Elasticity

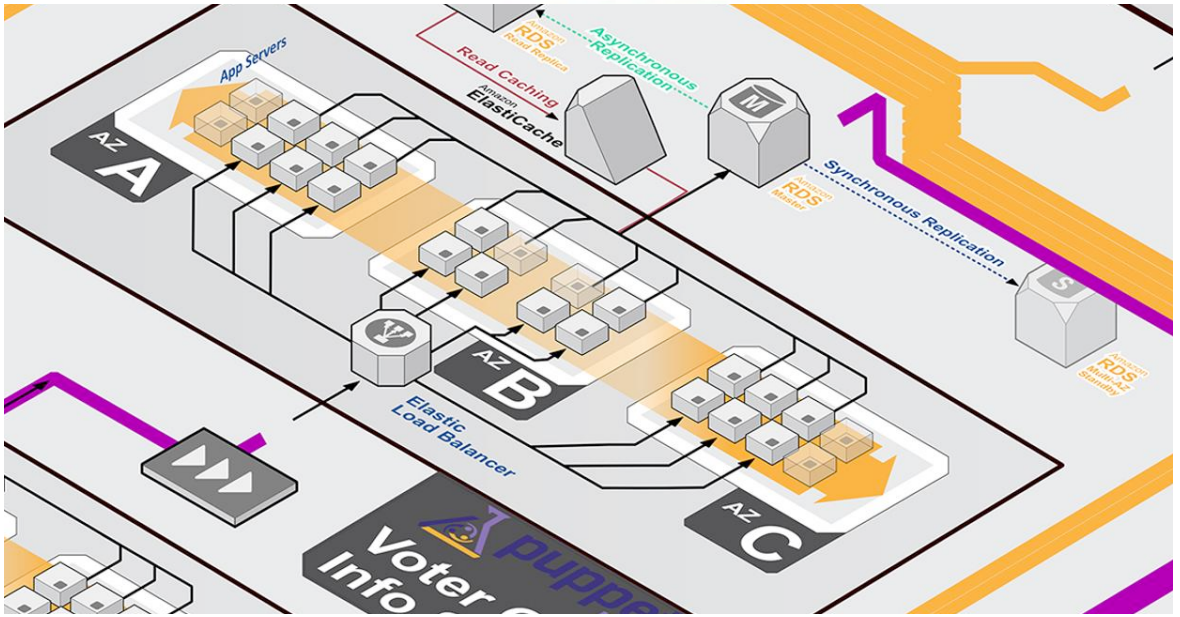
15-719:
Advanced Cloud Computing

Garth Gibson
Greg Ganger
Majd Sakr

Advanced Cloud Computing Elasticity Readings

- Req'd Ref 1: "Dynamically Scaling Applications in the Cloud." Luis M. Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. ACM SIGCOMM Computer Communications Review (CCR), v 41, n 1, 2011. http://www.sigcomm.org/ccr/papers/2011/January/1925861.1925869
- Opt Ref 2: "Jockey: guaranteed job latency in data parallel clusters." Andrew D. Ferguson, Peter Bodik, Srikanth Kandula, Eric Boutin, and Rodrigo Fonseca. 7th ACM European conf. on Computer Systems (EuroSys '12), 2012. http://doi.acm.org/10.1145/2168836.2168847
- Opt Ref 3: "Split/Merge: System Support for Elastic Execution in Virtual Middleboxes." Shriram Rajagopalan, Dan Williams, Hani Jamjoom, Andrew Warfield. 10th USENIX Symp. on Networked Systems Design and Implementation (NSDI '13), 2013. https://www.usenix.org/system/files/conference/nsdi13/nsdi13-final205.pdf
- Opt Ref 4: "ElasTraS: An elastic, scalable, and self-managing transactional database for the cloud."Sudipto Das, Divyakant Agrawal, Amr El Abbadi. ACM Trans. Database Syst. 38, 1, Article 5, 2013. http://doi.acm.org/http://dx.doi.org/10.1145/2445583.2445588

Web Servers

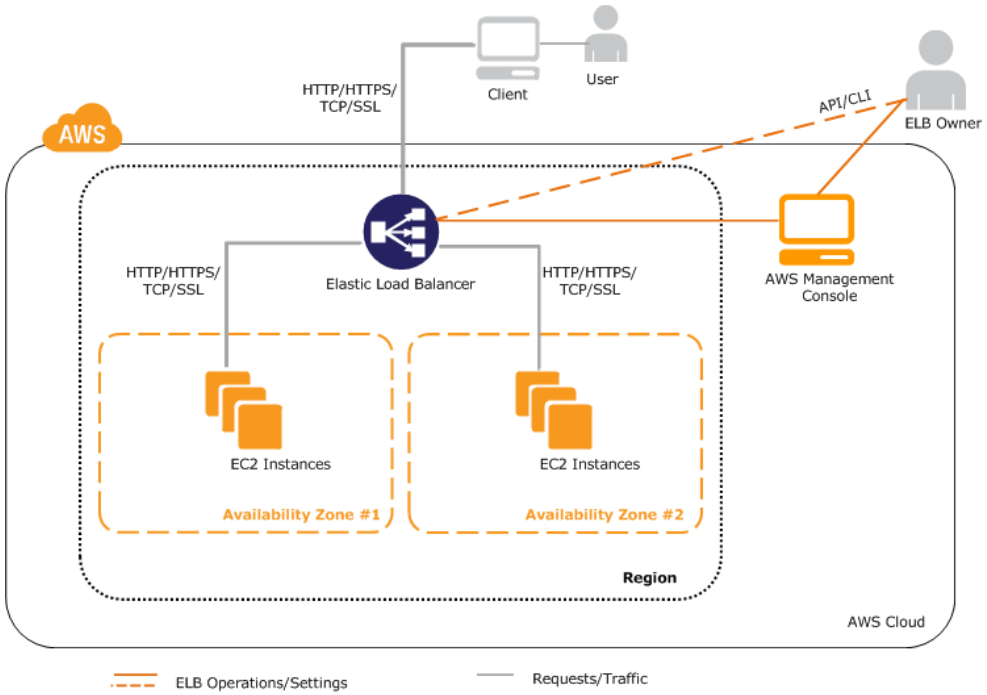


- The “killer app”
- Online retail service
- Most cycles spent on web page rendering
- Customers use http to browser inventory, viewing pictures, lists, technical specs, price comparisons before requesting a product be purchased and shipped to them
- Number of machines needed fluctuates with customer interest, changing with inventory and marketing – want to pay only for need

Basic service parallelization: Load Balancer

- Incoming stream of independent user requests broken into multiple separate streams, one per allocated machine
- Load balancer is not necessarily elastic
  - AWS charges for CloudWatch to monitor your EC2 instances
  - AutoScaling of the EC2 instances and reconfiguring of AWS Elastic Load Balancer at no add'l charge

- docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/SvcIntro_arch_workflow.html

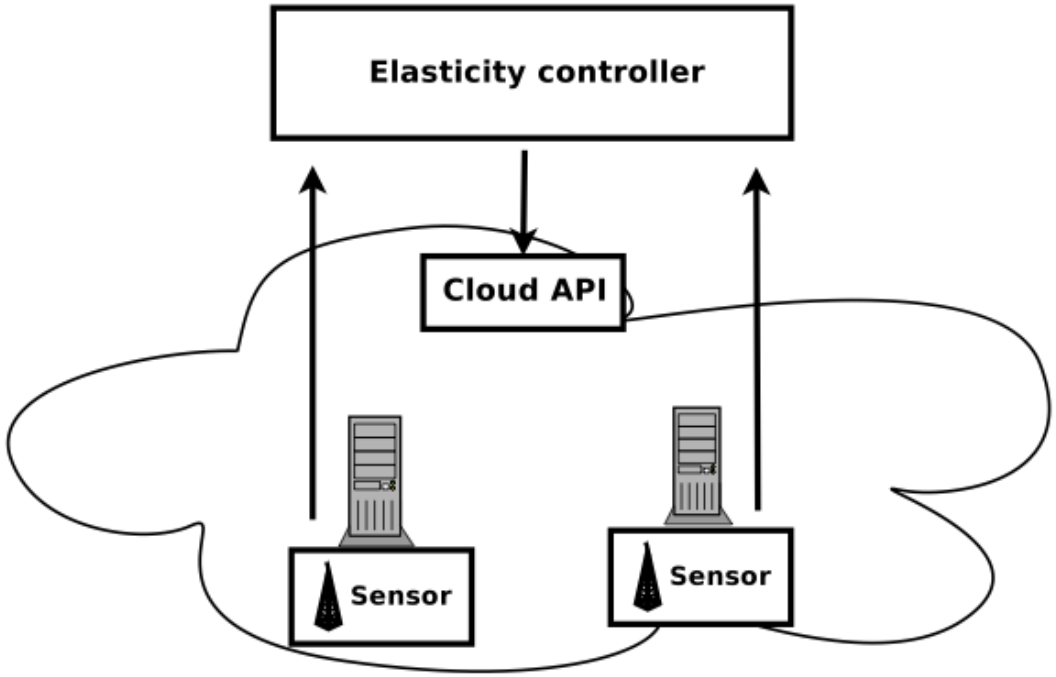


Load Balancing Mechanism

- DNS load balancing
  - out of band, can distribute arbitrary bandwidth (not limited by bandwidth of a router)
  - Tells client a binding of a name to an IP (list), which makes dynamic changing of the binding hard, or at least up to the client
- Router balancing of connection
  - TCP or SSL both start a connection, flow data, then close the connection
  - A router on the path can make a decision for the entire life of the connection
    - for some applications this might be long, making dynamic balance hard
- Router balancing of requests embedded in a connection
  - If router understands protocol embedded in connection flow, it can make routing decisions for specific requests
  - Most dynamic approach, but requires the most processing in the router
  - HTTP/HTTPS are the typical protocols that are recognized by load balancers

How is elasticity provided to web servers?

- Abstractly, an elasticity controller monitors allocated machines
- When overloaded,
  add load capability
- When under used,
  delete load capability
- Adding/deleting done
  by cloud framework



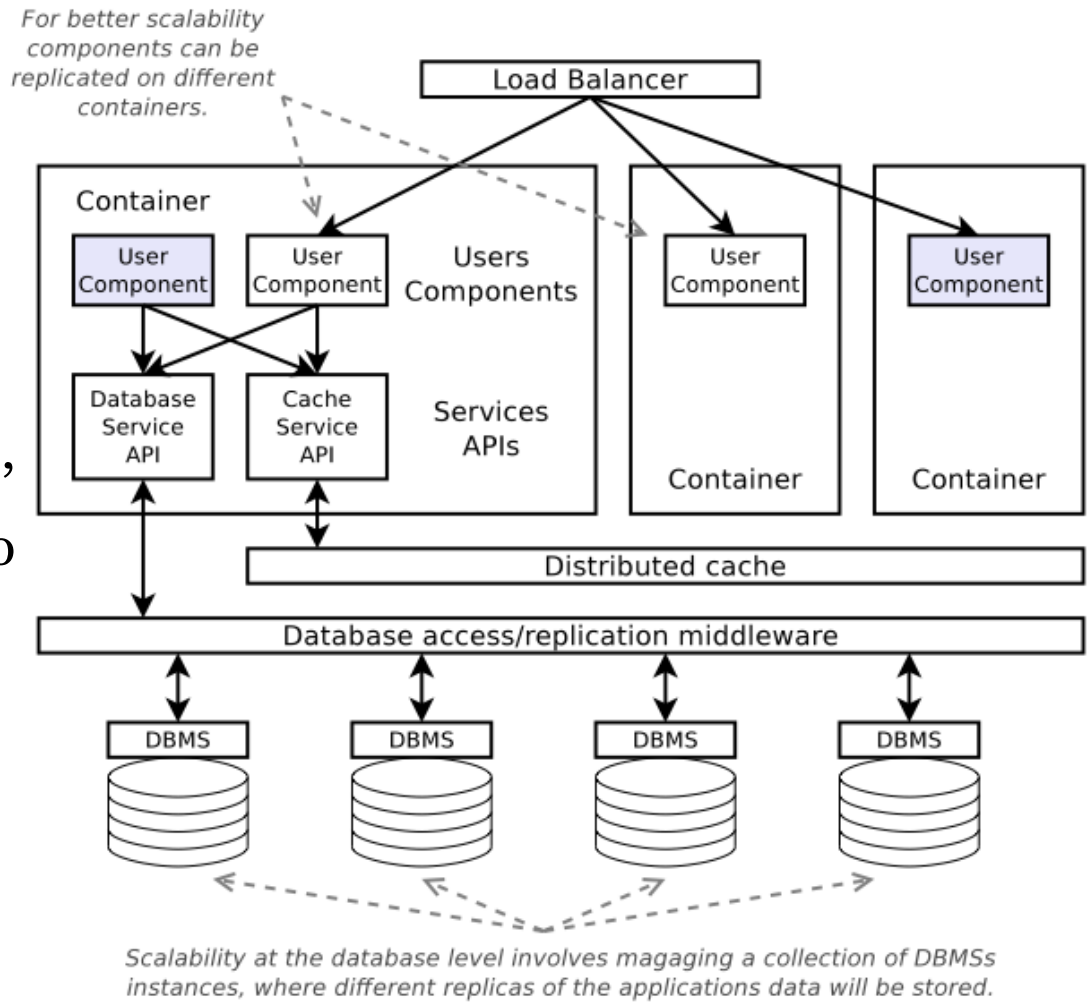
Elasticity: Scale-out or Scale-up?

- Horizontal scaling (Scale-out)
  - Adding more identically resourced instances
  - Most use of elasticity is done this way
- Vertical scaling (Scale-up)
  - Resizing the resources allocated to an existing instance
  - Does your (IaaS) OS accept and utilize more resources on the fly?
    - More network bandwidth is probably easy
    - More memory is harder but possible (eg. VM ballooning)
    - Changing cores is even harder
  - PaaS containers might hide resource representation
    - Eg. could provide more MapReduce slots in same machine

Elasticity Controller Capability

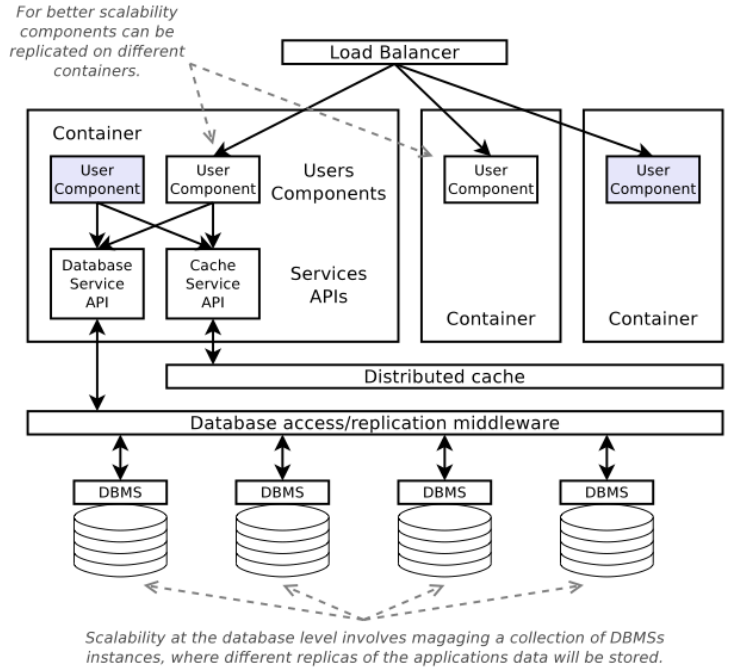
- User takes monitoring offered, defines rules for when to take actions (models the application)
- Monitoring
  - Monitor resource usage at specified instances, threshold individually or in total
  - Monitor request sequence, looking for patterns to reconfigure for predicted load
- Triggering
  - Trigger on simple conditions, thresholds, on monitored instances, request stream
  - Trigger on schedule (simple prediction)
  - Trigger on complex formula of many monitored instances (a model of overall service quality)
- Actions:
  - Launch single instances, or identical instances, or modify existing instances
  - Execute sequence of launches or modifications according to a dependency graph or workflow
  - Execute programs that implement a more abstract action (launch and configure multi-tier service)
- If a system is composed of a scalable set of processes feeding/interacting with another scalable set of processes implementing a very different function, are they monitored and scaled independently or is their elasticity inherently understood by one elasticity controller?
  - How good are your models for how you application actually depends on workload and resources?

Two tier services



- Most cycles in web servers, but want to take orders too
- Originally order taking wasn't even in cloud
  - Just send message
- Now frontends (web) & backends (database)
- Elasticity is certainly possible in IaaS – but database scaling is not simple replication of identical web server, so user scripts complex

Two tier web server scaling

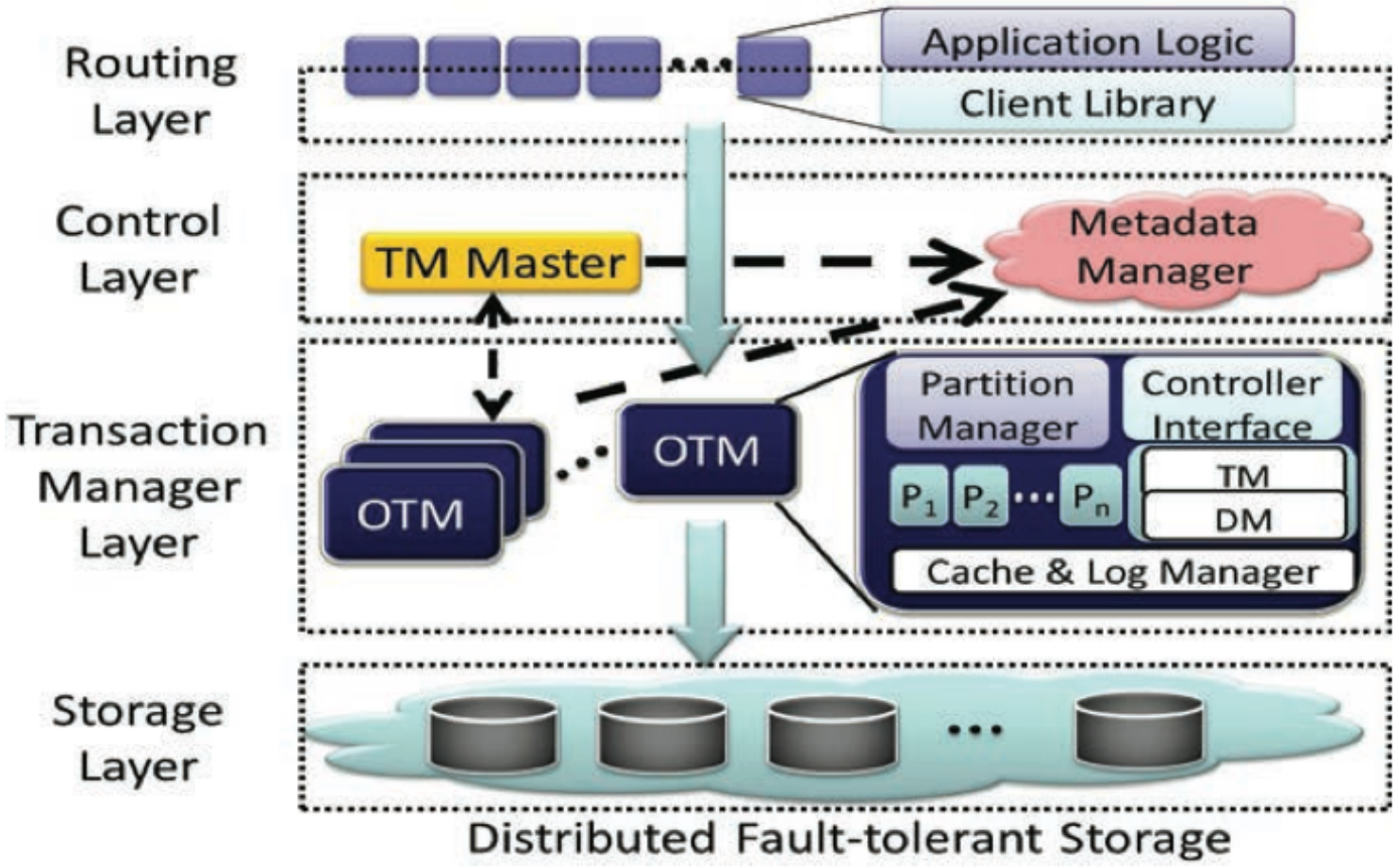


- Ahhah, PaaS where P == Web Service
- E.g. Google's AppEngine
  - provides a WSGI/CGI web server framework
- Built-in elastic load balancing and scheduled actions for containers
  - Most invoked servers have to complete in less than 60 seconds
- Built in persistent key-value store (datastore) & non-persistent memcache for simple database tier
- Users can instantiate Backends: bigger, long running, billed differently
  - Not autoscaled, but user code can request (actuate) horizontal scaling
  - Used for running traditional database services, whose scaling is still hard

What about a scalable relational database?

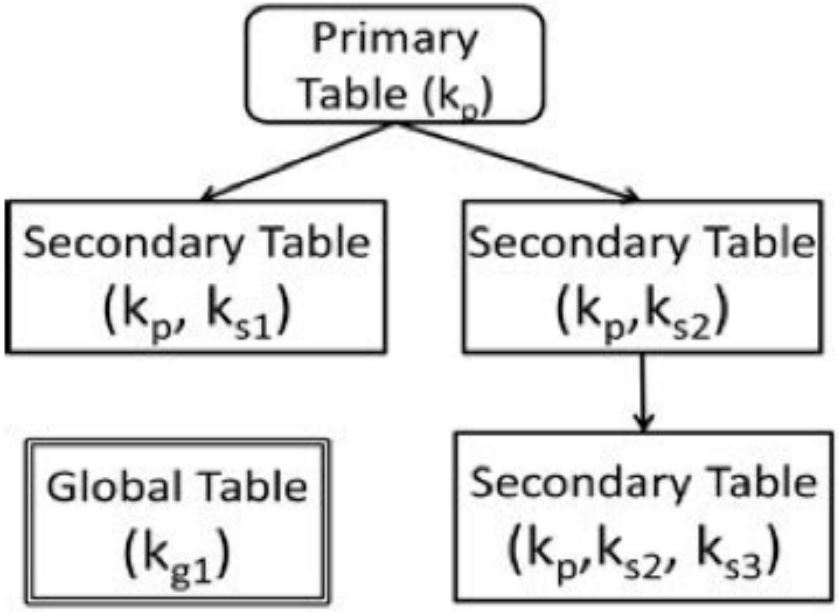
- Not a feature of traditional database, where users owned machines
- With a few possibly important restrictions, pretty easy (e.g. ElasTraS)
- Separate data at rest from ongoing or recent access & mutation
  - Data at rest is stored in (non-elastic) distributed pay-for-use storage (HDFS)
  - Recent access & mutation servers are elastic (called Owning Transaction Managers)
- Database can be partitioned but all transactions restricted to one partition
  - Distributed transactions usually block on locks and bottleneck performance scaling
- Elastic controller is also fault-tolerance manager
  - Servers can shutdown (flush to storage) or start up when controller re-assigns partitions
  - For the controller itself, reliability provided by replication (Zookeeper)

ElasTraS architecture scales OTM machines



Primary restriction: limitations on transactions

- Eliminate distributed transactions by rule – each transaction can touch only data from one partition
- But want dynamic repartitioning
- So establish a lot of mini-partitions allowing each OTM to manage many
- ElasTraS data model expresses this as the tree rooted at each row of the primary table



(a) Tree Schema.

- All transactions restricted to the tree beneath one row in the root table

Migrating OTMs with minimal pause

- First, push completed work to shared storage to reduce OTM state
- Next, "fuzzy migrate" of OTM state between servers
- Finally, stop processing requests for final part of migration

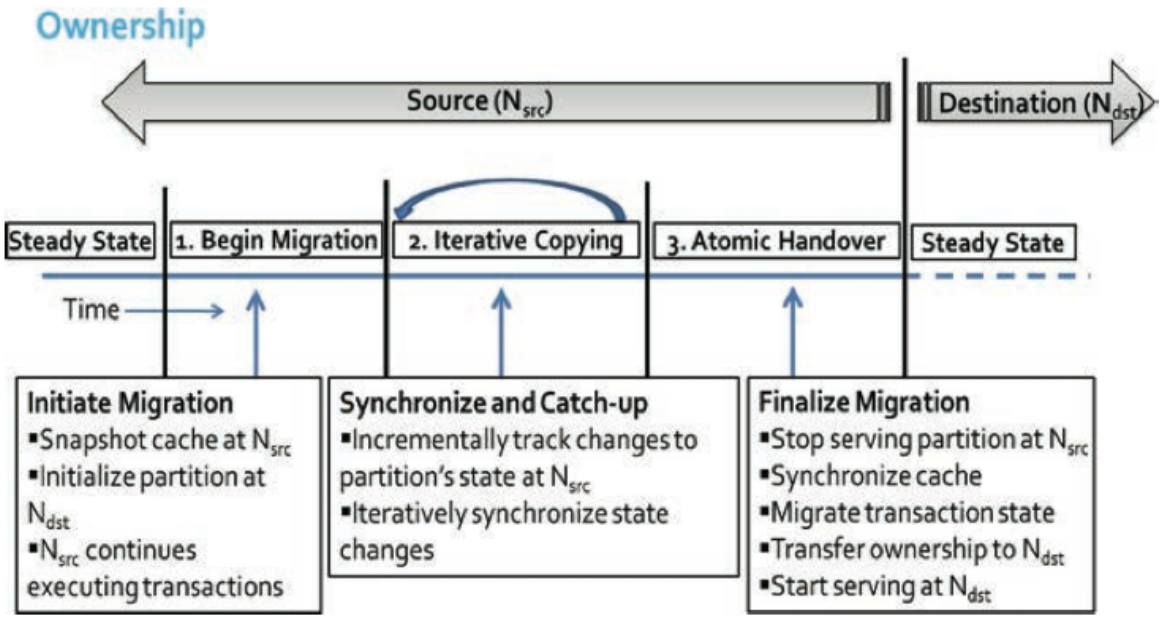


Fig. 3. Migration timeline for Albatross (times not drawn to scale).

Scaling the virtual network

- Load balancing is an example of a network middlebox
  - Eg., intrusion detection software, protocol accelerators, etc
  - Services that forward or mutate a network flow
- Scaling middleboxes is often overlooked, but may need its own tier
  - Especially if the function can be CPU intensive (ie., intrusion detection)
- Basic approach: split the flows
  - Eg. OpenFlow allows network switches to have flows defined & switched
  - “Load balancing the load balancers”
- Working with network switches & routers facilitates bandwidth allocation as well

Next up

- Next lecture will be on Programming Frameworks, part 2
- After that we will delve into cloud storage systems