Inside LiveJournal's Backend

or,

“holy hell that's a lot of hits!”

April 2004

Brad Fitzpatrick

brad@danga.com

Danga Interactive

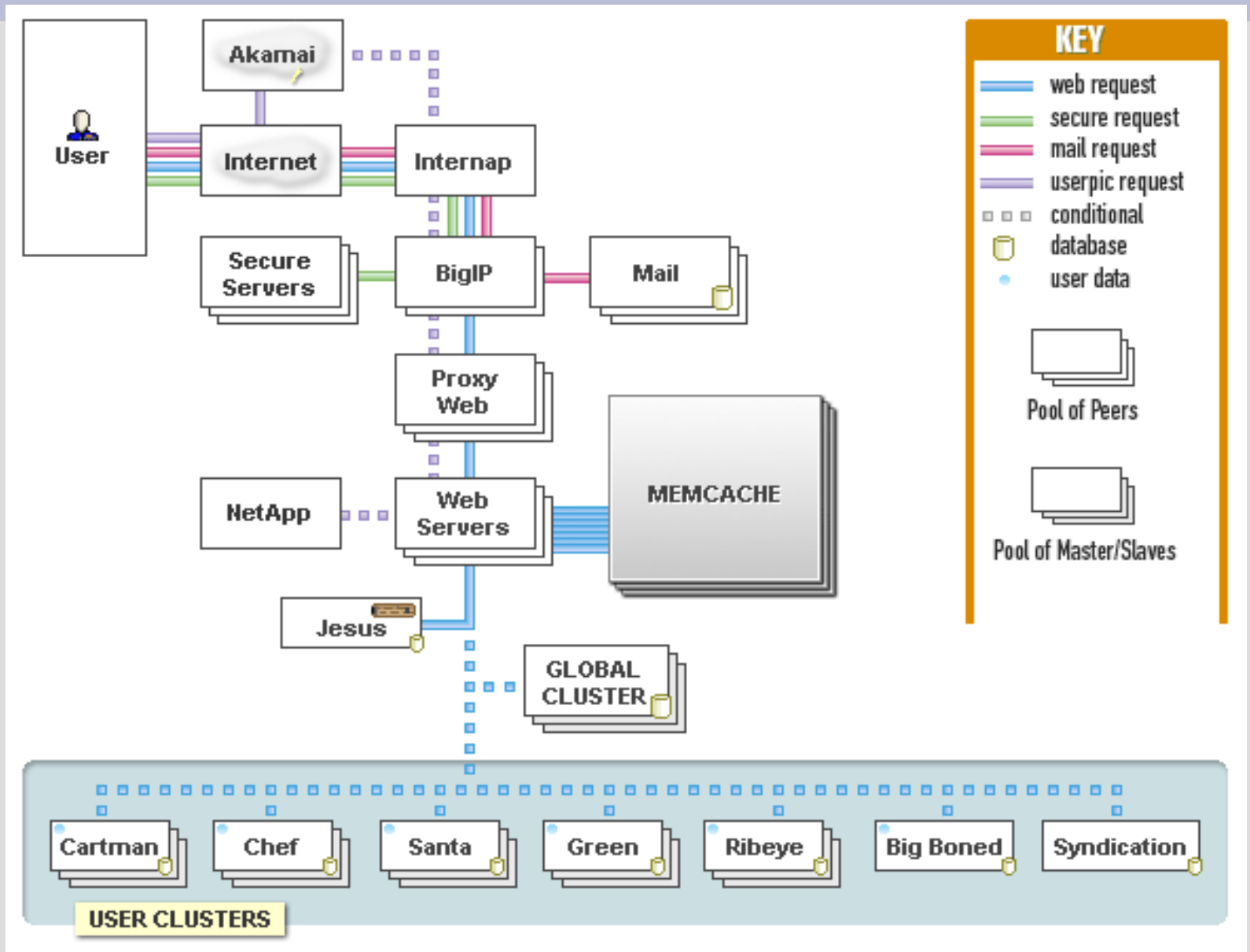
danga.com / livejournal.com

LiveJournal Overview

- college hobby project, Apr 1999
- blogging, forums
- aggregator, social-networking ('friends')
- 2.8 million accounts; ~half active
- 40-50M dynamic hits/day. 700-800/second at peak hours
- why it's interesting to you...
  - 60+ servers
  - lots of MySQL usage

LiveJournal Backend

(as of a few months ago)

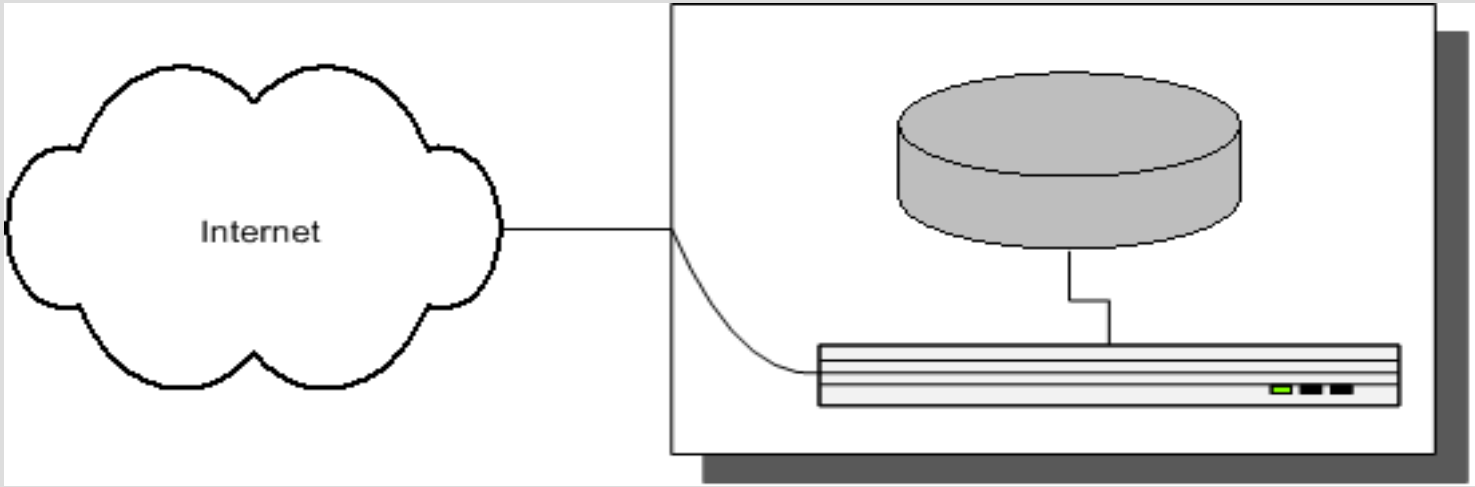


Backend Evolution

- From 1 server to 60+....
  - where it hurts
  - how to fix
- Learn from this!
  - don't repeat my mistakes
  - can implement our design on a single server

One Server

- shared server
- dedicated server (still rented)
  - still hurting, but could tune it
  - learn Unix pretty quickly (first root)
  - CGI to FastCGI
- Simple

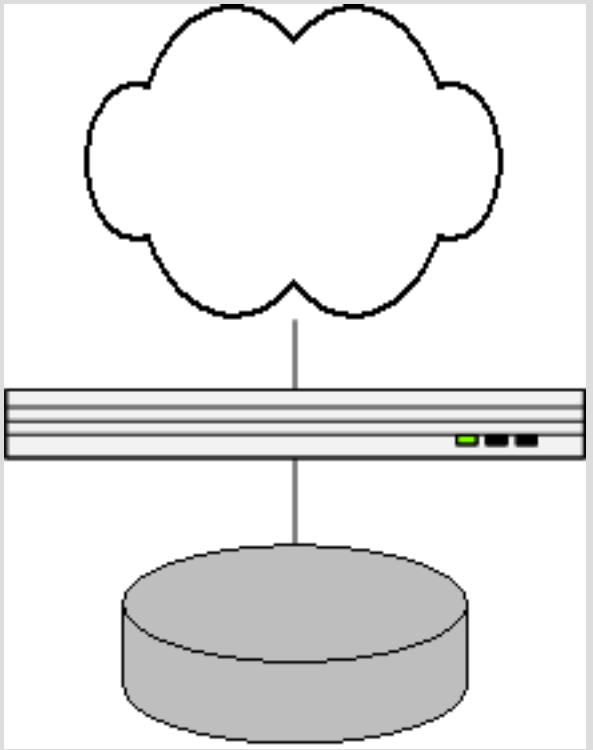


One Server - Problems

- Site gets slow eventually.
  - reach point where tuning doesn't help
- Need servers
  - start “paid accounts”

Two Servers

- Paid account revenue buys:
  - Kenny: 6U Dell web server
  - Cartman: 6U Dell database server
    - bigger / extra disks
- Network simple
  - 2 NICs each
- Cartman runs MySQL on internal network

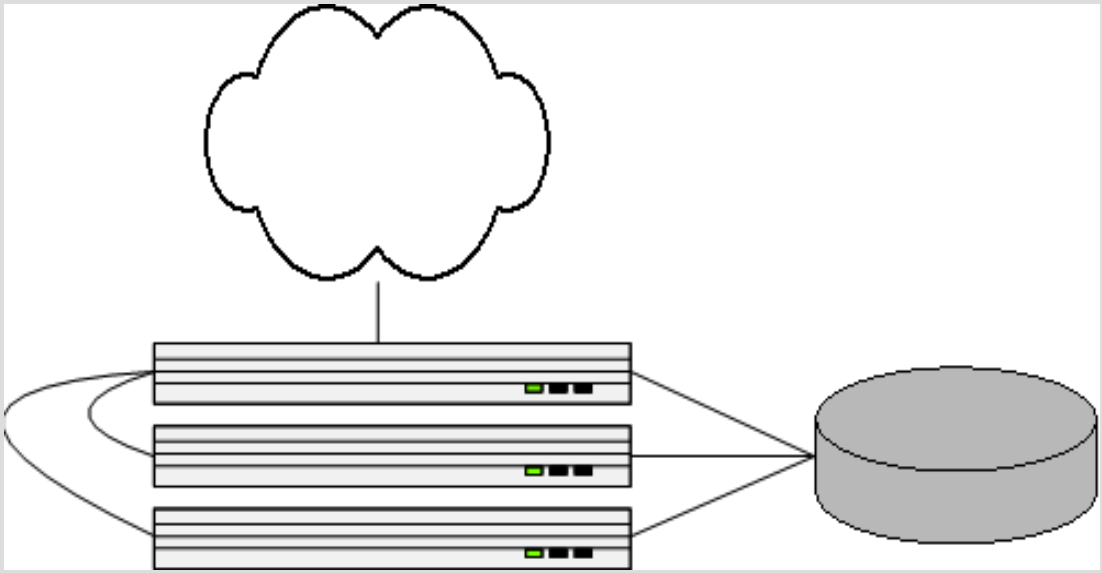


Two Servers - Problems

- Two points of failure
- No hot or cold spares
- Site gets slow again.
  - CPU-bound on web node
  - need more web nodes...

Four Servers

- Buy two more web nodes (1U this time)
  - Kyle, Stan
- Overview: 3 webs, 1 db
- Now we need to load-balance!
  - Kept Kenny as gateway to outside world
  - mod_backhand amongst 'em all



mod_backhand

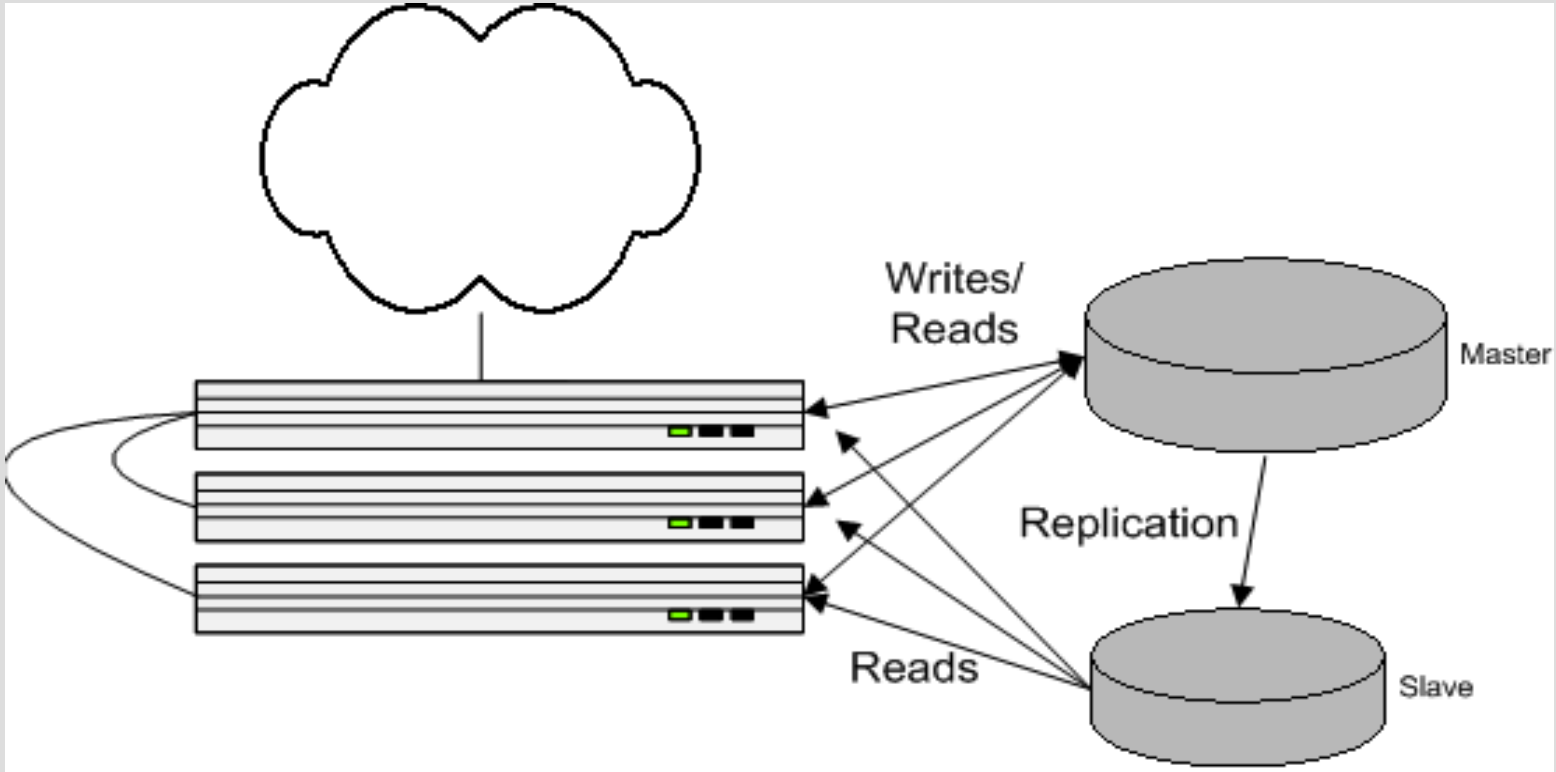
- web nodes broadcasting their state
  - free/busy apache children
  - system load
  - ...
- internally proxying requests around
  - network cheap

Four Servers - Problems

- Points of failure:
  - database
  - kenny (but could switch to another gateway easily when needed, or used heartbeat, but we didn't)
- Site gets slow...
  - IO-bound
  - need another database server ...
  - ... how to use another database?

Five Servers
introducing MySQL replication

- We buy a new database server
- MySQL replication
- Writes to Cartman (master)
- Reads from both

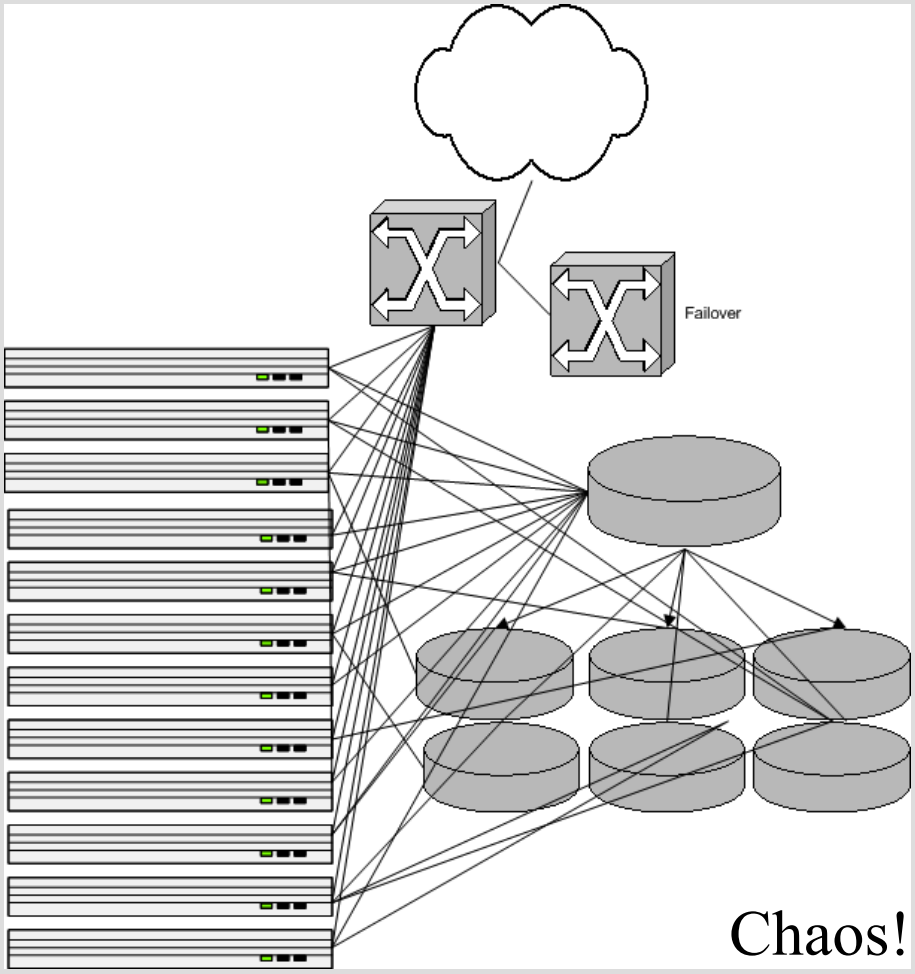


Replication Implementation

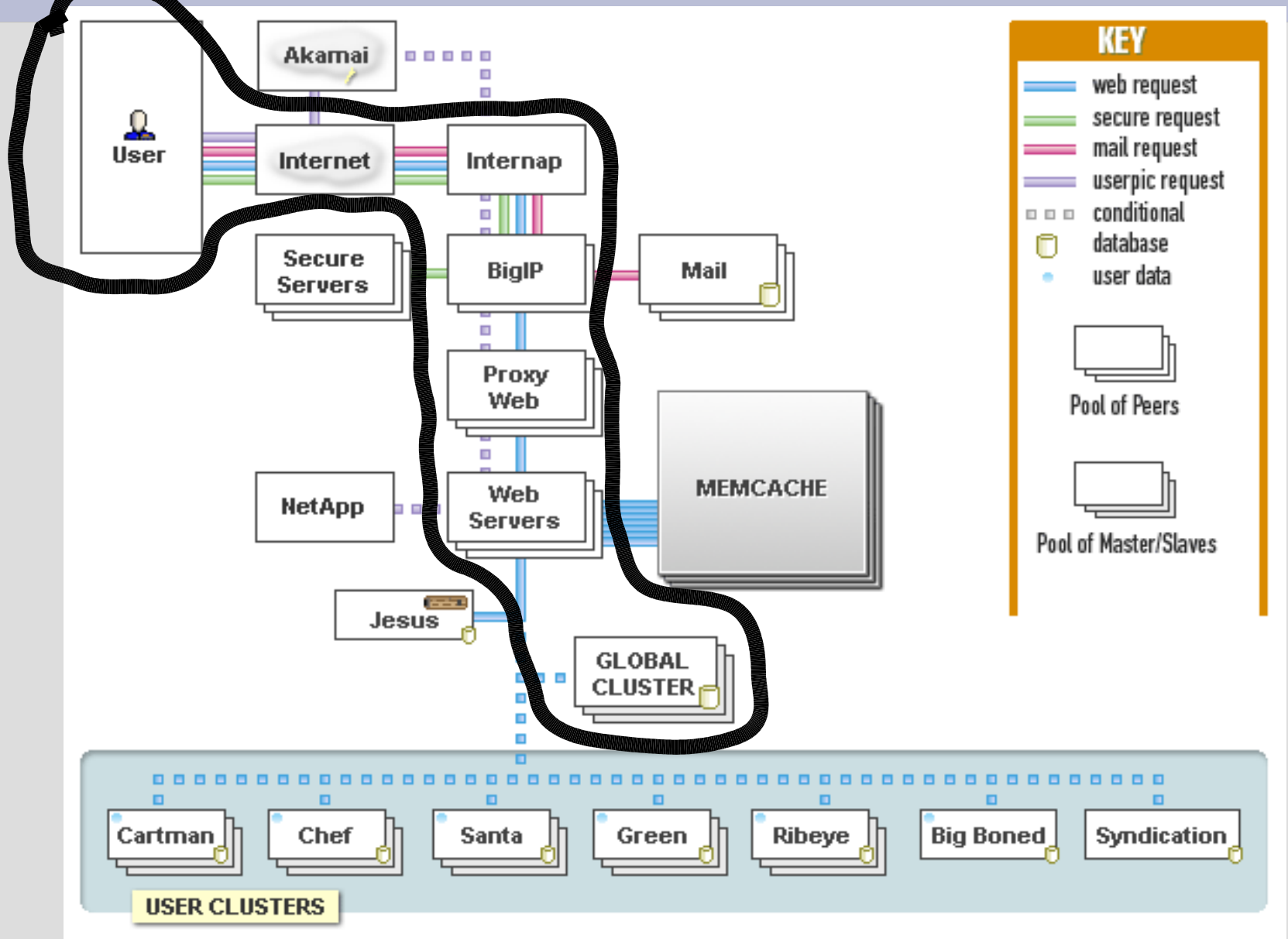
- get_db_handle() : $dbh
  - existing
- get_db_reader() : $dbr
  - transition to this
  - weighted selection
- permissions: slaves select-only
  - mysql option for this now
- be prepared for replication lag
  - easy to detect in MySQL 4.x
  - user actions from $dbh, not $dbr

More Servers

- Site's fast for a while,
- Then slow
- More web servers,
- More database slaves,
- ...
- IO vs CPU fight
- BIG-IP load balancers
  - cheap from usenet
  - two, but not automatic fail-over (no support contract)
  - LVS would work too



Where we're at...



Problems with Architecture

or,

"This don't scale..."

- Slaves upon slaves doesn't scale well...
  - only spreads reads
  - databases eventual consumed by writing
    - 1 server: 100 reads, 10 writes (10% writes)
    - Traffic doubles: 200 reads, 20 writes (10% writes)
      - imagine nearing threshold
    - 2 servers: 100 reads, 20 writes (20% writes)
- Database master is point of failure
- Reparenting slaves on master failure is tricky

Spreading Writes

- Our database machines already did RAID
- We did backups
- So why put user data on 6+ slave machines? (~12+ disks)
  - overkill redundancy
  - wasting time writing everywhere

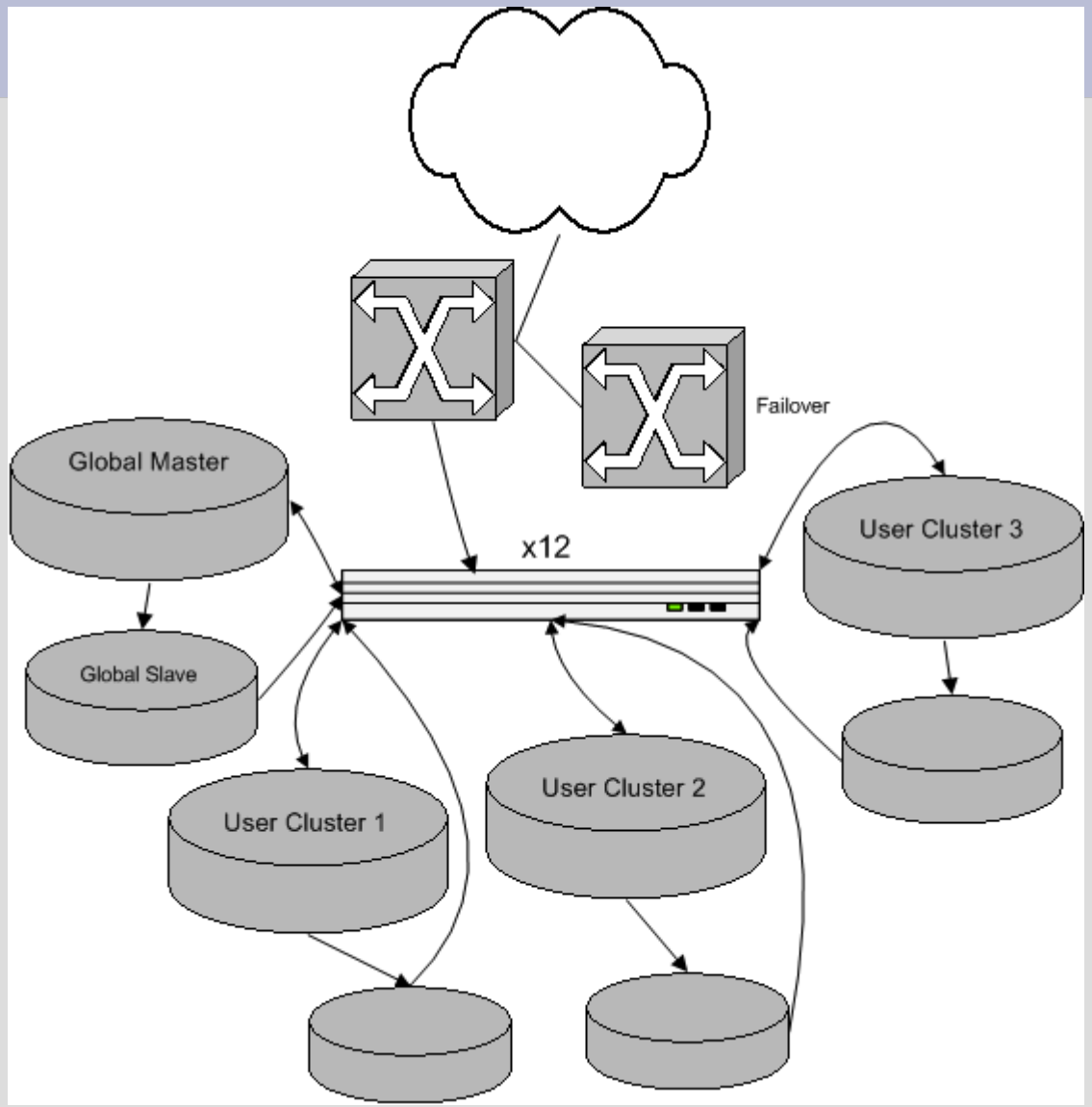
Introducing User Clusters

- Already had get_db_handle() vs get_db_reader()
- Specialized handles:
- Partition dataset
  - can't join. don't care. never join user data w/ other user data
- Each user assigned to a cluster number
- Each cluster has multiple machines
  - writes self-contained in cluster (writing to 2-3 machines, not 6)

User Cluster Implementation

- $u = LJ::load_user(“brad”)
  - hits global cluster
  - $u object contains its clusterid
- $dbcm = LJ::get_cluster_master($u)
  - writes
  - definitive reads
- $dbcr = LJ::get_cluster_reader($u)
  - reads

User Clusters



- almost resembles today's architecture

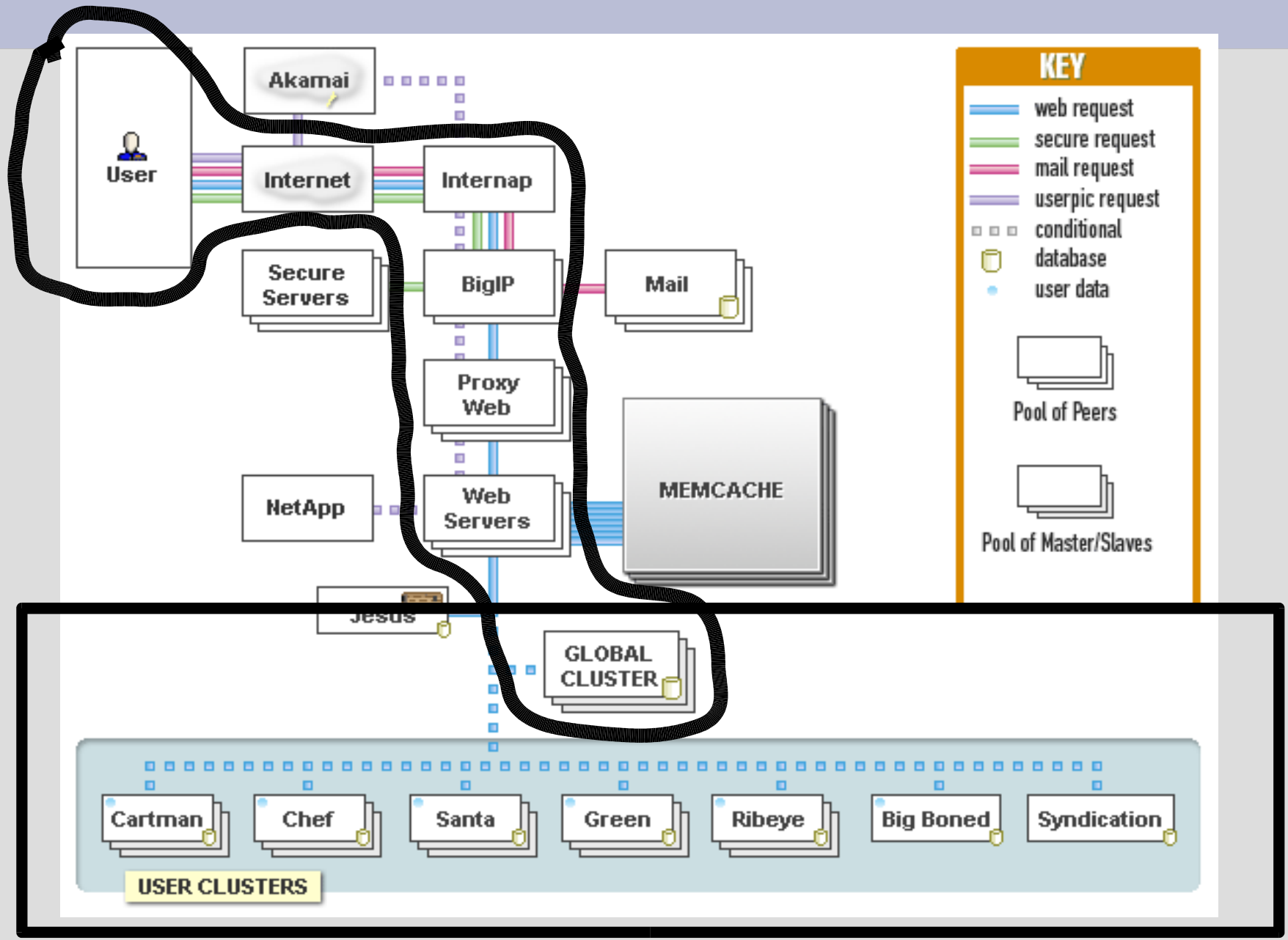
User Cluster Implementation

- per-user numberspaces
  - can't use AUTO_INCREMENT
  - avoid it also on final column in multi-col index: (MyISAM-only feature)
    - CREATE TABLE foo (uid INT, postid INT AUTO_INCREMENT, PRIMARY KEY (userid, postid))
- moving users around clusters
  - balancing disk IO
  - balance disk space
  - monitor everything
    - cricket
    - nagios
    - ...whatever works

Subclusters

- easy at this point; APIs already exist
- multiple databases per real cluster
  - lj_50
  - lj_51
  - lj_52
  - ...
- MyISAM performance hack
- incremental maintenance

Where we're at...



Points of Failure

- 1 x Global master
  - lame
- *n* x User cluster masters
  - *n* x lame.
- Slave reliance
  - one dies, others reading too much

Solution?

Master-Master Clusters!

- two identical machines per cluster
  - both “good” machines
- do all reads/writes to one at a time, both replicate from each other
- intentionally only use half our DB hardware at a time to be prepared for crashes
- easy maintenance by flipping the active in pair
- no points of failure

Master-Master Prereqs

- failover can't break replication, be it:
  - automatic (be prepared for flapping)
  - by hand (probably have other problems)
- fun/tricky part is number allocation
  - same number allocated on both pairs
  - cross-replicate, explode.
- strategies
  - odd/even numbering (a=odd, b=even)
    - if numbering is public, users suspicious
      - where's my missing _____ ?
      - solution: prevent enumeration. add gibberish 'anum' = rand (256). visiblenum = (realid << 8 + anum). verify/store the anum
  - 3rd party arbitrator for synchronization

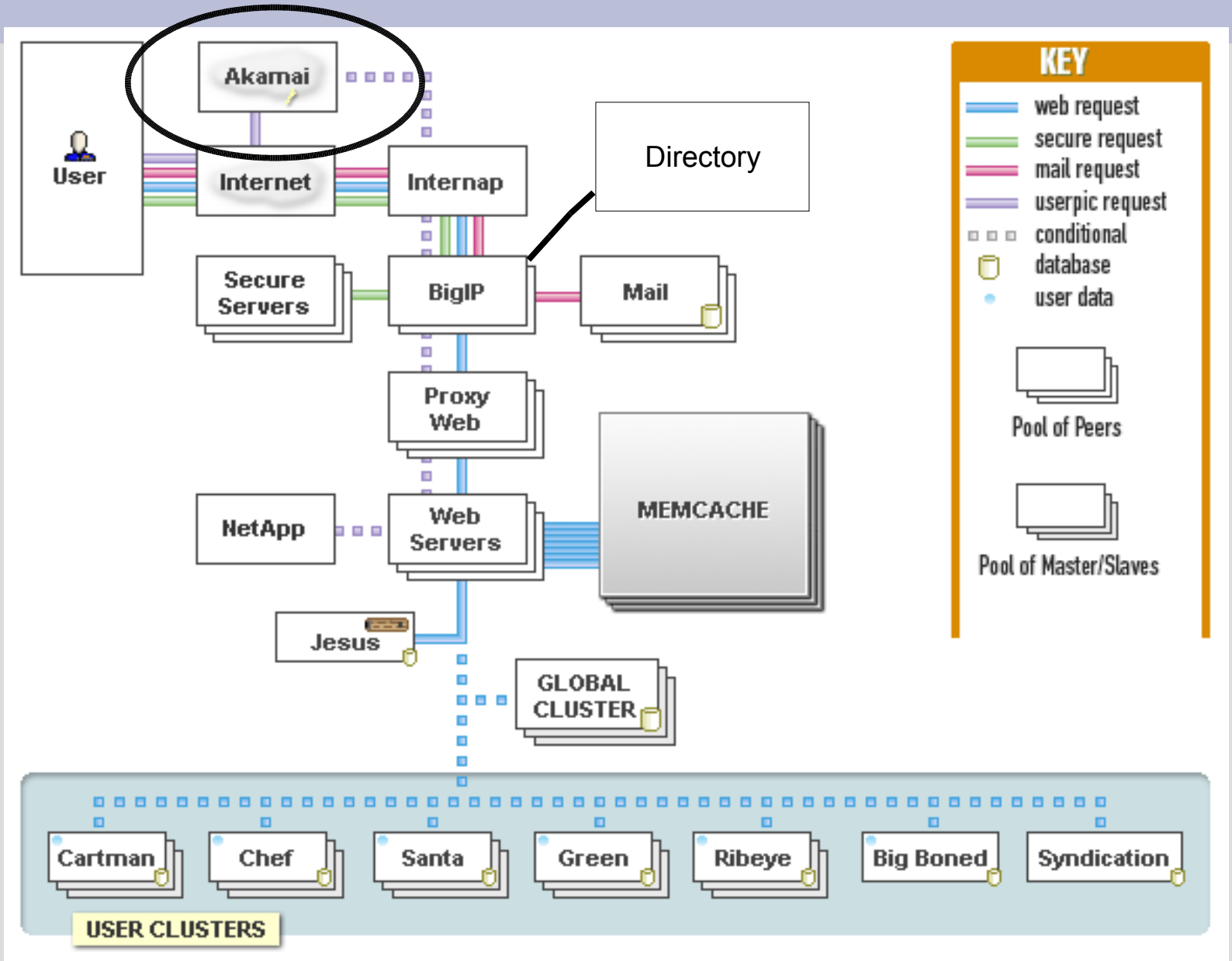
Cold Co-Master

- inactive pair isn't getting reads
- after switching active machine, caches full, but not useful (few min to hours)
- switch at night, or
- sniff reads on active pair, replay to inactive guy

Summary Thus Far

- Dual BIG-IPs (or LVS+heartbeat, or..)
- 30-40 web servers
- 1 “global cluster”:
  - non-user/multi-user data
  - what user is where?
  - master-slave (lame)
    - point of failure; only cold spares
    - pretty small dataset (<4 GB)
      - MySQL cluster looks potentially interesting
      - or master-election
- bunch of “user clusters”:
  - master-slave (old ones)
  - master-master (new ones)
- ...

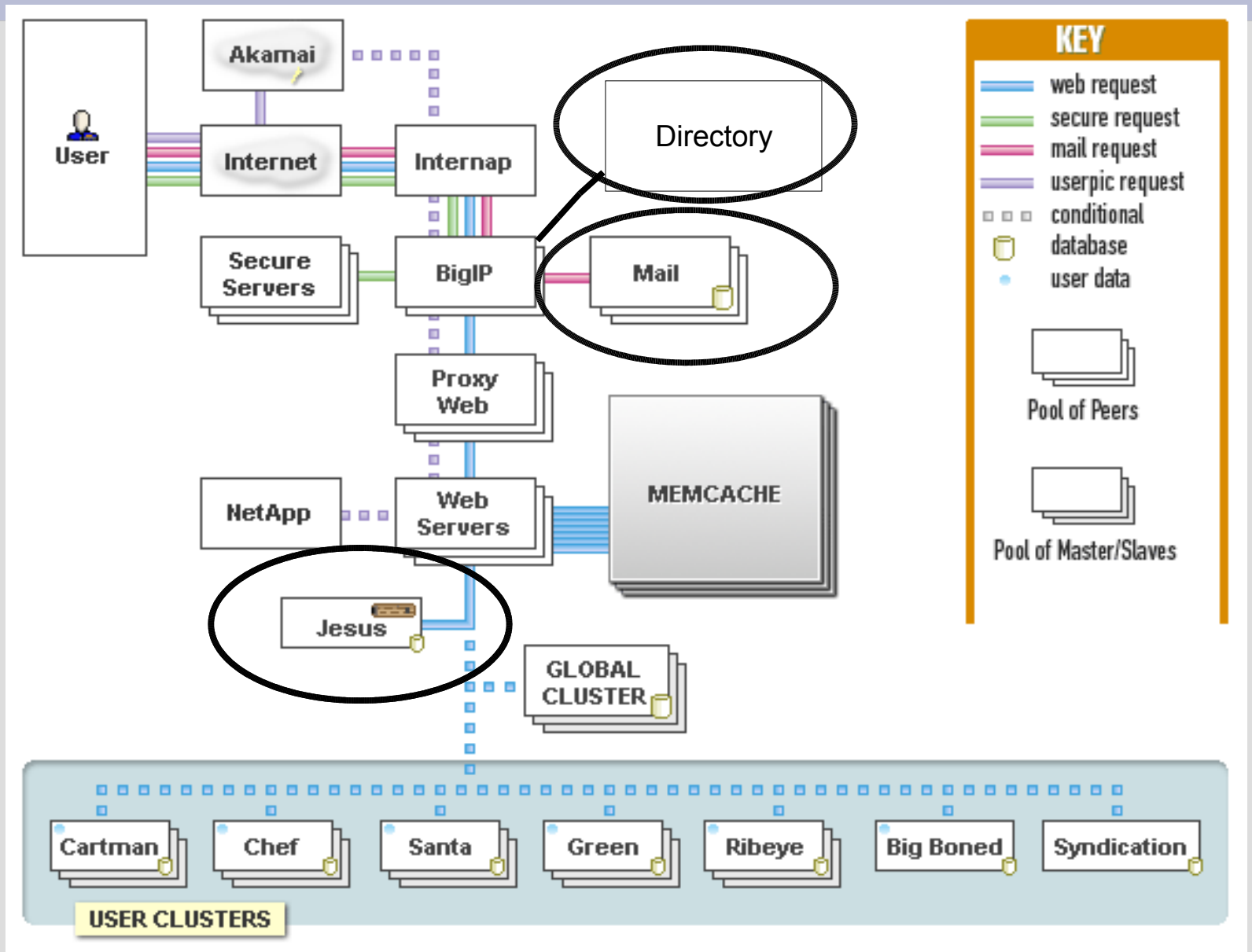
Static files...



Dynamic vs. Static Content

- static content
  - images, CSS
  - TUX, epoll-thttpd, etc. w/ thousands conns
  - boring, easy
- dynamic content
  - session-aware
    - site theme
    - browsing language
  - security on items
  - deal with heavy processes
- CDN (Akamai / Speedera)
  - static easier, APIs to invalidate
  - security: origin says 403 or 304

Misc MySQL Machines (Mmm...)



MyISAM vs. InnoDB

- We use both
- This is all nicely documented on mysql.com
- MyISAM
  - fast for reading xor writing,
  - bad concurrency, compact,
  - no foreign keys, constraints, etc
  - easy to admin
- InnoDB
  - ACID
  - good concurrency
- Mix-and-match. Design for both.

Directory & InnoDB

- Directory Search
  - multi-second queries
  - many at once
  - InnoDB!
  - replicates subset of tables from global cluster
  - some data on both global and user
    - write to both
    - read from directory for searching
    - read from user cluster when loading use data

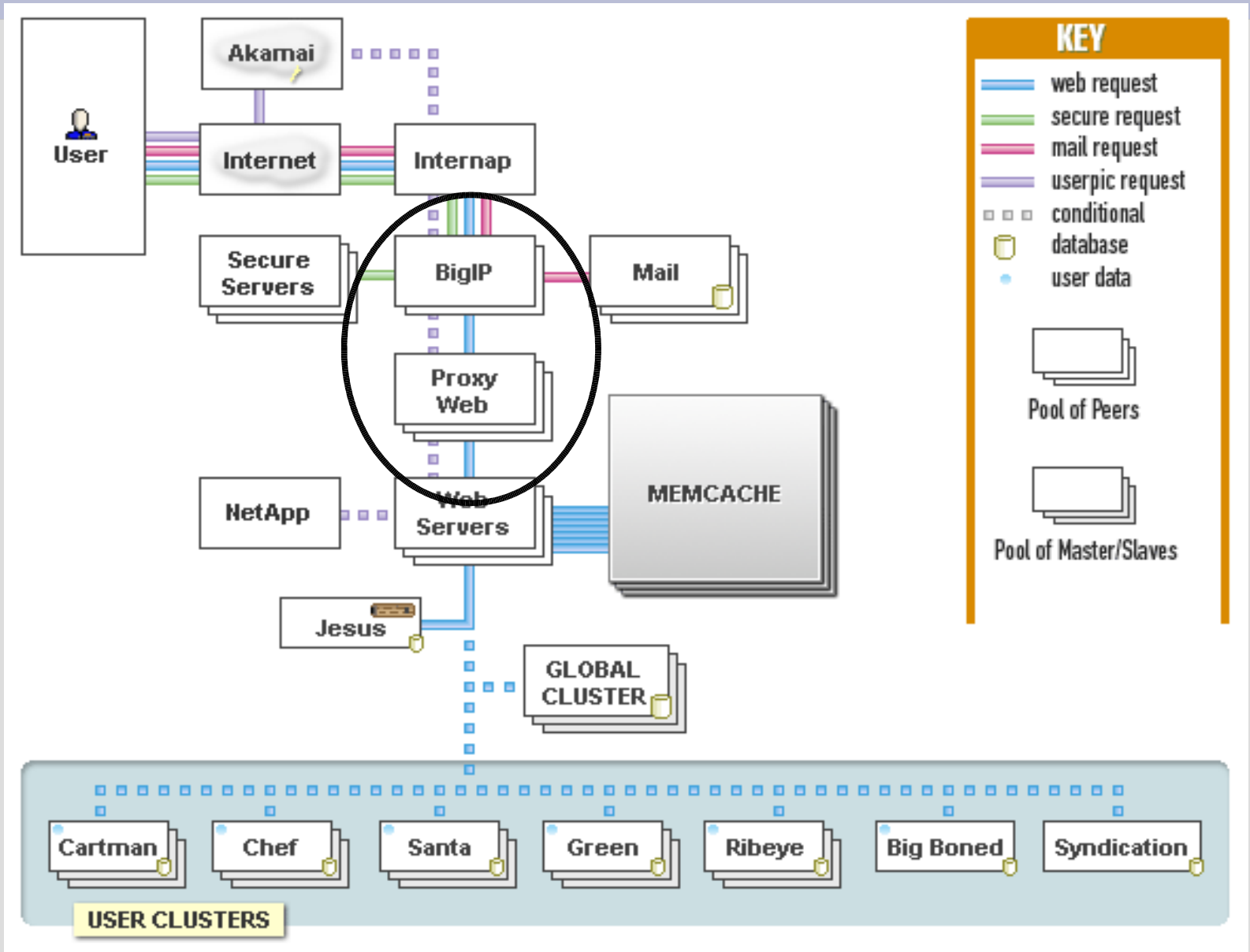
Postfix & MySQL

- Postfix
  - 4 servers: postfix + mysql maps
  - replicating one table: email_aliases
- Secondary Mail Queue
  - async job system
  - random cluster master
  - serialize message.

Logging to MySQL

- mod_perl logging handler
- new table per hour
  - MyISAM
- Apache access logging off
  - diskless web nodes, PXE boot
  - apache error logs through syslog-ng
- INSERT DELAYED
  - increase your insert buffer if querying
- minimal/no indexes
  - table scans are fine
- background job doing log analysis/rotation

Load Balancing!



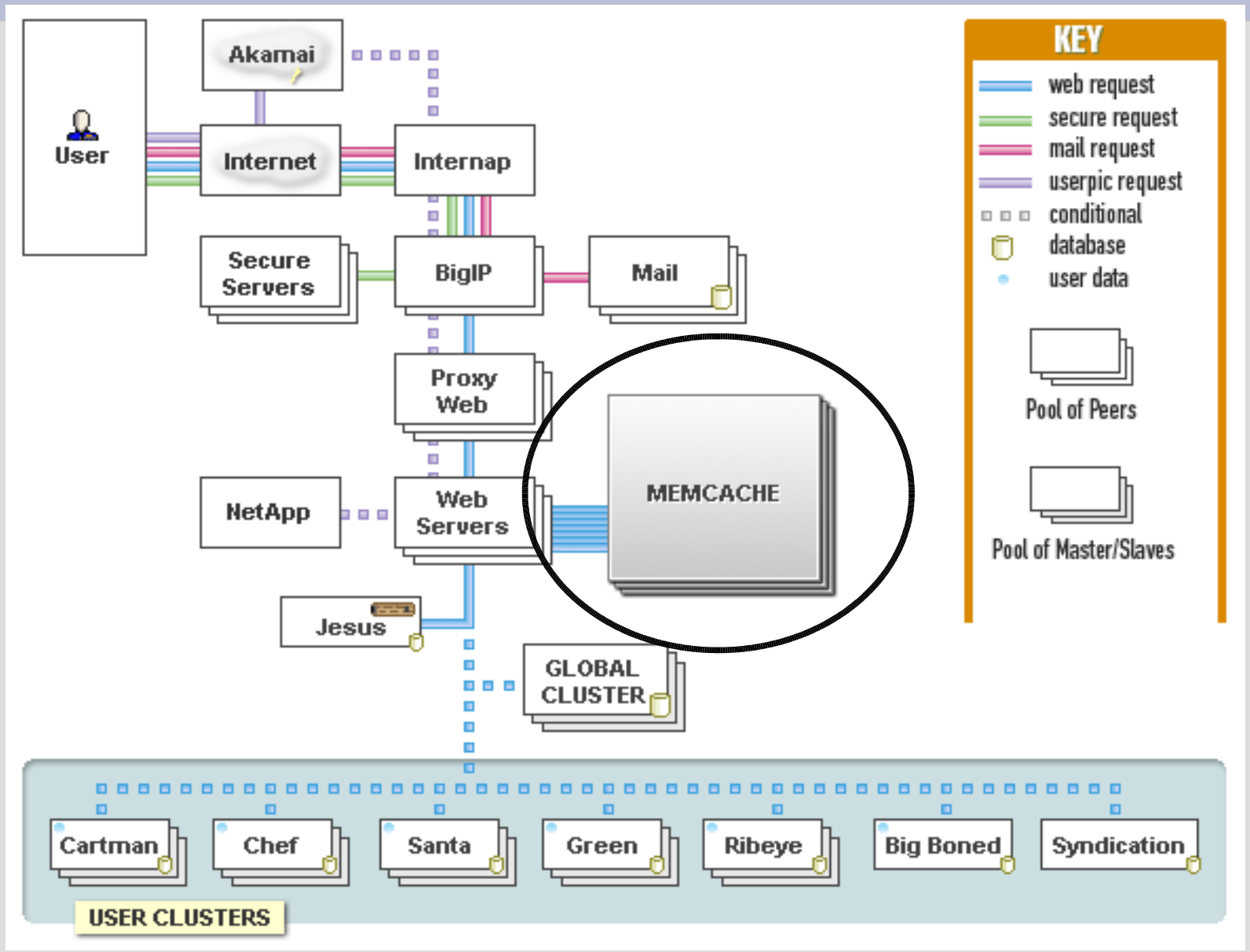
Web Load Balancing

- slow client problem (hogging mod_perl/php)
- BIG-IP [mostly] packet-level
- doesn't buffer HTTP responses
- BIG-IP can't adjust server weighting quick enough
  - few ms to multiple seconds responses
- mod_perl broadcasting state
  - Inline.pm to Apache scoreboard
- mod_proxy+mod_rewrite
  - external rewrite map (listening to mod_perl broadcasts)
  - map destination is [P] (mod_proxy)
- Monobal

DBI::Role – DB Load Balancing

- Our library on top of DBI
  - GPL; not packaged anywhere but our cvs
- Returns handles given a role name
  - master (writes), slave (reads)
  - directory (innodb), ...
  - cluster<n>{,slave,a,b}
  - Can cache connections within a request or forever
- Verifies connections from previous request
- Realtime balancing of DB nodes within a role
  - web / CLI interfaces (not part of library)
  - dynamic reweighting when node down

Caching!


User
Akamai
Internet
Internap
Secure Servers
BigIP
Mail
Proxy Web
MEMCACHE
NetApp
Web Servers
Jesus
GLOBAL CLUSTER
Cartman
Chef
Santa
Green
Ribeye
Big Boned
Syndication
USER CLUSTERS
KEY
web request
secure request
mail request
userpic request
conditional
database
user data
Pool of Peers
Pool of Master/Slaves

Caching

- caching's key to performance
- can't hit the DB all the time
  - MyISAM: r/w concurrency problems
  - InnoDB: good concurrency for disk
  - MySQL has to parse your query all the time
    - better with new MySQL binary protocol
- Where to cache?
  - mod_perl caching (address space per apache child)
  - shared memory (limited to single machine, same with Java/C#/Mono)
  - MySQL query cache: flushed per update, small max size
  - HEAP tables: fixed length rows, small max size

memcached

http://www.danga.com/memcached/

- our Open Source, distributed caching system
- run instances wherever there's free memory
  - requests hashed out amongst them all
  - choose to rehash or not on failure
- no “master node”
- protocol simple and XML-free; clients for:
  - perl, java, php, python, ruby, ...
- In use by:
  - LiveJournal, Slashdot, Wikipedia, ...
- People speeding up their:
  - websites, mail servers, ...

memcached – speed

- C
  - prototype Perl version proved concept, dog slow
- async IO, event-driven, single-threaded
- libevent (epoll, kqueue, select, poll...)
  - run-time mode selection
- lockless, refcounted objects
- slab allocator
  - glibc malloc died after 7~8 days
  - slabs: no address space fragmentation ever.
- O(1) operations
  - hash table inside
  - Judy didn't work (malloc problems?)
- multi-server parallel fetch (can't do in DBI?)

LiveJournal and memcached

- 10 unique hosts
  - none dedicated
- 28 instances
- 30 GB of cached data
- 90-93% hit rate
  - not necessarily 90-93% less queries:
    - FROM foo WHERE id IN (1, 2, 3)
    - would be 3 memcache hits; 1 mysql query
  - 90-93% potential disk seeks?
- 12 GB machine w/ five 2GB instances
  - left-over 'big' machines from our learn-to-scale-out days

What to Cache

- Everything?
- Start with stuff that's hot
- Look at your logs
  - query log
  - update log
  - slow log
- Control MySQL logging at runtime
  - can't
    - help me bug them.
  - sniff the queries! Net::Pcap
    - tool to be released? bug me.
- canonicalize and count
  - name queries: SELECT /* name=foo */

Caching Disadvantages

- updating your cache
  - decide what's important
  - when to do a clean read (from DB) vs potentially-dirty read (from memcached)
- more crap to admin
  - but memcached is easy
  - one real option: memory to use
- disable rehashing, or be prepared
  - small, quick objects
    - “time user #123 last posted = *t*”
  - heavy objects with unlimited lifetime, containing small item too
    - “last 100 recent post ids for user #123, as of time *t*”
  - application can detect problems

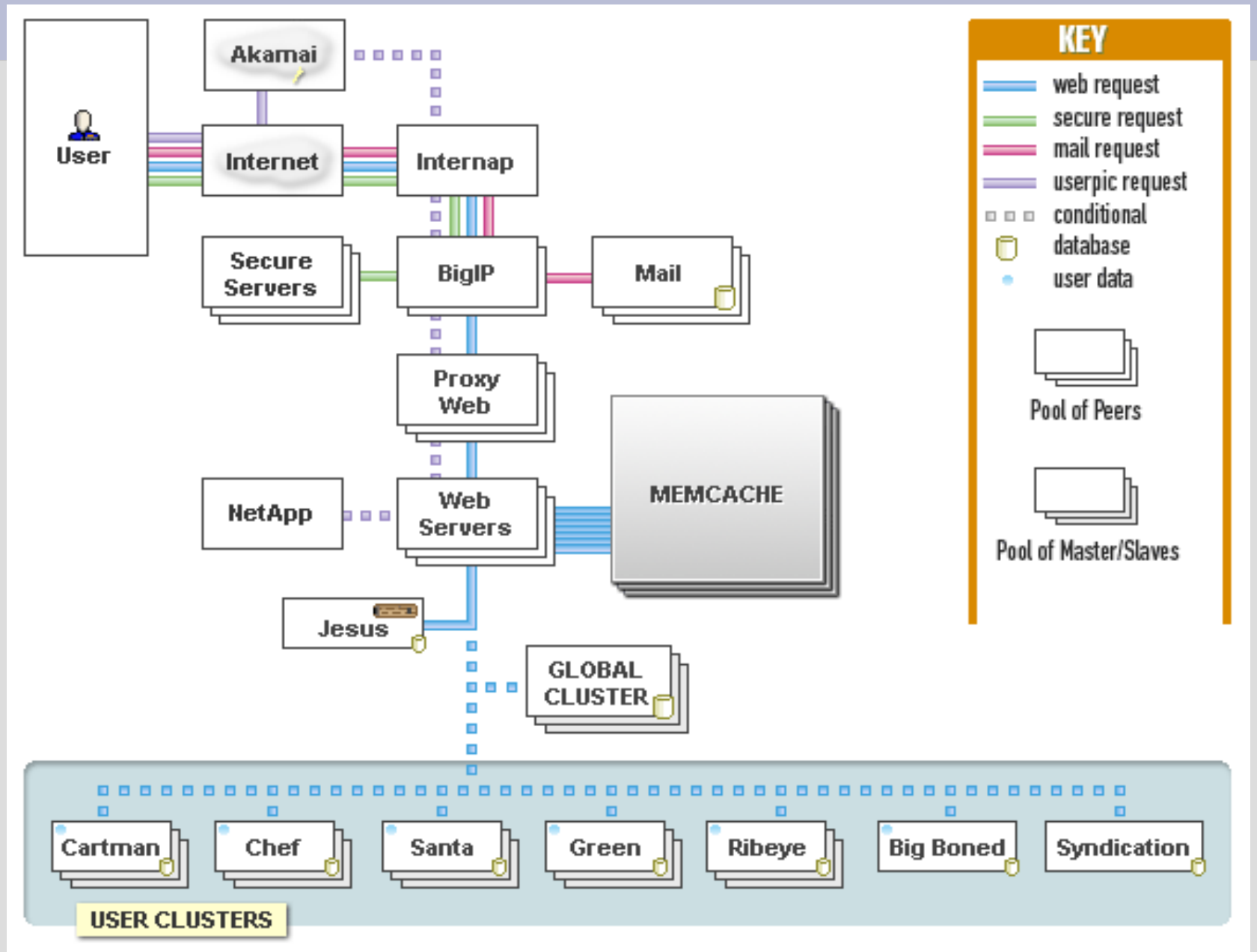
MySQL Persistent Connection Woes

- connections == threads == memory
  - (until MySQL 5.x? thanks, Brian!)
- max threads
  - limit max memory
- with 10 user clusters:
  - Bob is on cluster 5
  - Alice on cluser 6
  - Do you need Bob's DB handles alive while you process Alice's request?
- Major wins by disabling persistent conns
  - still use persistent memcached conns

Software Overview

- Linux 2.4
  - database servers
- Linux 2.6
  - web nodes; memcached (epoll)
  - experimenting on dbs w/ master-master
- Debian woody
  - moving to sarge
- BIG-IPs
  - got new ones, w/ auto fail-over
  - management so nice, anti-DoS
- mod_perl

Questions?


User
Akamai
Internet
Internap
Secure Servers
BigIP
Mail
Proxy Web
NetApp
Web Servers
MEMCACHE
Jesus
GLOBAL CLUSTER
Cartman
Chef
Santa
Green
Ribeye
Big Boned
Syndication
USER CLUSTERS
KEY
web request
secure request
mail request
userpic request
conditional
database
user data
Pool of Peers
Pool of Master/Slaves

Thank you!

Questions to...
brad@danga.com

Slides linked off:
http://www.danga.com/words/