CLUSTERING CAS for High Availability

Eric Pierce, University of South Florida

Overview

- High Availability Basics
- Before Clustering CAS
- Failover with Heartbeat
- Ticket Registry
- Load Balancing
- CAS at USF

HA is all about risk

- Make a list of *possible* Single-Points-of-Failure
  - Single connections to ANYTHING (Power, Network, etc)
  - Not just your servers – think about the datacenter
  - Try to quantify for management
    - How likely is this failure?
    - If it happens, how long will it take to fix?
    - How much will we lose while it is down?
  - Don't forget the human element!

Mitigating the risk

- Make a list of possible solutions
  - There are multiple ways to combat most SPoFs
  - Assign a relative cost score to each
    - The scoring system depends on your resources
    - Some things are easy to implement, but expensive
    - Cheaper solutions are (usually) more time-consuming
- Work with management
  - What risks are they willing to accept?

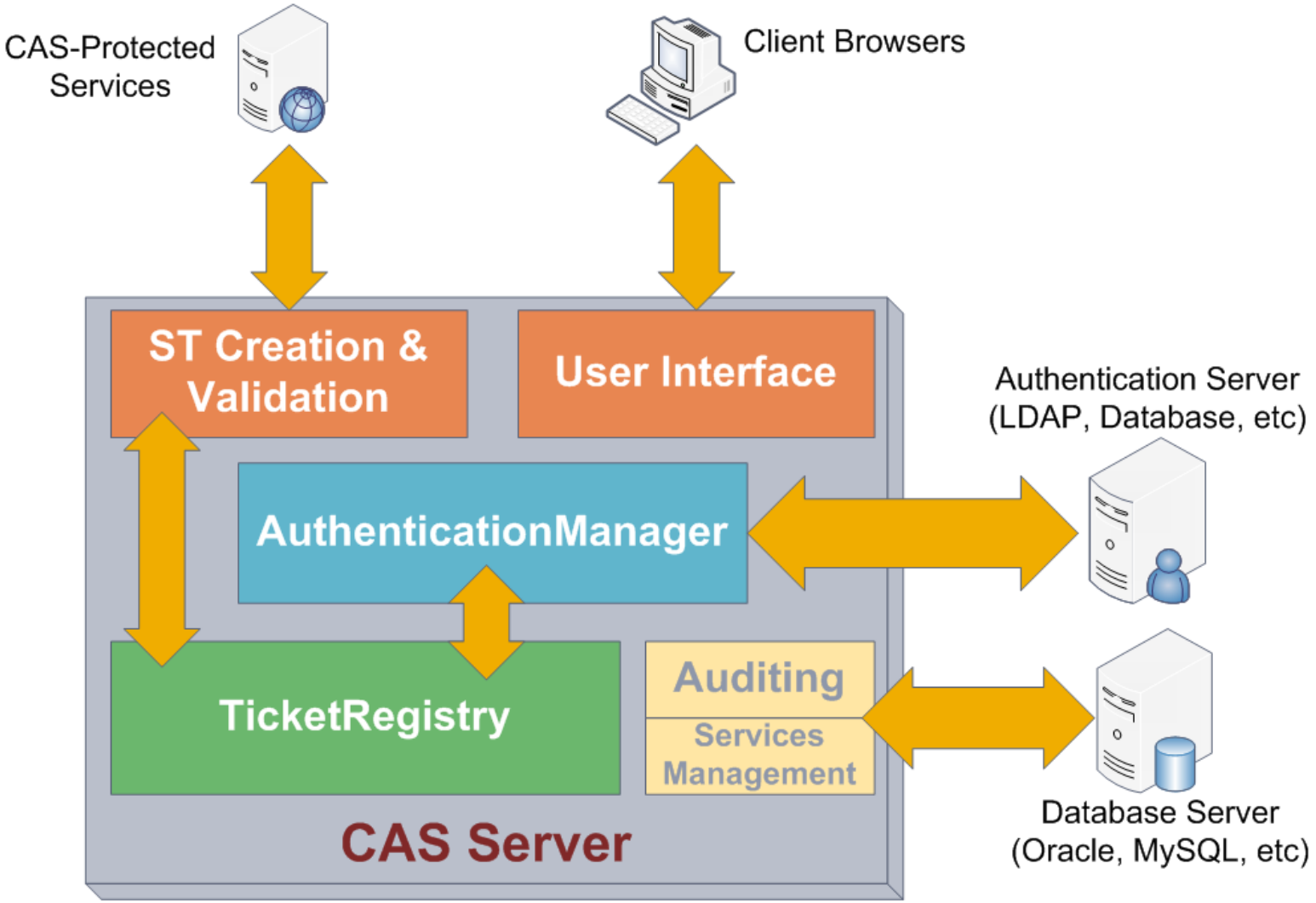
Why Cluster CAS?

- CAS is the central hub to all your web applications
- Without CAS, no one can use any applications
- A single machine is not enough

Before Clustering CAS

- CAS Architecture
- Authentication
- Service Management and Auditing

A Single CAS Server



Before Clustering CAS

Authentication Source

- Active Directory
  - Multiple Domain Controllers
- LDAP replication
  - Multi-Master replication
- Kerberos
  - JAAS can query multiple KDCs
- Database
  - Replication abilities product-specific

Before Clustering CAS

- Service Management
  - Storage Options
    - Database
    - LDAP
  - Service Registry is reloaded on all cluster nodes on a regular basis (since 3.3.4)
- Auditing & Statistics
  - Storage Options
    - Database
    - Local File

Both are optional, but recommended for production

CAS Failover with Heartbeat

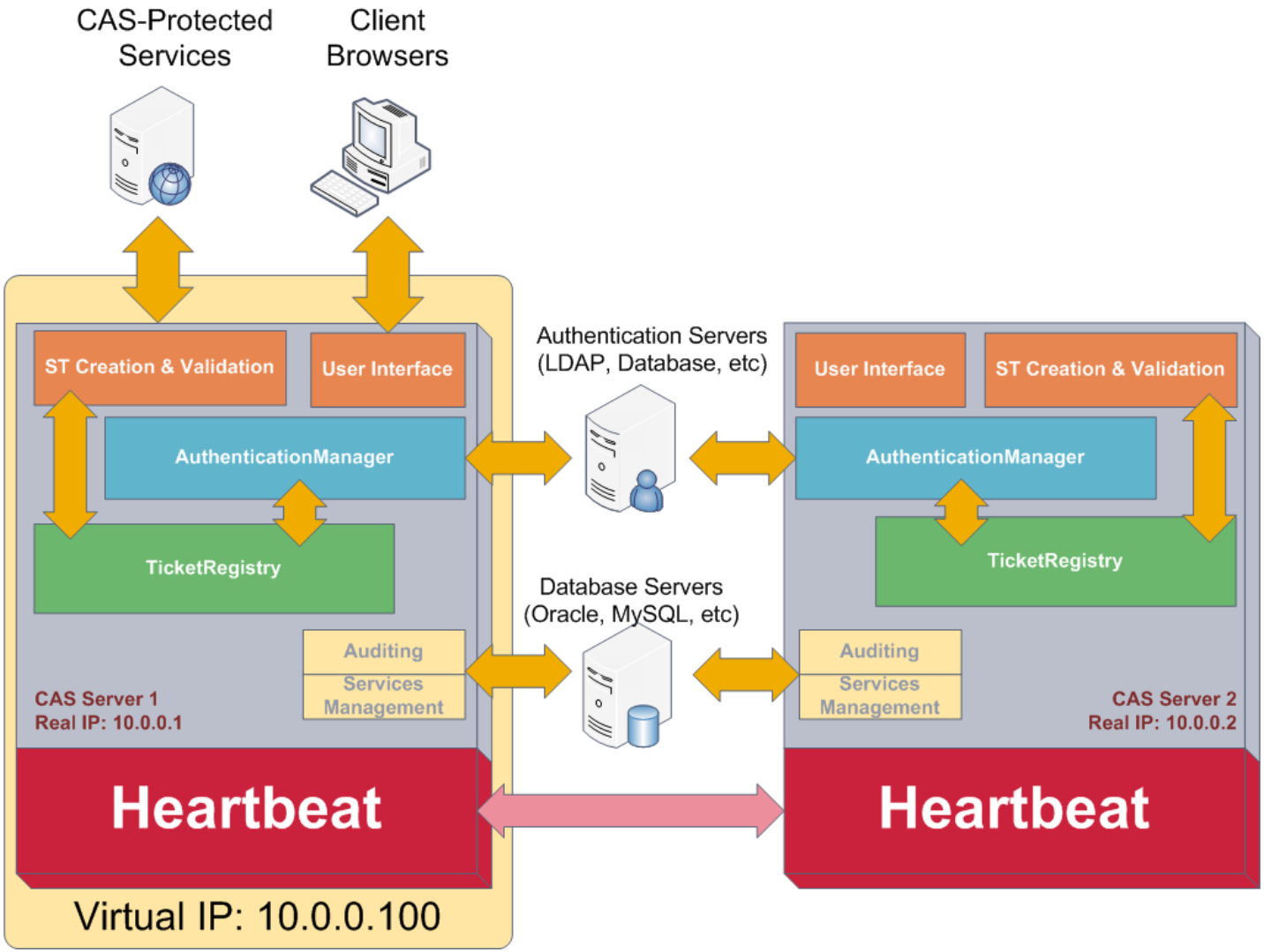
- Heartbeat
- Failover versus Load Balancing

Heartbeat

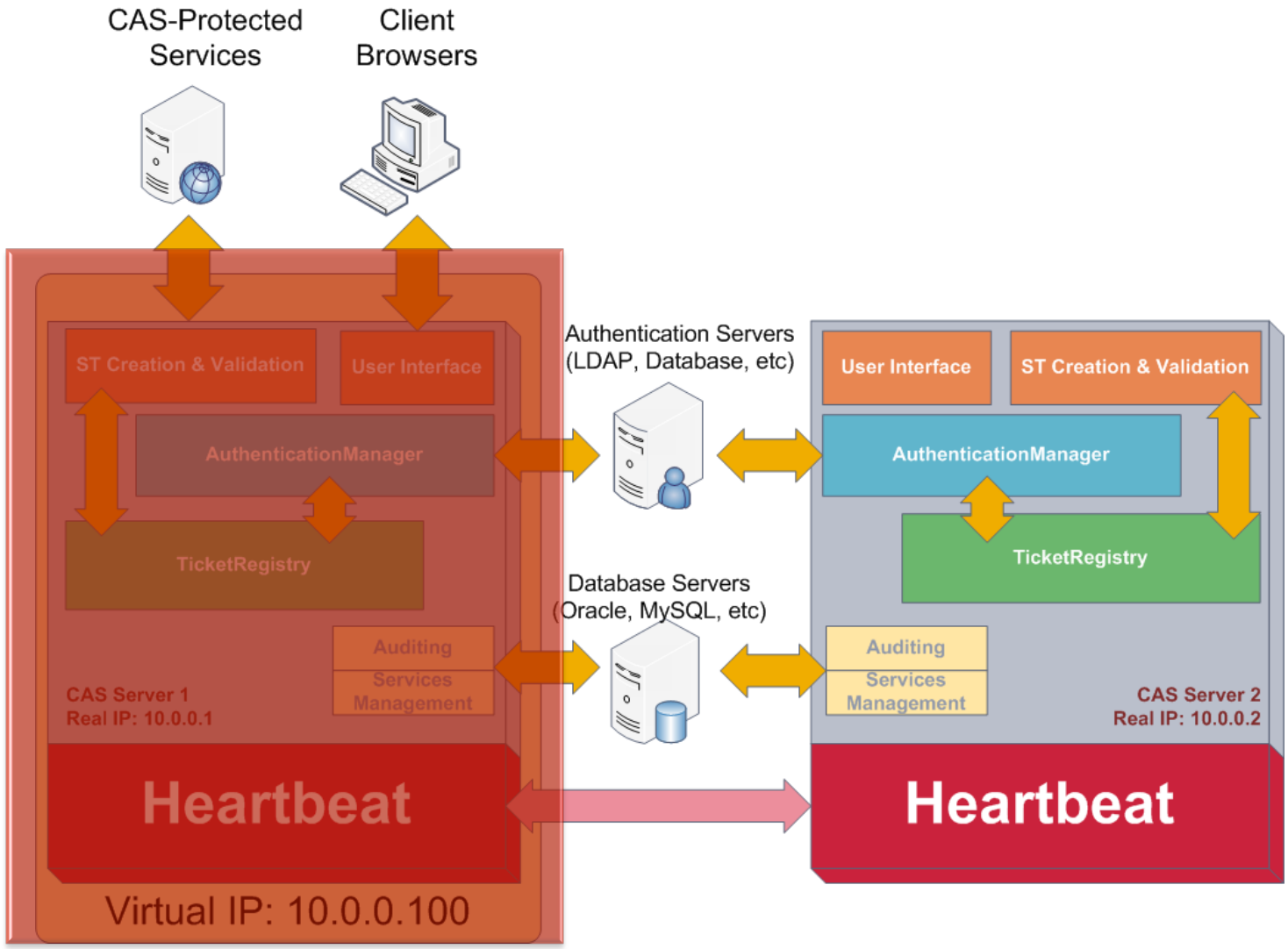
http://www.linux-ha.org

- Part of the Linux-HA Project
  - Runs on most Unix-based Operating Systems
- Provides communication layer between cluster nodes
- Sends regular 'heartbeat' between nodes to test health
- Cluster Resource Manager handles starting/stopping resources
- CRM from Heartbeat has spun-off to a separate project:
  - Pacemaker - http://clusterlabs.org

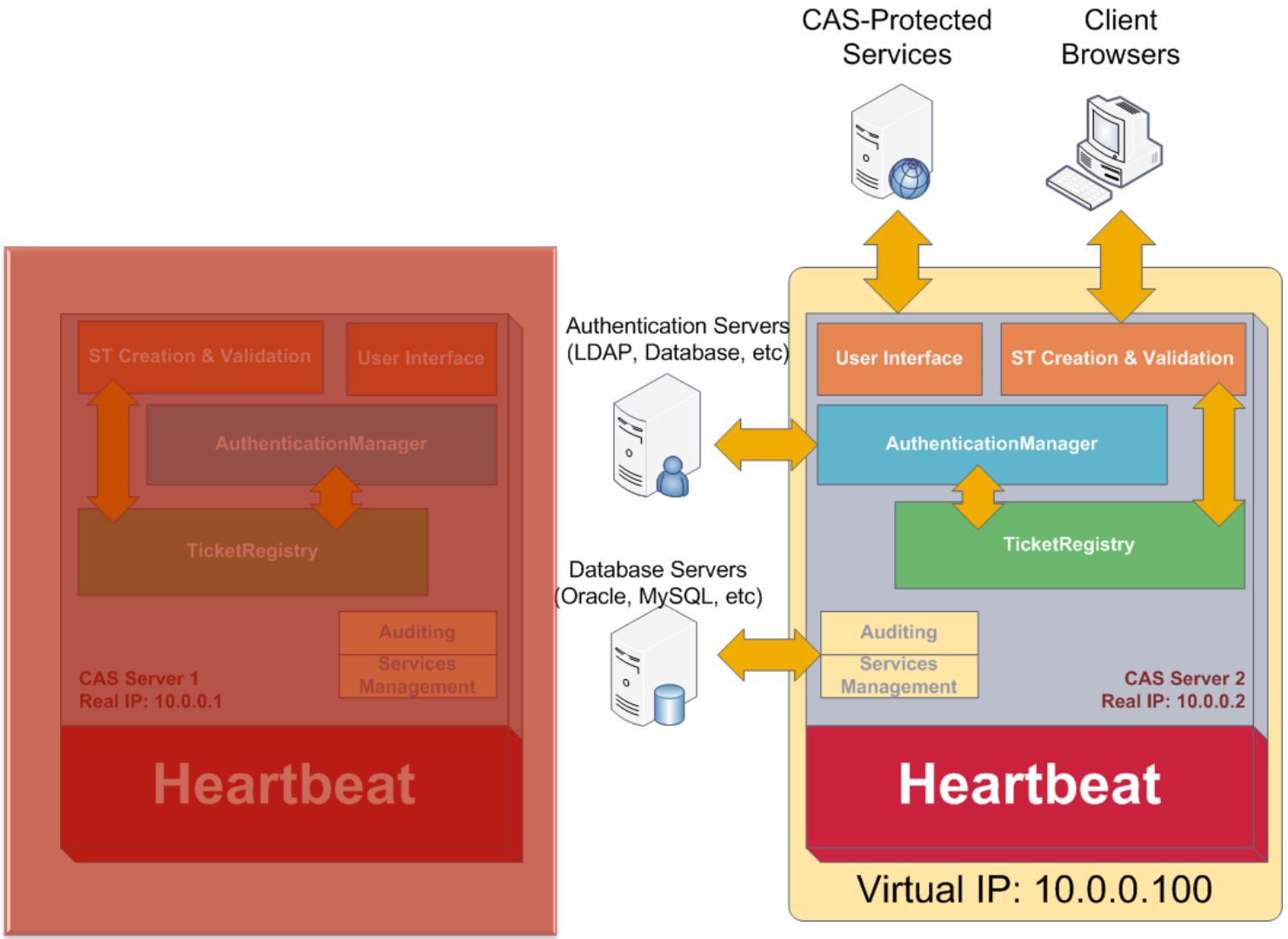
CAS failover with Heartbeat



CAS failover with Heartbeat



CAS failover with Heartbeat

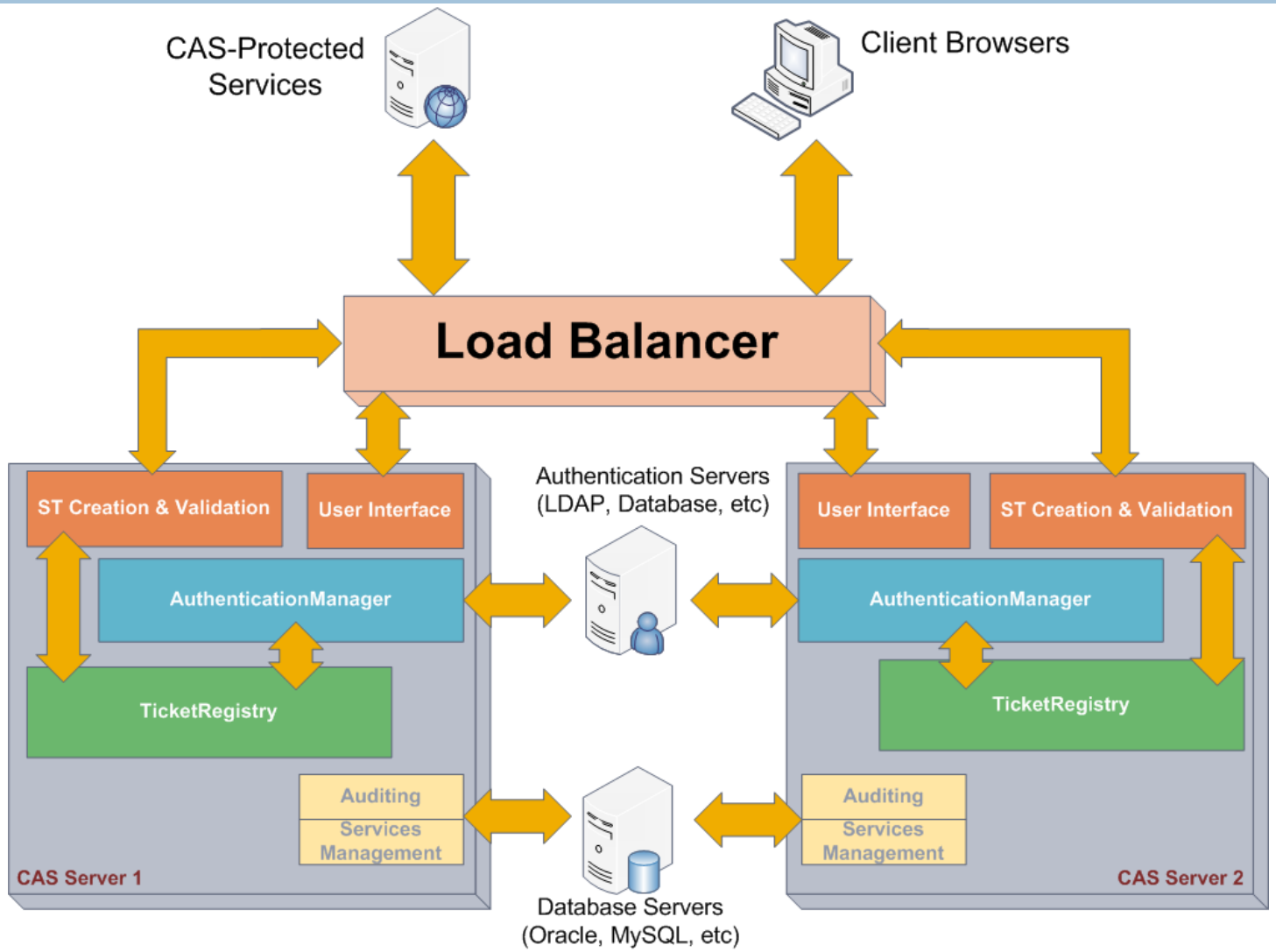


Pros & Cons of Failover

- Very easy to configure
  - Linux distros include all you need
  - GUI and CLI clients for setup & management
- No changes to CAS configuration required

- User Experience
  - All TGTs & STs are lost on failover
  - Users must re-authenticate after failover
- Wasted Resources
  - If both servers are up, one is totally idle

Load balancing to the rescue?



Load balancing to the rescue?

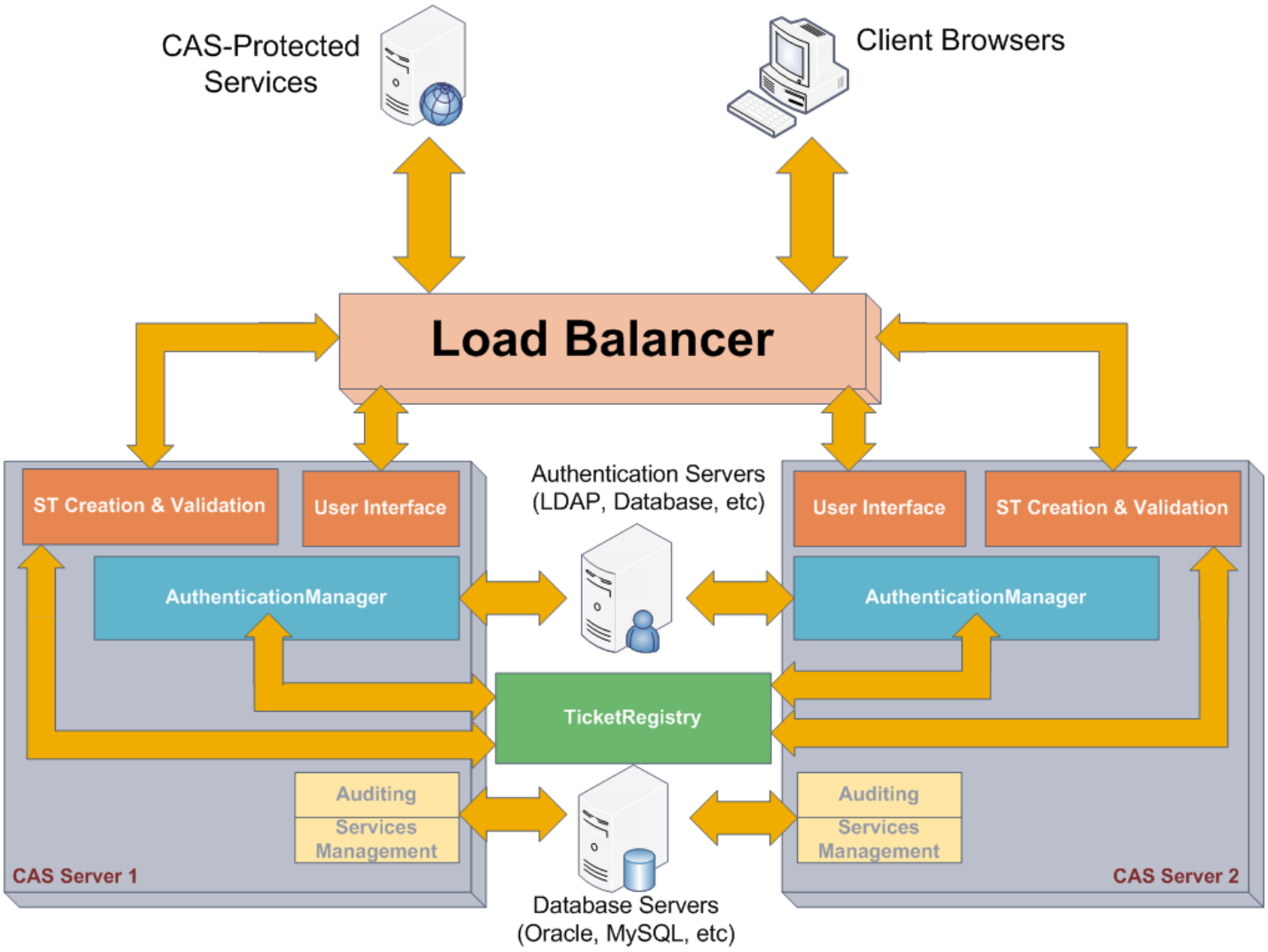
- Resource Usage improves
  - Both servers are now utilized 100% of the time
  - Hardware SSL on the LB *might* improve performance

- User Experience is ***worse***
  - Half (on average) of all ticket verifications fail
  - The TicketRegistry is not shared between servers

Shared Ticket Registry

- JBOSS Cache
- Memcached
- Java Persistence API

Shared Ticket Registry

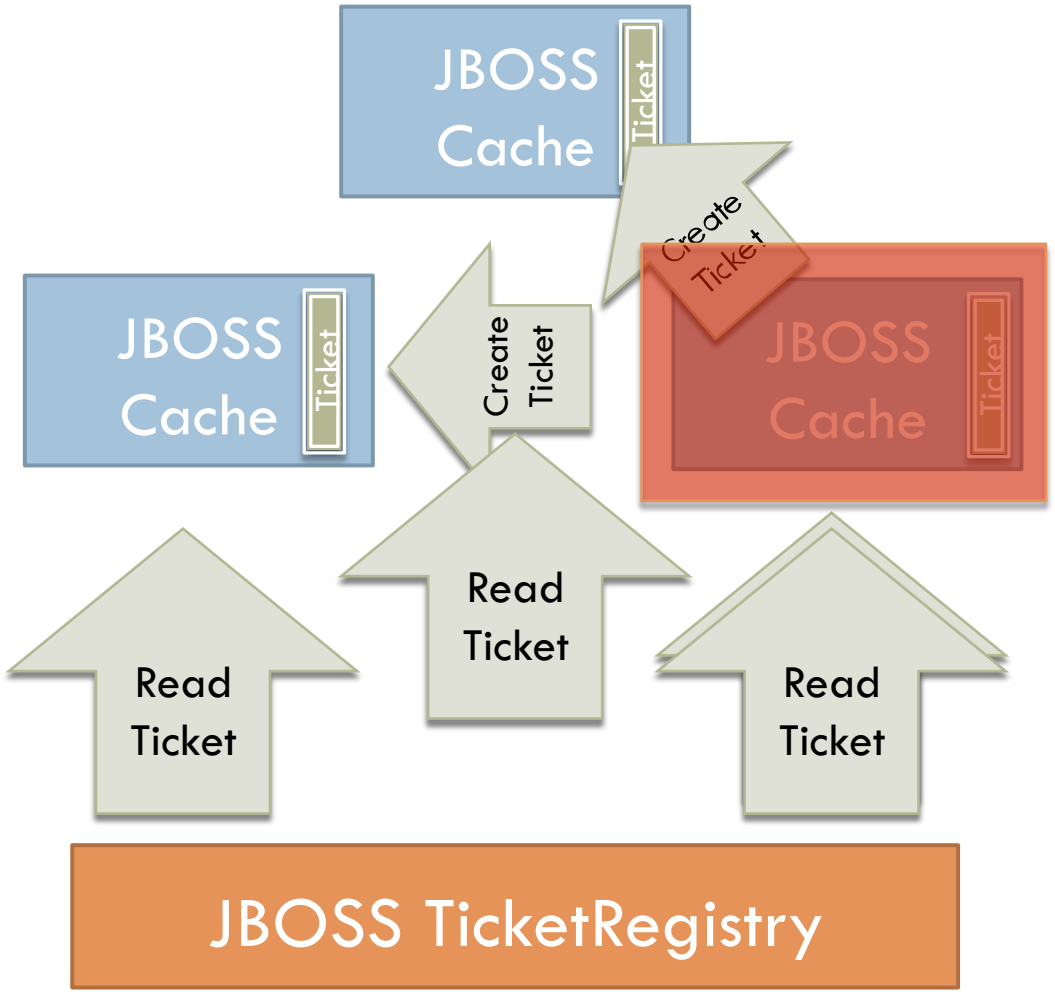


JBOSS Cache http://jboss.org/jbosscache

- Clustered cache service
- Distributes cache changes using JGroups
- Cache storage is *not* persistent in default config
  - JDBC and flat-file storage available for persistence
- Details on setting up `JBossCacheTicketRegistry` are available at the Jasig Wiki:

http://www.ja-sig.org/wiki/display/CASUM/Clustering+CAS

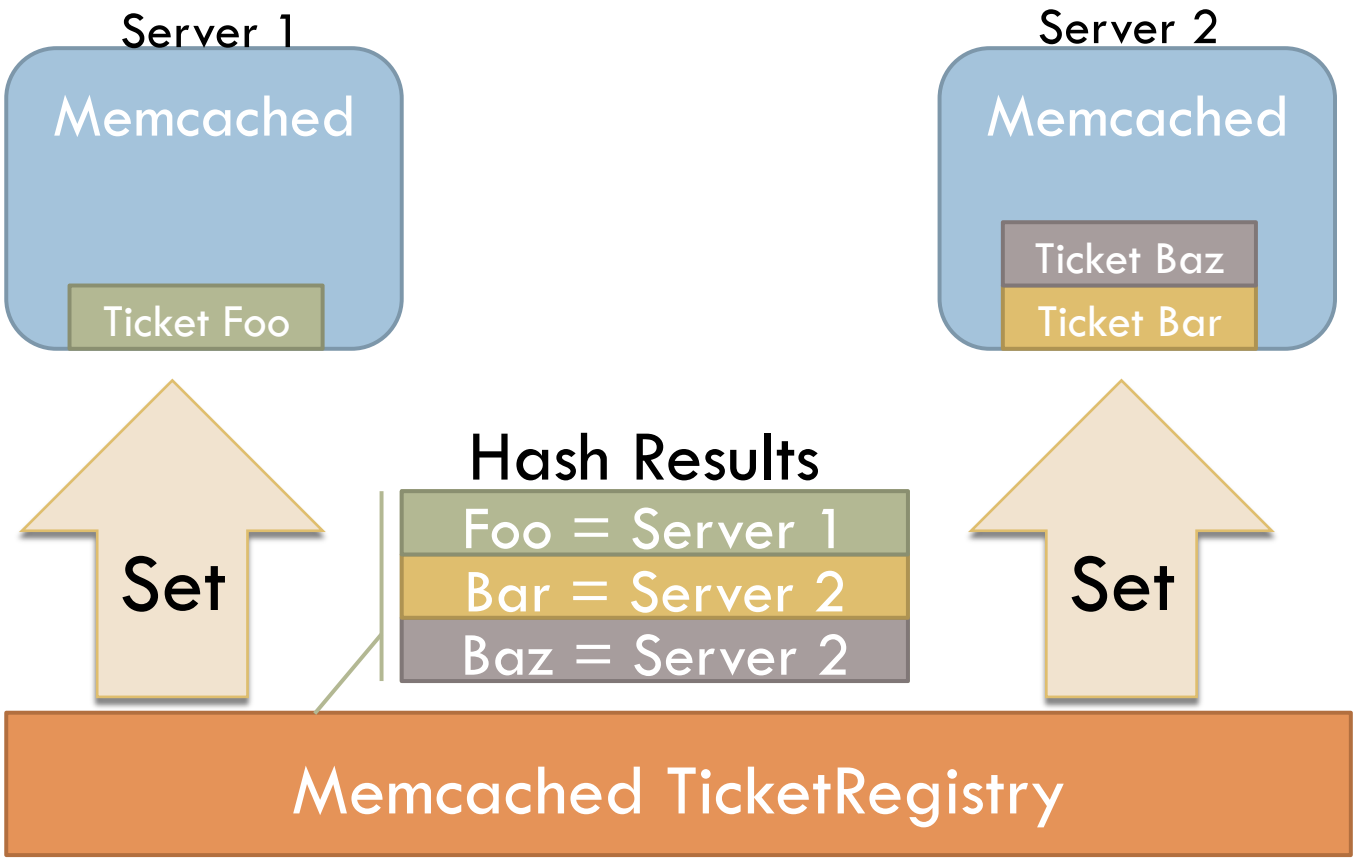
JBOSS Cache


JBOSS Cache
Ticket
Create Ticket
JBOSS Cache
Ticket
Create Ticket
JBOSS Cache
Ticket
Read Ticket
Read Ticket
Read Ticket
JBOSS TicketRegistry

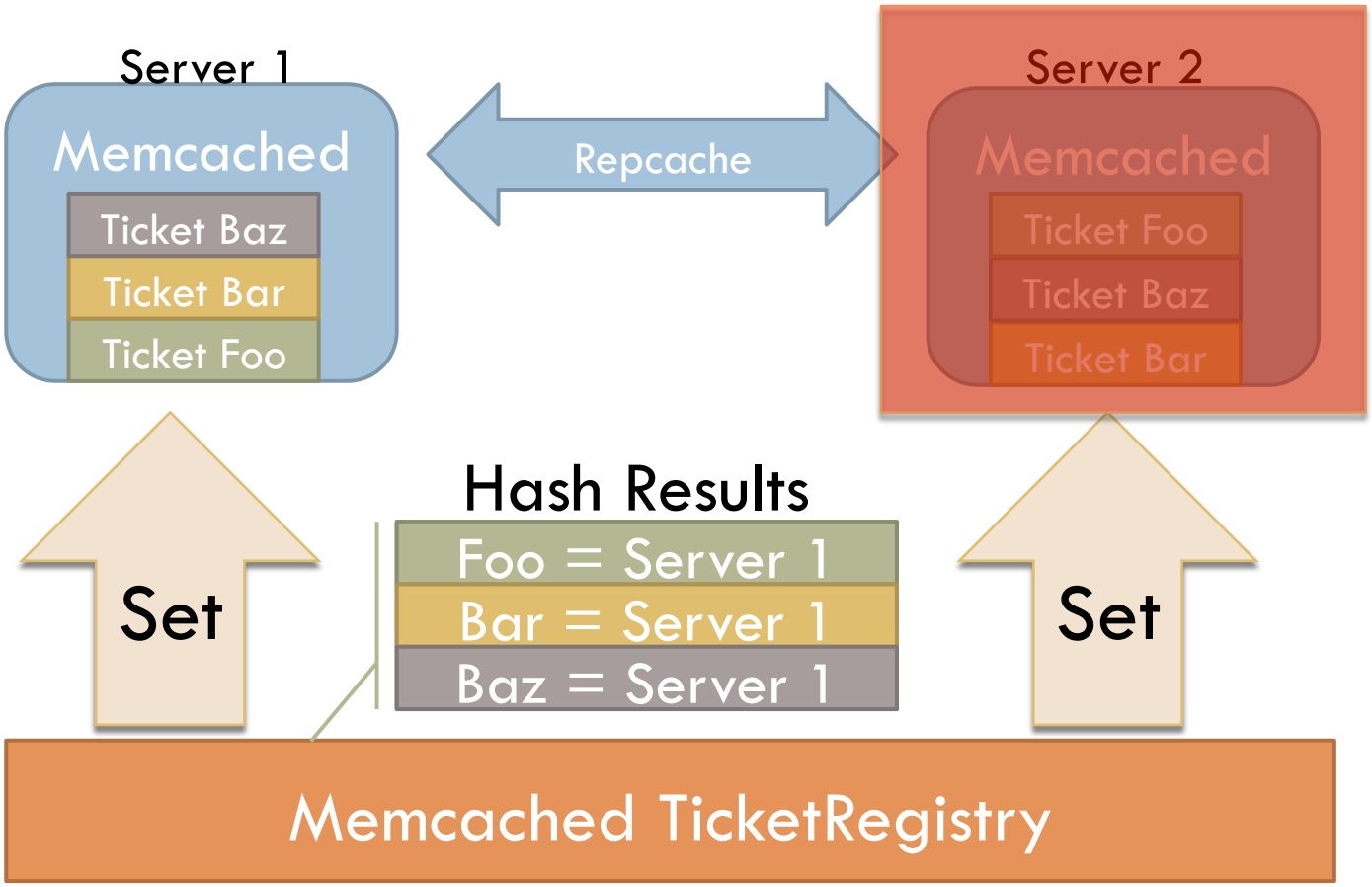
Memcached http://memcached.org

- Distributed caching system
- Hashing algorithm selects which node to store data on
- Cache is stored in memory
  - Cache storage is *not* persistent
  - Oldest objects are removed when cache is filled
- Simple, lightweight and fast
- Repcached patch adds 2-server data replication
  - http://repcached.lab.klab.org/
  - Project stagnate?

Memcached


Server 1
Memcached
Ticket Foo
Server 2
Memcached
Ticket Baz
Ticket Bar
Hash Results
Foo = Server 1
Bar = Server 2
Baz = Server 2
Set
Set
Memcached TicketRegistry

Memcached with Repcache



JPA Ticket Registry

- Tickets are stored in a database
  - Storage *is* persistent
  - Database HA is a necessity!
- Performance is can be very good
  - Dependant on the speed of the db configuration
- Registry Cleaning
  - Deadlocks have been an issue with the default cleaner
  - CAS 3.4 introduces LockingStrategy

JdbcLockingStrategy

- Cleaner attempts to ensure exclusive access to the DB before removing any expired tickets
- Uses a database table to hold lock state
- Only one node can clean the registry at a time
- Lock can be set by any node after expiration time

Which one should I use?

- JBoss Cache
  - Very flexible but complicated
  - Good option for clusters >2 nodes
- Memcached
  - Easiest option for a 2-node cluster
  - Status of repcache project is a concern
- JPA
  - Best data integrity/reliability
  - Obvious choice if you already have an HA database
  - Best choice for very long ticket lifetimes (Remember me)
  - Needs CAS 3.3.4 or newer (3.4 would be best)

Load Balancing

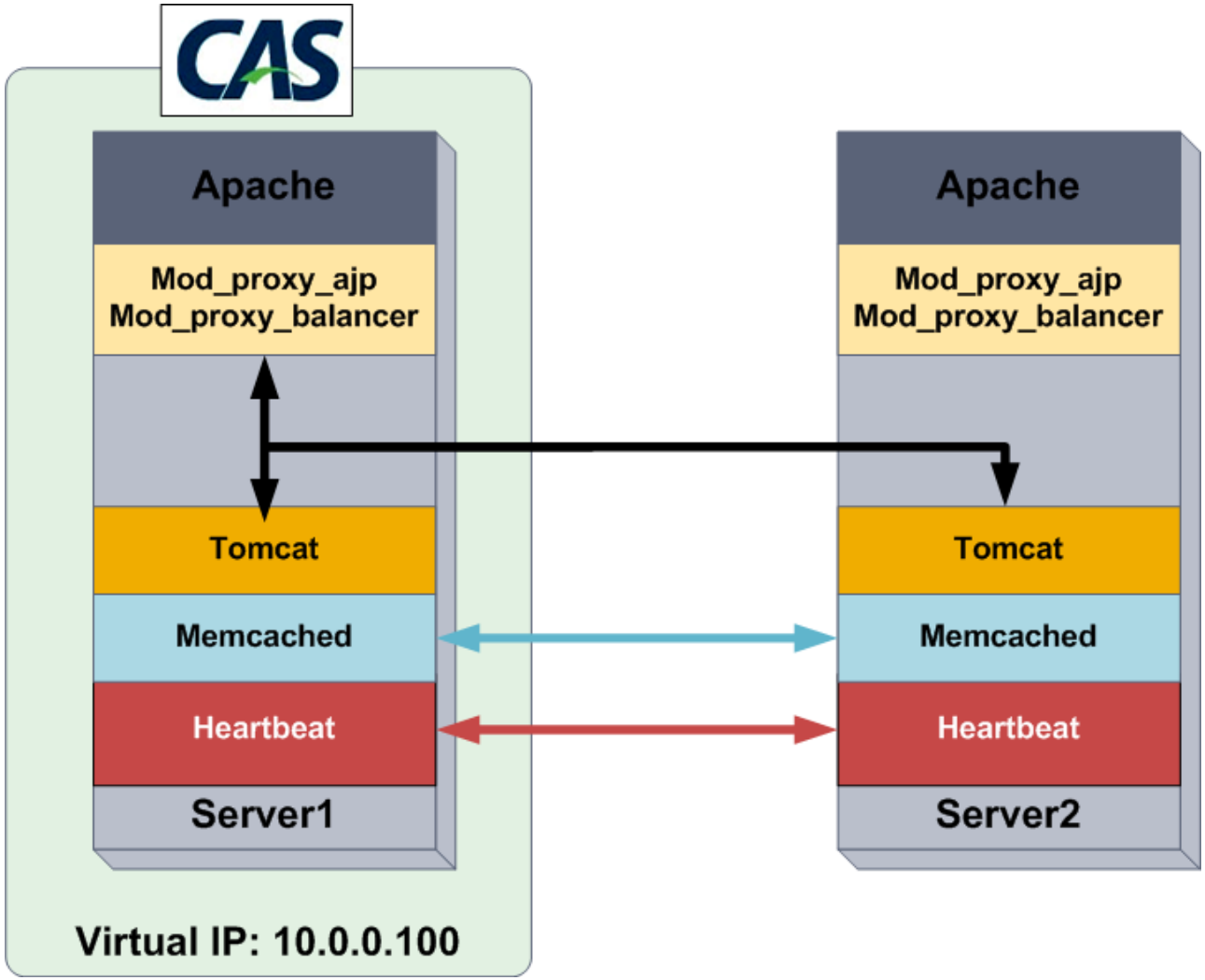
- Load Balancing with Free software
- Hardware vs. Software Load Balancing
- N-to-N Cluster

Software Load Balancing

- Combination of Apache modules
  - mod_proxy_ajp
  - mod_proxy_balancer
- Simple to configure:

```
ProxyPass /cas balancer://mycluster
<Proxy balancer://mycluster>
        BalancerMember ajp://server1:8009/cas
        BalancerMember ajp://server2:8009/cas
</Proxy>
```

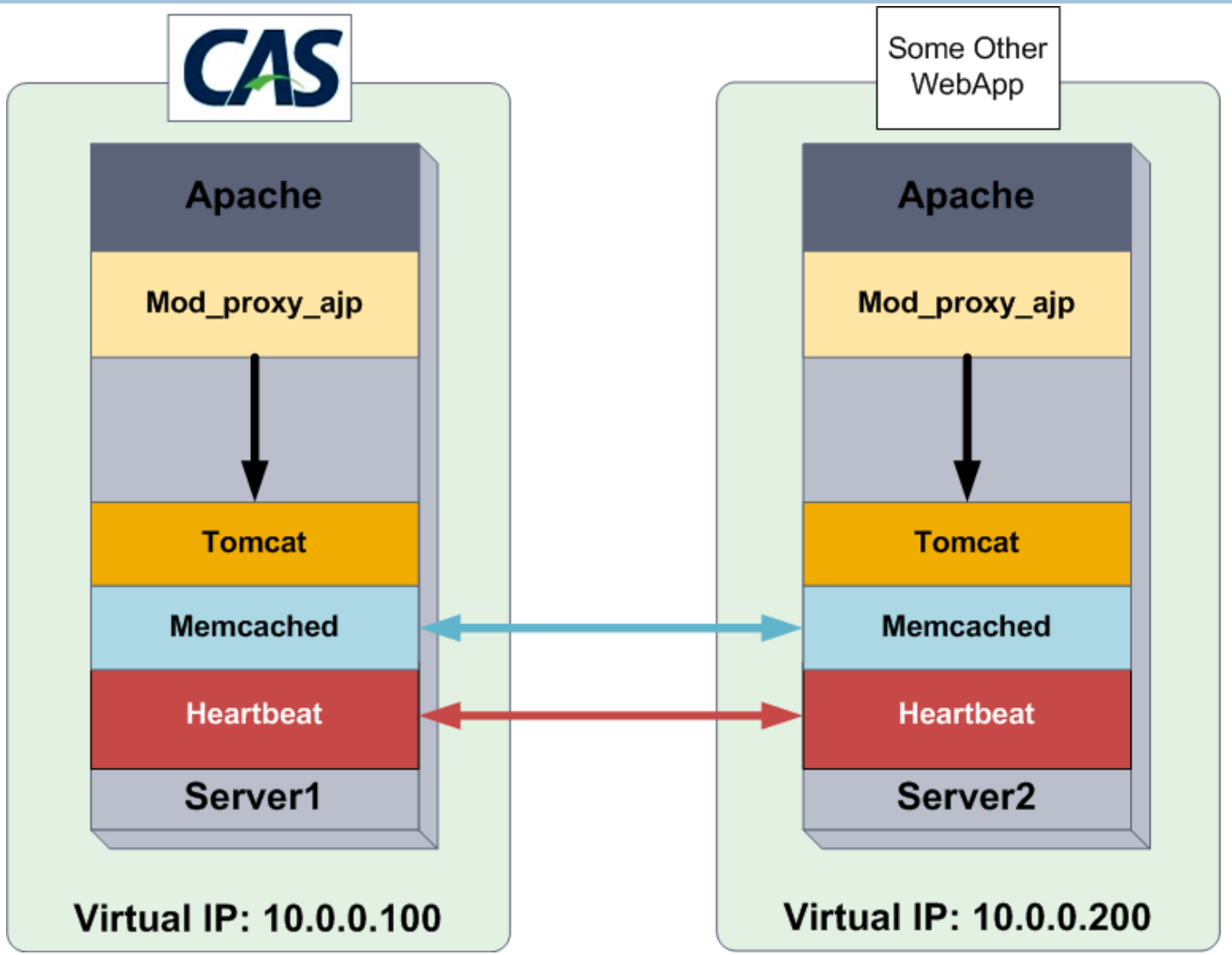
Software Load Balancing



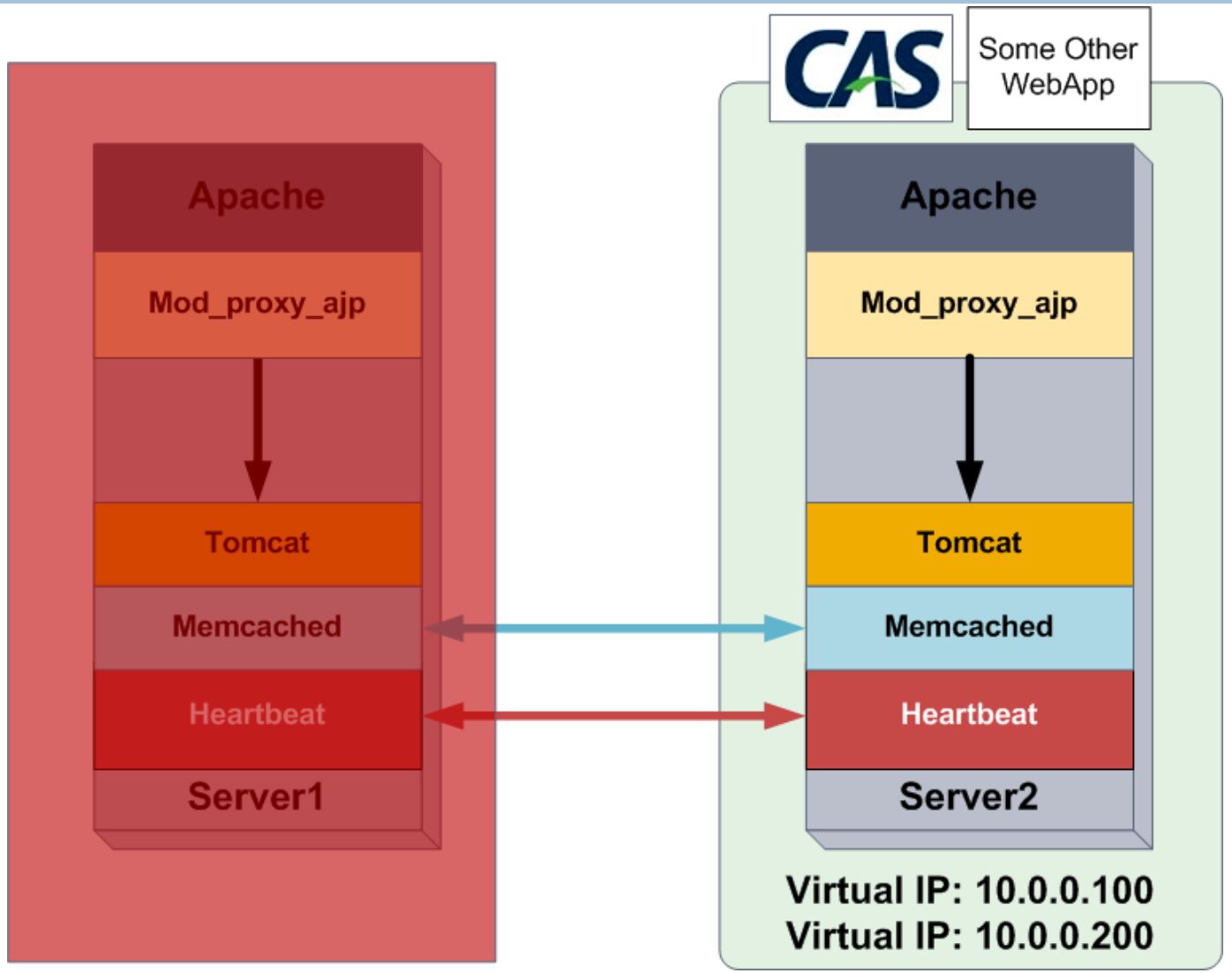
Hardware vs. Software LB

- Hardware
  - High Performance
  - SSL off-load
  - Can be expensive
  - Need multiple devices for HA
- Software
  - Free (as in Speech & Beer)
  - Very configurable

N-to-N Cluster



N-to-N Cluster



Tomcat Sessions

- CAS Clustering wiki page recommends session replication
- You don't need it
  - Adds complexity
  - Session is only used for storing the webflow state
- Change WEB-INF/cas-servlet.xml:

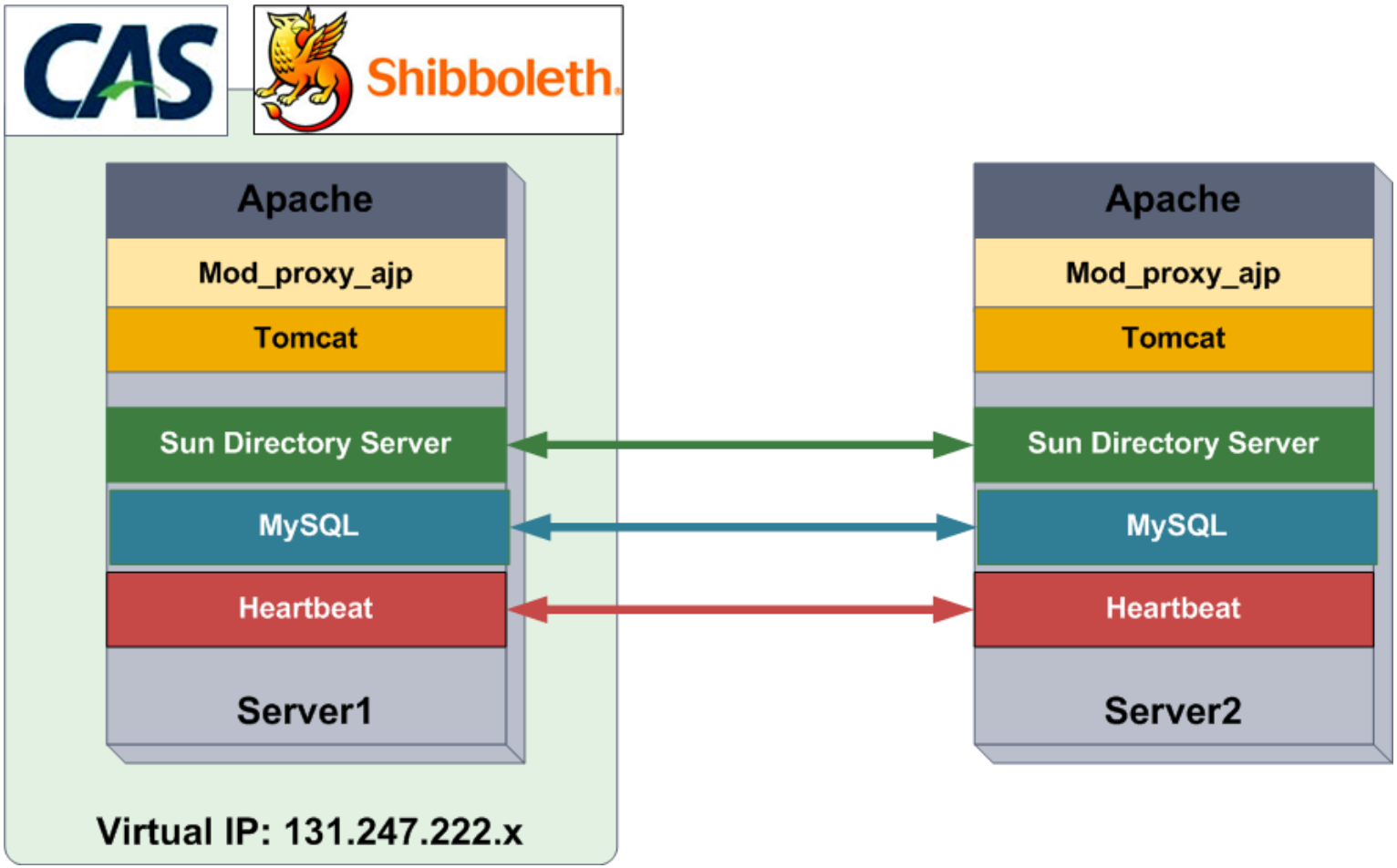
```
<flow:executor id="flowExecutor" registry-
  ref="flowRegistry" repository-type="client">
```

CAS at USF

USF CAS Cluster (v1)

- In service Feb. 2008 – Oct. 2009
- Failover Cluster using Heartbeat
- Default (non-shared) Ticket Registry
- Apache/Tomcat shared by CAS and Shibboleth IdP
- Service Registry & Auditing use MySQL
  - Master-Master Replication

USF CAS Cluster (v1)

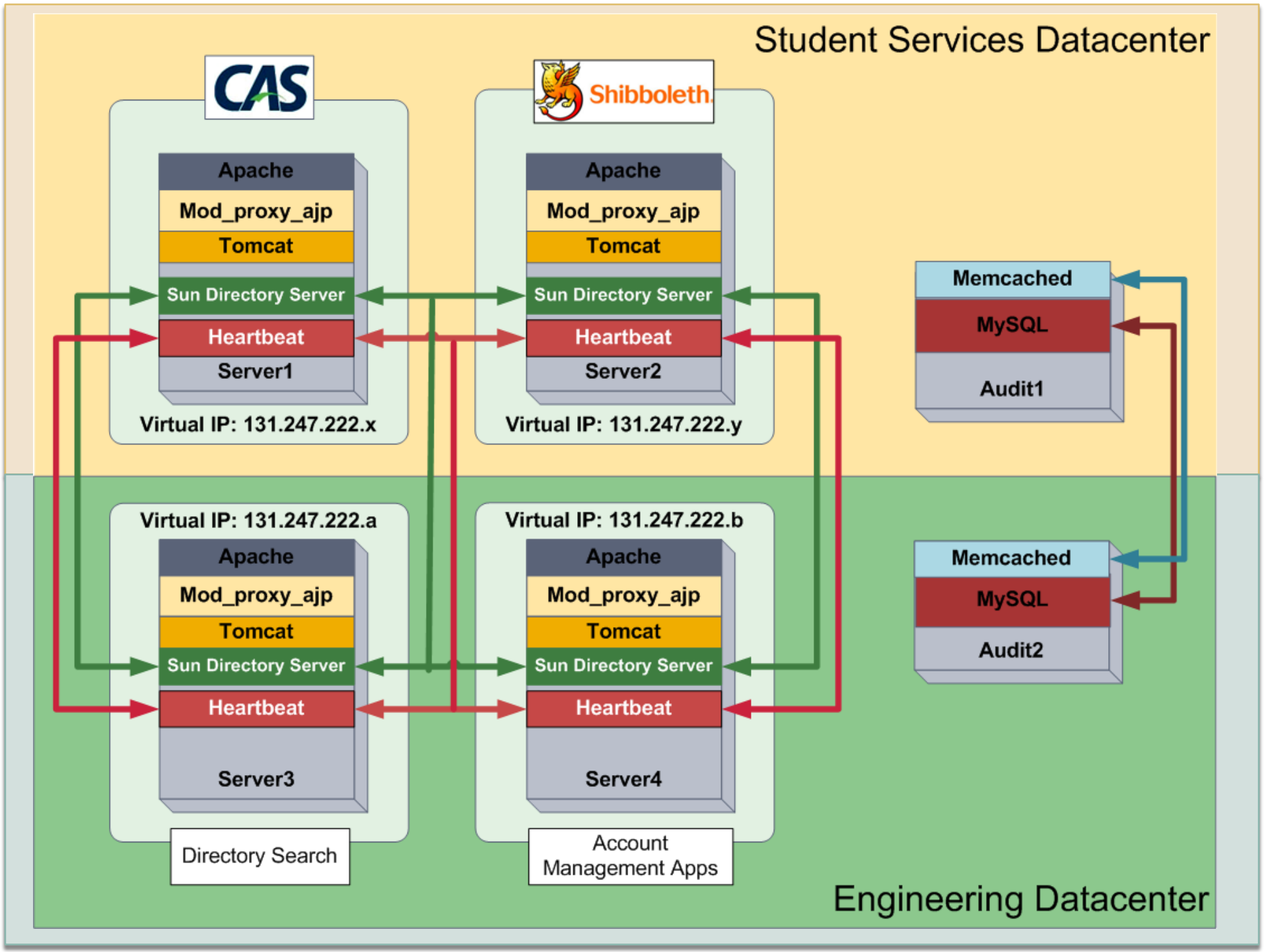


Problems with version1

- Location
  - Servers were in the same (poorly outfitted) server room
- Performance
  - During high-load, CAS & Shibboleth were a bit slow
- User Experience
  - All tickets were lost on failover, forcing users to login again

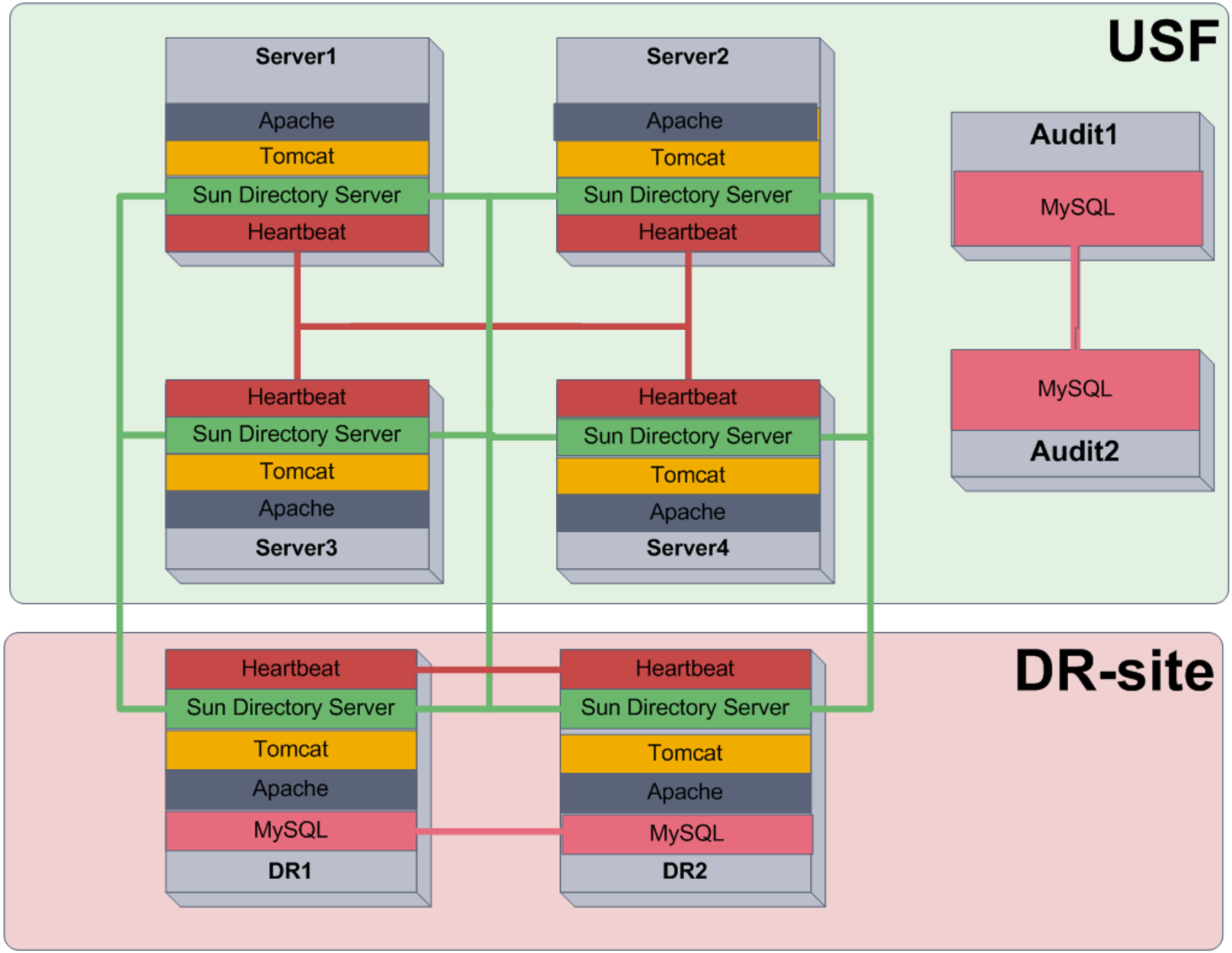
USF CAS Cluster (v2)

- In production since Oct. 2009
- 4-node N-to-N Cluster using Heartbeat/Pacemaker
- Geographically separated (~1KM apart)
- Memcached Ticket Registry (Repcache)
- CAS, Shibboleth and other webapps have 'dedicated' machines
- Service Registry & Auditing on dedicated hardware


Student Services Datacenter
CAS
Shibboleth
Apache
Mod_proxy_ajp
Tomcat
Sun Directory Server
Heartbeat
Server1
Virtual IP: 131.247.222.x
Apache
Mod_proxy_ajp
Tomcat
Sun Directory Server
Heartbeat
Server2
Virtual IP: 131.247.222.y
Memcached
MySQL
Audit1
Virtual IP: 131.247.222.a
Apache
Mod_proxy_ajp
Tomcat
Sun Directory Server
Heartbeat
Server3
Directory Search
Virtual IP: 131.247.222.b
Apache
Mod_proxy_ajp
Tomcat
Sun Directory Server
Heartbeat
Server4
Account Management Apps
Memcached
MySQL
Audit2
Engineering Datacenter

Future Additions

- Hardware Load Balancing
- Off-campus Disaster-Recovery site
  - Currently in Tallahassee
  - Moving it farther North
- Persistent Ticket Storage
  - ‘Remember Me’ function is highly requested
  - JPA or JBOSS Cache with persistent storage


USF
Server1
Apache
Tomcat
Sun Directory Server
Heartbeat
Server2
Apache
Tomcat
Sun Directory Server
Heartbeat
Audit1
MySQL
MySQL
Audit2
Heartbeat
Sun Directory Server
Tomcat
Apache
Server3
Heartbeat
Sun Directory Server
Tomcat
Apache
Server4
DR-site
Heartbeat
Sun Directory Server
Tomcat
Apache
MySQL
DR1
Heartbeat
Sun Directory Server
Tomcat
Apache
MySQL
DR2

Questions?

ERIC PIERCE
epierce@usf.edu