SOMETHING SOMETHING COORDINATOR NODES

Cheating Our Way to Better Performance

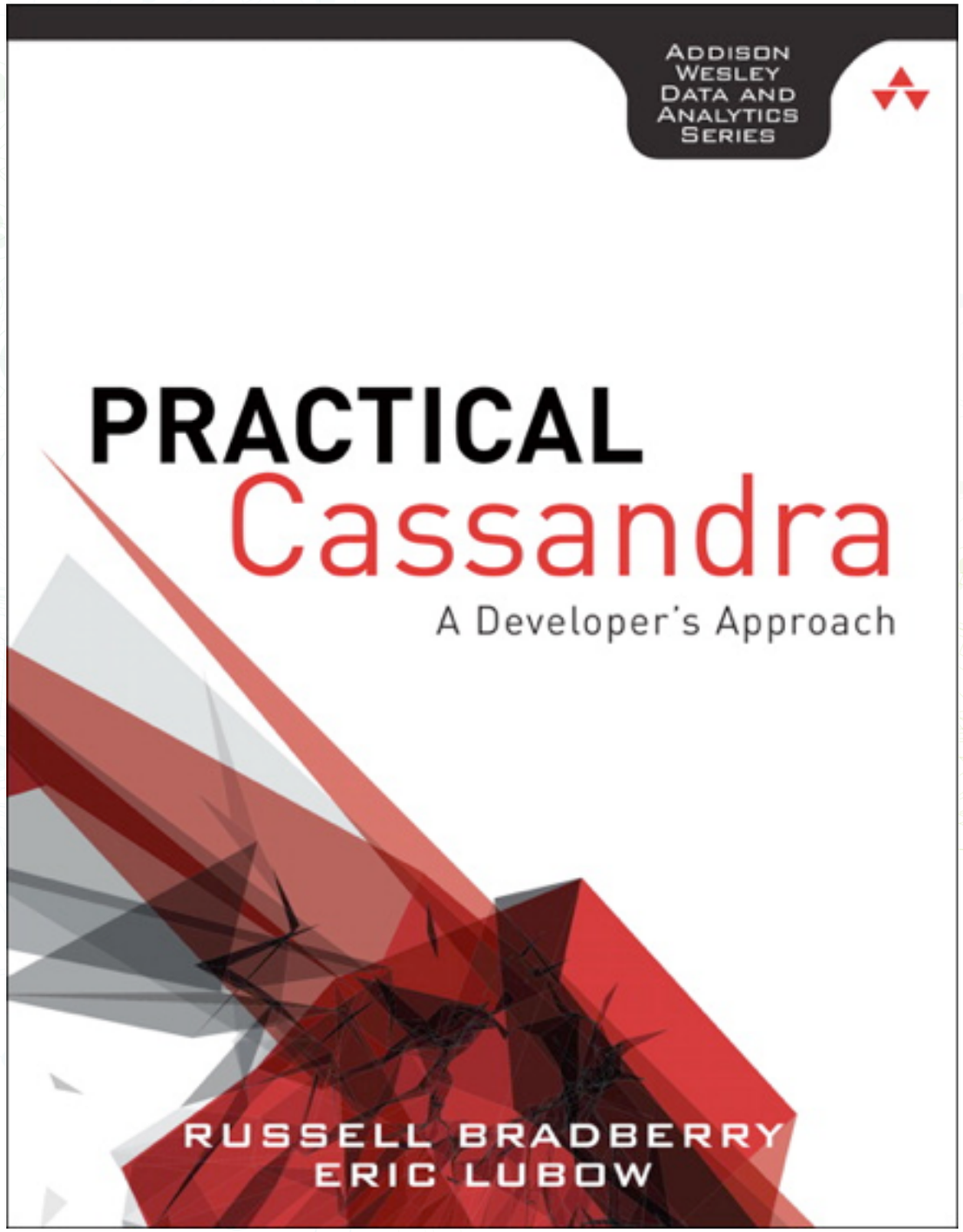
LEARN DATA MODELING BY EXAMPLE

JON HADDAD

GO TO ROOM 210A NOW!

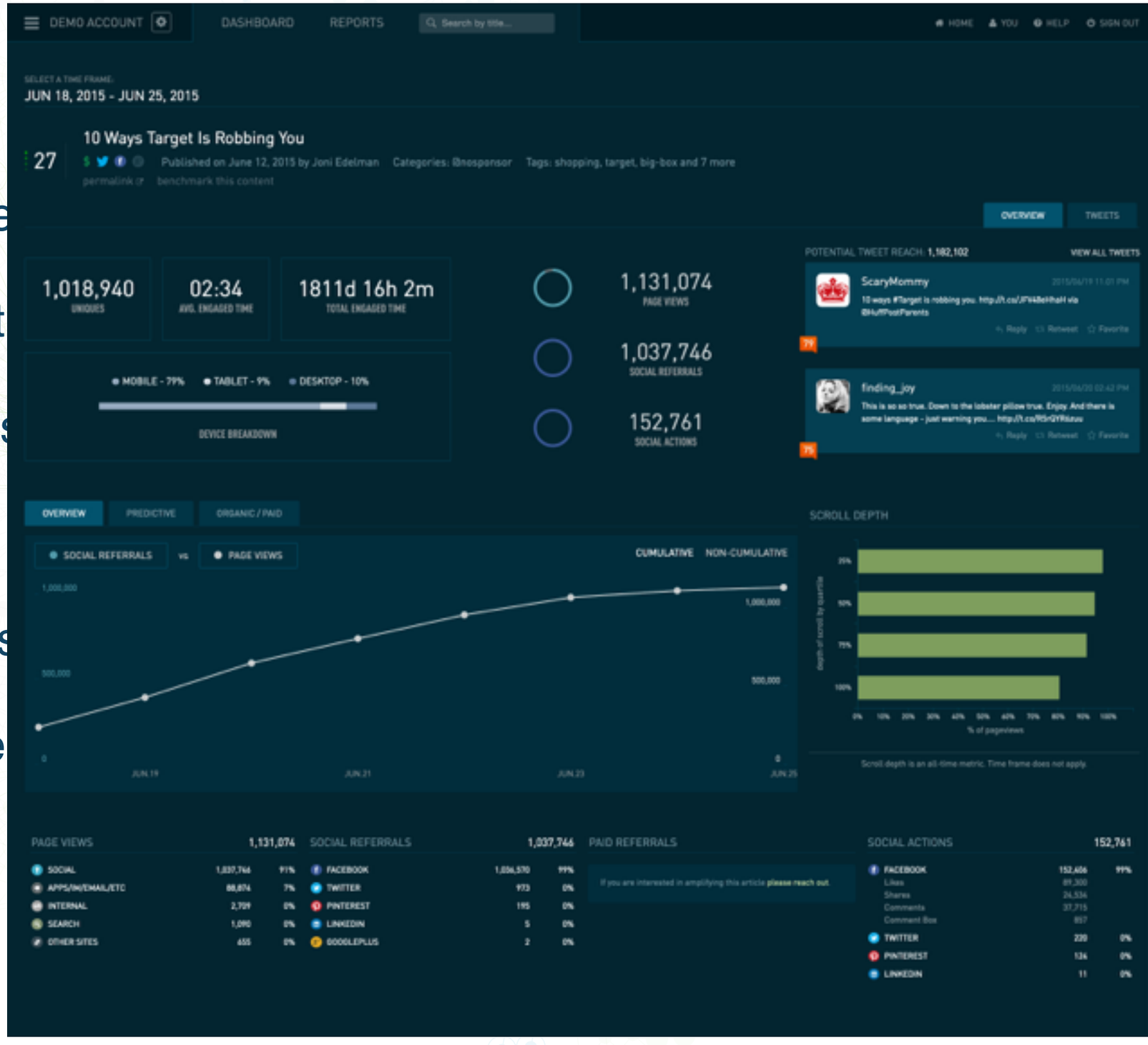
PERSONAL VANITY

- CTO of SimpleReach
- Co-Author of Practical Cassandra
- Skydiver, Mixed Martial Artist, Motorcyclist, Dog Dad (IG: @charliedognyc), NY Giants fan



SIMPLEREACH

- Help marketers organize
- Identify the best content
- Use engagement metrics
- Workflow solution
- Stream processing inges
- Many metrics, time slice
- Multiple data stores



CONCEPTS YOU SHOULD UNDERSTAND

1. Thick clients and thin clients
2. CPU utilization and load average
3. Database tuning may not have anything to do with the database

WHAT IS A FAT CLIENT?

A fat client (also called heavy, rich, or thick client) is a computer (client) in client–server architecture or networks that typically provides rich functionality independent of the central server.

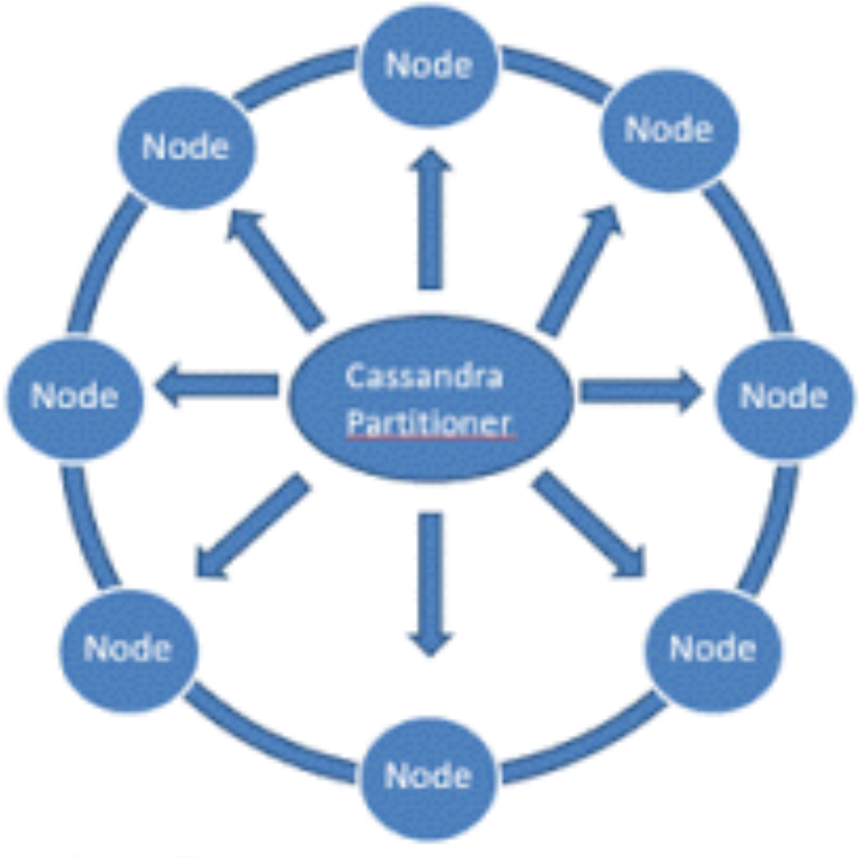
— Wikipedia

THIN CLIENTS AND THICK CLIENTS

- Thin clients typically can’t operate without the “server”
- Thick clients try to do more locally compared to thin clients which try to do more remotely
- Thick clients require more resources, but fewer servers.
- Thin clients require fewer resources and more servers.

ONCE MORE, BUT WITH CASSANDRA

- Fat clients have the Cassandra binary, but no data
- Data nodes are denser and more focused on storage
- For context, we can call them proxy nodes
- Proxy nodes are more compute hea
- Fat clients only handles coordinatio



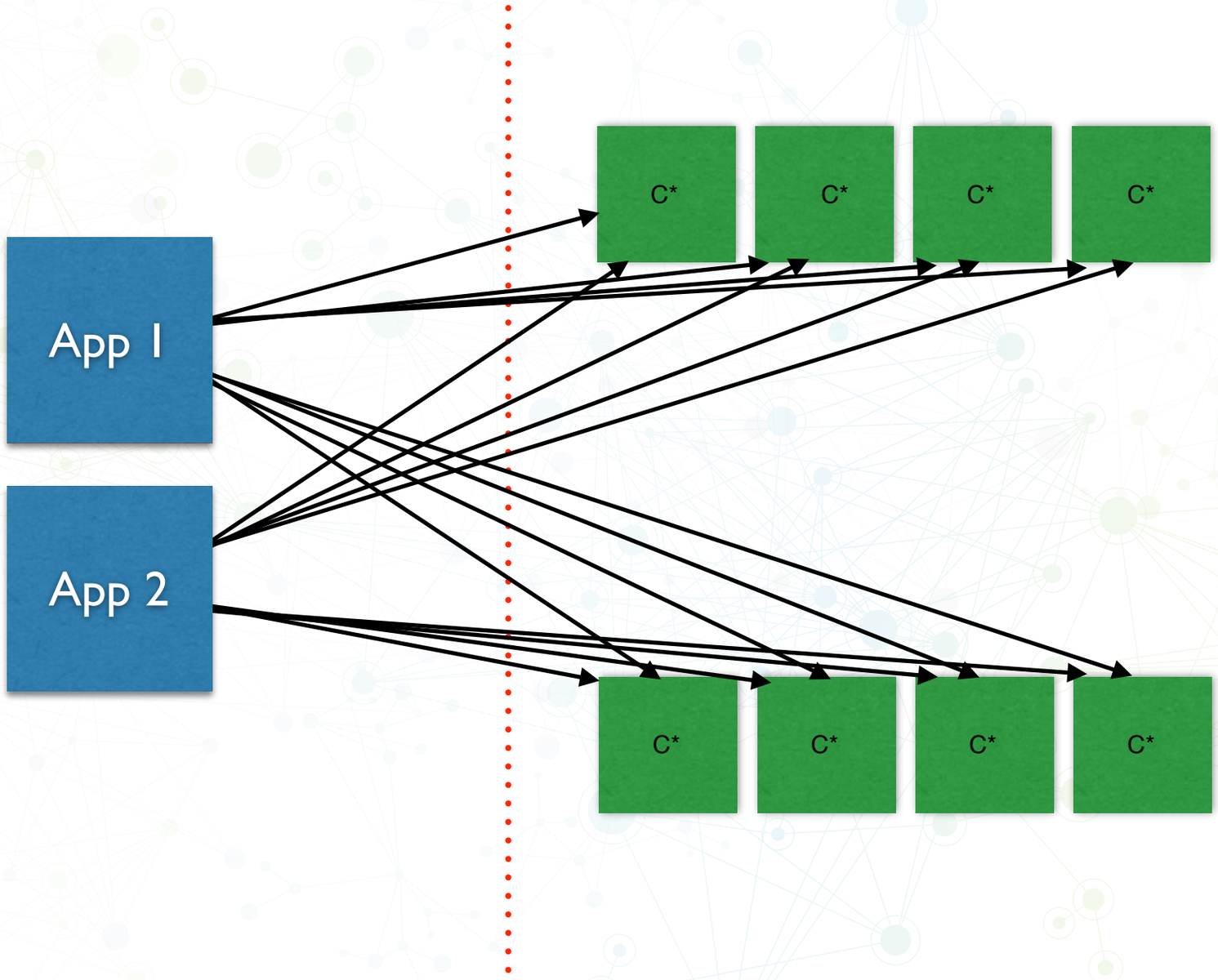
WHAT'S REALLY GOING ON HERE?

- Fat clients are effectively just changing settings
  - < Cassandra 2.2 -Djoin_ring=false (hack)
- No data on the nodes, just coordination responsibility
- Intentionally sidestepping Cassandra homogenous nature in favor of performance
- Can be lots of room for adding proxy nodes without incurring additional performance loss from increasing the ring size
- Reduces per node work on the data nodes

NORMAL CASSANDRA SETUP

APPLICATION TIER

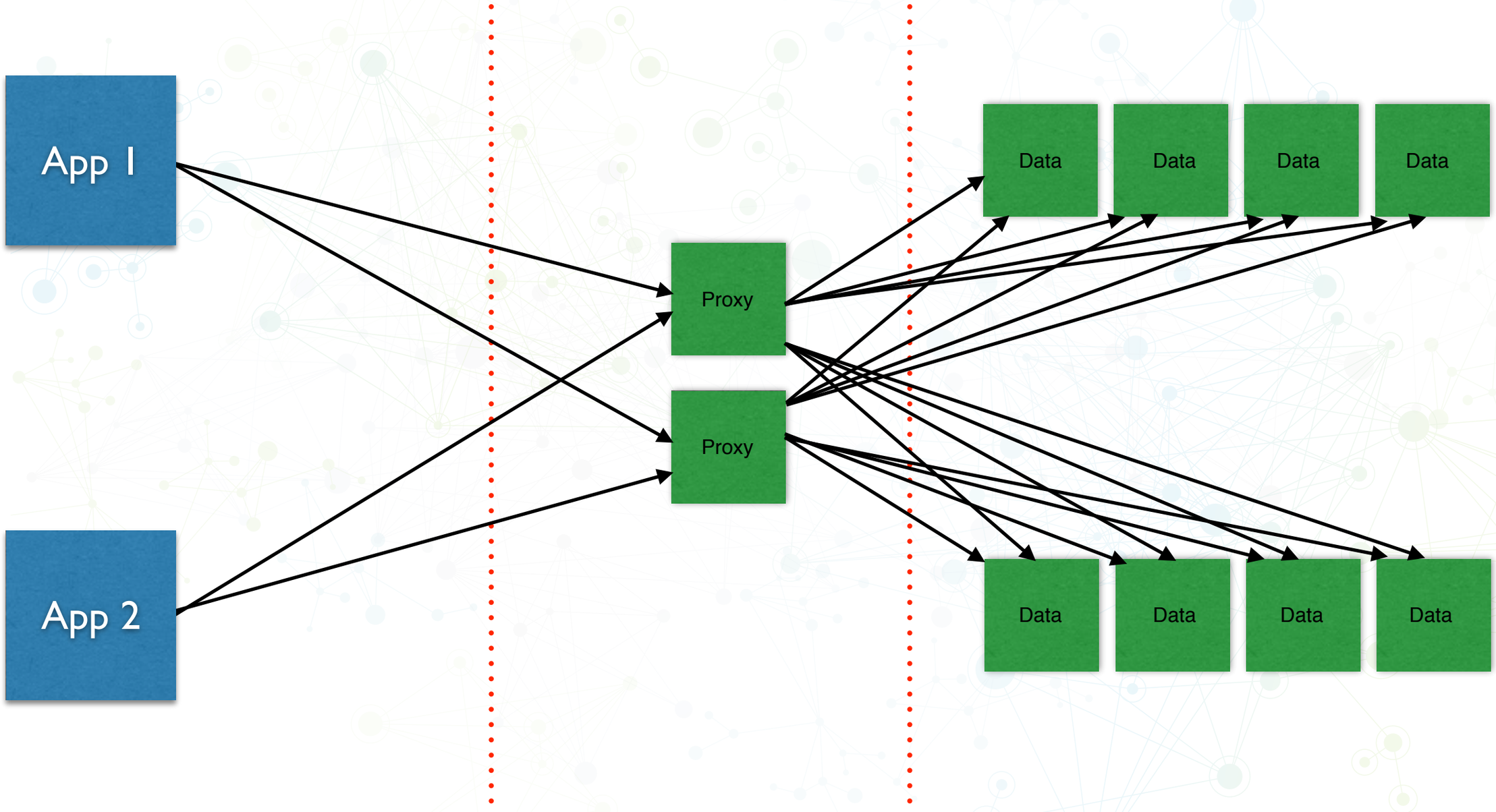
CASSANDRA CLUSTER




BRACE YOURSELVES
THE NEW HOTNESS IS COMING

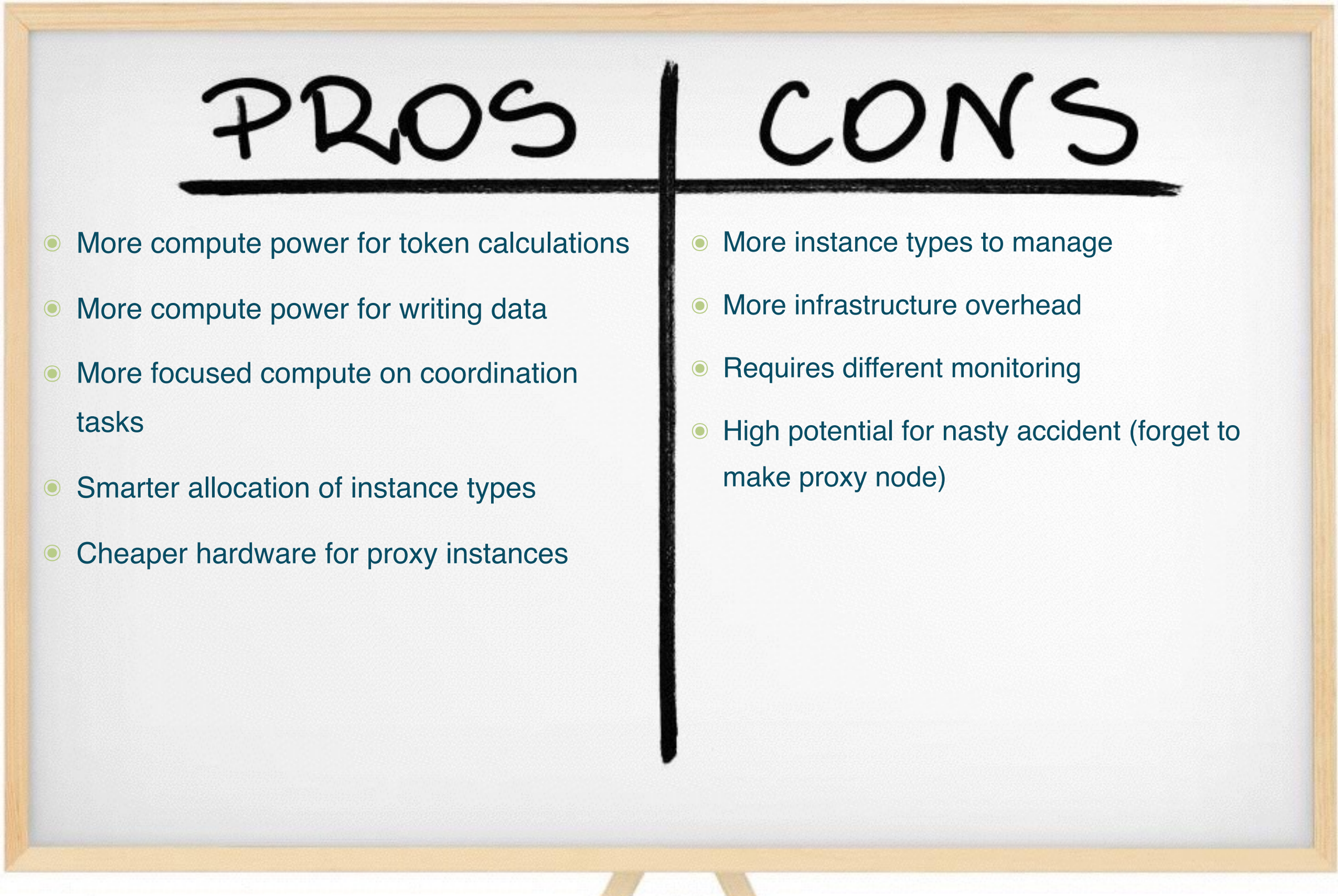
CASSANDRA PROXY TIER SETUP

APPLICATION TIER PROXY TIER CASSANDRA CLUSTER

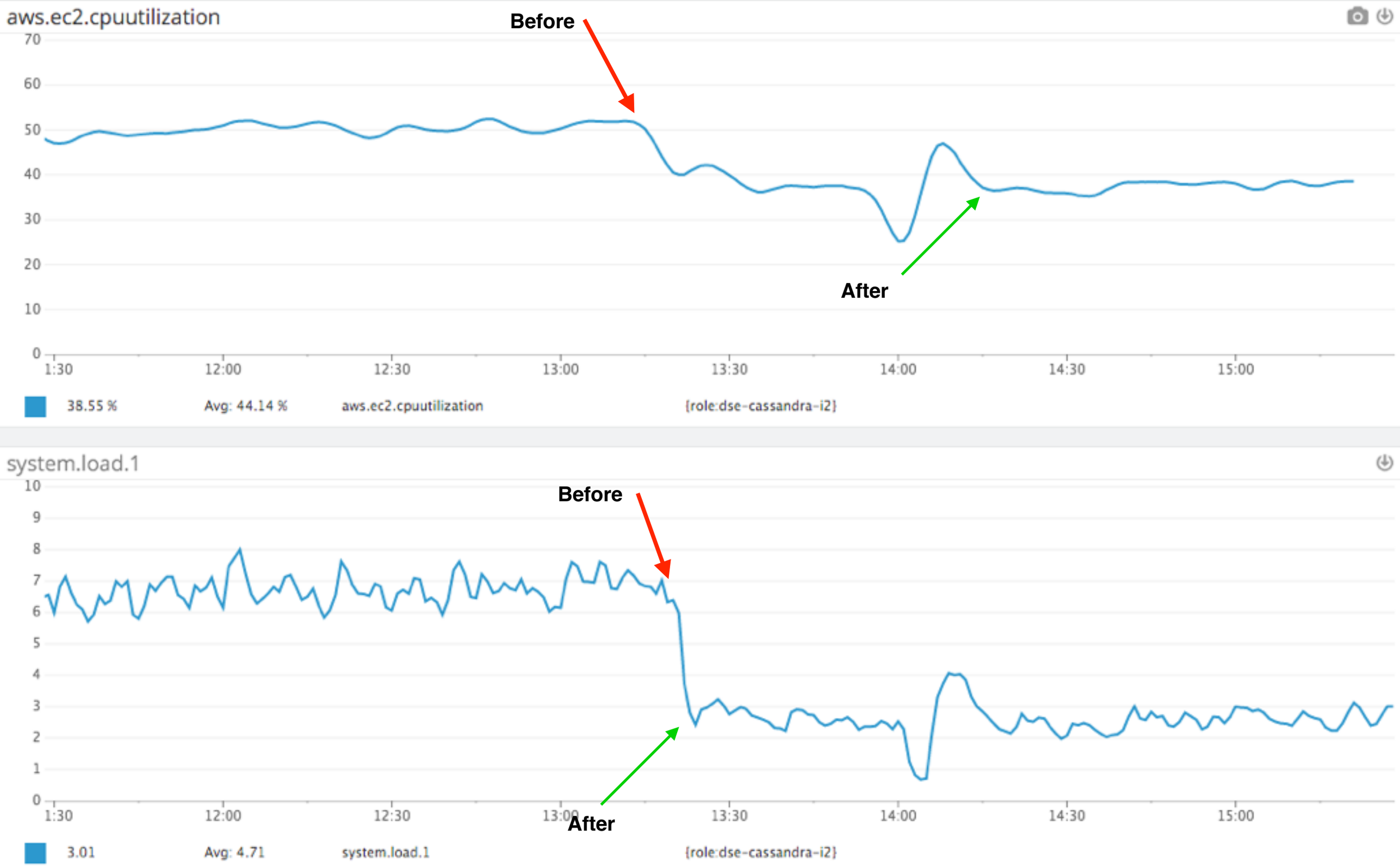


Proxy Nodes: c3.4xlarge **Data Nodes: c3.4xlarge**

TRADEOFFS


PROS
CONS
More compute power for token calculations
More compute power for writing data
More focused compute on coordination tasks
Smarter allocation of instance types
Cheaper hardware for proxy instances
More instance types to manage
More infrastructure overhead
Requires different monitoring
High potential for nasty accident (forget to make proxy node)

AVERAGE CLUSTER CPU UTILIZATION



HOW DID WE DO?

HOW DO WE KNOW?

- Why are we talking about CPU utilization and load average
- Terminology is important
- Understanding gains/losses is important
- Let's talk about CPU utilization and load average

WHAT IS LOAD AVERAGE?

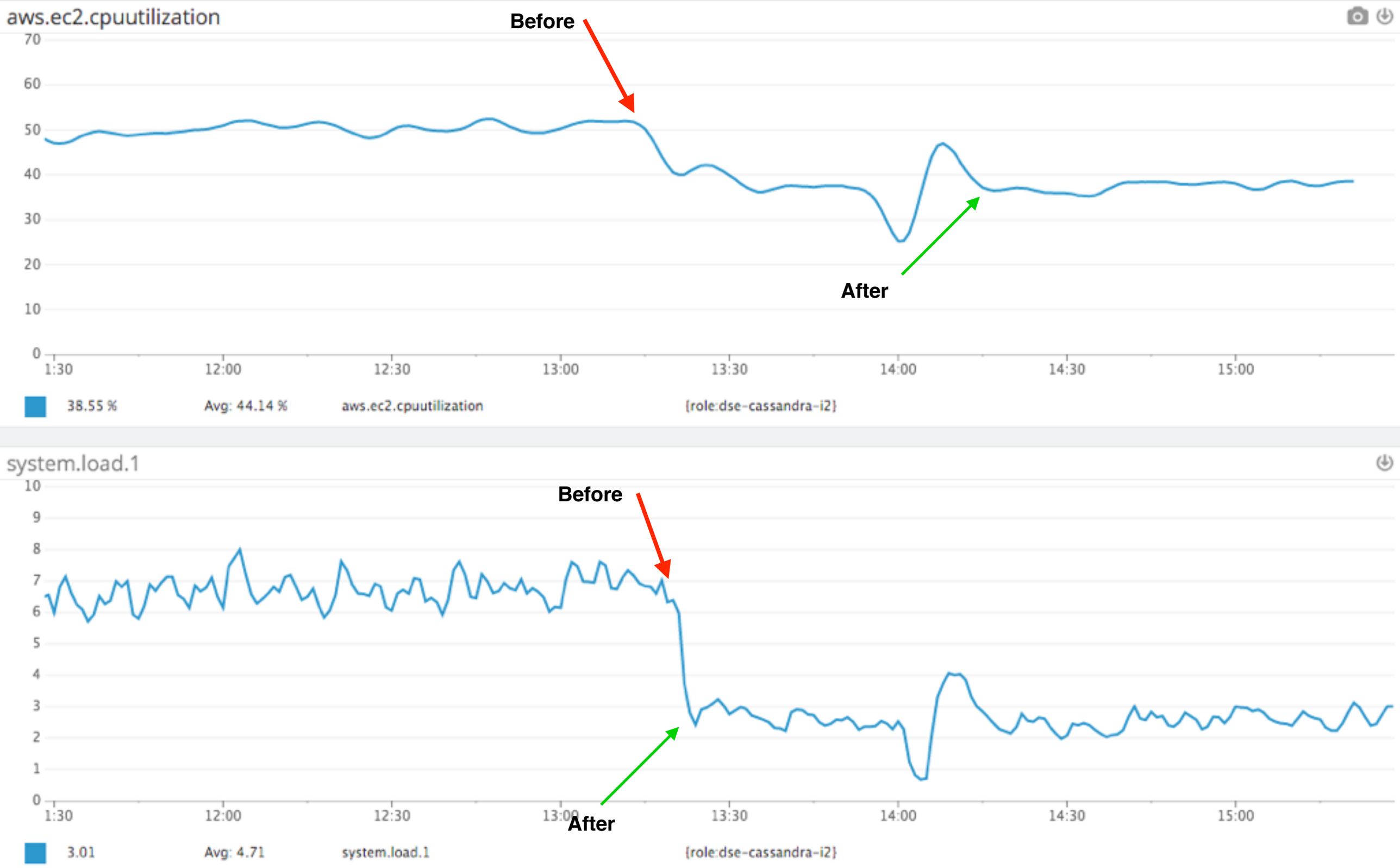
- Let’s use a traffic analogy for load average
- Imagine you are the bridge operator of a single lane bridge (single CPU):
  - 0.00 means there's no traffic on the bridge at all. In fact, between 0.00 and 1.00 means there's no backup, and an arriving car will just go right on.
  - 1.00 means the bridge is exactly at capacity. All is still good, but if traffic gets a little heavier, things are going to slow down.
  - over 1.00 means there's backup. How much? Well, 2.00 means that there are two lanes worth of cars total -- one lane's worth on the bridge, and one lane's worth waiting. 3.00 means there are three lane's worth total -- one lane's worth on the bridge, and two lanes' worth waiting. Etc.
- Best load average for a single CPU system is between 0.7 and 0.8 (headroom)
- Different for multi-core systems

NOTES ABOUT CPU UTILIZATION

- Each core on a CPU has it's own utilization graph
- CPU utilization isn't straight forward
- Assume you have a single core processor fixed at a frequency of 2.0 GHz. CPU utilization in this scenario is the percentage of time the processor spends doing work (as opposed to being idle). If this 2.0 GHz processor does 1 billion cycles worth of work in a second, it is 50% utilized for that second.
- Current multiple cores processors exist with dynamically changing frequencies, hardware multithreading, and shared caches all of which effect reporting.
- Resource sharing makes monitoring CPU utilization difficult


LETS SEE THAT AGAIN
BUT NOW IN REAL TIME
quickmeme.com

AVERAGE CLUSTER CPU UTILIZATION



WHAT ACTUALLY HAPPENED 1/2

- Batches work better when there is a coordinator dispatching batches to the correct data node without additional processing on the part of the data node
- One of the many downsides of vnodes is massive coordination requirements
- Removing coordination responsibilities from data nodes makes them more performant
  - less context switching
  - less network traffic/gossip/GC
  - less CPU utilization

WHAT ACTUALLY HAPPENED 2/2

- 30 nodes * 256 tokens/node = 7,680 token ranges
- Queries go through a nearly 8,000 item list, slow, context switch, lots of GCable objects
- Considering just reads, at 30k requests per second, this is a significant reduction in work on a per query basis
- We are able to tune the JVMs differently

RESULTS

RESULTS

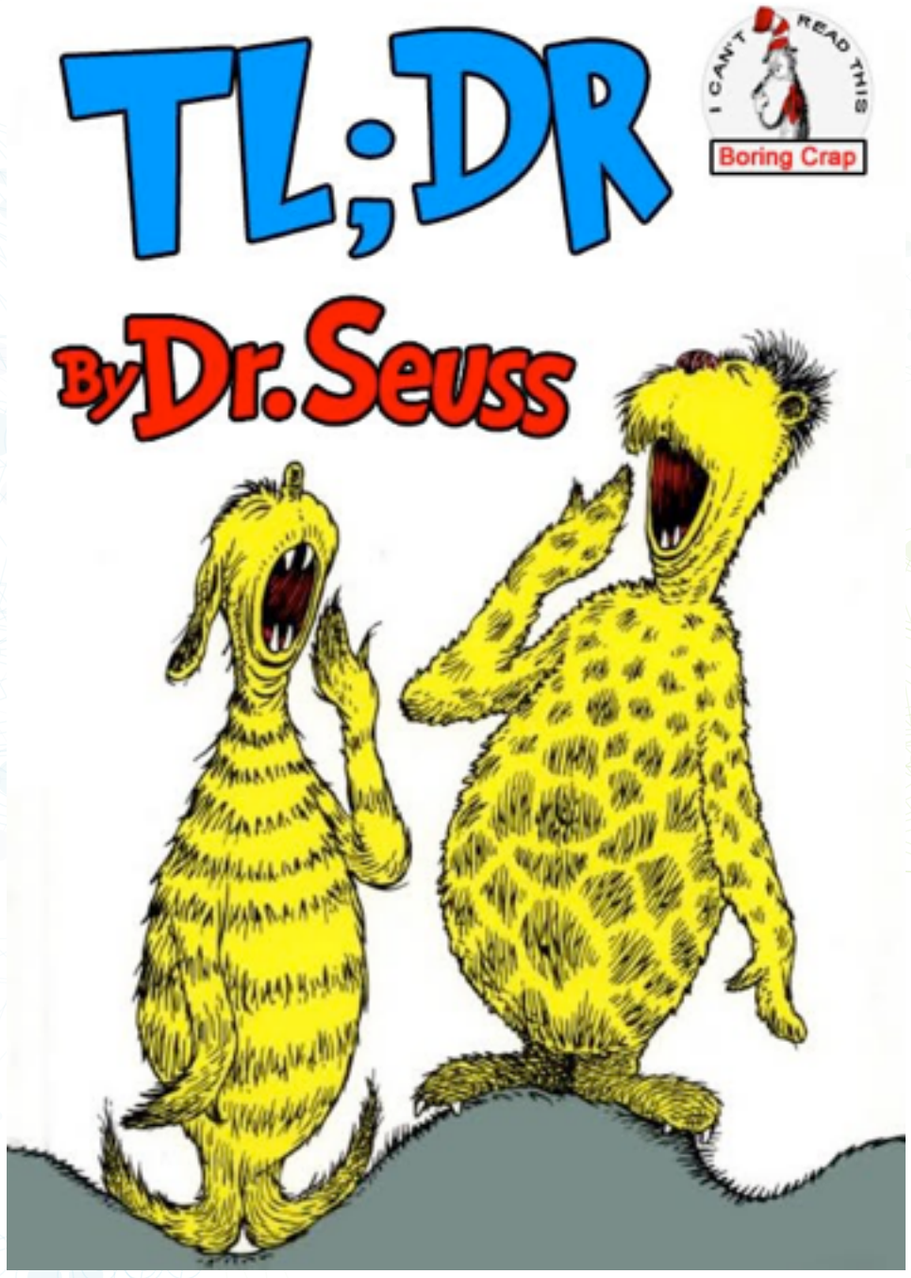
- Went from 72 nodes down to 30 nodes
- All data is now stored on AWS ST1 EBS volumes
  - Works best for write heavy workloads
- Roughly 300% increase in available and burstable capacity
- Less footprint to watch over; fewer machines, more roles

FEATURE NOT HACK

- Command line option or cassandra.yaml option for coordinator only mode
- Code path short cuts for performance
- Specific JMX beans around query coordination
- Allow query mutation by coordinator nodes (Lua?)

WHAT DID I SAY?

- Fat clients can save you money
- Don't start out with complexity
- Know the basics
- Know what your baseline measurements are
- Monitor everything
- Sometimes database tuning doesn't require making changes to the database



QUESTIONS IN LIFE ARE GUARANTEED, ANSWERS AREN'T.

Eric Lubow

@elubow