7200 RPM Samsung SpinPoint F4 HD332GJ, Linux Paolo Bertasi, Federica Bogo, Marco Bressan and Enoch Peserico Univ. Padova, Italy	2011, 334 GB **psort** 2.7 Ghz AMD Sempron, 4 GB RAM, 5x320 GB 7200 RPM Samsung SpinPoint F4 HD332GJ, Linux Paolo Bertasi, Federica Bogo, Marco Bressan and Enoch Peserico Univ. Padova, Italy
Minute	2009, 500 GB **Hadoop** 1406 nodes x (2 Quadcore Xeons, 8 GB memory, 4 SATA) Owen O'Malley and Arun Murthy Yahoo Inc.	2011, 1353GB **TritonSort** 52 nodes x (2 Quadcore processors, 24 GB memory, 16x500GB disks) Cisco Nexus 5096 switch Alex Rasmussen, Michael Conley, George Porter, Amin Vahdat, University of California, San Diego
Joule 10^8 recs	2011, 1,430 Joules **FAWNSort** 69,900 records sorted / joule Intel Core i5-2400S 2.5 GHz, 16GB RAM, Nsort, 7 x 120 GB Intel Series 510 series SSDs Padmanabhan Pillai, Michael Kaminsky, Michael A. Kozuch, Intel Labs Pittsburgh Vijay Vasudevan, Lawrence Tan, David Andersen Carnegie Mellon University	2011, 1,430 Joules **FAWNSort** 69,900 records sorted / joule Intel Core i5-2400S 2.5 GHz, 16GB RAM, Nsort, 7 x 120 GB Intel Series 510 series SSDs Padmanabhan Pillai, Michael Kaminsky, Michael A. Kozuch, Intel Labs Pittsburgh Vijay Vasudevan, Lawrence Tan, David Andersen Carnegie Mellon University
Joule 10^9 recs	2010, 27.9 kJoules **Nsort** 35,800 records sorted / joule Intel Atom 330 1.6GHz, 4GB RAM, 4 x Super Talent UltraDrive GX MLC 256GB Andreas Beckmann, Ulrich Meyer Goethe University Frankfurt am Main, Germany Peter Sanders, Johannes Singler Karlsruhe Institute of Technology, Germany	2010, 25.1 kJoules **EcoSort** 39,900 records sorted / joule Intel Atom 330 1.6GHz, 4GB RAM, 4 x Super Talent UltraDrive GX MLC 256GB Andreas Beckmann, Ulrich Meyer Goethe University Frankfurt am Main, Germany Peter Sanders, Johannes Singler Karlsruhe Institute of Technology, Germany
Joule 10^{10} recs	2011, 1,900 kJoules **Nsort** 5,273 records sorted / joule 2 x Intel Xeon X5550 2.67 GHz, 48 GB RAM 8 x 1TB Seagate 7200RPM SATA HDD Andreas Beckmann, Ulrich Meyer Goethe University Frankfurt am Main, Germany Peter Sanders, Johannes Singler Karlsruhe Institute of Technology, Germany	2010, 572 kJoules **DEMSort** 17,500 records sorted / joule Intel Atom 330 1.6GHz, 4GB RAM, 4 x Super Talent UltraDrive GX MLC 256GB Andreas Beckmann, Ulrich Meyer Goethe University Frankfurt am Main, Germany Peter Sanders, Johannes Singler Karlsruhe Institute of Technology, Germany
Joule 10^{12} recs	2011, 132 MJoules **TritonSort** 7,595 records sorted / joule 52 nodes x (2 Quadcore processors, 24 GB memory, 16x500GB disks) Cisco Nexus 5096 switch Alex Rasmussen, Michael Conley, George Porter, Amin Vahdat, University of California, San Diego	2011, 103 MJoules **TritonSort** 9,700 records sorted / joule 52 nodes x (2 Quadcore processors, 24 GB memory, 16x500GB disks) Cisco Nexus 5096 switch Alex Rasmussen, Michael Conley, George Porter, Amin Vahdat, University of California, San Diego

Inefficiency of deployed scale-out systems

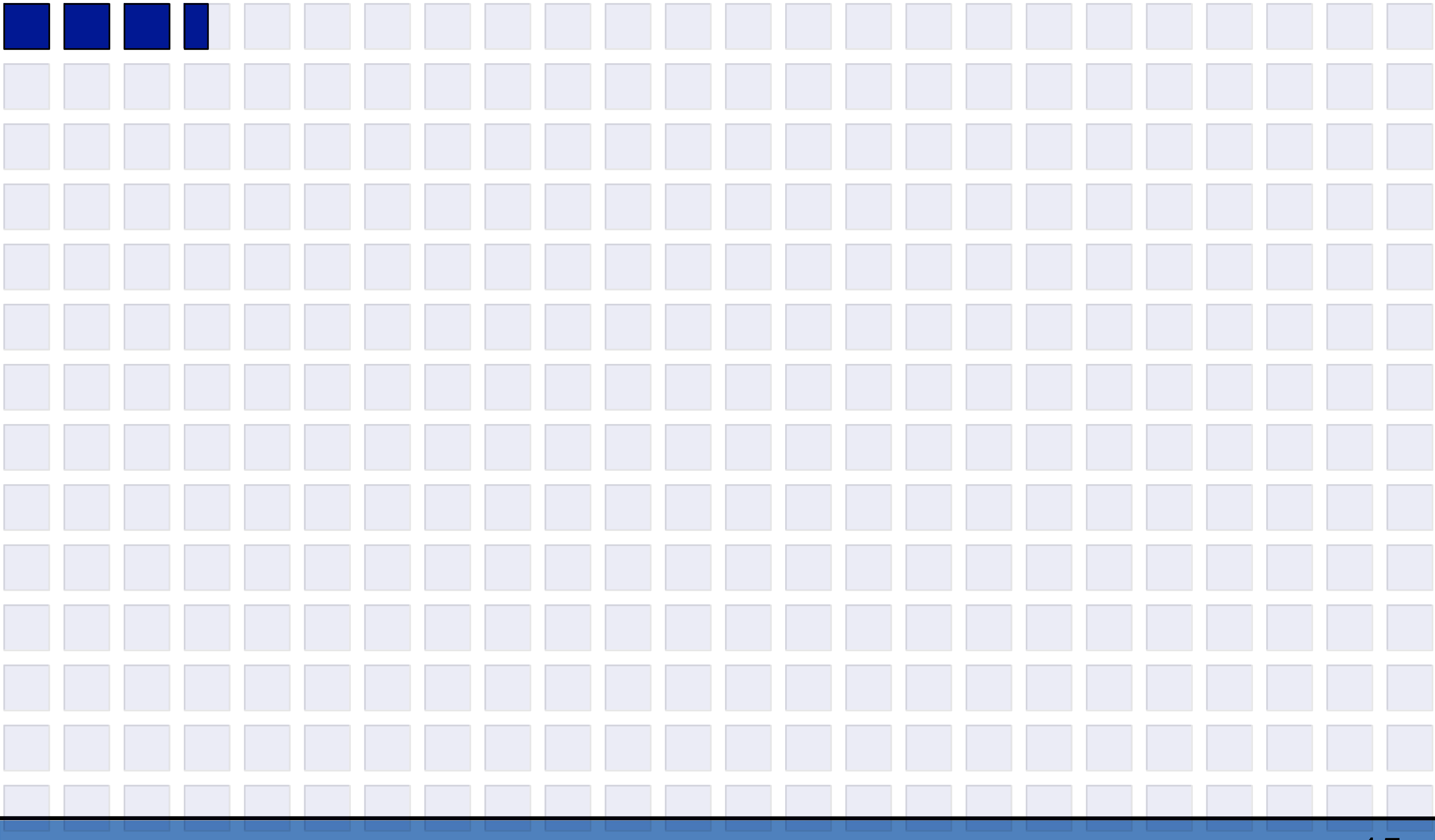
- Analysis of GraySort contest results*
  - On average: 94% disk IO idle; 33% of CPU idle
- Case study: 2009 Yahoo! Hadoop Cluster
  - Sorted 100TB with 3,452 nodes in ≈3 hours
  - 1% disk efficiency

* *Anderson and Tucek, "Efficiency matters!" SIGOPS OSR 44, 1 (March 2010)*

3452 nodes at 1% efficiency



35 nodes at 100% efficiency



TritonSort Goals

- Build a highly efficient DISC system that improves per-node efficiency by an order of magnitude vs. existing systems
  – Through balanced hardware and software
- Secondary goals:
  – Completely “off-the-shelf” components
  – Focus on I/O-driven workloads (“Big Data”)
  – Problems that don’t come close to fitting in RAM
  – Initially sorting, but have since generalized

Outline

- Define hardware and software balance
- TritonSort design
  - Highlighting tradeoffs to achieve balance
- Evaluation with sorting as a case study

Building a "Balanced" System

- *Balanced hardware* drives all resources as close to 100% as possible
  - Removing any resource slows us down
  - Limited by commodity configuration choices
- *Balanced software* fully exploits hardware resources

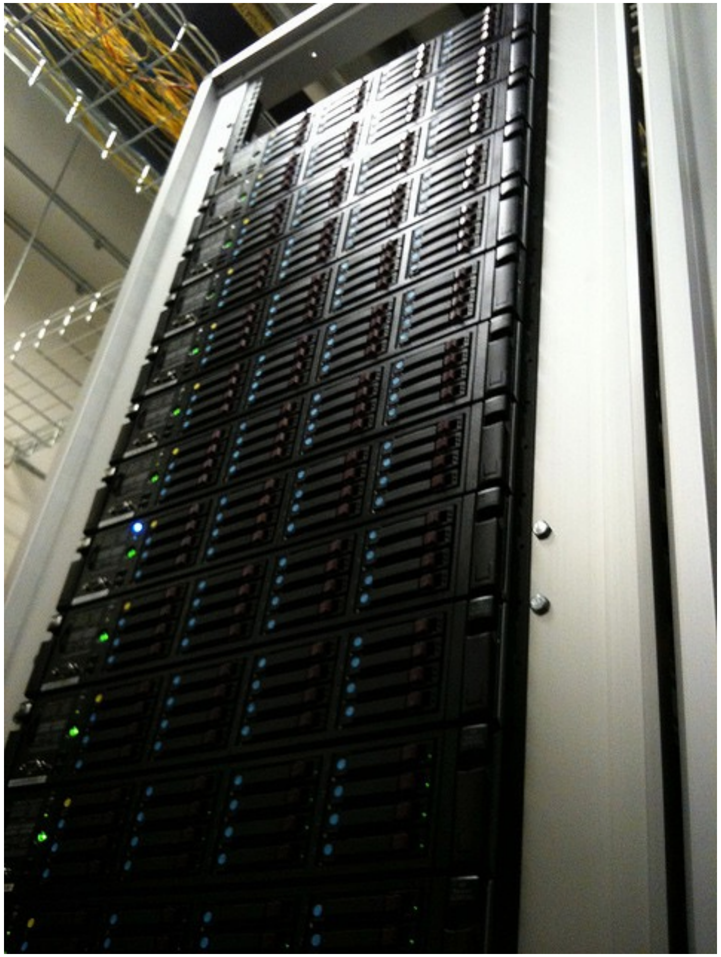
Hardware Selection (in 2010)

- Designed for I/O-heavy workloads
  - Not just sorting
- Static selection of resources:
  - Network/disk balance
    - 10 Gbps / 80 MBps ≈ 16 disks
  - CPU/disk balance
    - 2 disks / core = 8 cores
  - CPU/memory
    - Originally ~1.5GB/core… later 3 GB/core

Resulting Hardware Platform

52 Nodes:

- Xeon E5520, 8 cores
  (16 with hyperthreading)
- 24 GB RAM
- 16 7200 RPM hard drives
- 10 Gbps NIC
- Cisco Nexus 5020
  10 Gbps switch



Software Architecture

- Staged, pipeline-oriented dataflow system
- Program expressed as digraph of stages
  - Data stored in buffers that move along edges
  - Stage's work performed by *worker* threads
- Platform for experimentation
  - Easily vary:
    - Stage implementation
    - Size and quantity of buffers
    - Worker threads per stage
    - CPU and memory allocation to each stage

Why Sorting?

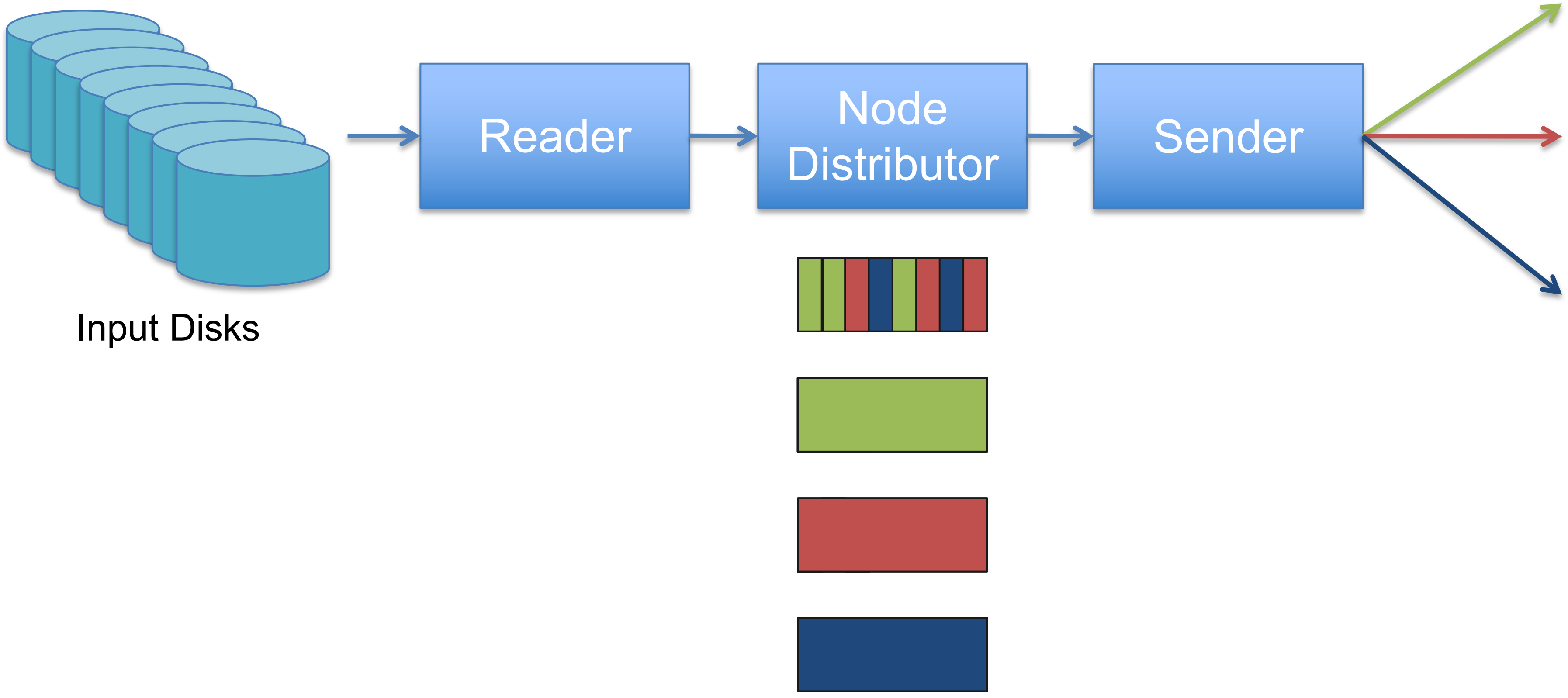
- Easy to describe
- Industrially applicable
- Uses all cluster resources

Current TritonSort Architecture

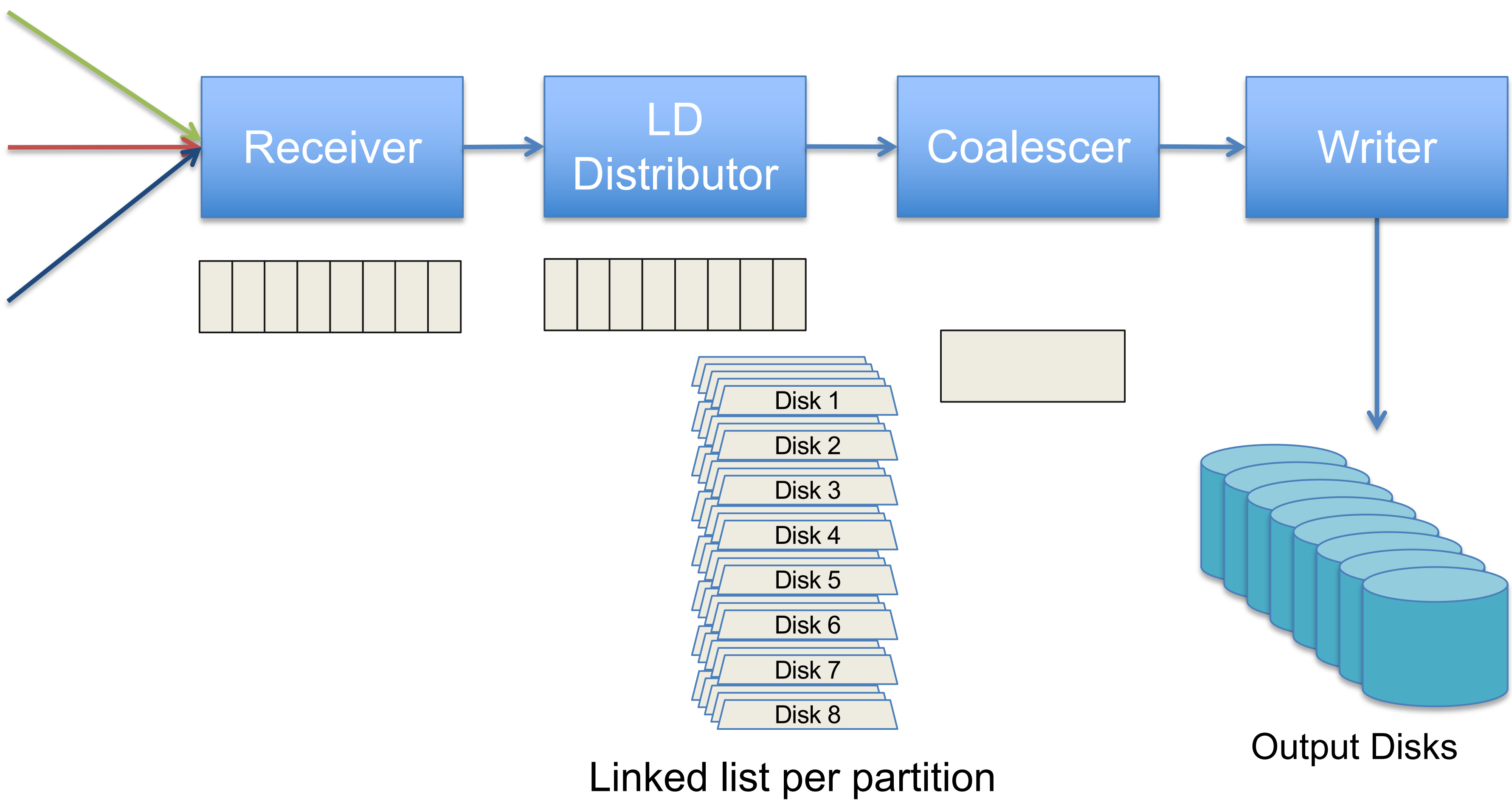
- External sort – two reads, two writes*
  - Don't read and write to disk at same time
    - Partition disks into input and output
- Two phases
  - **Phase one**: route tuples to appropriate on-disk partition (called a "logical disk") on appropriate node
  - **Phase two**: sort all logical disks in parallel

* A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. CACM, 1988.

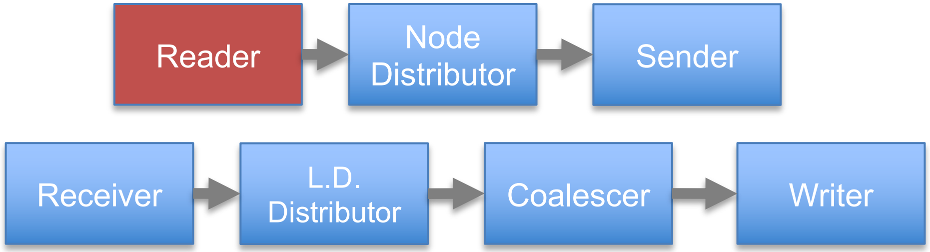
Architecture Phase One



Architecture Phase One

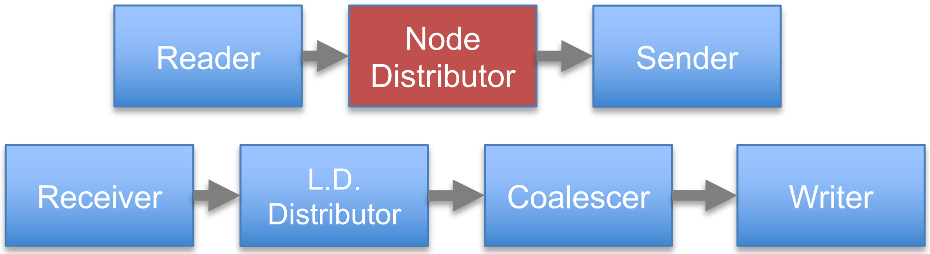


Reader



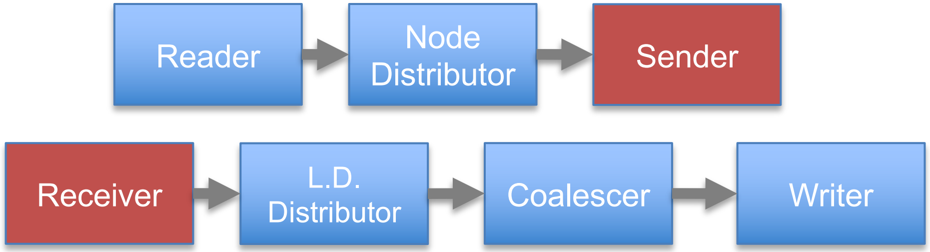
- 100 MBps/disk * 8 disks = 800 MBps
- No computation, entirely I/O and memory operations
  - Expect most time spent in iowait
  - 8 reader workers, one per input disk
    - ✓ All reader workers co-scheduled on a single core

NodeDistributor



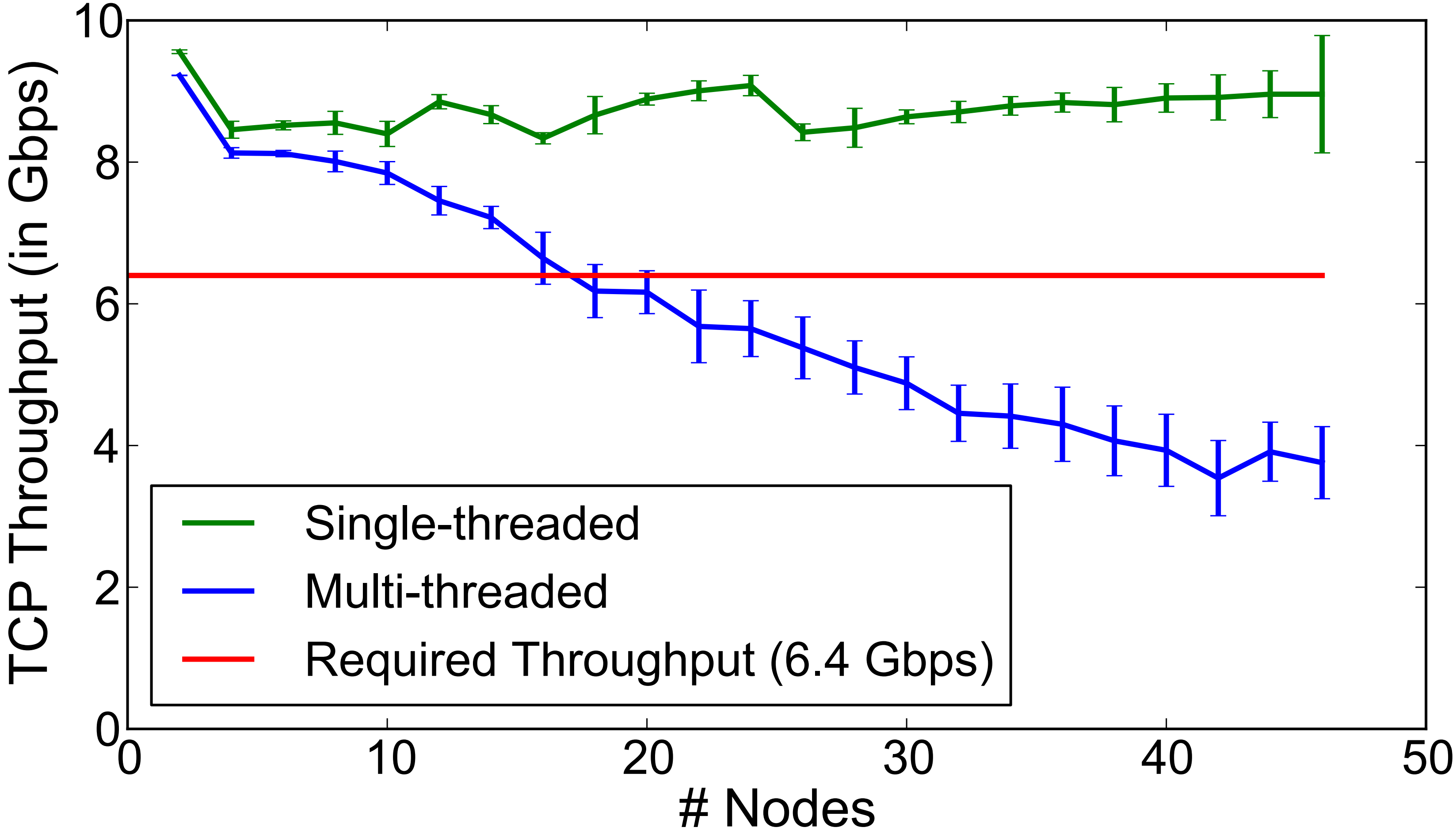
- Appends tuples onto a buffer per destination node
- Memory scan + hash per tuple
- 300 MBps per worker
  - Need three workers to keep up with readers

Sender & Receiver

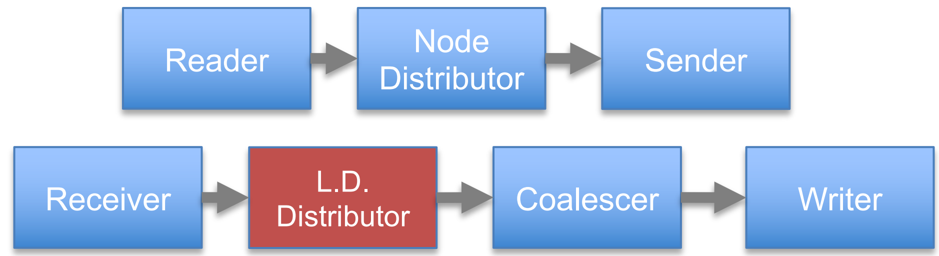


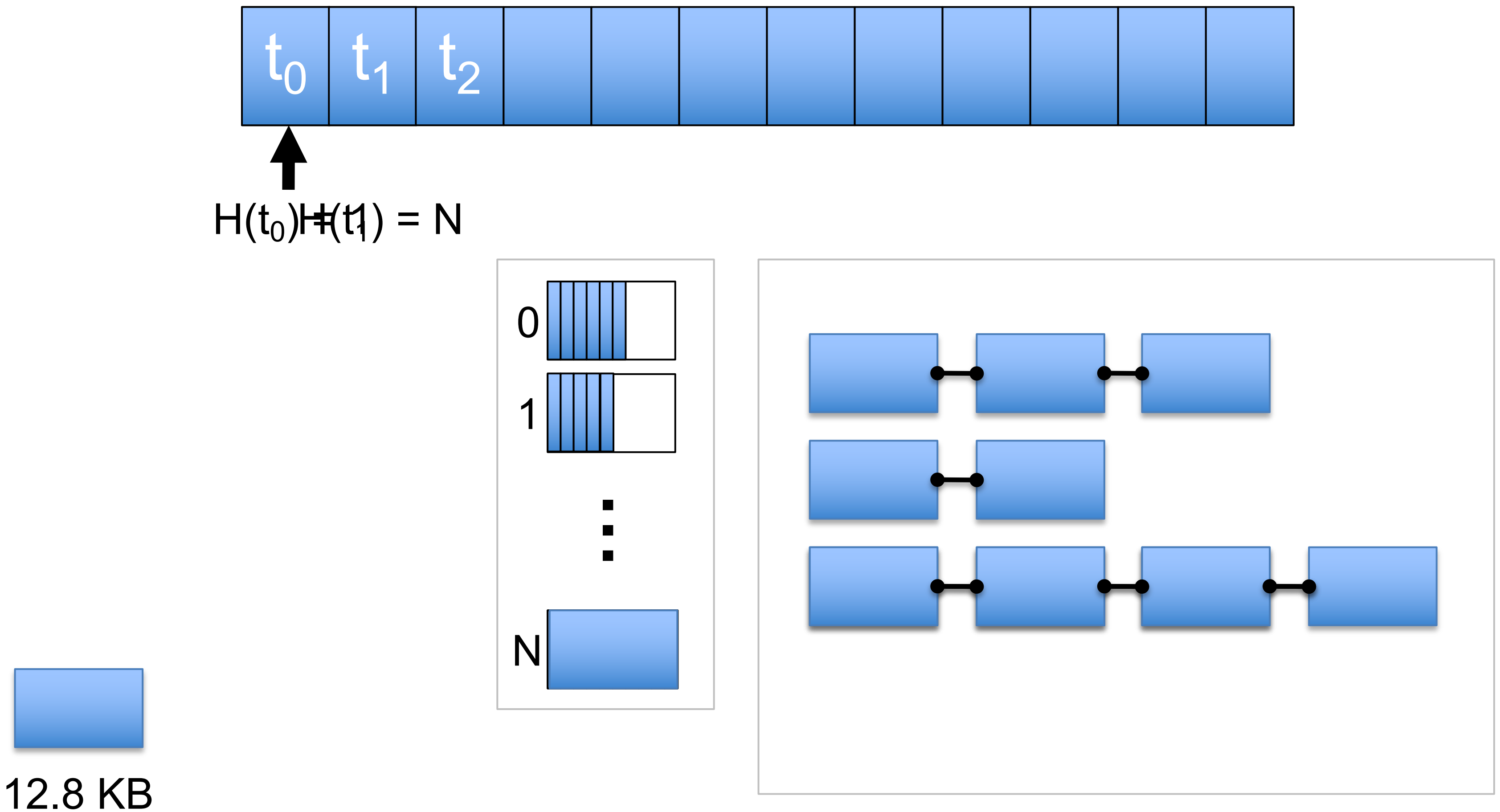
- 800 MBps (from Reader) is 6.4 Gbps
  – All-to-all traffic
- Must keep downstream disks busy
  – Don’t let receive buffer get empty
  – Implies strict socket send time bound
- Multiplex all senders on one core (single-threaded tight loop)
  – Visit every socket every 20 μs
  – Didn’t need epoll()/select()

Balancing at Scale

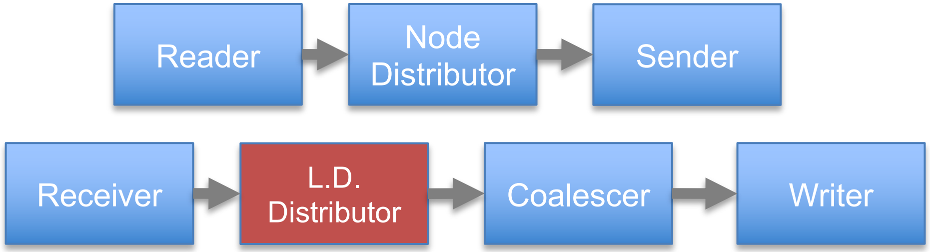


Logical Disk Distributor



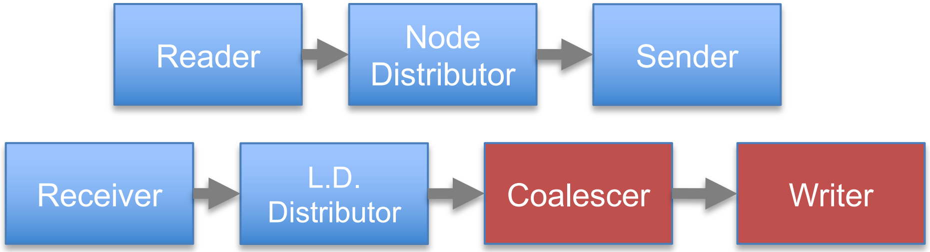


Logical Disk Distributor



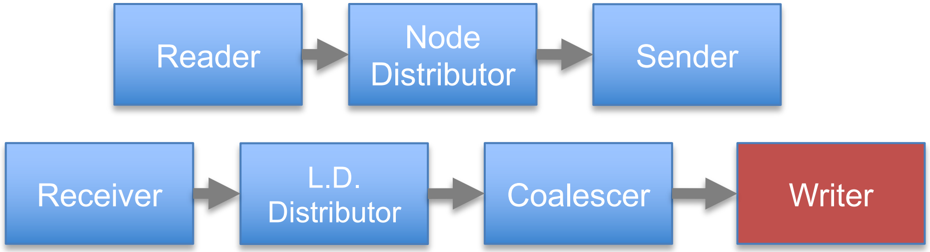
- Data non-uniform and bursty at short timescales
  - Big buffers + burstiness = head-of-line blocking
  - Need to use *all* your memory *all* the time
- **Solution**: Read incoming data into smallest buffer possible, and form chains

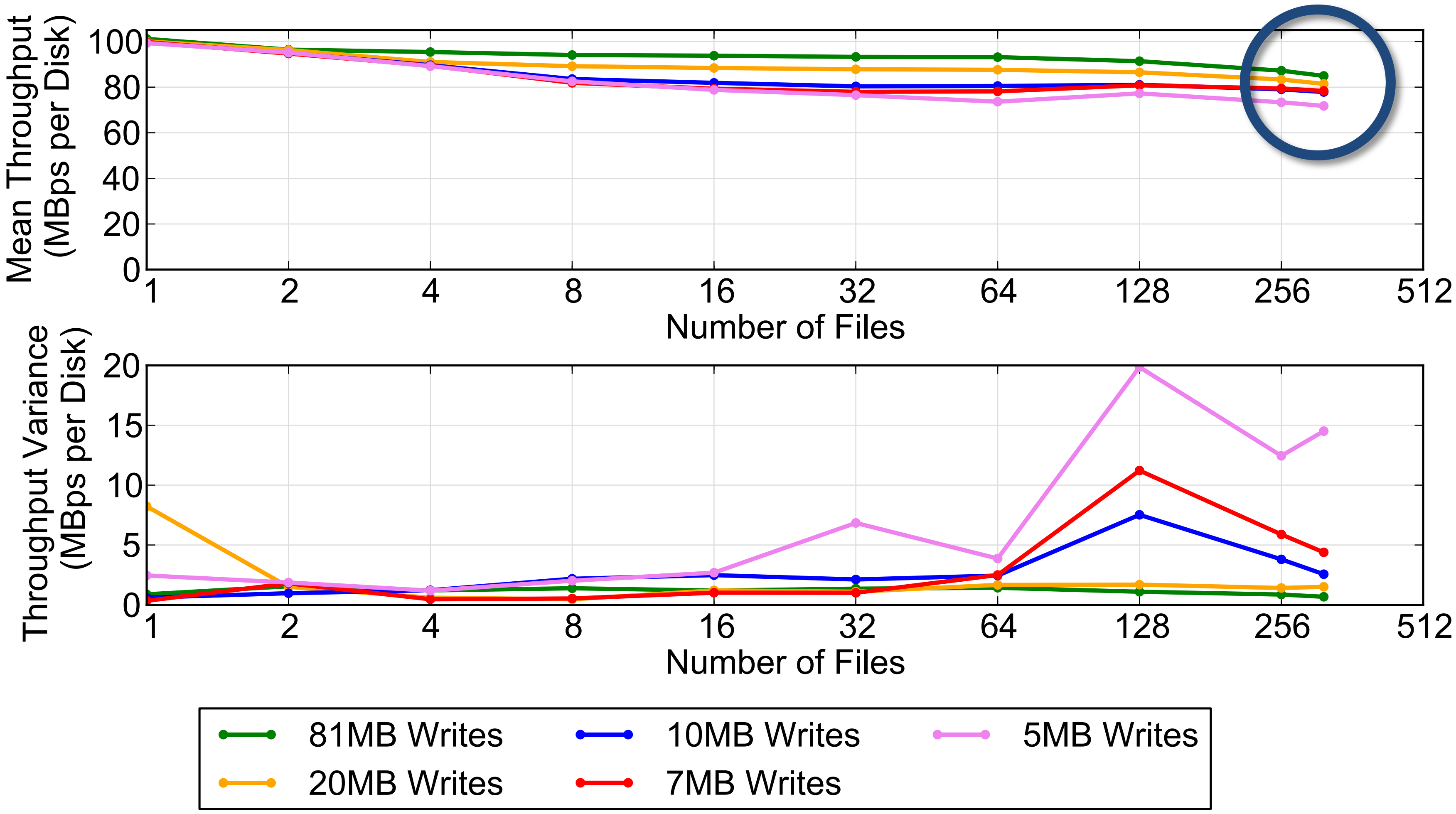
Coalescer & Writer



- Copies tuples from LDBuffer chains into a single, sequential block of memory
- Longer chains = larger write before seeking = faster writes
  - Also, more memory needed for LDBuffers
- Buffer size limits maximum chain length
  - How big should this buffer be?

Writer

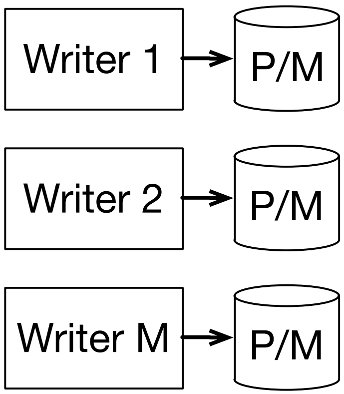

Reader
Node Distributor
Sender
Receiver
L.D. Distributor
Coalescer
Writer


Mean Throughput (MBps per Disk)
100
80
60
40
20
0
1
2
4
8
16
32
64
128
256
512
Number of Files
Throughput Variance (MBps per Disk)
20
15
10
5
0
1
2
4
8
16
32
64
128
256
512
Number of Files
81MB Writes
10MB Writes
5MB Writes
20MB Writes
7MB Writes

Appending records to partitions

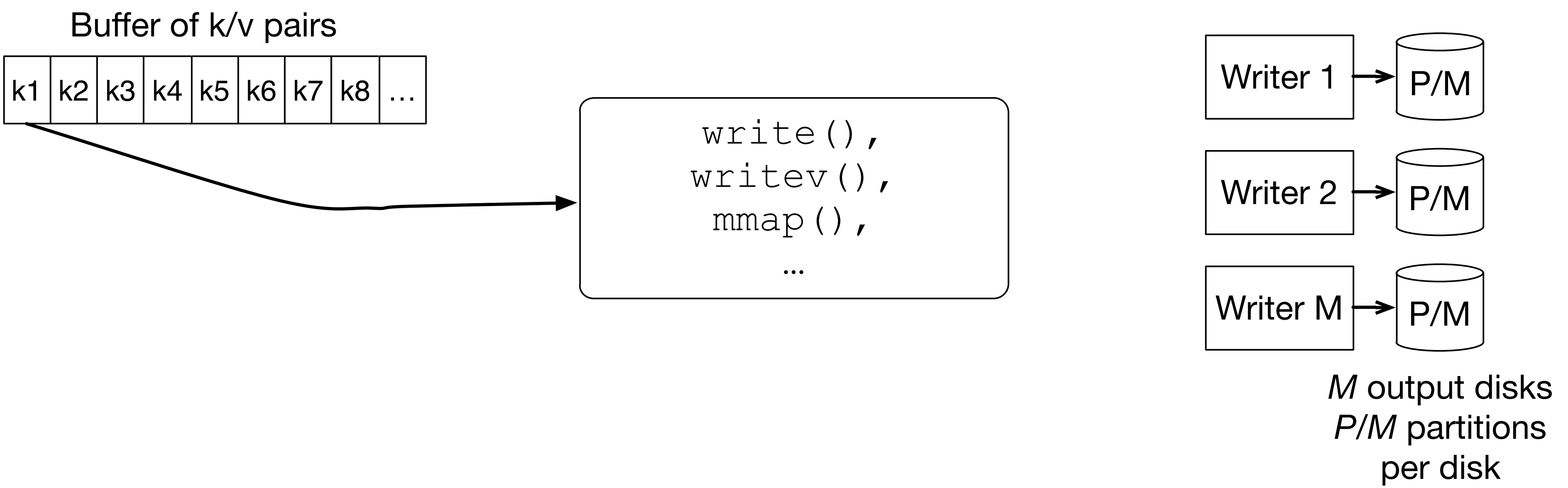
Buffer of k/v pairs

k1	k2	k3	k4	k5	k6	k7	k8	…



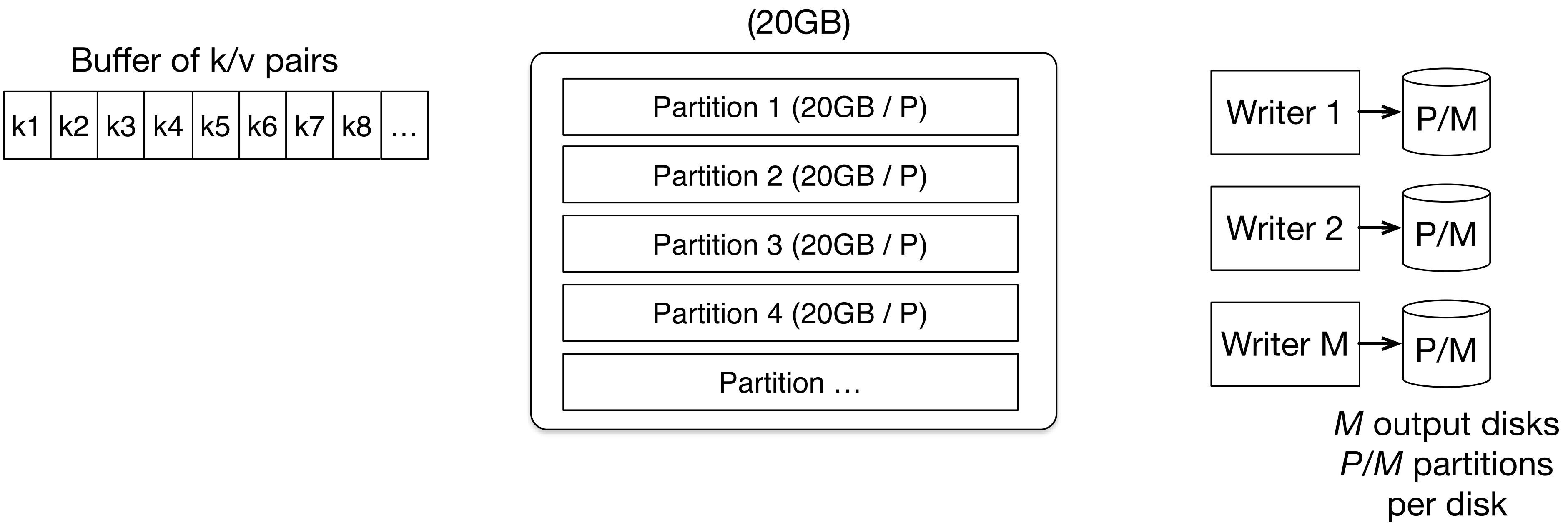
M output disks
P/M partitions
per disk

Approach #1: Delegate to OS

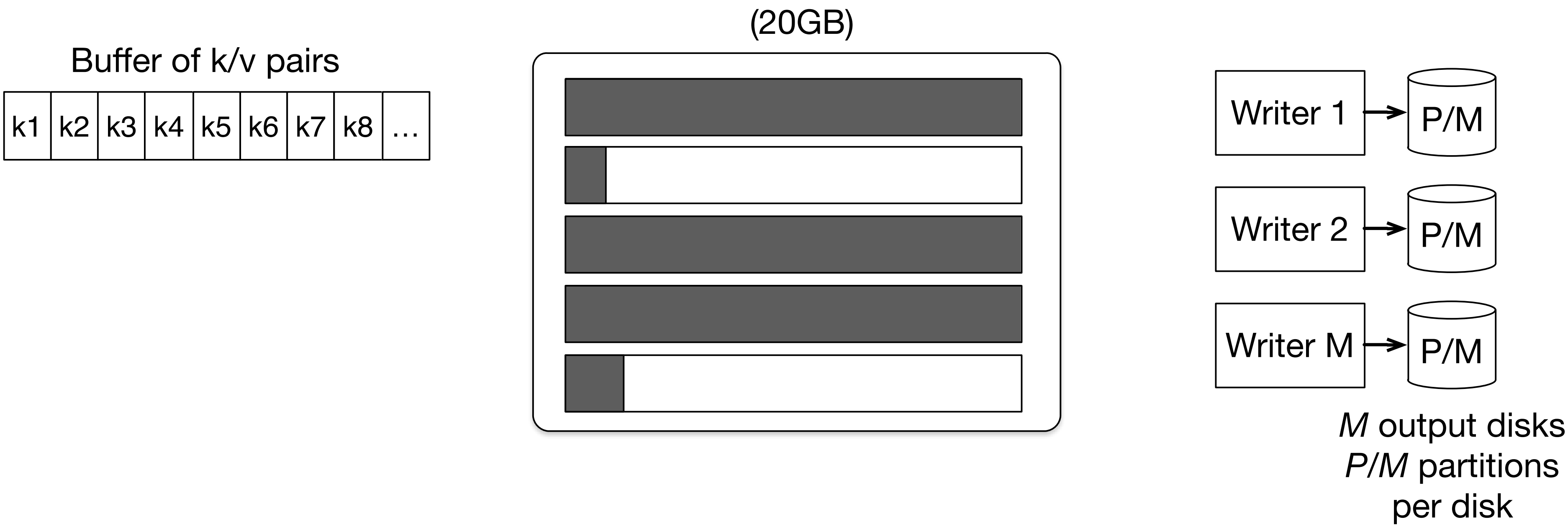


– Low performance due to insufficient batching

Approach #2: Per-partition buffers

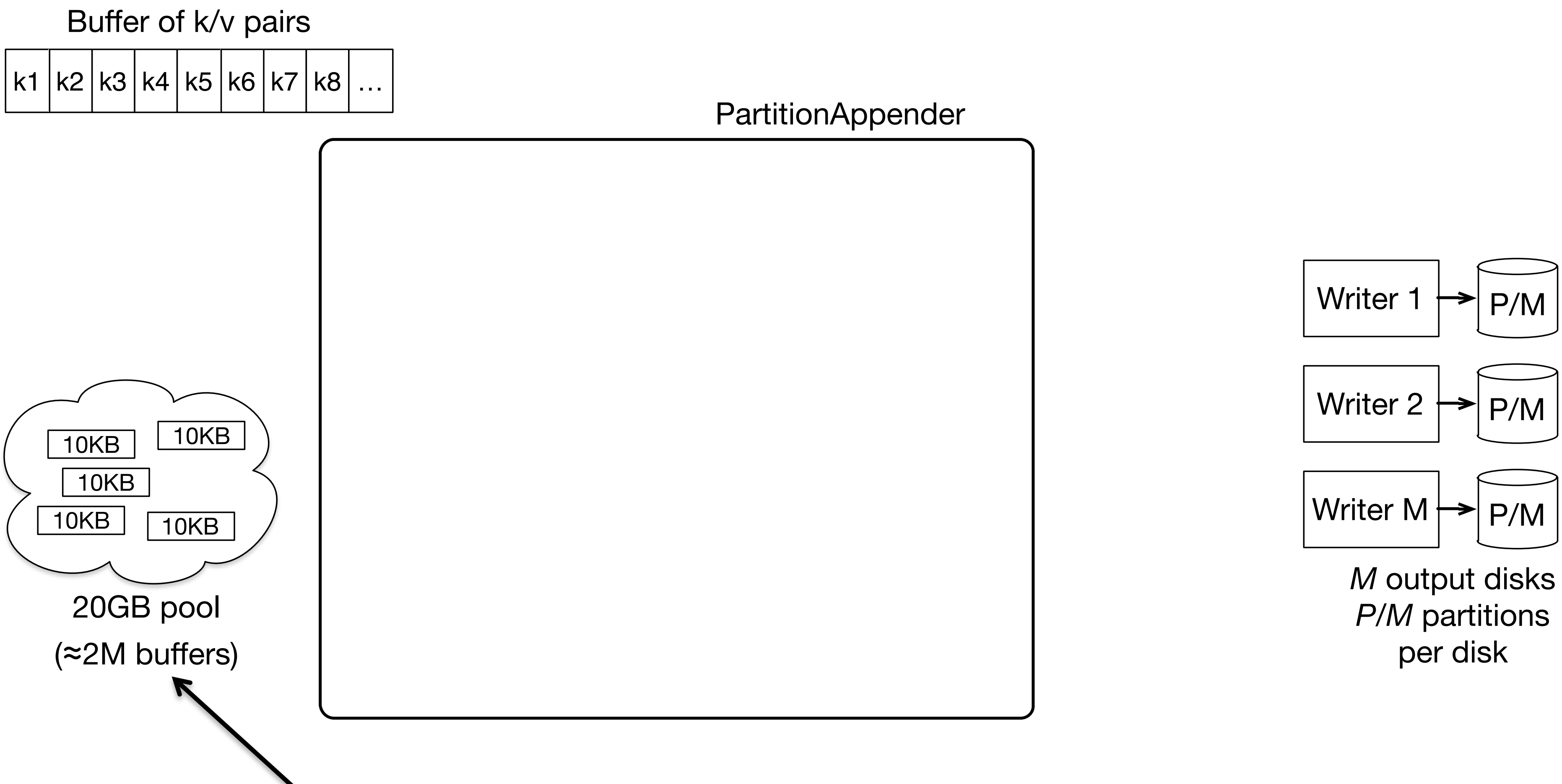


Approach #2: Per-partition buffers



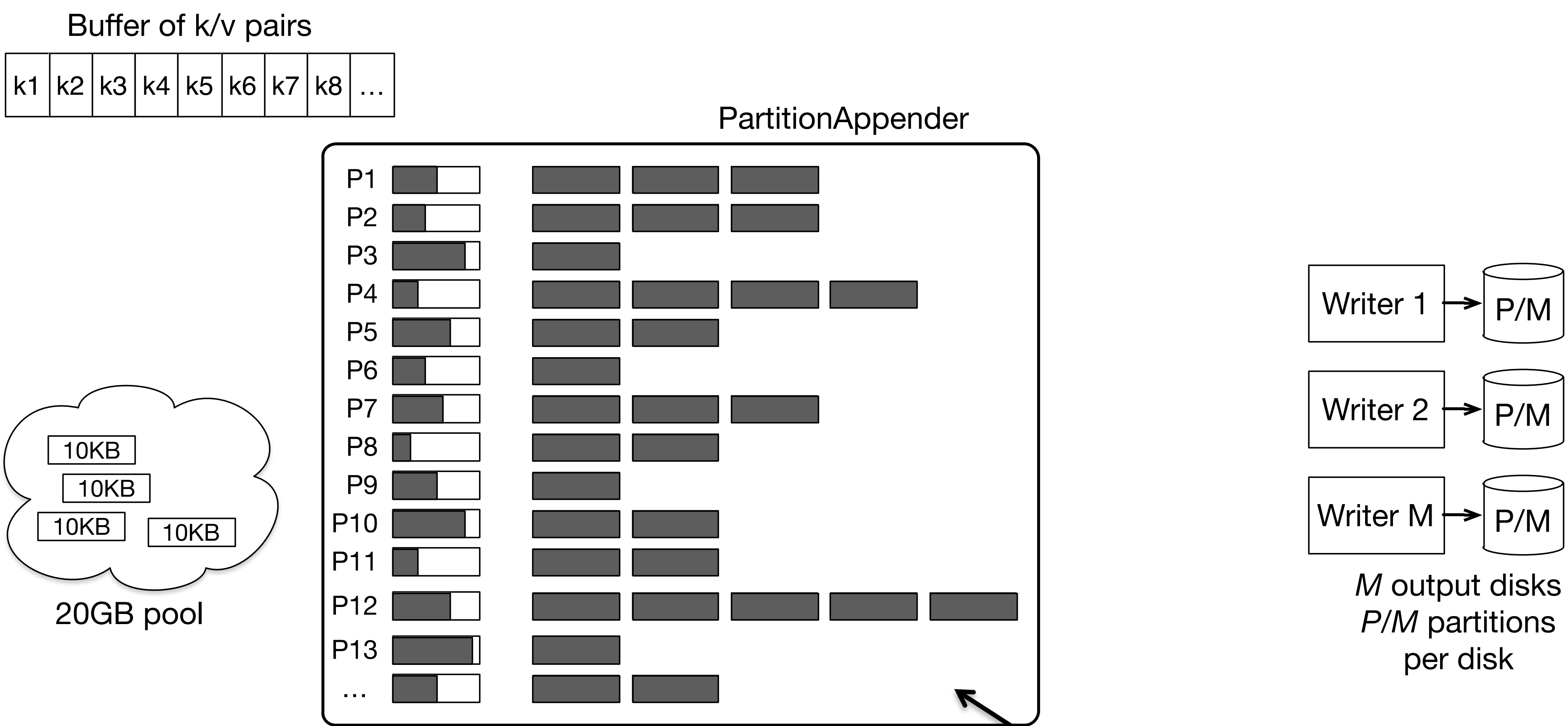
– Non-uniform arrivals result in “hot” buffers

TritonSort: Load-balancing across partitions



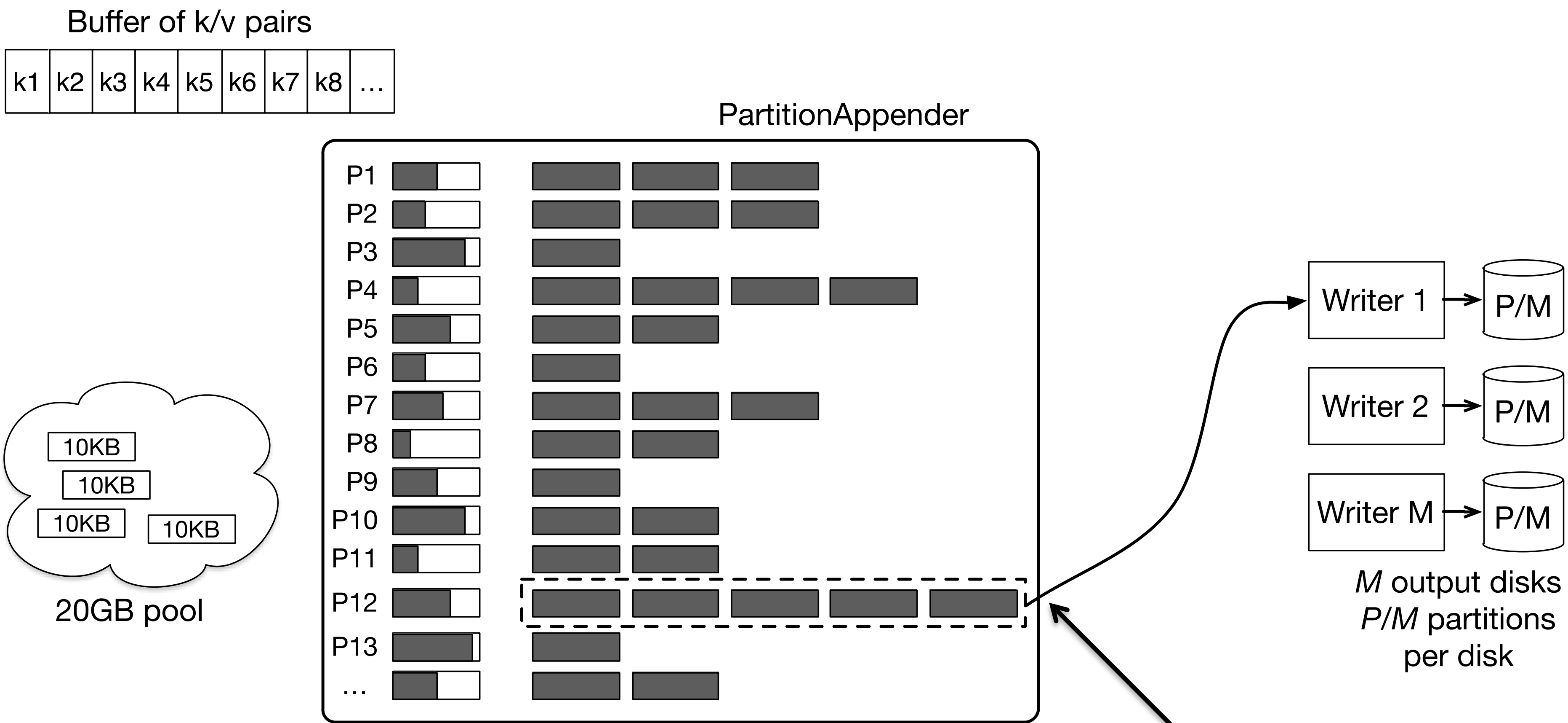
Fine-grained allocation of small buffers to partitions

TritonSort: Load-balancing across partitions



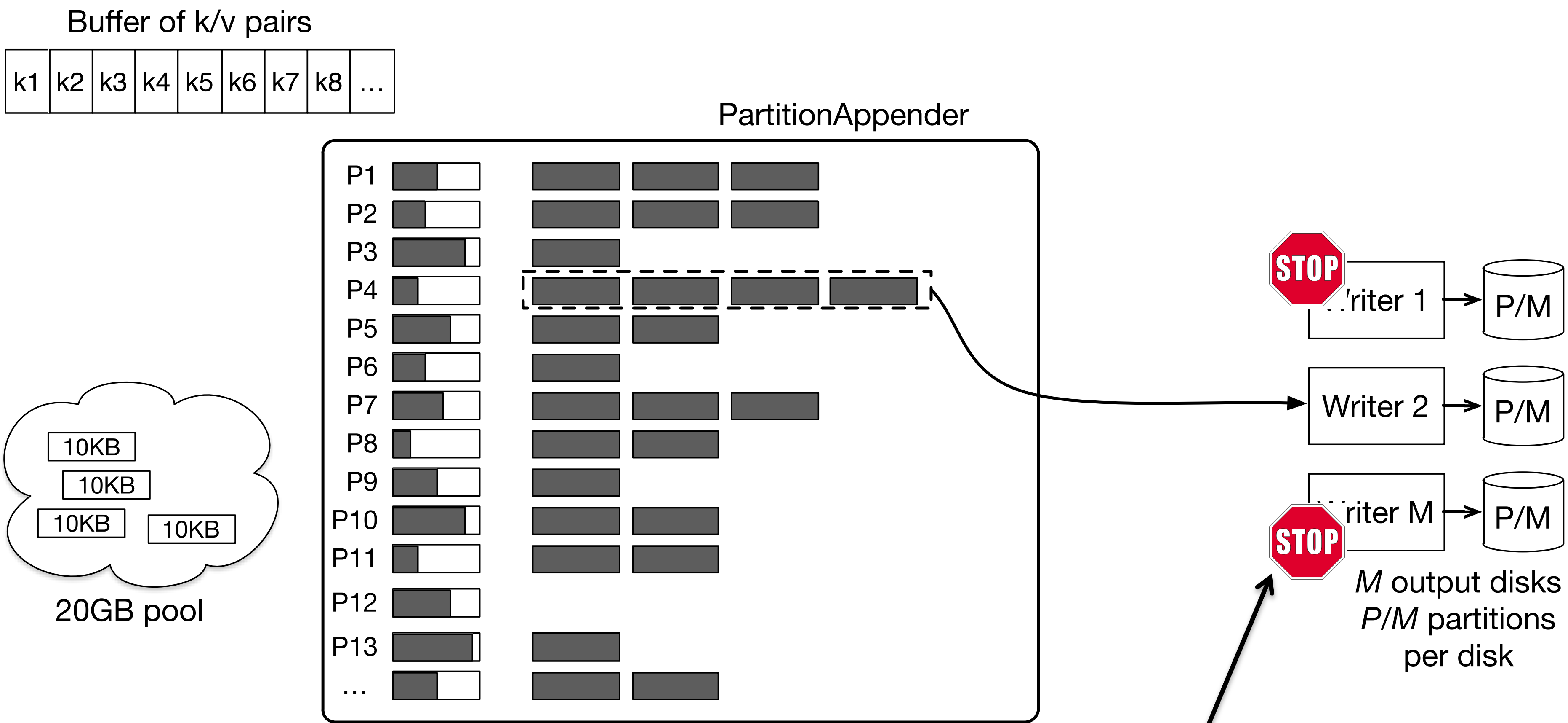
One "chain" of buffers per partition

Handling "hot" partitions



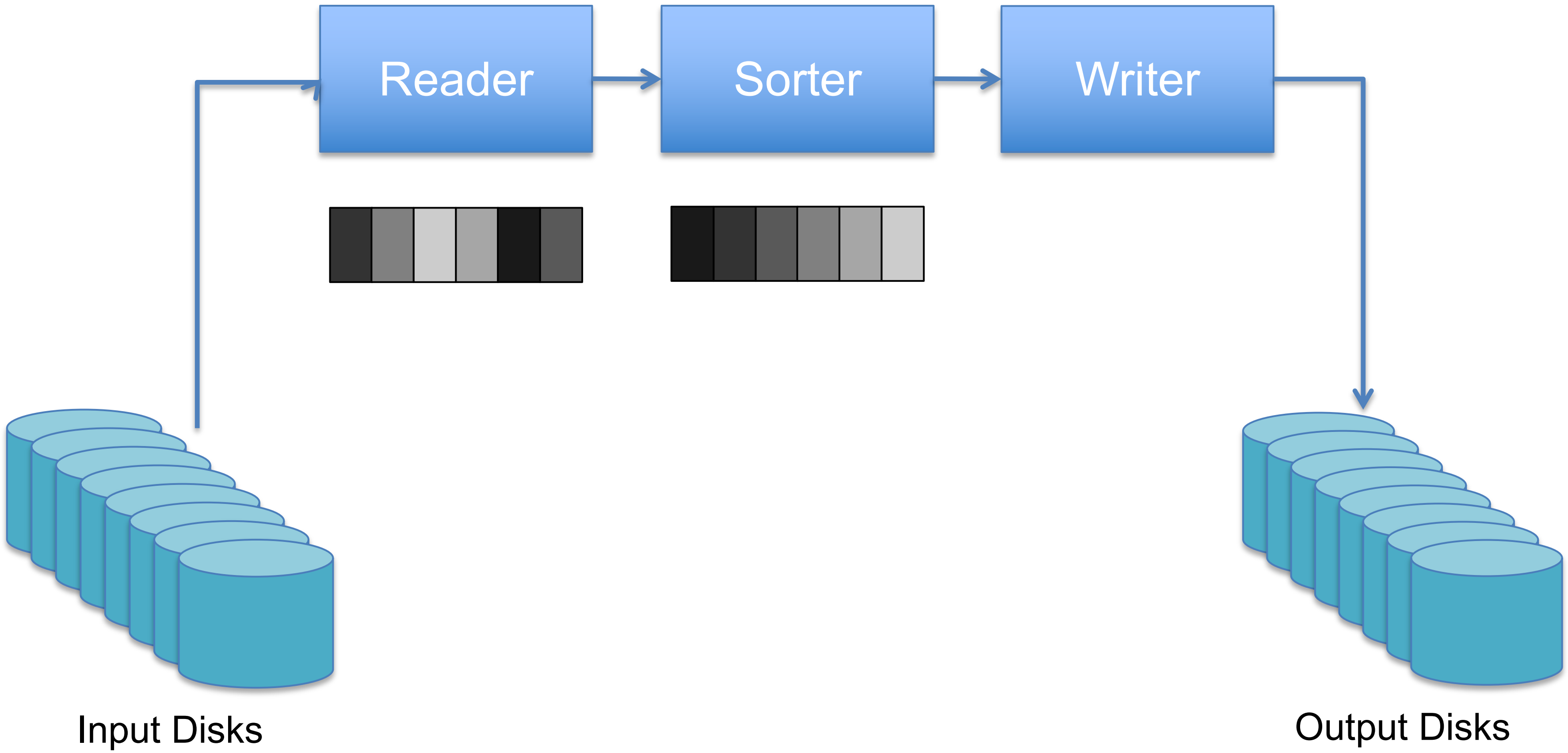
Largest chain = Largest possible write

Handling slow disks



Slow disks accept writes less often, leading to larger writes

Architecture Phase Two



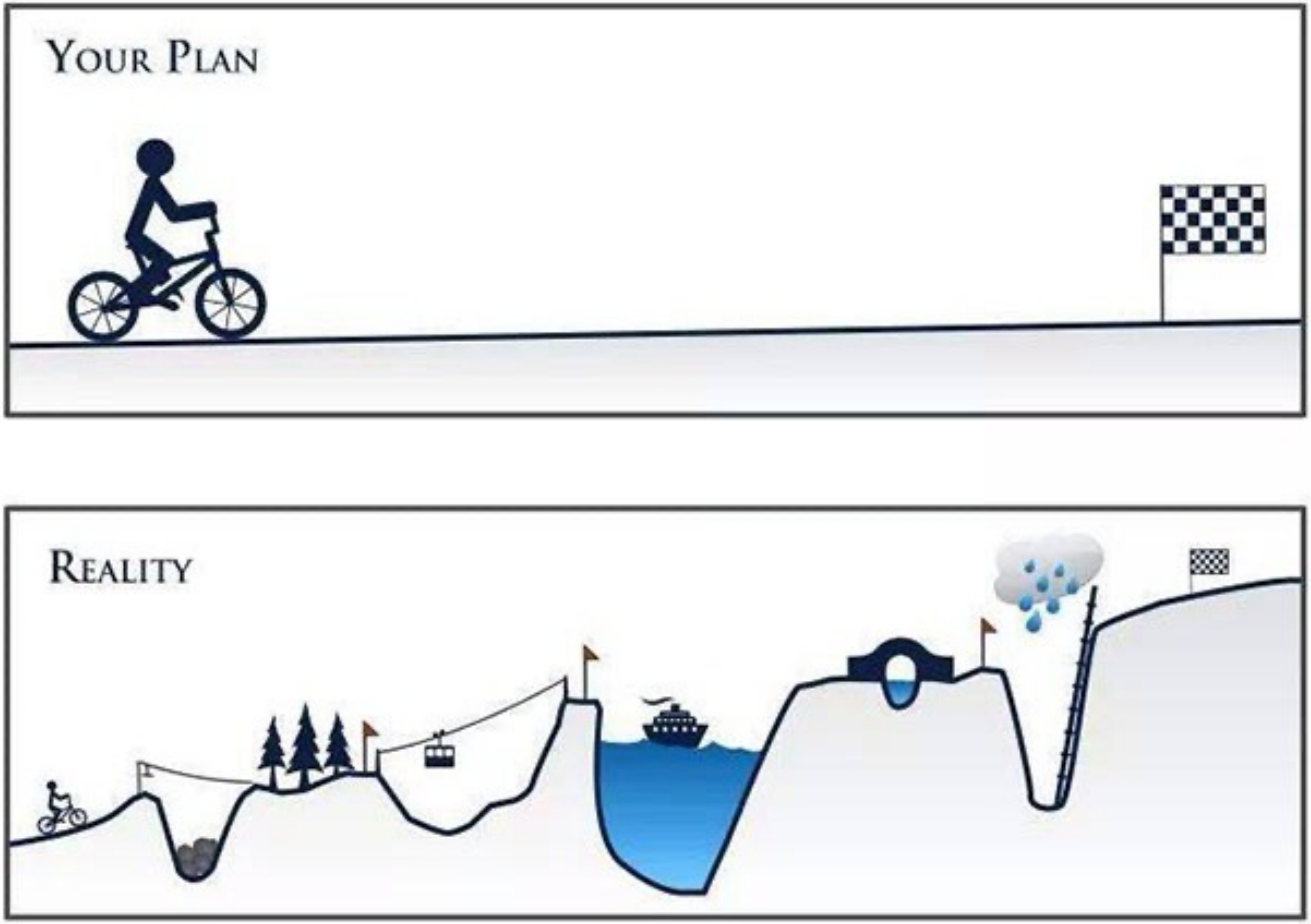
Evaluation

	100TB GraySort		100TB JouleSort
2009	Hadoop MapReduce	0.578 TB/min 3,452 nodes (2.8 MB/sec/node)	
2010	TritonSort	0.725 TB/min 52 nodes (232.4 MB/sec/node)	Created in 2010
2011	TritonSort		TritonSort
2012	TritonSort		TritonSort
2013	Hadoop MapReduce	1.42 TB/min 2,200 nodes (11.3 MB/sec/node)	TritonSort

Going after CloudSort

- Our team, with lead student Mike Conley, ported Themis to Amazon's Cloud Infrastructure
- Goal:
  - Learn how to migrate a system designed for **dedicated** resources to an **on-demand service**
  - Break the record using cloud computing

The project in a nutshell...

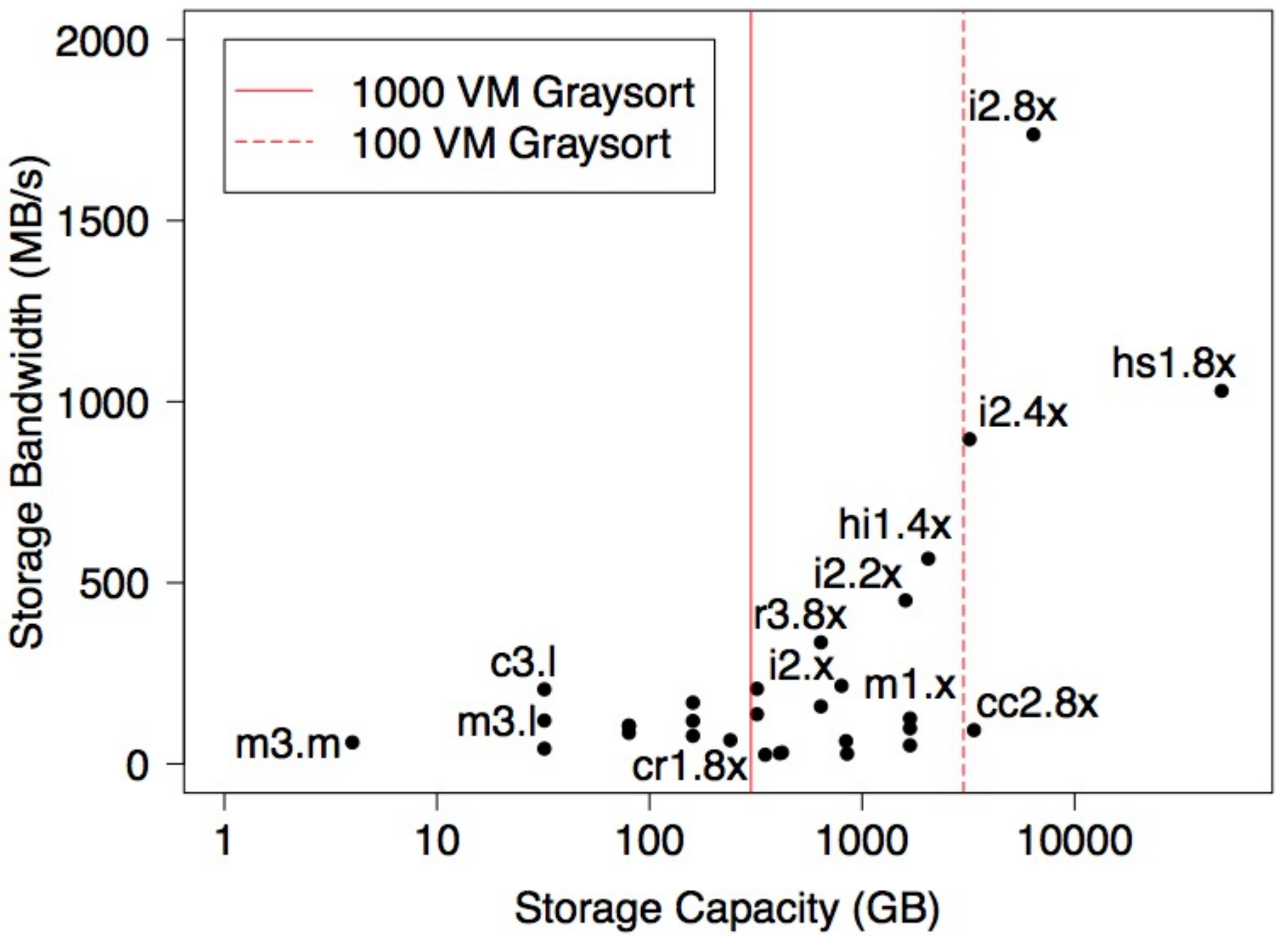


Key challenges

- "The cloud" often doesn't have any rain
  - We frequently couldn't get enough nodes ☹
- Performance(N-node cluster) !=

  N * Performance(1 node)
- Network bandwidth is not good

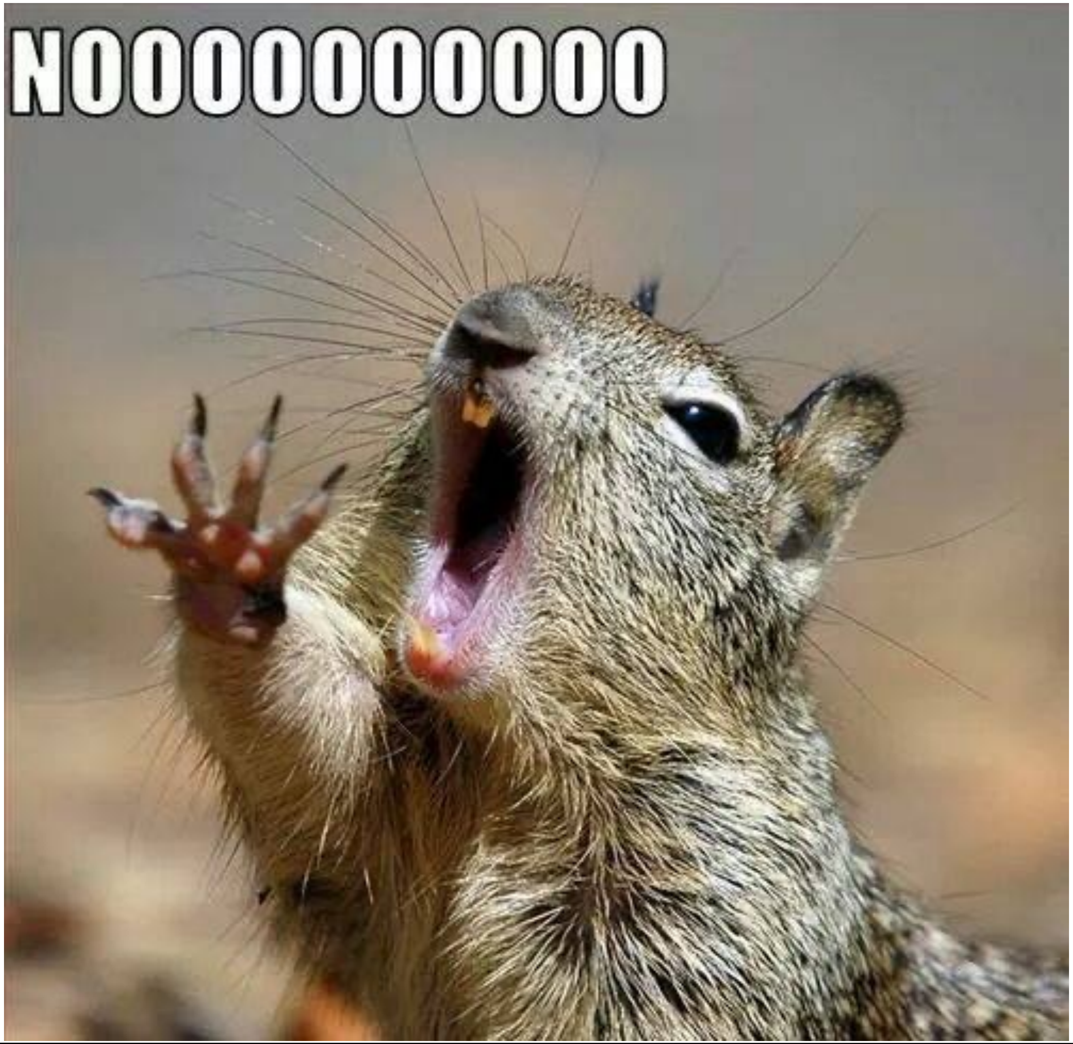
Characterizing each type of node



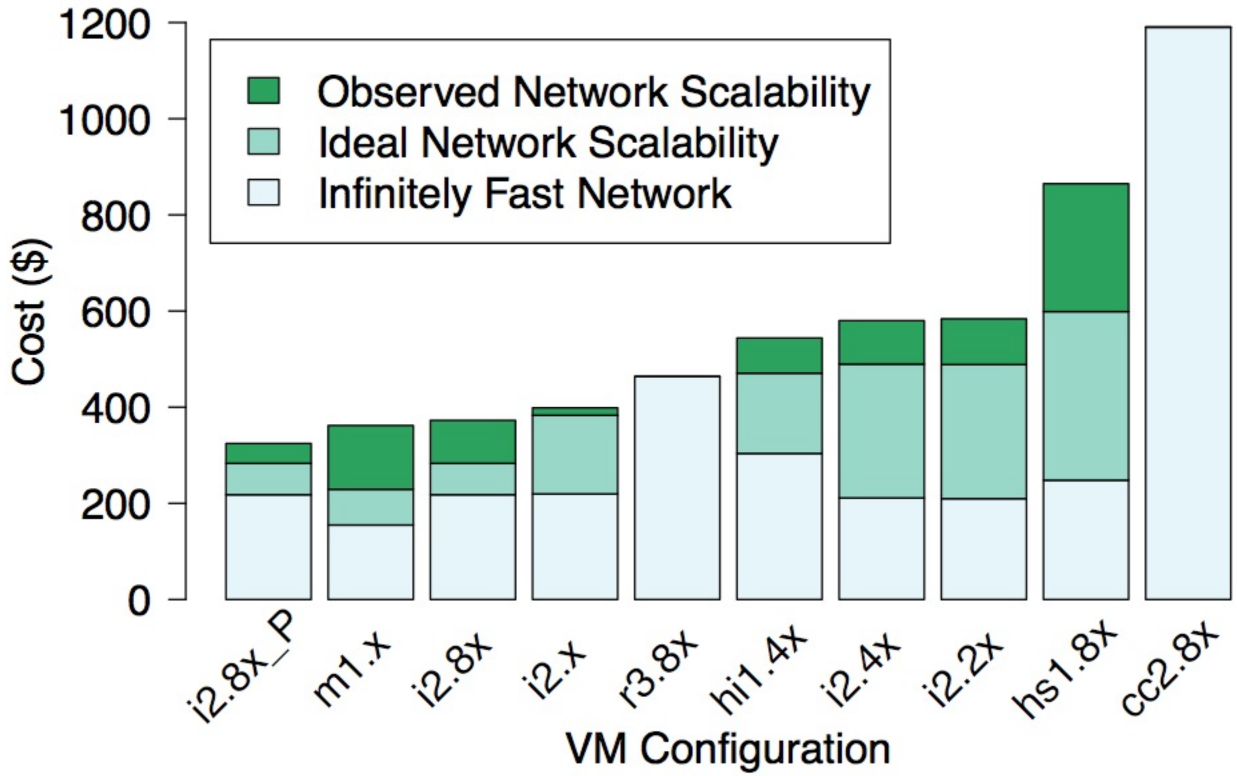
What a 100TB sort *should* cost

Instance type	Num nodes to sort 100 TB	Sort cost ($)
c3.large	9,375	28
m3.large	9,375	65
m3.medium	75,000	66
m1.xlarge	179	155
i2.4xlarge	94	211
i2.8xlarge	47	218
hs1.8xlarge	7	248
cr1.8xlarge	1,250	2,966

But then... the network...



Factoring in the network



Results

Category	Previous record	UCSD 2014 Result
Indy GraySort	1.42 TB/min (2,100 nodes)	6.76 TB/min (178 nodes)
Daytona GraySort	1.42 TB/min (2,100 nodes)	4.35 TB/min (186 nodes)
Indy MinuteSort	1401 GB (256 nodes)	4094 GB (178 nodes)
Indy CloudSort	N/A	$449.53 (330 nodes)
Daytona CloudSort	N/A	$449.53 (330 nodes)

For more information

- TritonSort: A Balanced Large-Scale Sorting System, Alexander Rasmussen, George Porter, Michael Conley, Harsha V. Madhyastha, Radhika Niranjan Mysore, Alexander Pucher, and Amin Vahdat, Proceedings of the 8th ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA, March 2011.

- Themis: An I/O-Efficient MapReduce, Alexander Rasmussen, Michael Conley, Rishi Kapoor, Vinh The Lam, George Porter, and Amin Vahdat, Proceedings of the ACM Symposium on Cloud Computing (SOCC), San Jose, CA, October 2012.