ThreatMetrix®
The Digital Identity Company™

Keeping RAFT Afloat
Cloud Scale Distributed Consensus

Philip Haynes

YOW! Data
September 2016

CONSENSUS?

- To the general public, consensus is usually a good thing…..

The downside……

- On the other hand, what if the robotic consensus decides that humans should be exterminated….

- In IT reality of course, the role of consensus is much more subtle

What is a consensus algorithm then?

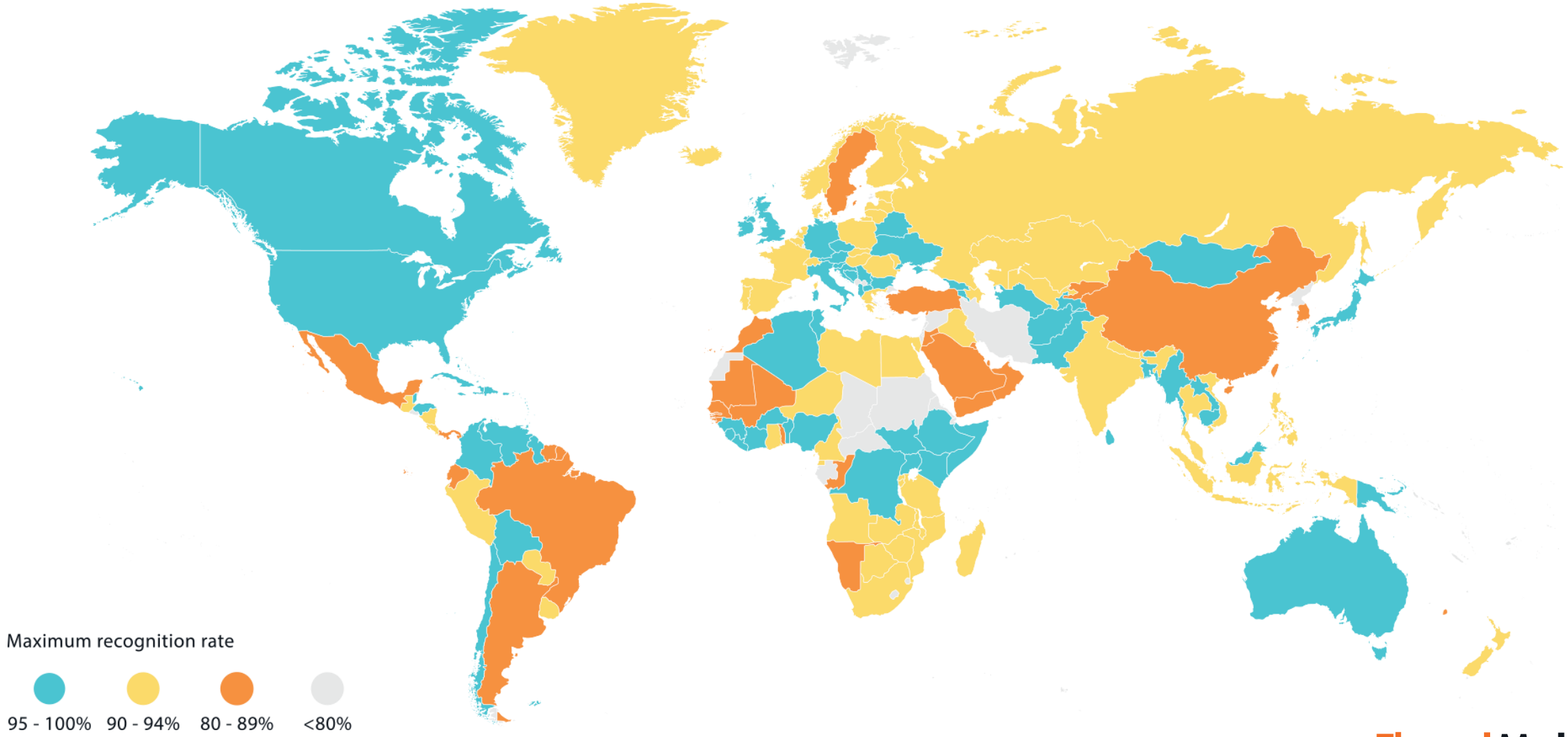
- The consensus problem is fundamental in the control of a multi agent system e.g. multiple servers
- A consensus problem requires agreement among a number of processes (or agents) for a single data value
- Some of the processes (agents) may fail or be unreliable in other ways, so consensus protocols must be fault tolerant
- One option is for all processes (agents) to agree on a majority value e.g. > half the votes

Hard Distributed Consensus is:

- Fundamental where a consistent view of a system in the presence of failure is required
  - Think financial records / TP systems
- Unfortunately perceived as being too difficult and costly for cloud scale – hence "eventually" consistent models
- Critical for simplifying big data processing and its analysis where:
  - Near enough is not good enough; and
  - Systems must always be up

Cloud scale at ThreatMetrix

Global Device Identity Recognition Rates



ThreatMetrix®

Cloud scale consensus requirements

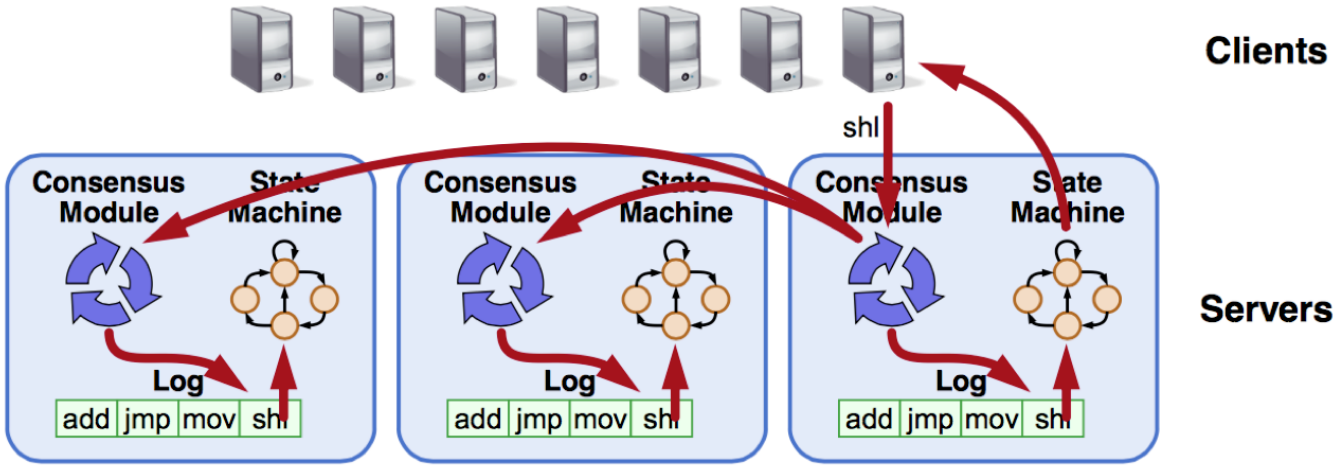
- > 100M digital identity requests per day. Internal SLA < 100ms. Multi-data center. > 400 node Cassandra cluster.
- New capability for operational fraud processing:
  - Capable of processing > 1B events per day
  - < 5 ms initial detection
  - Far fewer nodes than 400 (3-5); where
  - Results are evidentiary (i.e. consistency matters)
- We also care about availability and scalability

Building blocks for cloud scale distributed consensus

- Hardware aware programming methods
- Aeron messaging. Low latency reliable transport from Martin Thompson et. al.
- RAFT. New distributed consensus algorithm designed to be understandable (compared to Paxos etc.)
- Asynchronous replicated log system model
- Concern existed that distributed consensus is hard to implement*
- RocksDB. LSM database originally from Google, now Facebook
  - Designed for write heavy loads on SSD's

*Our local experiment continues to support this view.

RAFT – A replicated log

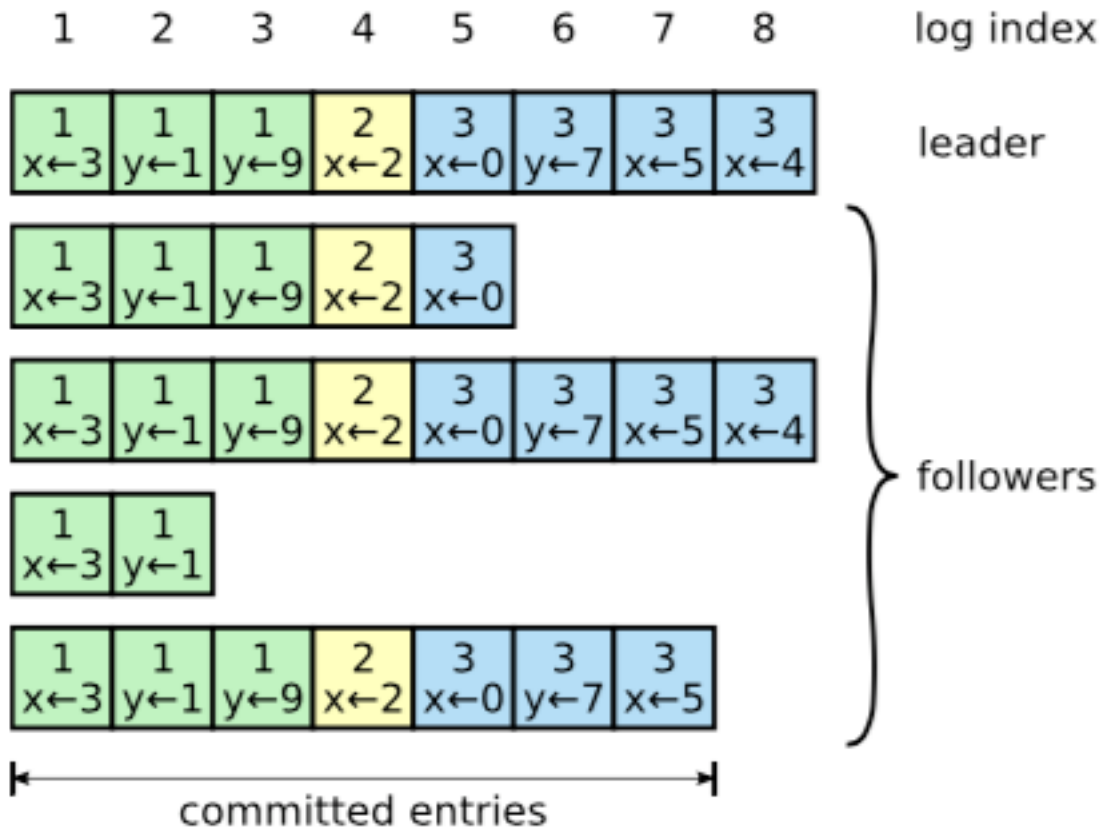


- Replicated log => replicated state machine
  - All servers execute same command in same order
- Consensus module ensures proper log replication
- Systems makes progress as long as the majority of servers are up
- Failure mode: fail-stop (not Byzantine), delayed loss messages

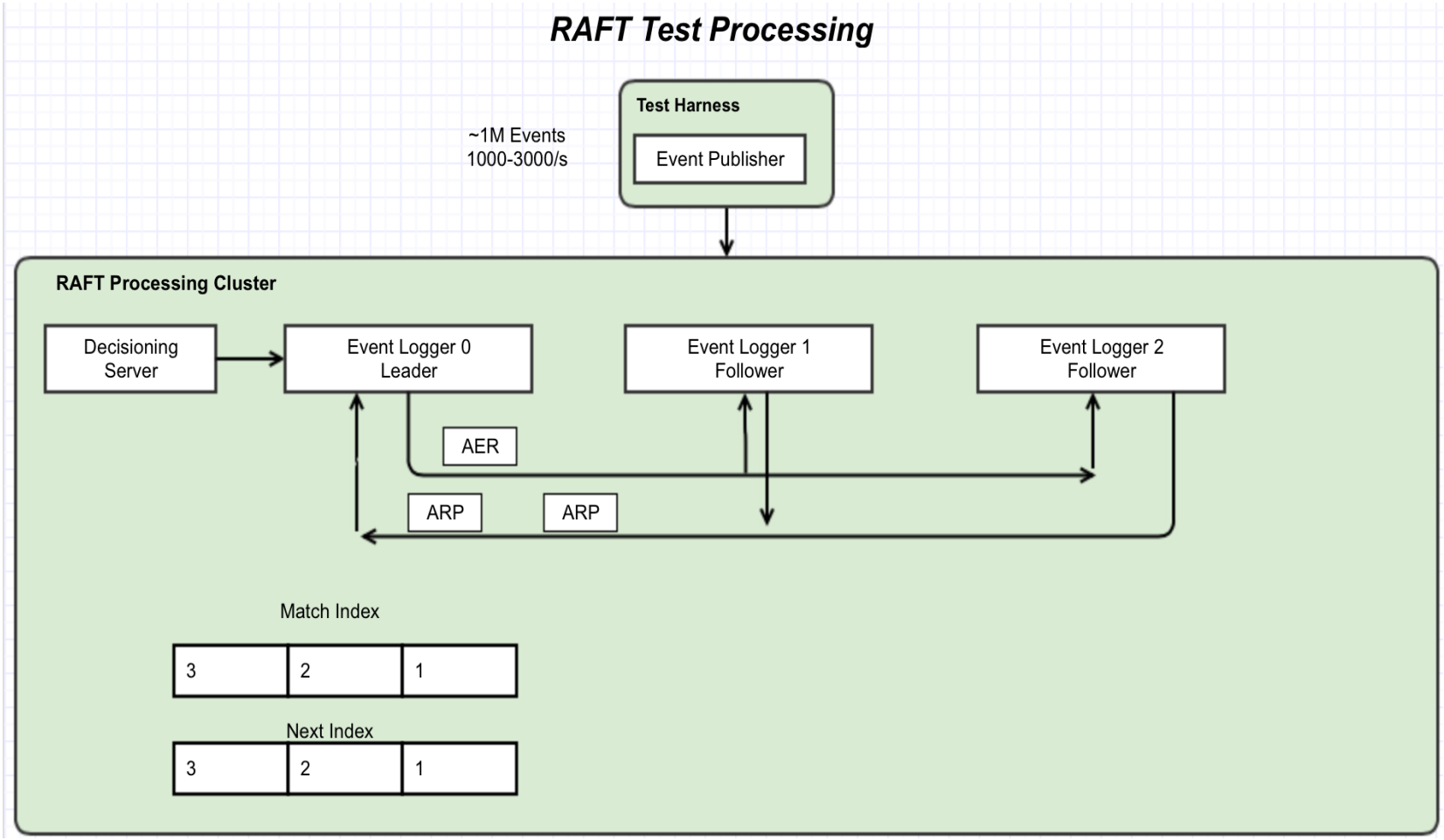
Replicate across a cluster with a leader adding concept of time (called a term)

Odd number of servers to support voting

1. Request Vote RPC to elect leader
2. AppendEntry RPC to replicate log entries
3. When majority of followers append entries the log entry is committed and the state machine may be applied



Initial RAFT implementation

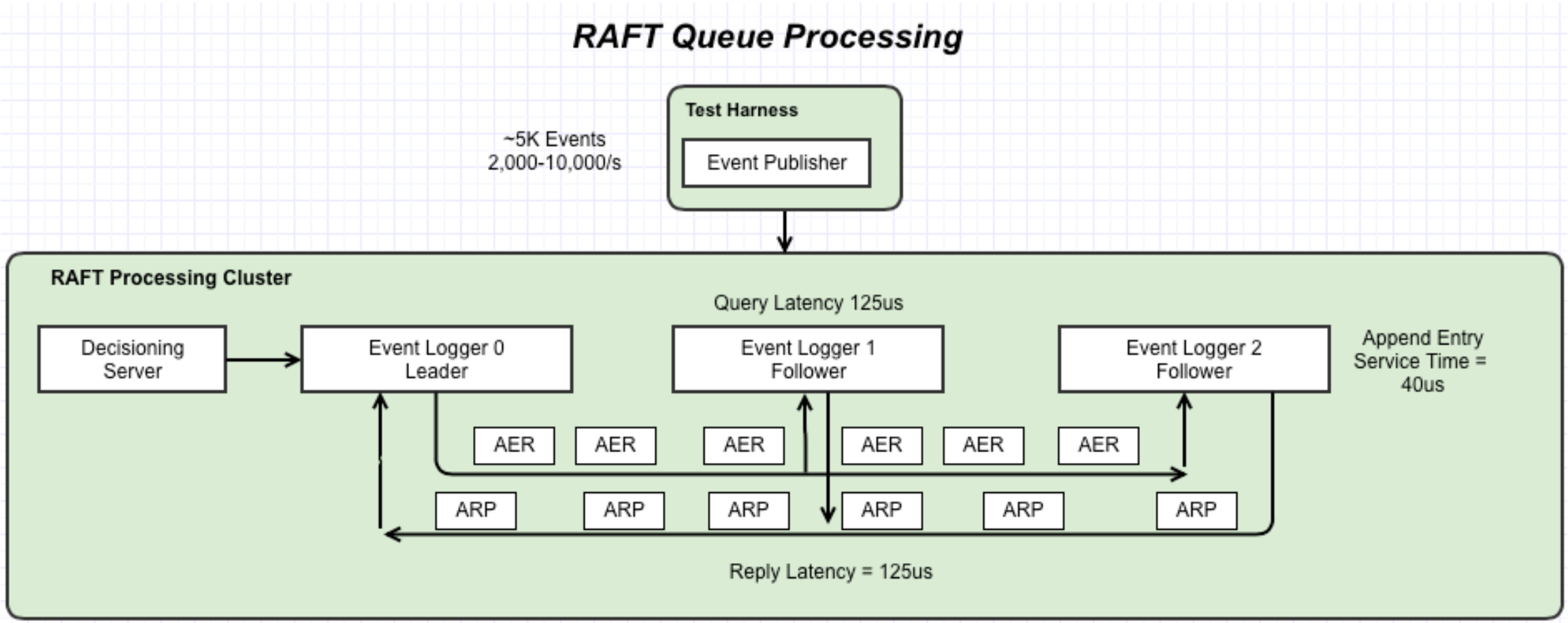


ThreatMetrix®

Key initial implementation failures

- Conceptual: Misunderstood that communication between leader and followers must be viewed as a queue of request and responses
- Flow control
  - Started seeing > 7000 messages in flight
  - When this happened the system collapsed
  - Requirement for flow control not understood on raft-dev
- State of Practice
  - TMX RAFT: > 3K msgs/s @ ~300us latency
  - Public: 20 per second, batched to 50ms over TCP/IP

Flow control for RAFT: The hypothesis

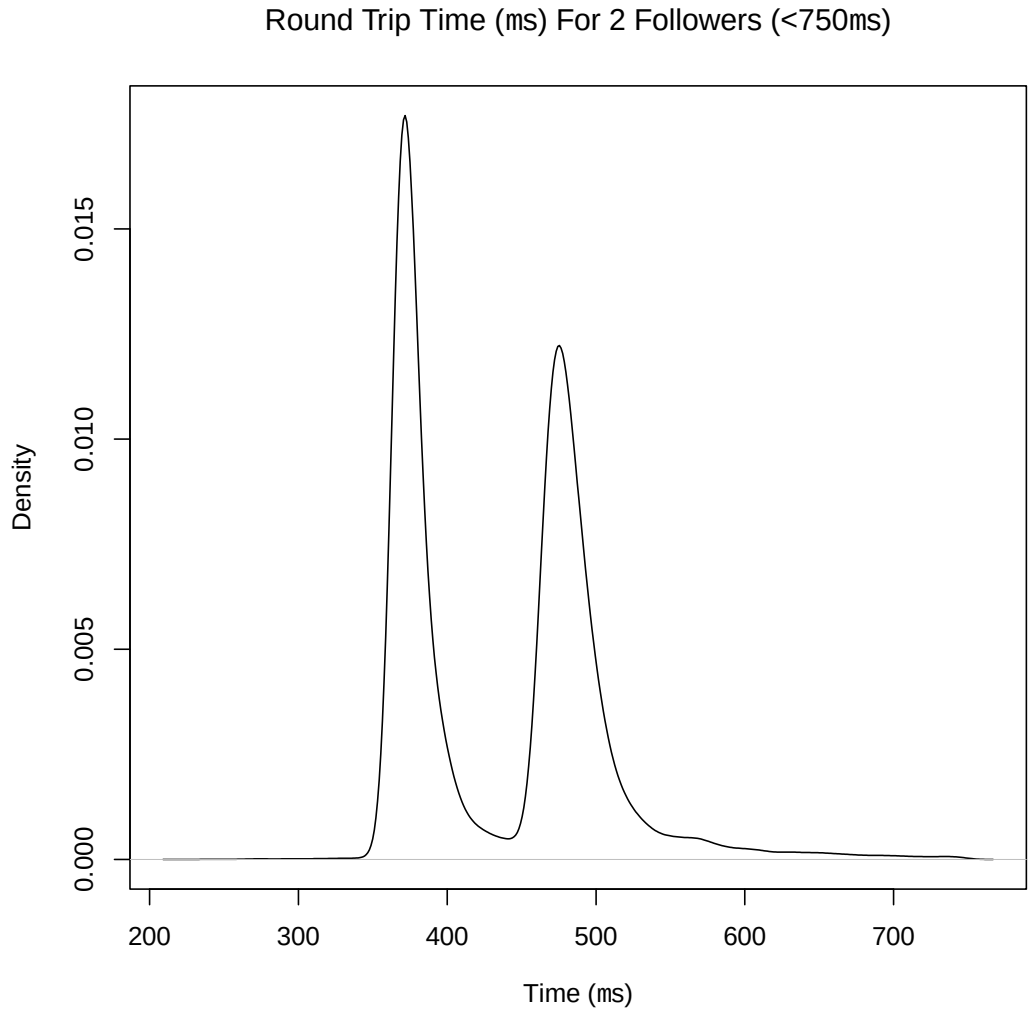


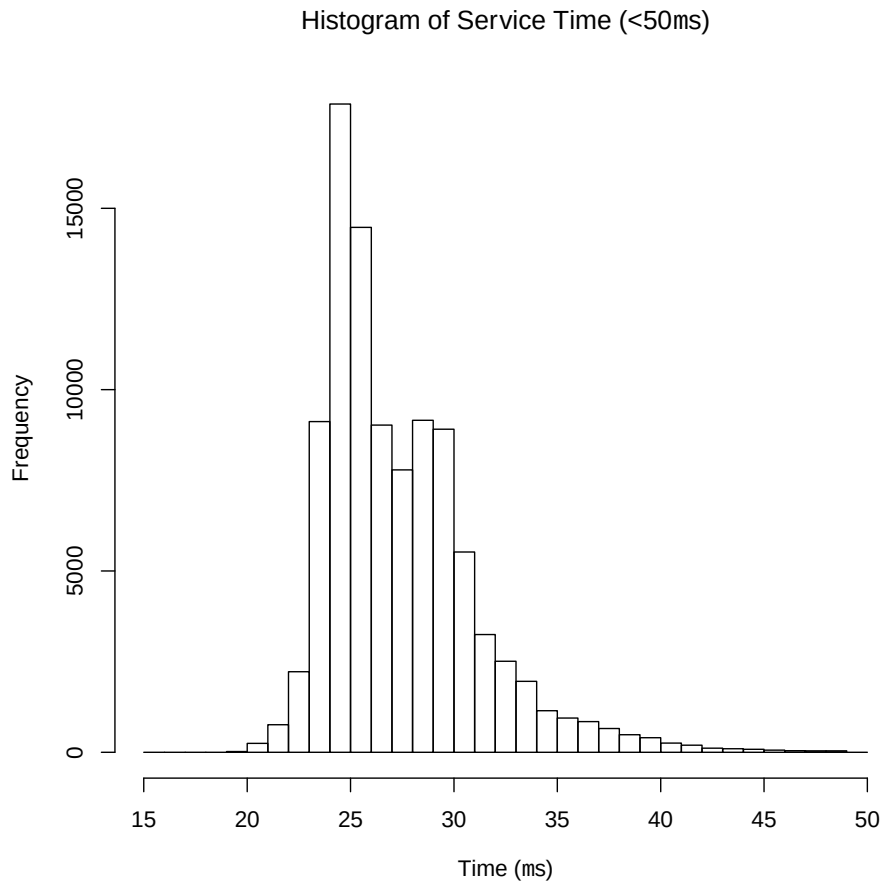
ThreatMetrix®

RAFT flow control: The implementation

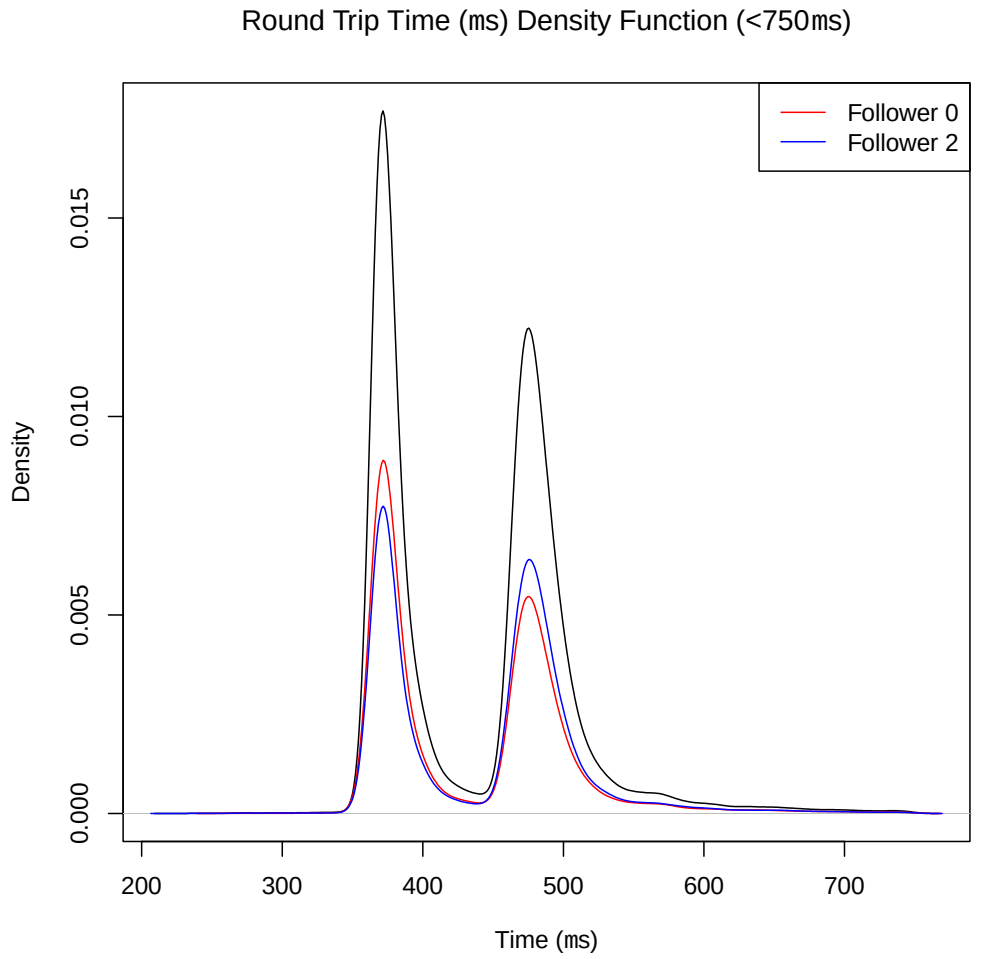
- Keep moving averages of round trip time and service time
- Keep a record of messages in flight
  (i.e. queue size and active nodes)
- Throttle when:
  - timeSinceLastReceived < heatBeatTimeOut; and
  - Queue size >= maxQueueSize; where
  - maxQueueSize = Math.*max(1* + (int)(minRoundTripLatency / serviceTime), 10);

Performance results: Round trip time and service time



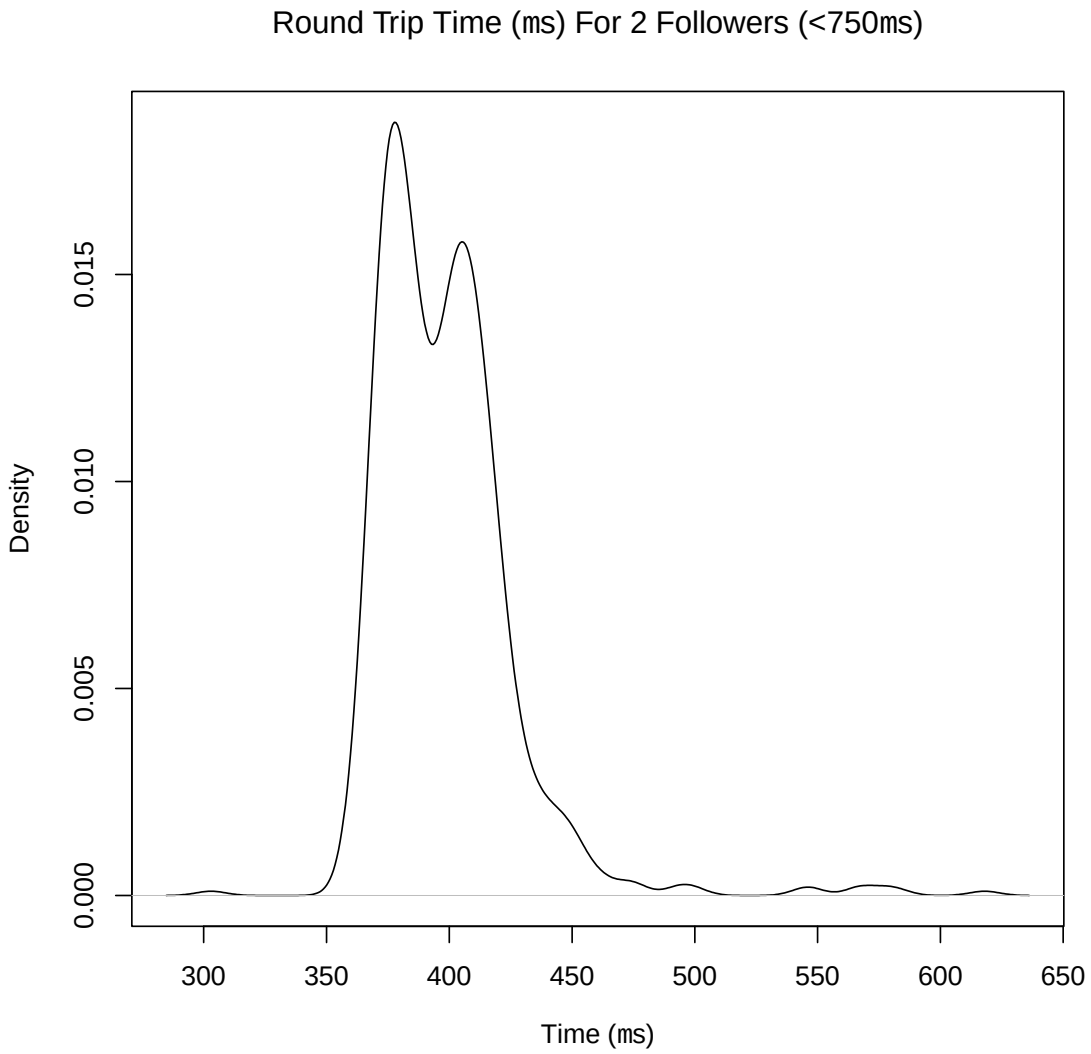


Round trip and service time analysis



ThreatMetrix®

Round trip final analysis



Getting it to really work

- Implementation attempted to utilize multicast to provide discovery across the cluster
  - Flow control interference between clusters
- Modified for independent RAFT cluster flow control
- RAFT messages prioritization over command messages
- Introduced flow control between the different clusters
- DTrace to identify and remove outliers (Units in nano’s)
- Now processing:
  - 1,600 events per second; to process
  - Creating and closing 1,600 cases per second

```
           value  ------------- Distribution ------------- count
            2048 |                                         0
            4096 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@           9345089
            8192 |@@@                                      967619
           16384 |@@@@@@                                   1721219
           32768 |@                                        341919
           65536 |                                         20138
          131072 |                                         9235
          262144 |                                         3256
          524288 |                                         19
         1048576 |                                         5
         2097152 |                                         37
         4194304 |                                         0
```

Conclusion

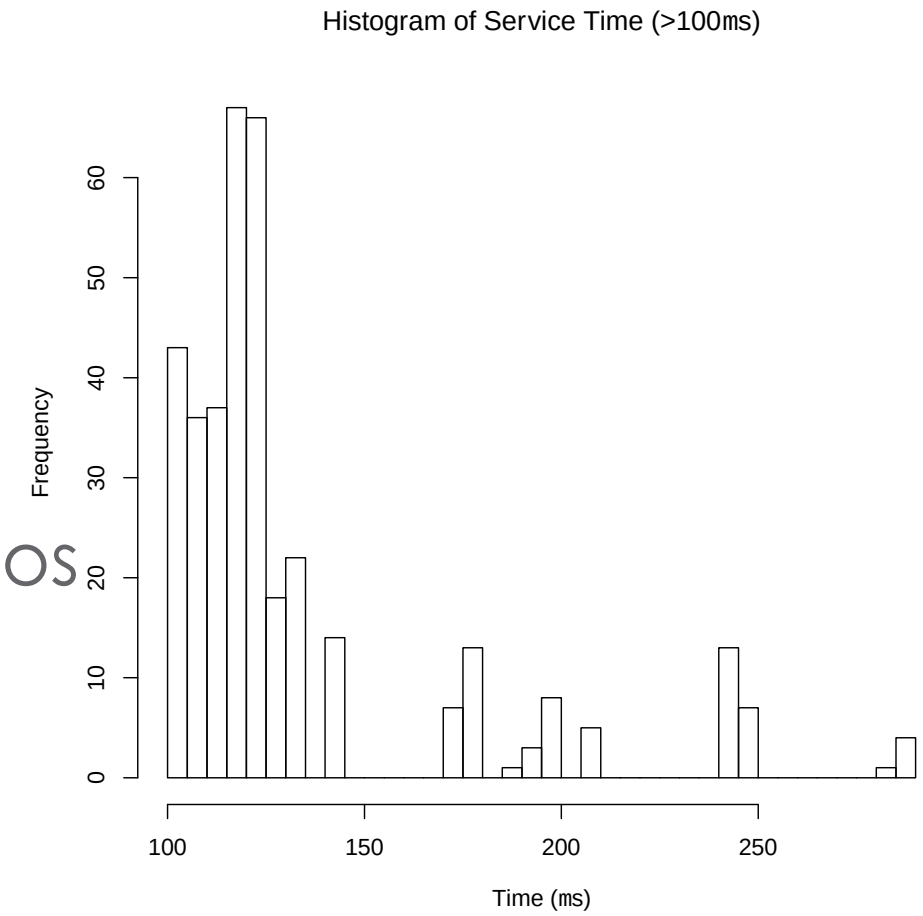
- Aeron and other hardware aware programming techniques are fundamental to reducing the cost for cloud scale services
- RAFT and DFSM's are fundamental for implementing transaction engines but are insufficient
- Cloud scale is fundamentally different scale to research systems
- Performance model and measure system processing

ThreatMetrix®

Questions?

Study limitations

- Yet to fully optimize the system
- Repeat on 10G hardware
- More than 4 followers
- Multi-data center issue
- Flow control during failure scenarios



Latency Curve