
UNORTHODOX PATHS
TO HIGH PERFORMANCE
ICAS

@ALEXRAS

TRIFACTA

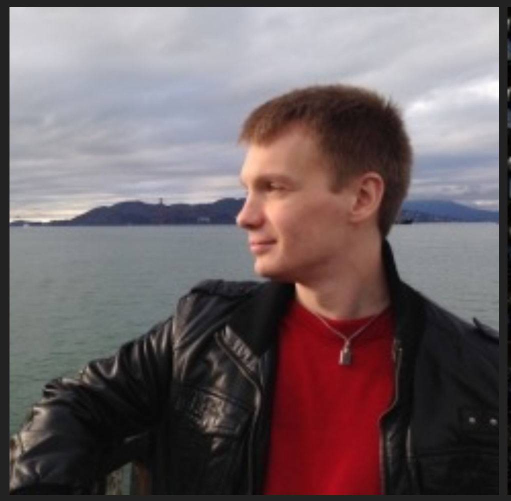
TRITONSORT (NSDI 2011)

Sort Really Fast

THEMIS (SOCC 2012)

MapReduce Really Fast








2010
2011
2014

THIS TALK:

HOW WE DID IT

~~LANGUAGE~~

~~FRAMEWORK~~

~~TOOLS~~

~~API~~

1. SOFTWARE-HARDWARE CO-DESIGN
2. BUILDING FOR EXPERIMENTATION
3. CAREFULLY MANAGING MEMORY


MOTIVATION

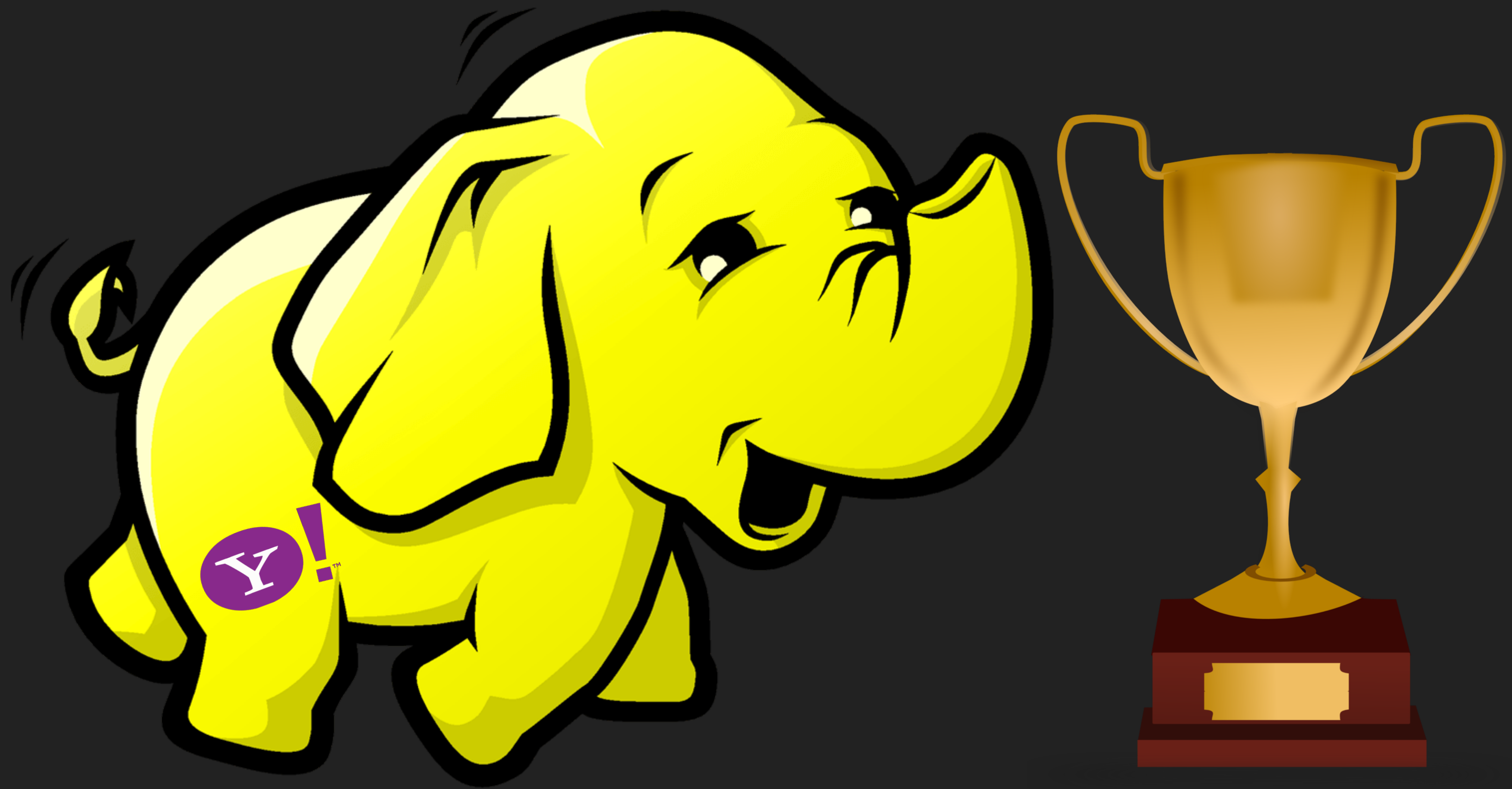
[SORT] IS AN EXCELLENT TEST OF THE INPUT-OUTPUT ARCHITECTURE OF A COMPUTER AND ITS OPERATING SYSTEM.

"A measure of transaction processing power"
Datamation 1985

THE SORT BENCHMARK

- SORTING K/V PAIRS (RECORDS)
- MANY CATEGORIES, VARIANTS
- TODAY: GRAYSORT (100TB)

2009: YAHOO! SORTS 100TB IN 173 MINUTES ON 3452 HADOOP NODES



578 GB PER MINUTE

9.6 GB PER SECOND

3452 NODES

2.79 MBPS PER NODE

THE GAP IN THEORETICAL PER NODE PERFORMANCE AND WHAT IS ACTUALLY ACHIEVED HAS BECOME GLARINGLY LARGE.

Anderson and Tucek
“Efficiency Matters!” SIGOPS OSR 2010

HOW CAN WE DO BETTER?

HARDWARE & SOFTWARE CO-DESIGNED FOR WORKLOAD


UNDERSTANDING THE PROBLEM

HOW TO MAXIMIZE PER-NODE SPEED?

~~CPU~~

~~RAM~~

NETWORK 10Gbps

DISK ~15 disks/NIC

- 8 CORES
- 24 GB RAM
- 16 DISKS
- 10 GBPS NIC

WHAT ARE THE EXPENSIVE OPERATIONS?


ROTATION
SEEKING

🐌 WRITING

☠️ SEEKING

KEEP OFF THE DISK

TWO READS +
TWO WRITES
PER RECORD

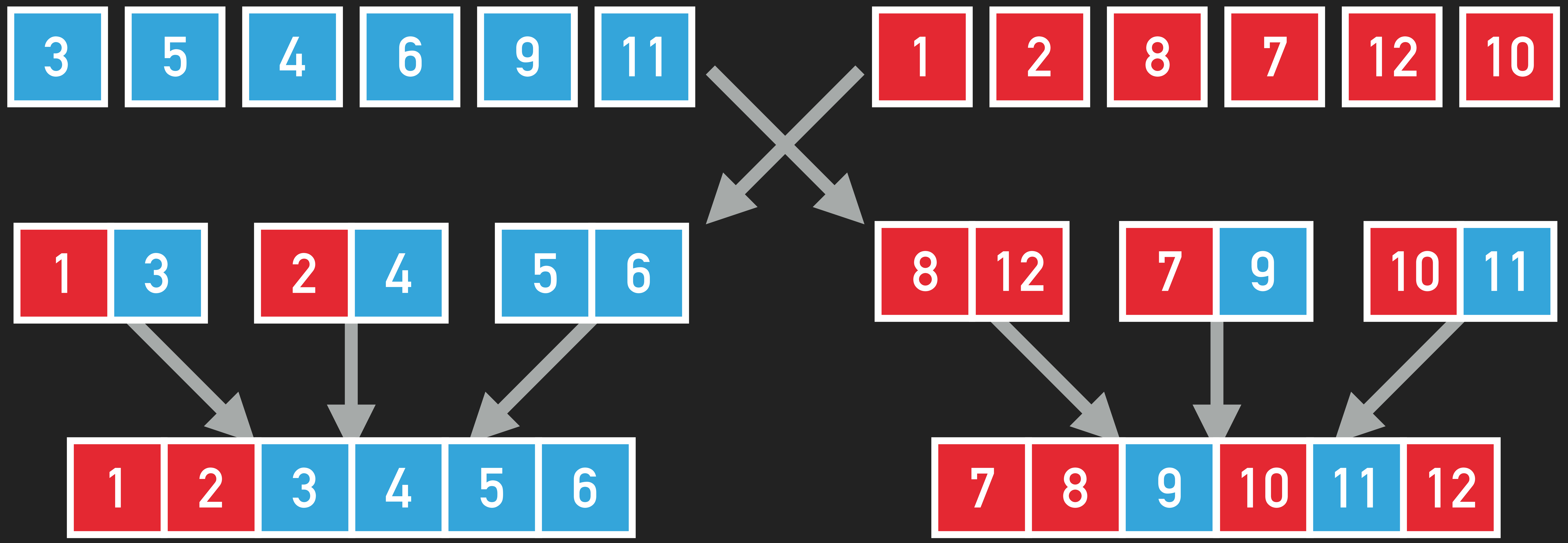
SEEK INFREQUENTLY

BIG READS

BIG WRITES

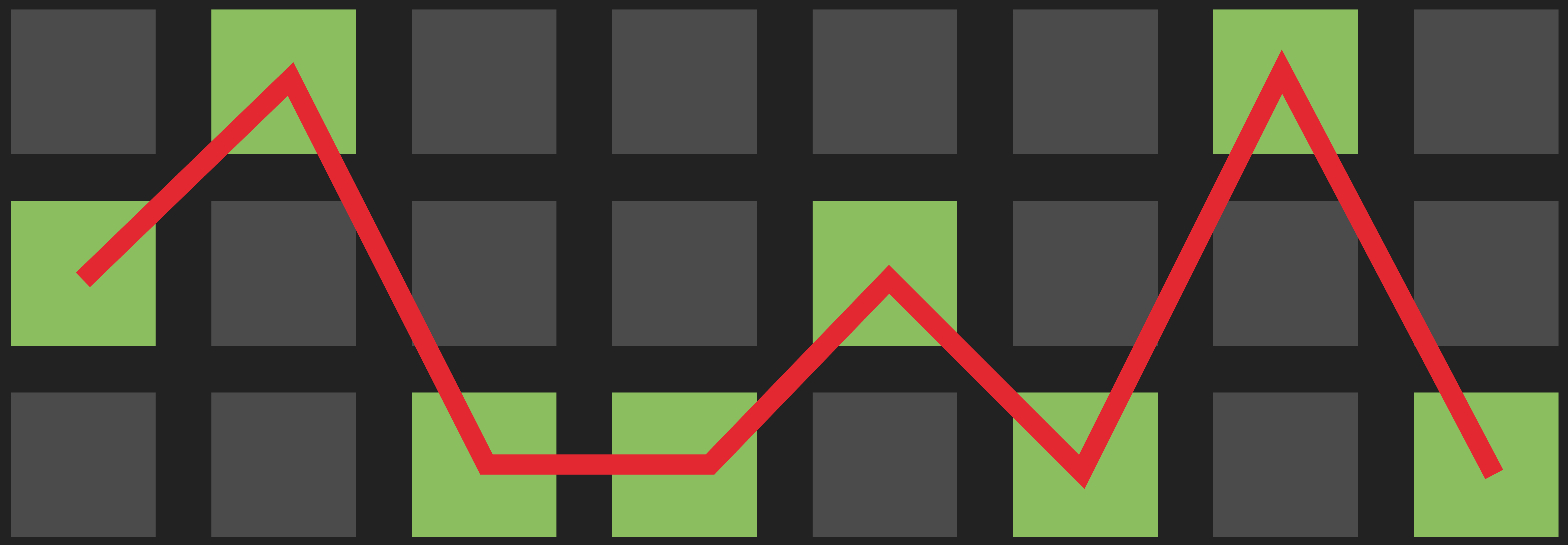
WHICH DISTRIBUTED SORTING ALGORITHM?

MERGESORT



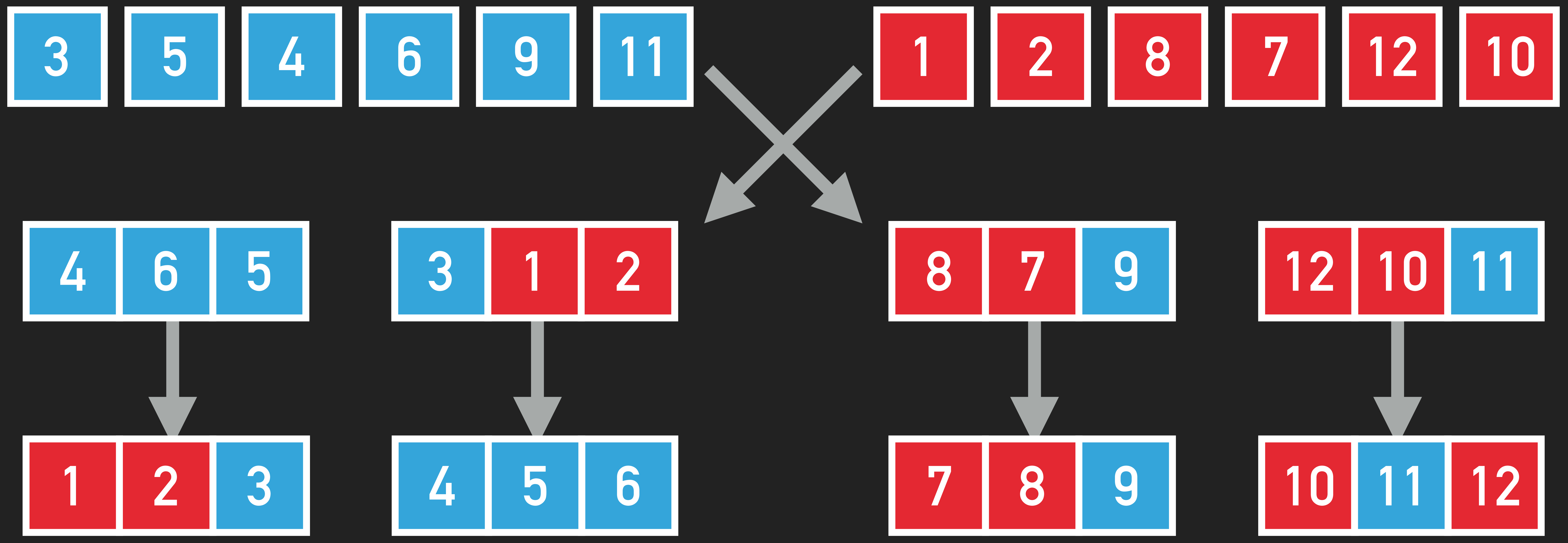
THE TROUBLE WITH MERGESORT

AT SCALE, MANY SORTED CHUNKS



FETCHING RANDOMLY CAUSES SEEKS

DISTRIBUTION SORT



DISTRIBUTION SORT

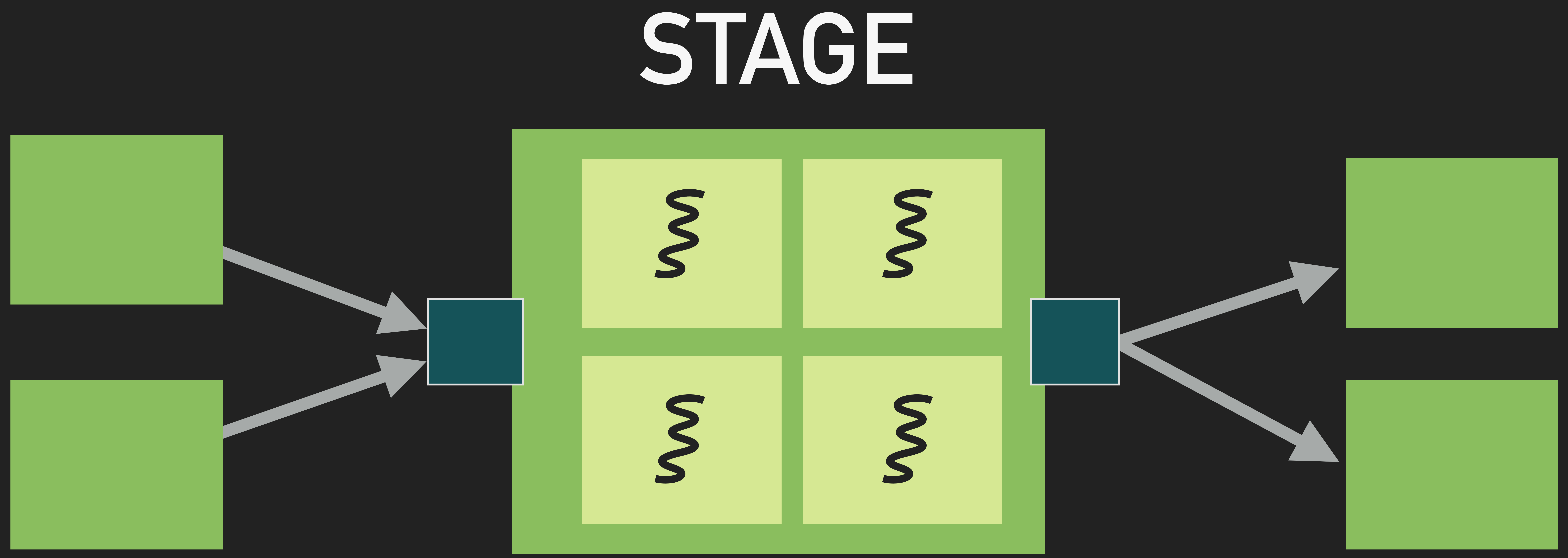
- BIG WRITES IN FIRST PASS
- SEQUENTIAL I/O IN SECOND PASS

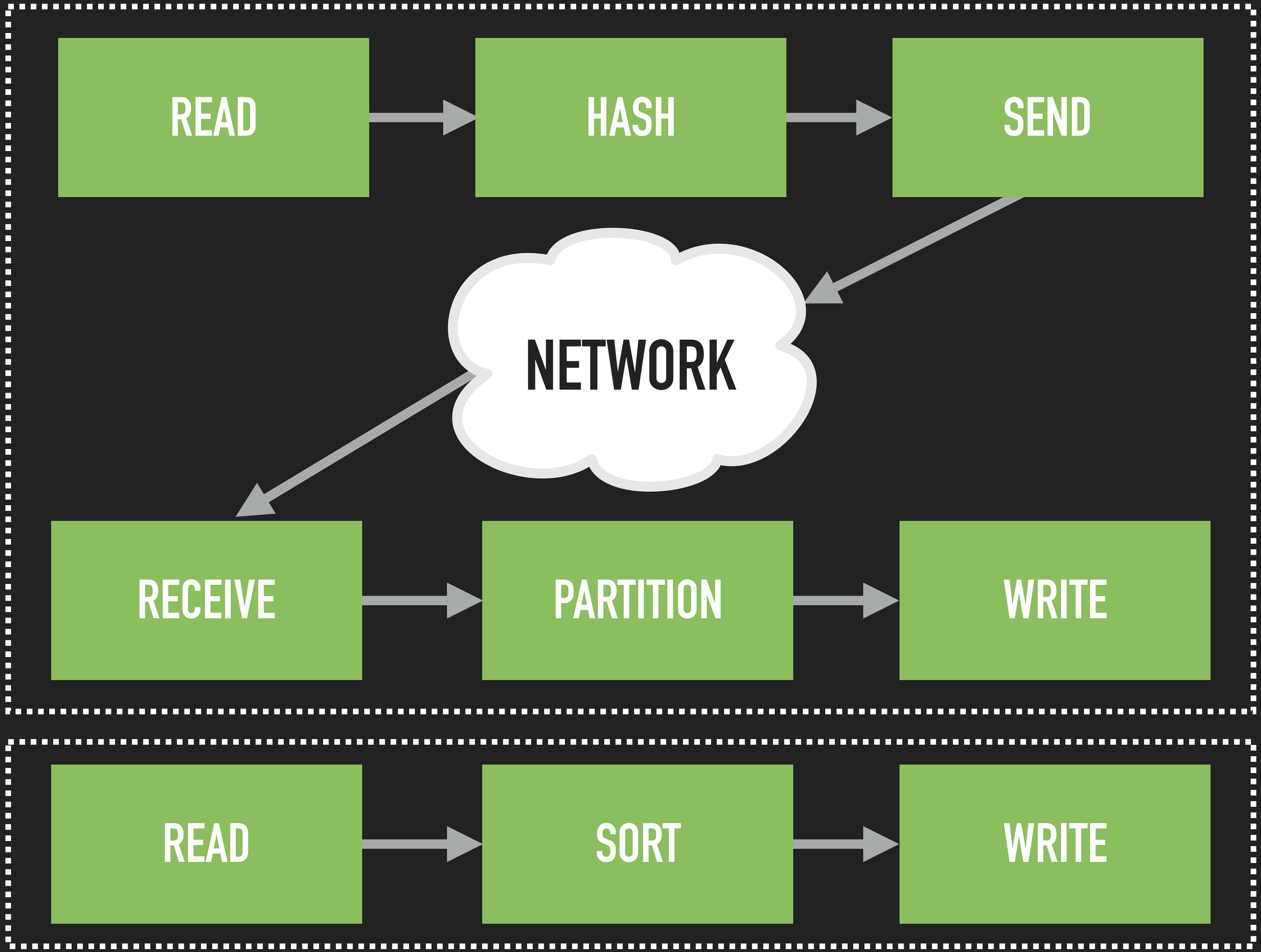

NOW, TO BUILDING!

IT WON'T JUST MAGICALLY BE FAST


EXPERIMENTATION!

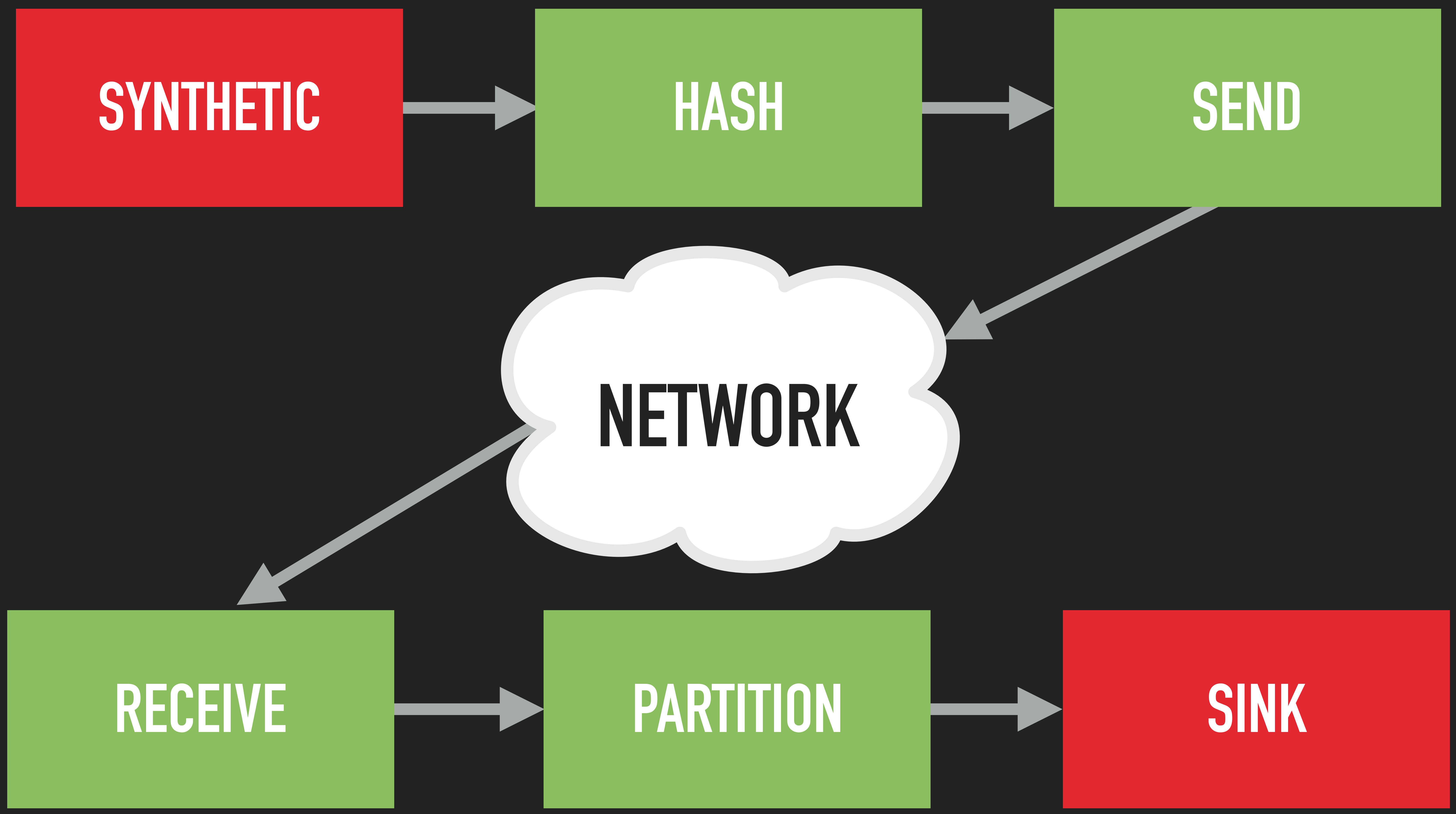
FLEXIBLE
MODULAR
GRAPHS


STAGE

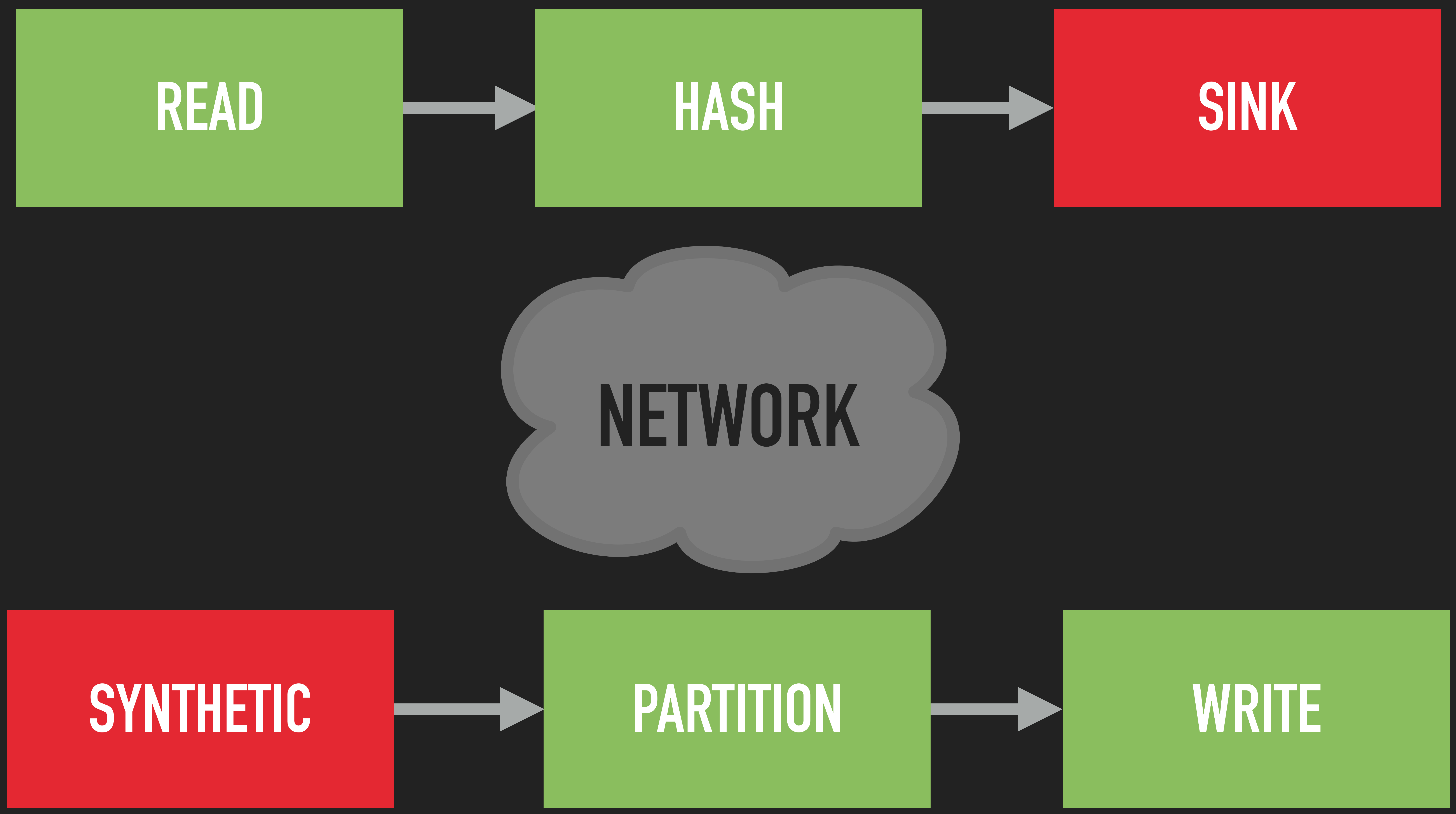


READ
HASH
SEND
NETWORK
RECEIVE
PARTITION
WRITE
READ
SORT
WRITE

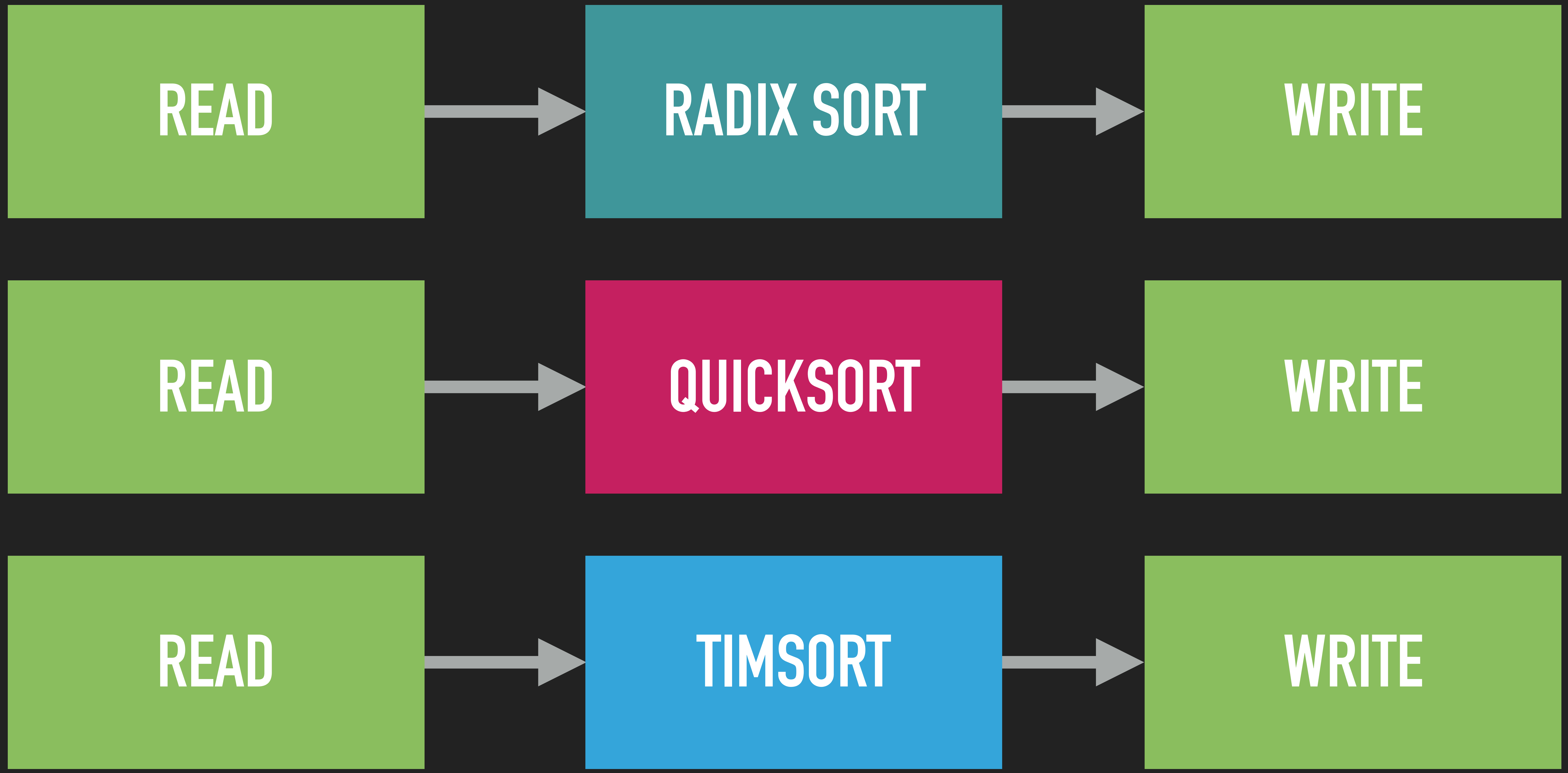
REALLY FAST DISKS?



REALLY FAST NETWORK?



DIFFERENT SORTS?



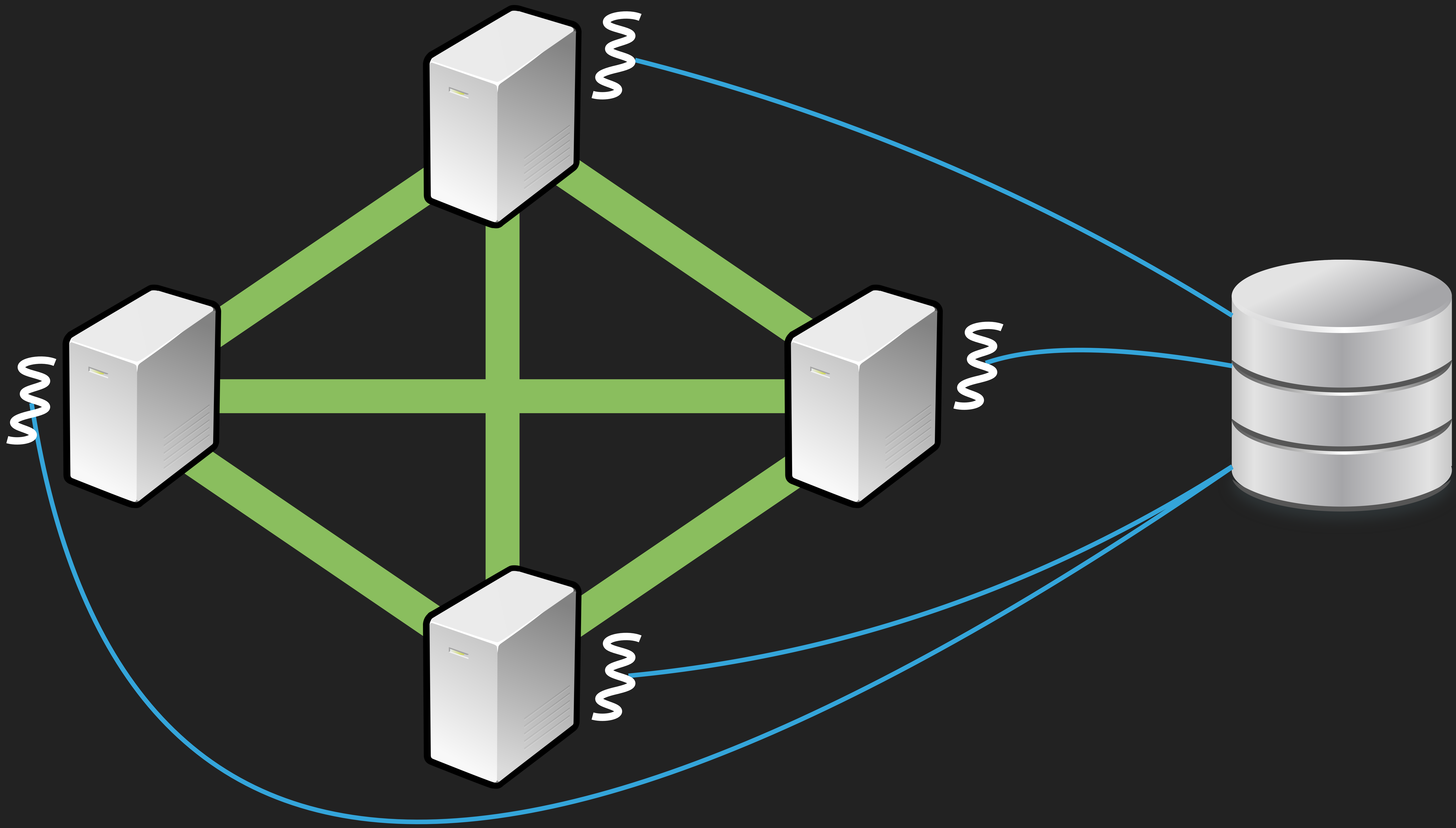

MEASUREMENT
40
30
20

HOW LARGE ARE WRITES?

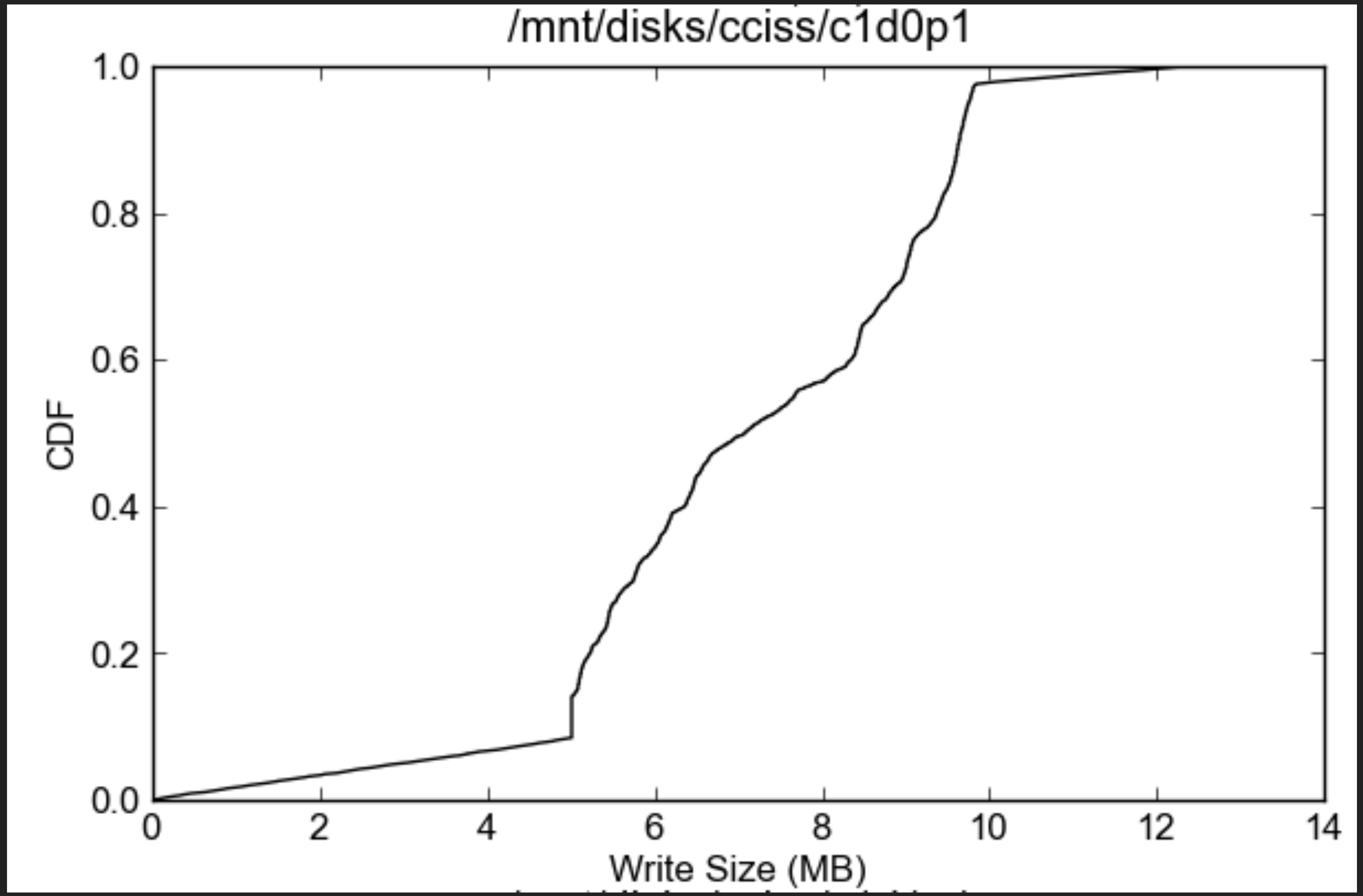
WHERE IS THE BOTTLENECK?

ARE STAGES BLOCKED? IDLE?

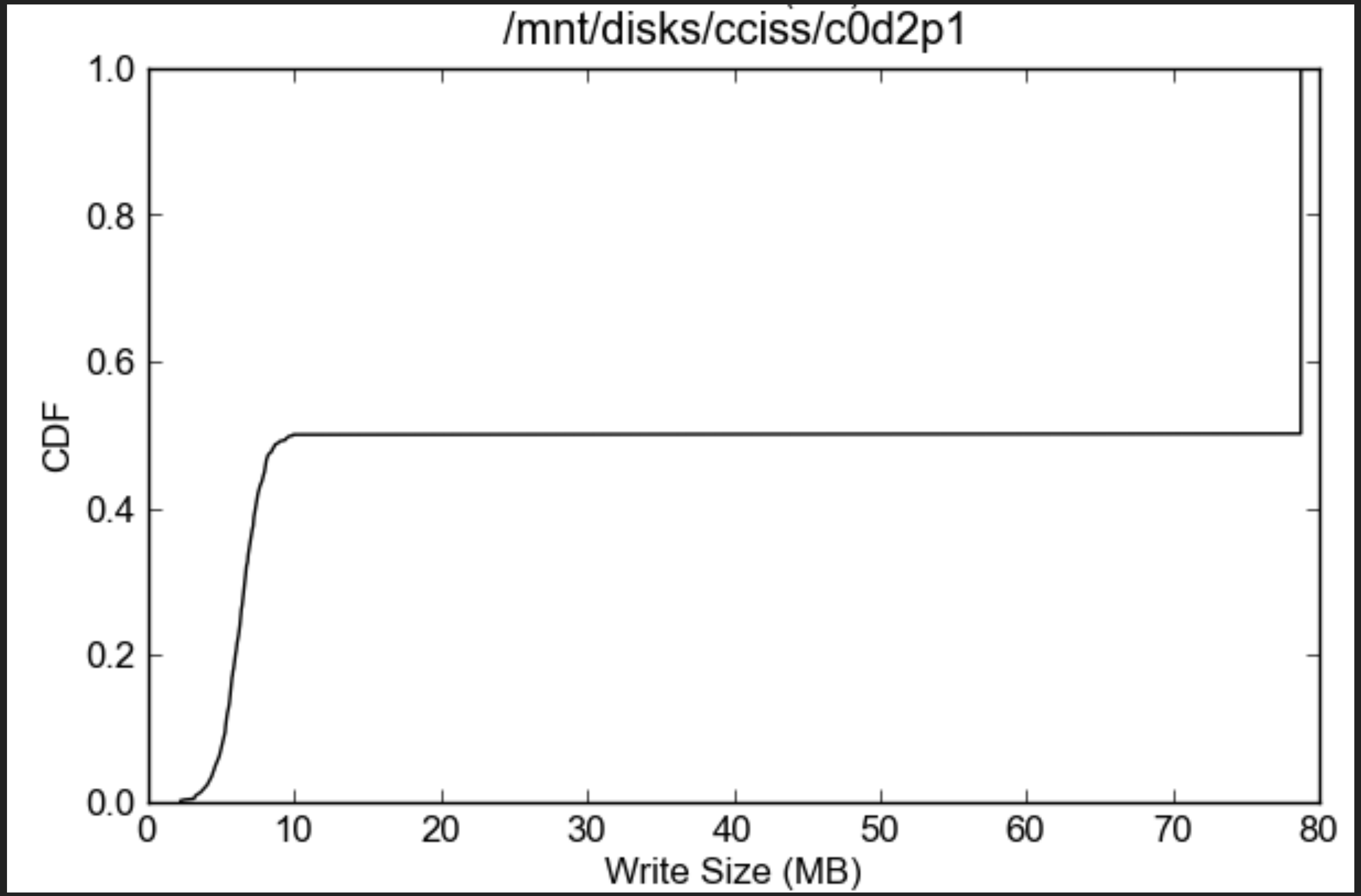

TONS AND TONS OF LOGS
ONE
FULL
CORD



AGGREGATES

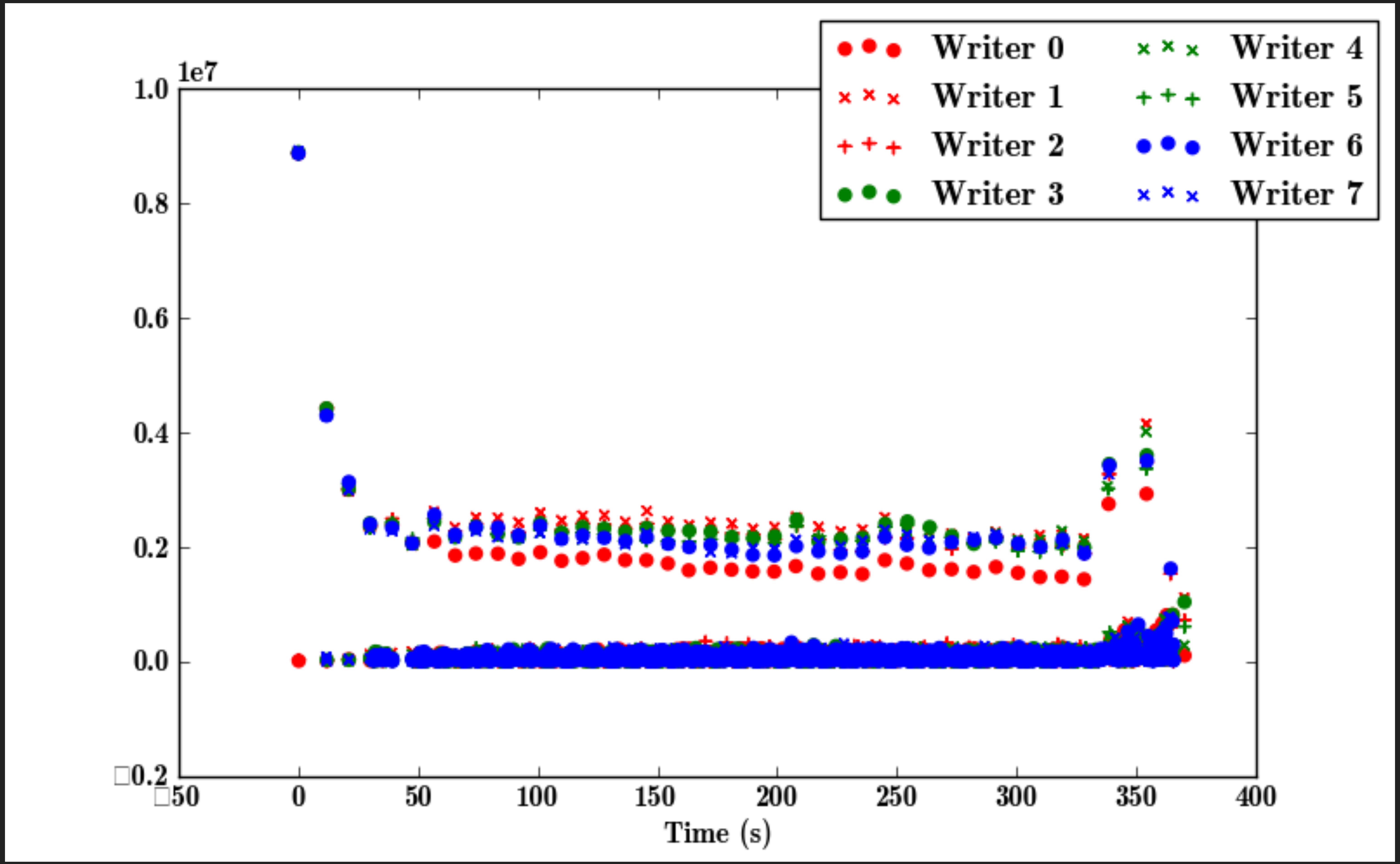


/mnt/disks/cciss/c1d0p1
CDF
Write Size (MB)
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14



/mnt/disks/cciss/c0d2p1
CDF
Write Size (MB)
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
70
80

TIME-SERIES



Writer 0
Writer 1
Writer 2
Writer 3
Writer 4
Writer 5
Writer 6
Writer 7
1e7
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−50
0
50
100
150
200
250
300
350
400
Time (s)

RUNTIME INFO

writer

Key	Worker ID								Total
	0	1	2	3	4	5	6	7	
bytes_consumed	7.88 GiB	7.89 GiB	7.89 GiB	7.90 GiB	7.87 GiB	7.88 GiB	7.89 GiB	7.89 GiB	63.09 GiB
bytes_produced	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B
bytes_written	7.88 GiB	7.89 GiB	7.89 GiB	7.90 GiB	7.87 GiB	7.88 GiB	7.89 GiB	7.89 GiB	63.09 GiB
bytes_written_after_direct_io_disabled	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B
queue_size	0	0	0	0	0	0	0	0	0
work_units_consumed	1873	1873	1874	1877	1870	1873	1873	1874	14987
work_units_produced	0	0	0	0	0	0	0	0	0

reader:local

Key	Worker ID								Total
	0	1	2	3	4	5	6	7	
bytes_consumed	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B
bytes_produced	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B
bytes_read	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B	0 B
work_units_consumed	0	0	0	0	0	0	0	0	0
work_units_produced	0	0	0	0	0	0	0	0	0

ORGANIZING LOGS

/2016/06/14/frob_widgets_1.tar.bz2

ORGANIZING LOGS

/2016/06/14/frob_widgets_1.tar.bz2

/cluster_nodes.txt

/node.conf

ORGANIZING LOGS

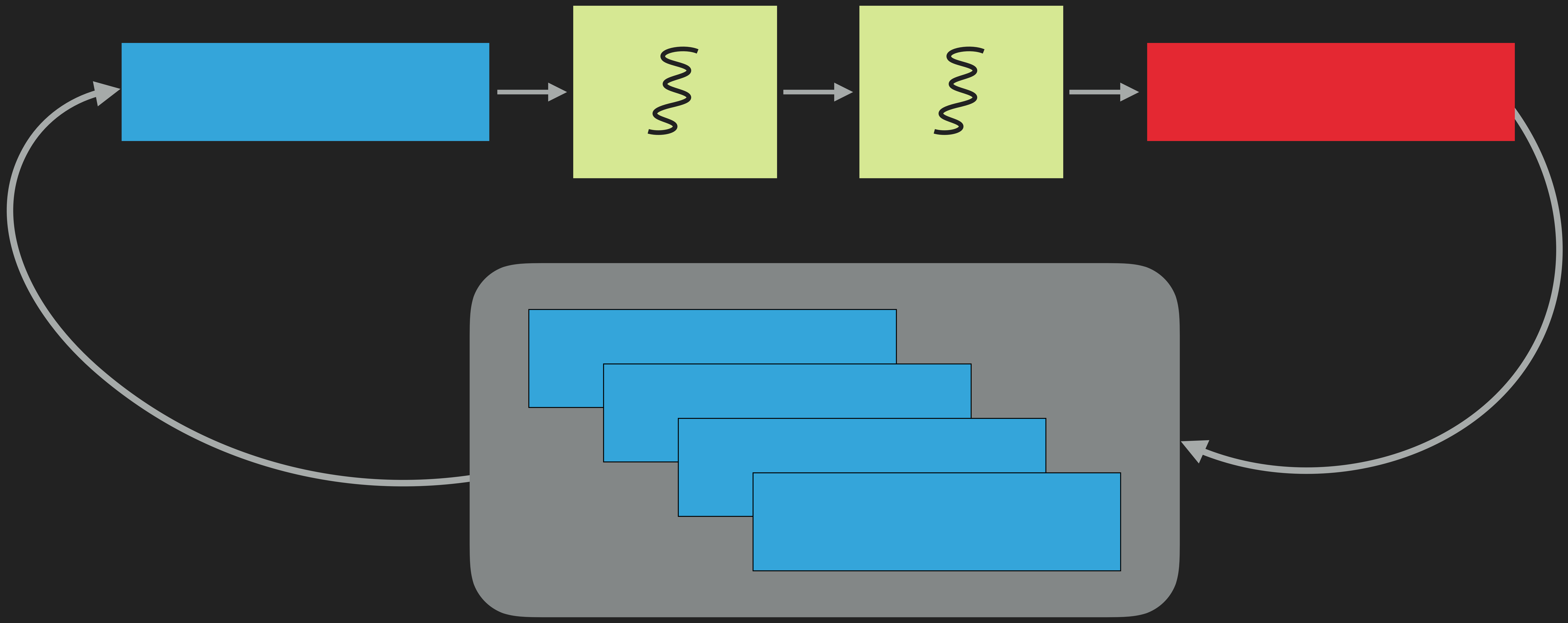
/2016/06/14/frob_widgets_1.tar.bz2

/cluster_nodes.txt

/node.conf

/notes.md

SMARTLY CONTROLLING MEMORY



BUFFER POOLS ARE FAST AND SIMPLE AND INFLEXIBLE

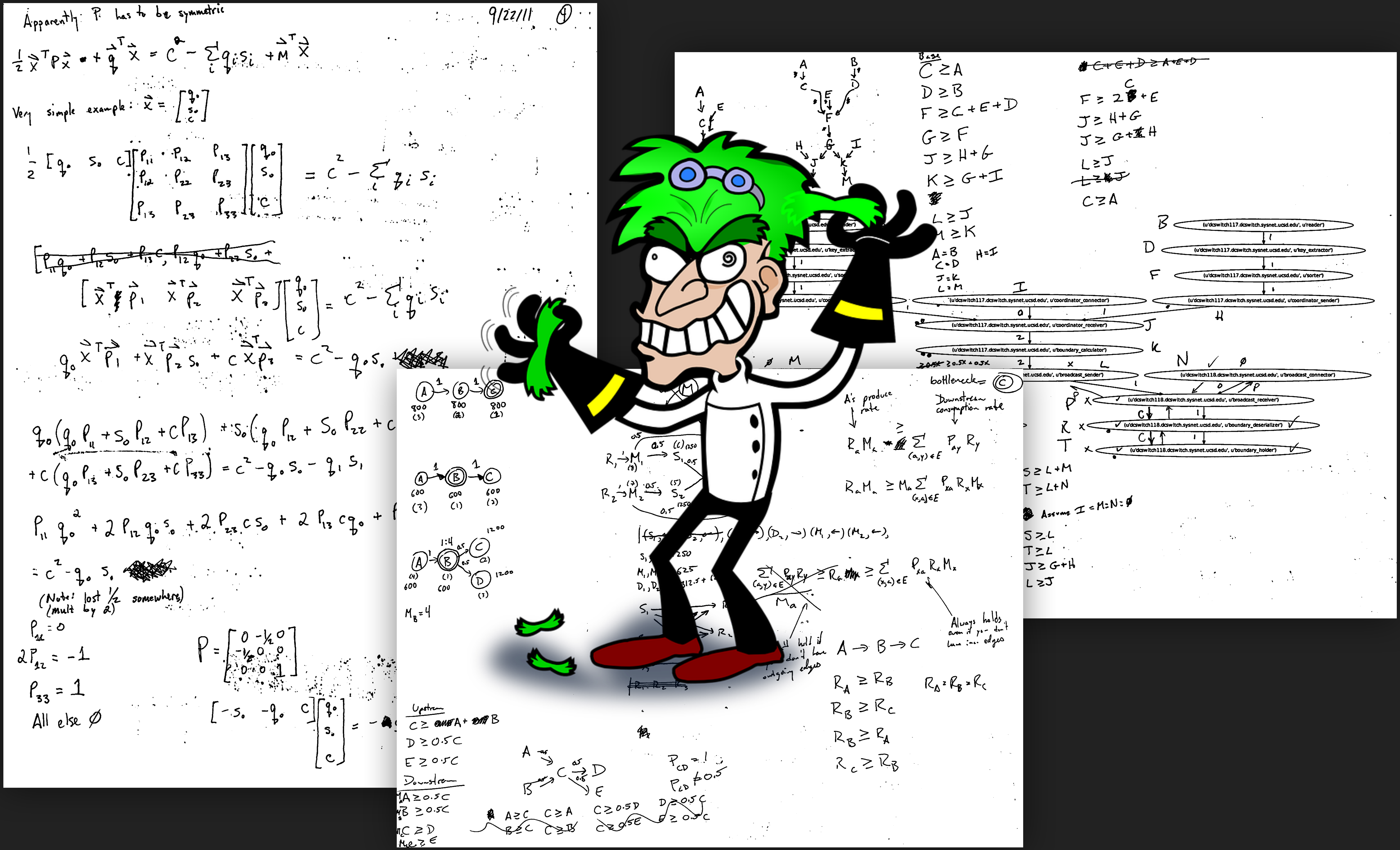
MALLOC IS

SIMPLE AND FLEXIBLE

AND DANGEROUS

By default, Linux follows an **optimistic** memory allocation strategy. This means that when malloc() returns non-NULL there is **no guarantee that the memory really is available**.

In case it turns out that the system is out of memory, one or more processes **will be killed** by the OOM killer.


Apparently P has to be symmetric
9/22/11
Very simple example:
(Note: lost 1/2 somewhere)
All else ∅
bottleneck
A's produce rate
Downstream consumption rate
Upstream
Downstream
Always holds even if you don't have inc. edges


ARE WE SOLVING
THE RIGHT PROBLEM?

WHAT IF MALLOC WAITED?

WAITING PROVIDES BACKPRESSURE

CALLERS CAN BE SCHEDULED

INTERFACE STAYS SIMPLE

DECISIONS CAN BE GLOBAL

POOLS
QUOTAS
CONSTRAINTS


WRAPPING UP

938 GB PER MINUTE

15.6 GB PER SECOND

52 NODES

300 MBPS PER NODE

6757 GB PER MINUTE

112.6 GB PER SECOND

178 NODES

632 MBPS PER NODE

LESSONS LEARNED:

BOTTLENECKS SHAPE YOUR ARCHITECTURE

STRUCTURE SOFTWARE FOR EXPERIMENTATION AND MEASUREMENT

SAVE YOUR LOGS
SAVE YOUR CONFIG
SAVE YOUR NOTES

SOMETIMES YOU NEED MORE CONTROL THAN THE OS WILL GIVE YOU

GOING FAST IS HARD

THANKS