TRITONSORT

A Balanced Large-Scale Sorting System

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, …

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

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

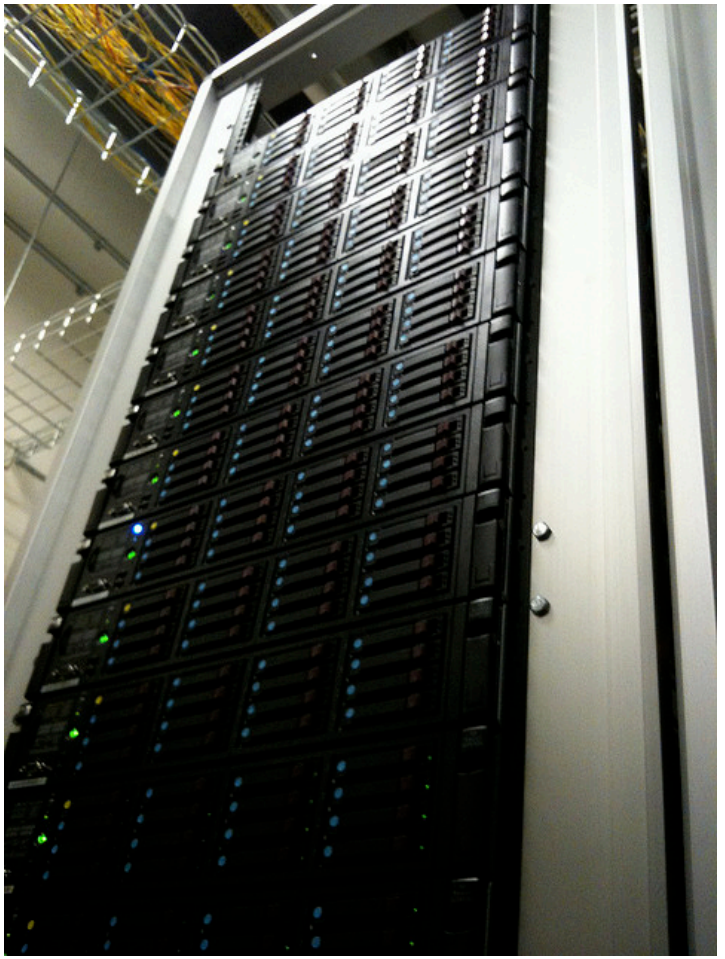
Hardware Selection

- 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?

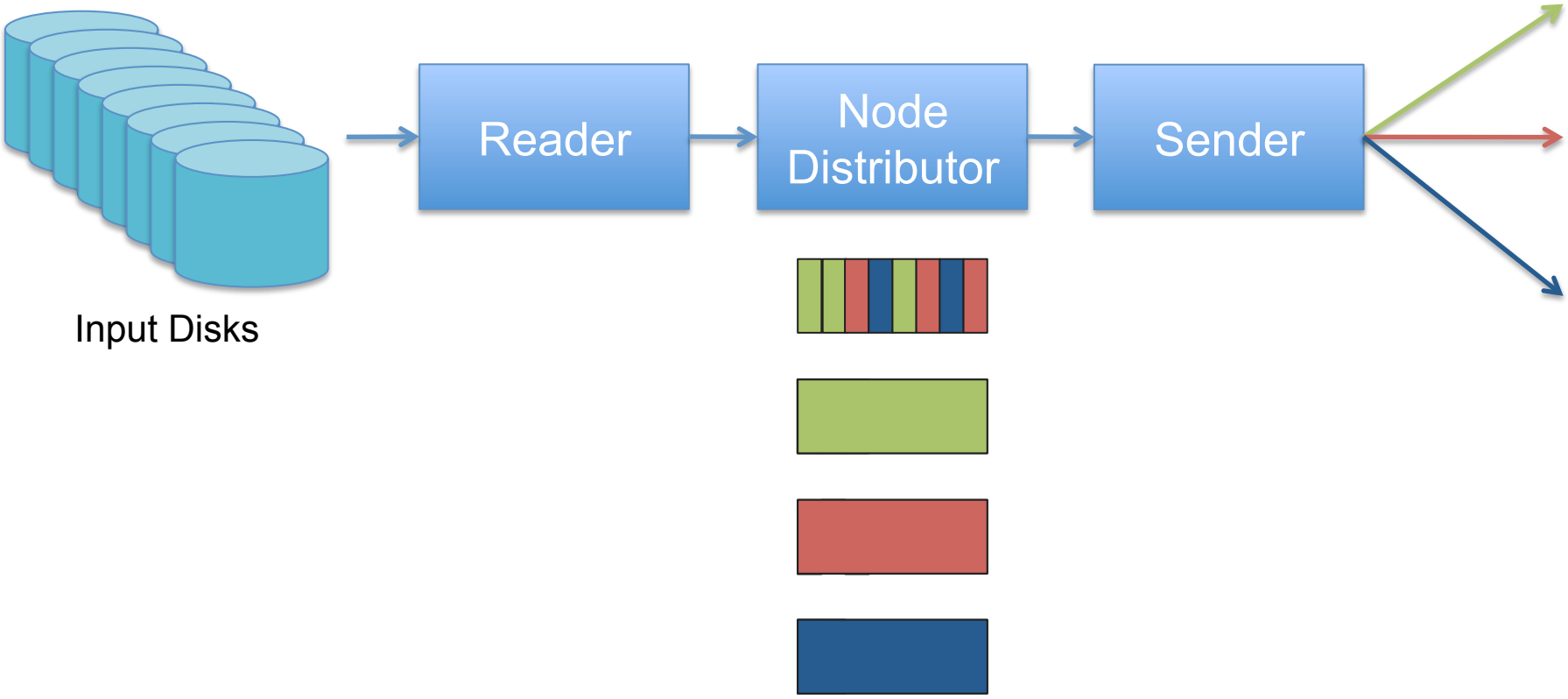
- Easy to describe
- Industrially applicable
- Uses all cluster resources

Current TritonSort Architecture

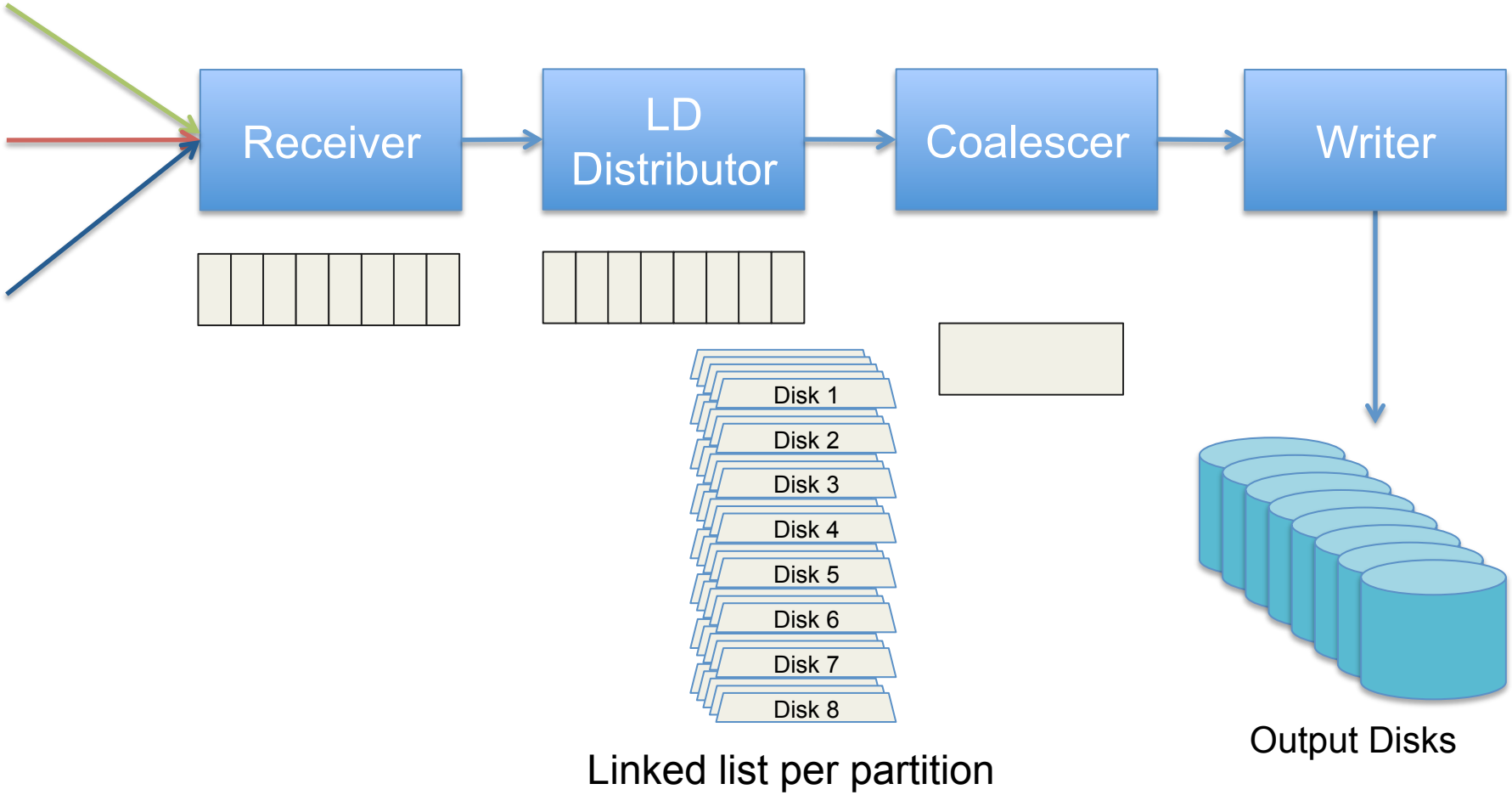
- External sort – two reads, two writes*
  - Don't read and write to disk at same time
    - Partition disks into input and output
- Two phases
  - **Phase one**: route tuples to appropriate on-disk partition (called a "logical disk") on appropriate node
  - **Phase two**: sort all logical disks in parallel

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

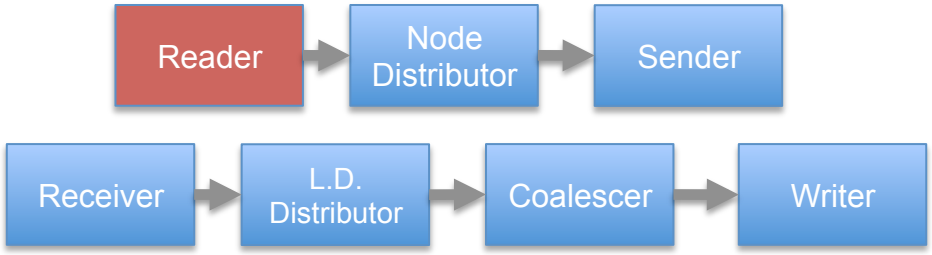
Architecture Phase One



Architecture Phase One

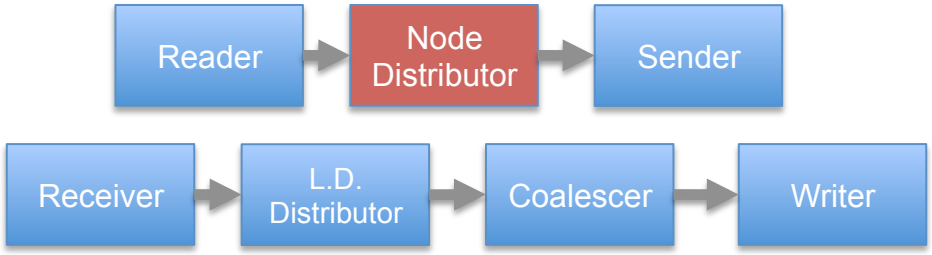


Reader



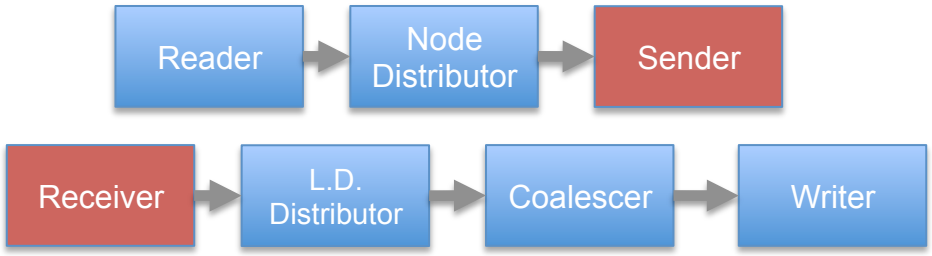
- 100 MBps/disk * 8 disks = 800 MBps
- No computation, entirely I/O and memory operations
  - Expect most time spent in iowait
  - 8 reader workers, one per input disk
    - ✓All reader workers co-scheduled on a single core

NodeDistributor



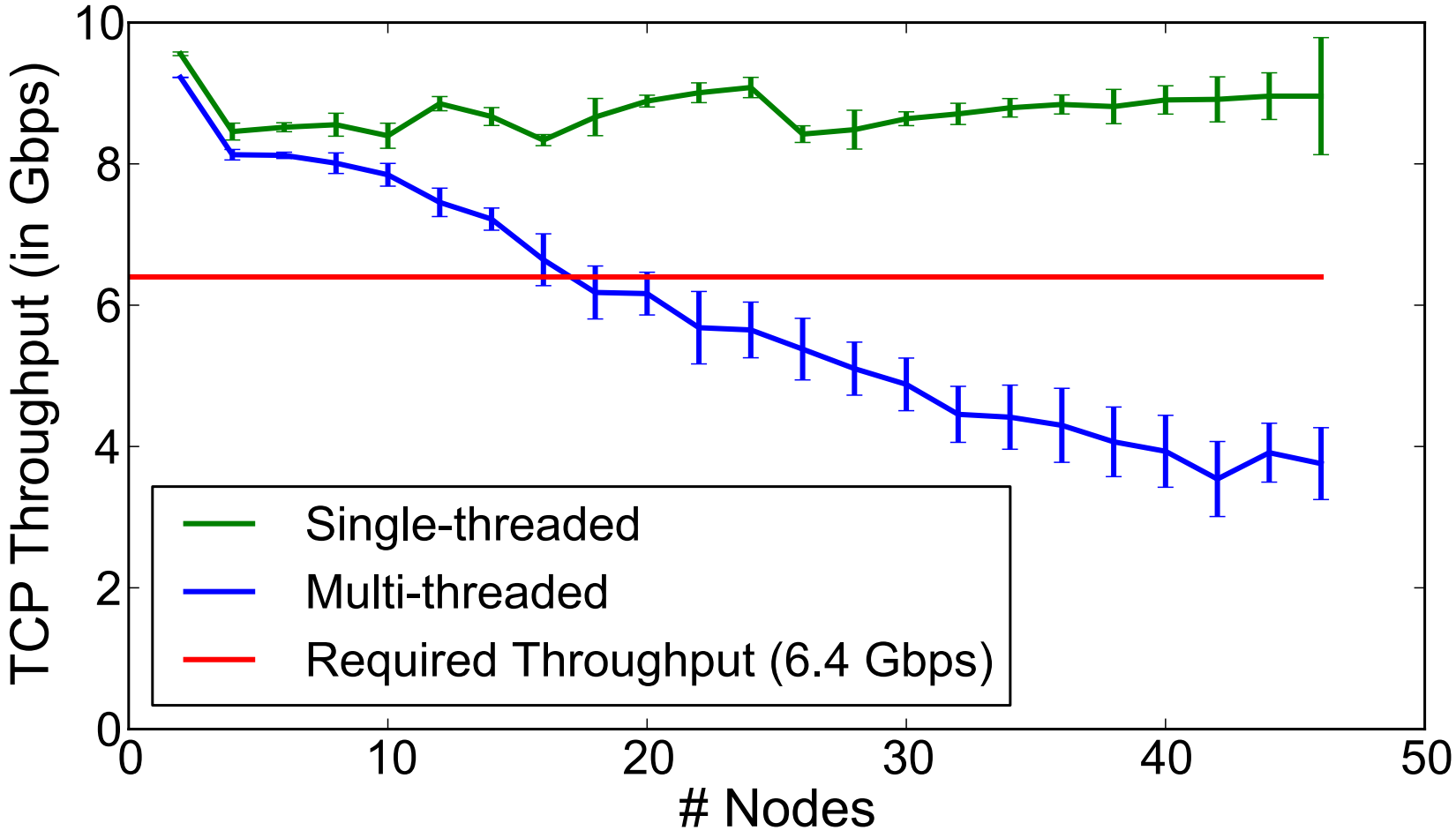
- Appends tuples onto a buffer per destination node
- Memory scan + hash per tuple
- 300 MBps per worker
  - Need three workers to keep up with readers

Sender & Receiver

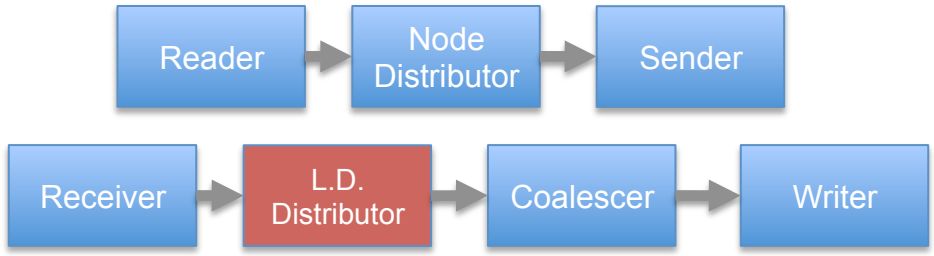


- 800 MBps (from Reader) is 6.4 Gbps
  - All-to-all traffic
- Must keep downstream disks busy
  - Don't let receive buffer get empty
  - Implies strict socket send time bound
- Multiplex all senders on one core (single-threaded tight loop)
  - Visit every socket every 20 µs
  - Didn't need epoll()/select()

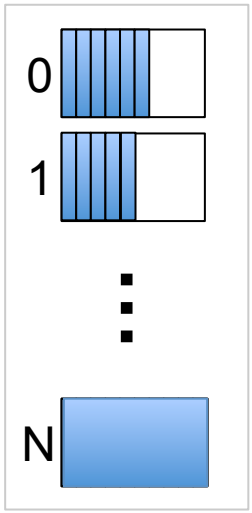
Balancing at Scale

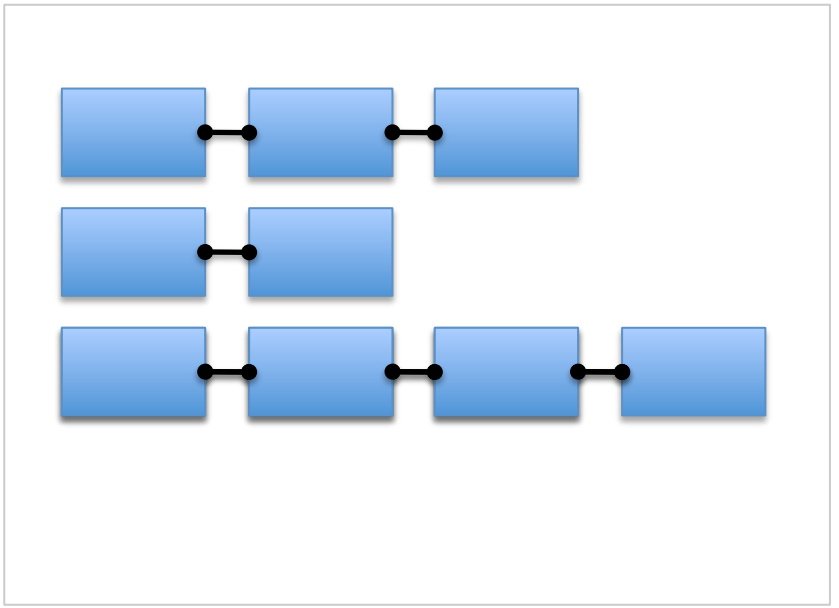


Logical Disk Distributor



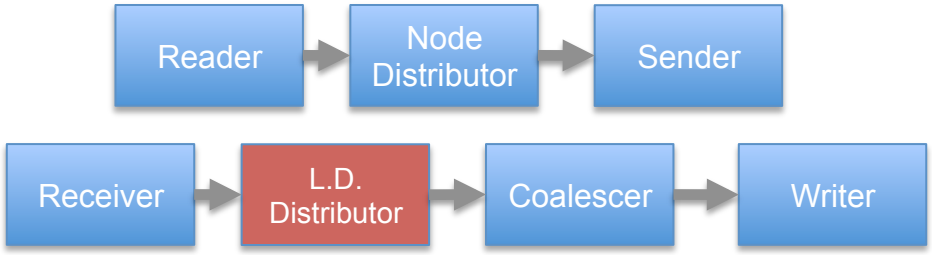
$H(t_0) \neq H(t_1) = N$





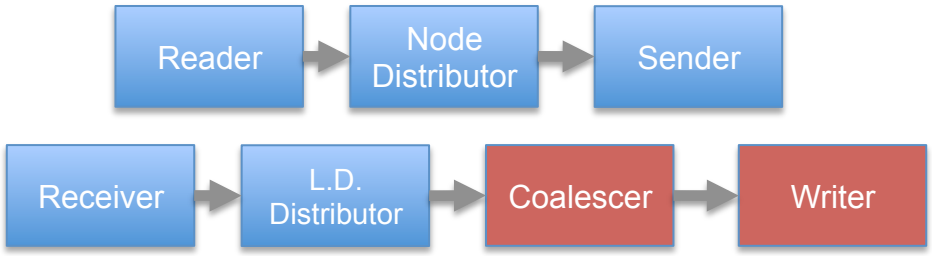
12.8 KB

Logical Disk Distributor



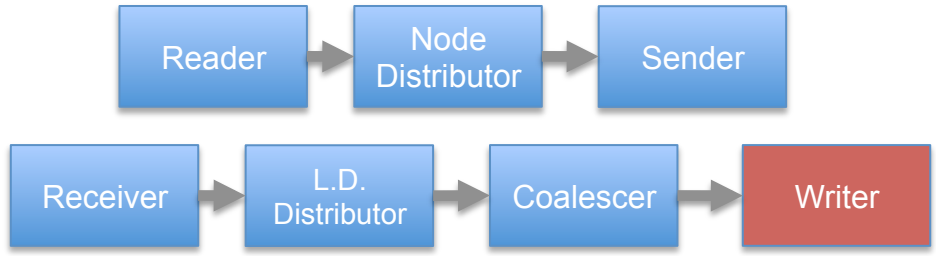
- Data non-uniform and bursty at short timescales
  - Big buffers + burstiness = head-of-line blocking
  - Need to use *all* your memory *all* the time
- **Solution**: Read incoming data into smallest buffer possible, and form chains

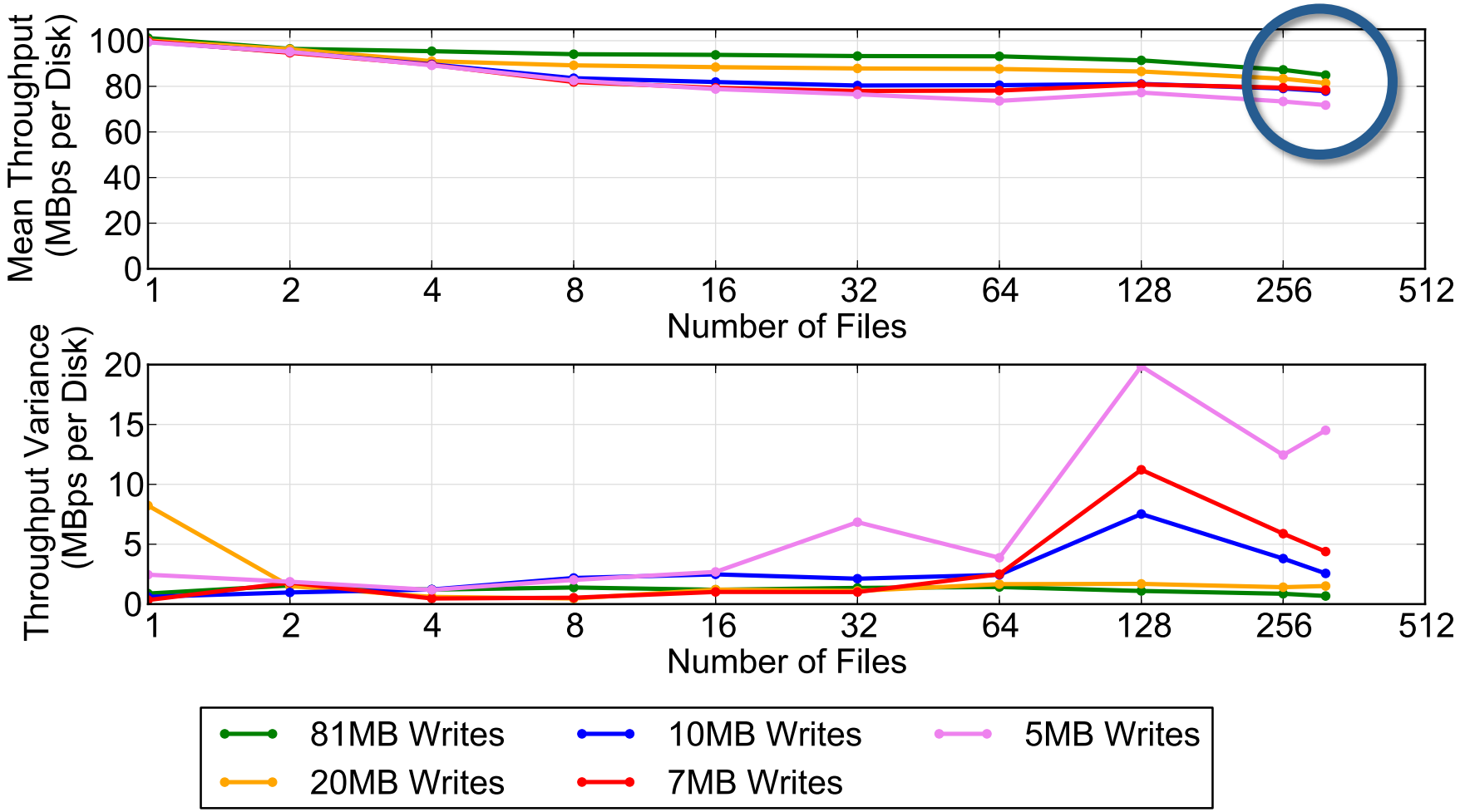
Coalescer & Writer



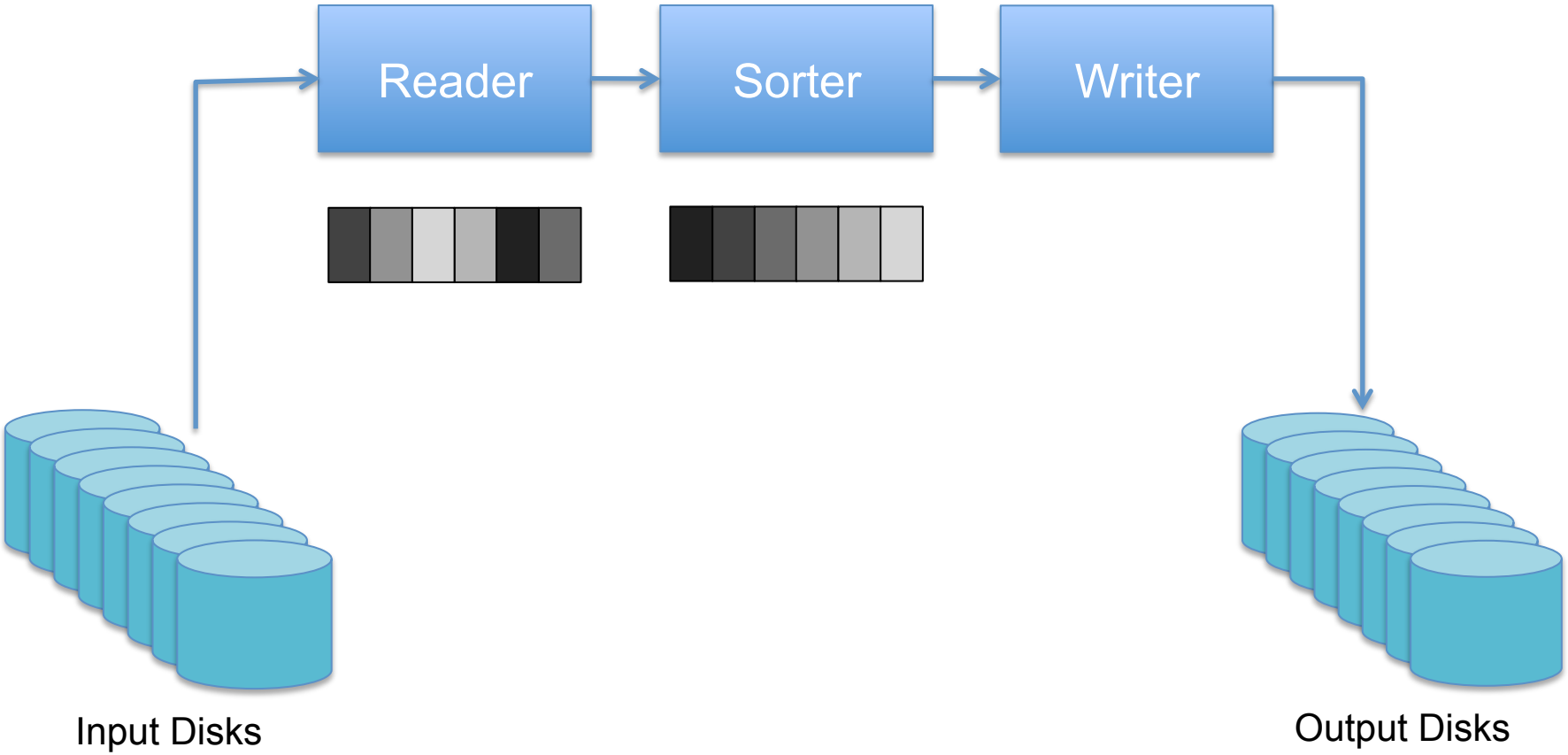
- Copies tuples from LDBuffer chains into a single, sequential block of memory
- Longer chains = larger write before seeking = faster writes
  - Also, more memory needed for LDBuffers
- Buffer size limits maximum chain length
  - How big should this buffer be?

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

Architecture Phase Two



Sort Benchmark Challenge

- Started in 1980s by Jim Gray, now run by a committee of volunteers
- Annual competition with many categories
  - GraySort: Sort 100 TB
- "Indy" variant
  - 10 byte key, 90 byte value
  - Uniform key distribution

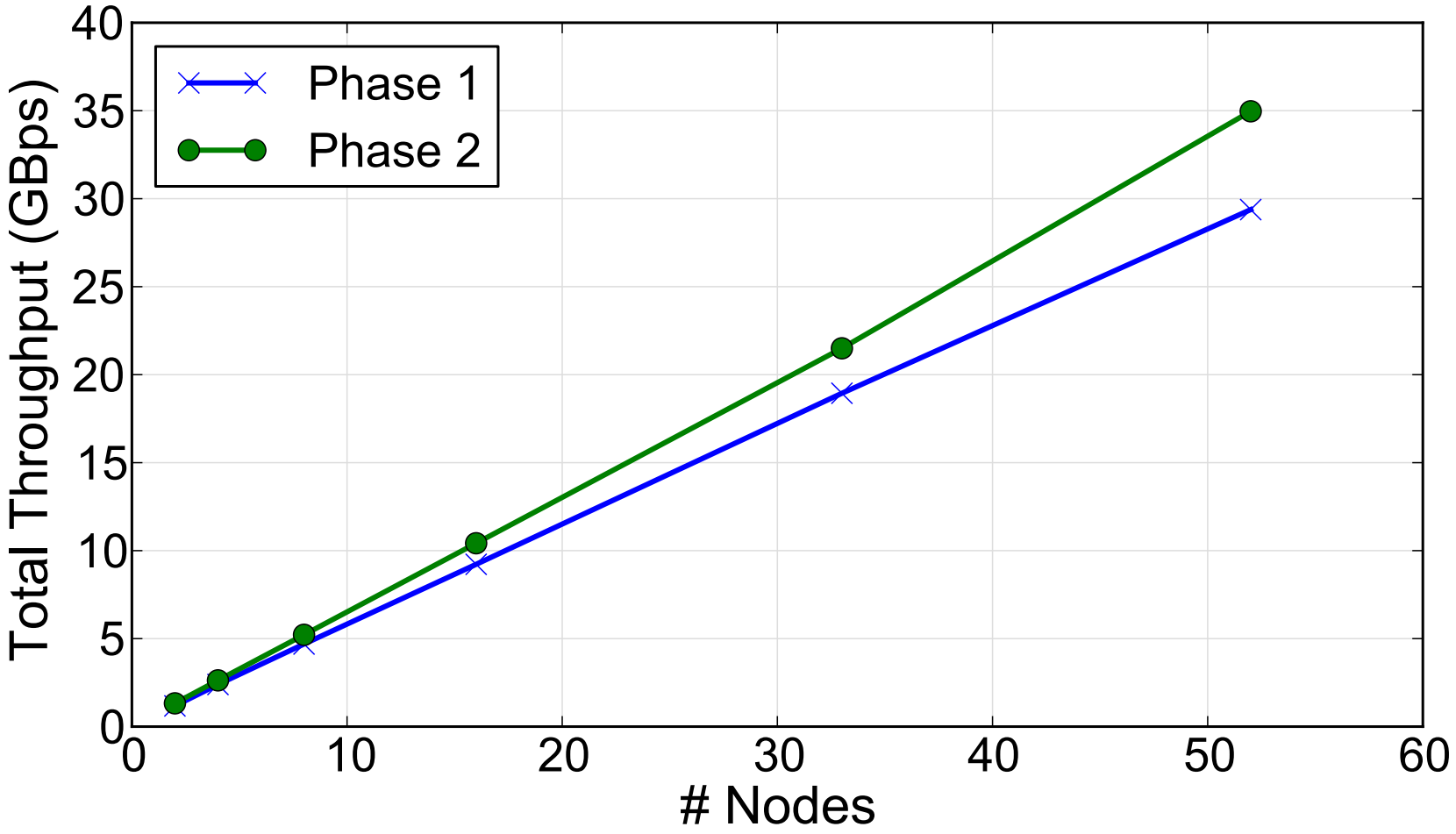
How balanced are we?

Worker Type	Workers	Total Throughput (MBps)	% Over Bottleneck Stage
Reader	8	683	13%
Node-Distributor	3	932	55%
LD-Distributor	1	683	13%
Coalescer	8	18,593	30,000%
Writer	8	601	0%
Reader	8	740	3.2%
Sorter	4	1089	52%
Writer	8	717	0%

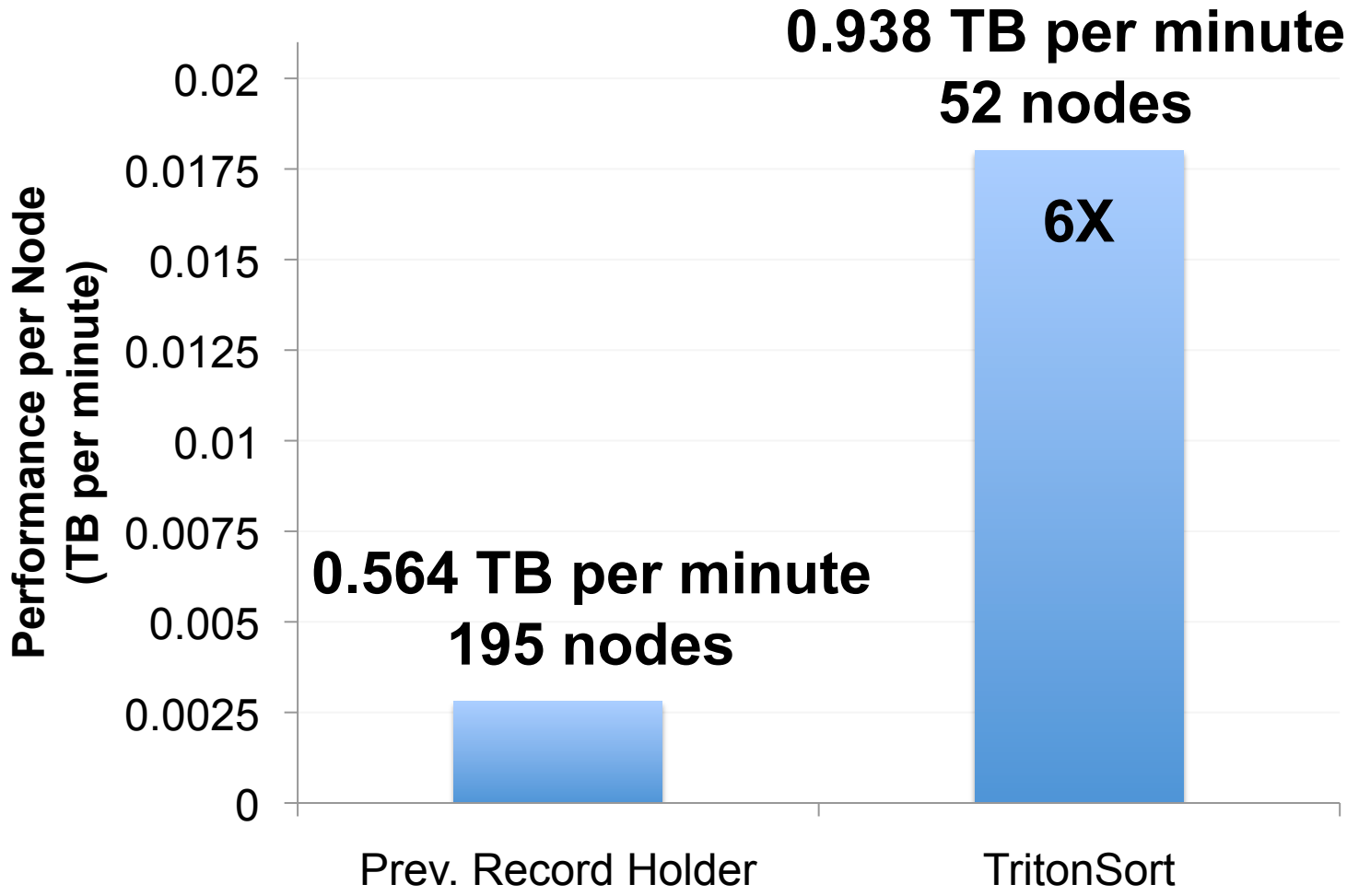
How balanced are we?

Phase	Resource Utilization			
	CPU	Memory	Network	Disk
Phase One	25%	100%	50%	82%
Phase Two	50%	100%	0%	100%

Scalability



Raw 100TB “Indy” Performance



Impact of Faster Disks

- 7.2K RPM → 15K RPM drives
- Smaller capacity means fewer LDs
- Examined effect of disk speed and # LDs
- Removing a bottleneck moves the bottleneck **somewhere else**

Intermediate Disk Speed (RPM)	Logical Disks Per Physical Disk	Phase One Throughput (MBps)	Phase One Bottleneck Stage	Average Write Size (MB)
7200	315	69.81	Writer	12.6
7200	158	77.89	Writer	14.0
15000	158	79.73	LD Distributor	5.02

Impact of Increased RAM

- Hypothesis that memory influences chain length, and thus write speed
- Doubling memory indeed increases chain length, but the effect on performance was minimal
- Increasing a non-bottleneck resource made it faster, **but not by much**

RAM Per Node (GB)	Phase One Throughput (MBps)	Average Write Size (MB)
24	73.53	12.43
48	76.43	19.21

Future Work

- Generalization
  - We have a fast MapReduce implementation
  - Considering other applications and programming paradigms
- Automatic Tuning
  - Determine appropriate buffer size & count, # workers per stage for reasonable performance
    - Different hardware
    - Different workloads

TritonSort – Questions?

- Proof-of-concept balanced sorting system
- 6x improvement in per-node efficiency vs. previous record holder
- Current top speed: 938 GB per minute
- Future Work: Generalization, Automation

http://tritonsort.eng.ucsd.edu/