CSE 403

Scale

Announcements

- Deliverables for 11/19 release (due 11:59PM)
  - Documents up to date: req, arch, design, schedule
  - Release notes
  - Status report for the **release** (not due until 11/20) -- content should be part of 11/21 presentation
- Presentations on 11/21: 7 minutes, same logistics as before. Deposit presentations (pdf) in dropbox by 11:59PM on 11/20
- Final release due on 12/03

Release notes

- A description of what you just released
  - High level, thematic description
  - Major functionality
  - Exceptions of what might not be working
  - How to access the release
- Approximately a “README”
- “We’re proud to announce release 1.4 of...”
- Largely for external consumption
- About 1 page

Project Presentations, 11/21 (7 minutes)

- Operations Review by manager or project manager-- similar to weekly status but for the release cycle
  - What did you say you were going to do? What did you do? (1 minute)
  - What are you going to do in the next cycle? (1 minute)
  - What are the issues? (1 minute)
  - **What's your current status on platforms and browsers?**
  - Report on a metric -- or if not in place, scale from 1(disaster) - 10 (exceeding) and why (1 minute)
- Demo (1 minute)
- Questions and Answers (1 minute)

Too much work?

- Project
- Homework
- Platform/browser issues?
- What to do?
  - Talk to instructor or TA
  - No need to suffer/"complain" in silence

One-on-one's

(tip of the day)

- Regularly scheduled meeting with your manager (about once a week)
- Communication mechanism
- Weekly status update
- Other (strategic) issues
- Don't discount the value
- Red flag if your boss isn't doing this?

CSE 403

Scale

Things are working and you are starting to grow...

- Congratulations! You've deployed your application!
- Things are working
- You are starting to get customers/traffic/users
- ...system is slowing down! or crashing!
- Good news: Success...You need to scale

Scale up

- Monitor and measure
- Fix the bugs
- Tune
- Replicate (stateless)
- Cache
- Database architecture

Measure and profile

- Unix command line tools (ps, top, vmstat,...)
- Munin
- Database tools
- Log files
- Django instrumentation tools

Munin

- CPU utilization
- Disk performance
- Database health
- Network utilization
- Webserver traffic
- Back end servers

Munin

(http://munin-monitoring.org/)

Code changes

- Fix bugs
- Improve code/Change algorithms
- Database indexing/query optimization
- Database tuning
- System tuning

Bigger machines

- Bigger CPU
- More CPUs
- Bigger caches
- More memory
- More bandwidth
- Bigger database

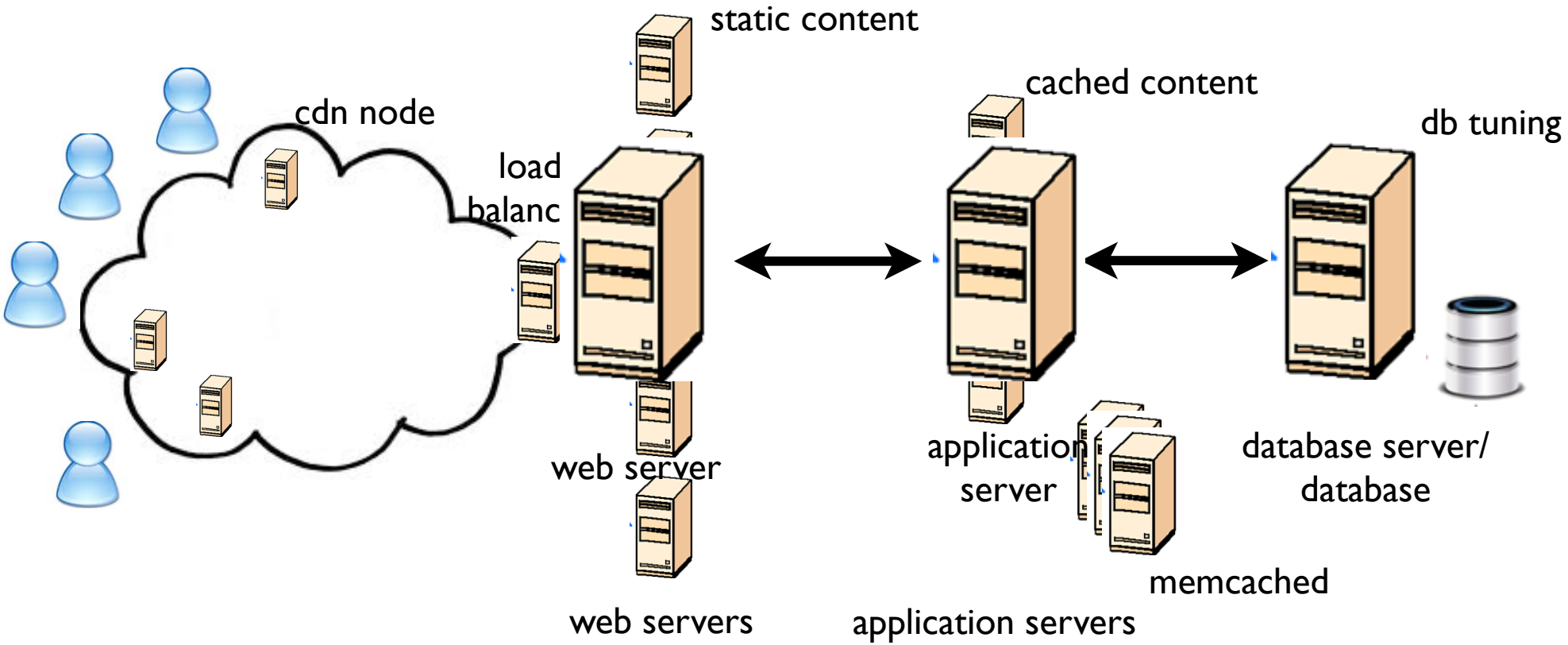
Scale out

- Add more web servers
- Add more application servers
- More processes/threads?
- Issues
  - Statefullness
  - Server affinity

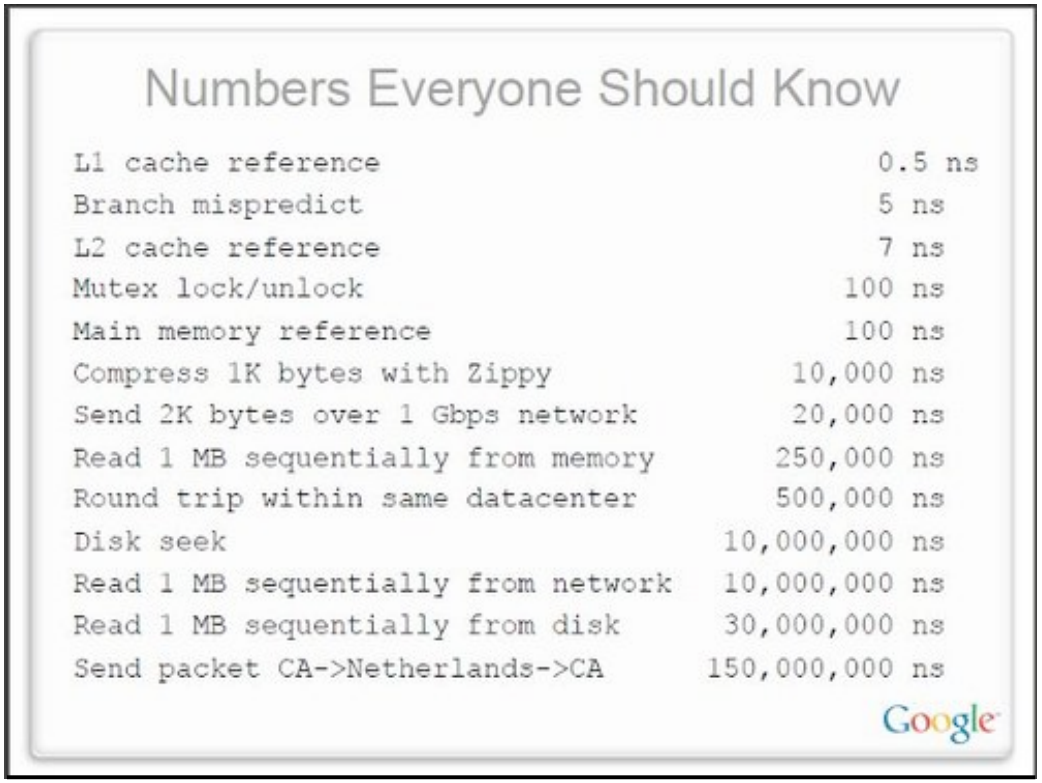
Caching

- Store read-only data in web server
- Content distribution network (read only data)
- Code changes to cache data in app server (avoid database round trips)
- Snippet caching (don't generate html on every request)
- Memcached (distributed shared memory as a key value store)
- In general, bring (read only) data closer to the user

Scaling up through caching



Aside: Rules of thumb



Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Google

But 2 years old!

When you outgrow a single database

What are you going to do?

Scaling databases (RDBMSes mostly)

- Likely culprit of scale, performance, response time issues
- Can you stay on a single DB?
  - Tuning, indices, bigger machines, more memory...
  - Off load work from the RDBMS
- No?
  - What to do now?
  - Add more databases and partition (but this is sort of hard!)

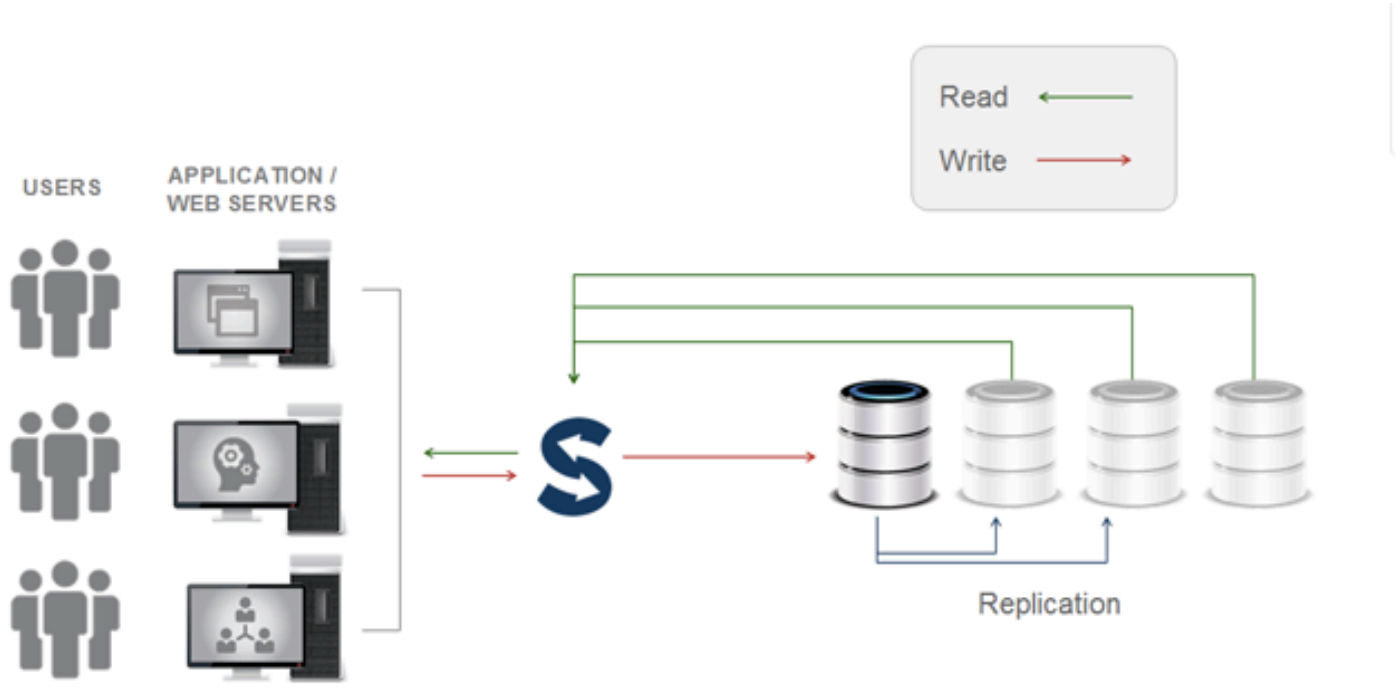
Scale up through replication

- Read/Write partitioning
- Horizontal scaling (sharding)
- Vertical scaling

Read/write partitioning

- Most accesses to your system might be reads (90% reads, 10% writes)
- Reads can proceed in parallel
- Writes block readers and writers
- Replicate databases with multiple copies that can be read from
- Allow writes only to one instance (master)
- Have to propagate writes to read copies
- Consistency issue?

Read/write partitioning



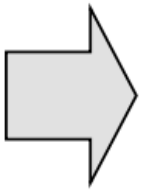
Horizontal partitioning (sharding)

- Separate data across multiple machines
- Schema replication
- Machines hold full rows of data
- Rows are split across machines
- Know which rows are on which machines ("hash a primary key to a machine")

Horizontal partitioning

Customers

SSN	Name	City
234234	Mary	Houston
345345	Sue	Seattle
345343	Joan	Seattle
234234	Ann	Portland
--	Frank	Calgary
--	Jean	Montreal



SSN	Name	City
234234	Mary	Houston
345345	Sue	Seattle

SSN	Name	City
345343	Joan	Seattle
234234	Ann	Portland

.....

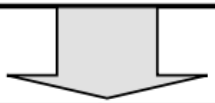
Vertical partitioning

- Separate data across multiple machines
- Machines hold different parts of the column space
- Access to subsets of the column space are are hopefully clustered (so queries don't span multiple machines)

Vertical partitioning

Resumes

SSN	Name	Address	Resume	Picture
234234	Mary	Huston	Clob1…	Blob1…
345345	Sue	Seattle	Clob2…	Blob2…
345343	Joan	Seattle	Clob3…	Blob3…
234234	Ann	Portland	Clob4…	Blob4…



SSN	Name	Address
234234	Mary	Huston
345345	Sue	Seattle
. . .		

SSN	Resume
234234	Clob1…
345345	Clob2…

SSN	Picture
234234	Blob1…
345345	Blob2…

Consistency

- Do users need to see a consistent view of the data over time?
- Do different users need to see a consistent view of the data?
- Weak consistency means no
- Eventually consistency (weak for awhile, but eventually unified)

Relaxing consistency requirements and not having transactions makes things easier

CAP theorem

- Data is **Consistent**
- Data is **Available** (every request receives a response)
- Data can be served in the event of a **Partition**

CAP Theorem: Can't satisfy all three

You're big

- If you have to do all this, you are bigger than 99% of all web applications
- Your engineering process has evolved, your team is much bigger
- But it's all understandable
- Large engineering headache
- What's the next scale up?

Megascale

- There are companies that have scaled to over 100 million users with a million CPUs
- Market caps of $100+ billions
- Engineering is very different. How is this done?
- Guest lecture next Monday (11/26): Ari Steinberg, formerly of Facebook, "Megascale Software Engineering at Facebook"

Take aways

- Scaling issues are "good" problems
- Don't over-plan or over-execute too early (think "pure thoughts")
- Buy bigger machines/more machines
- Caching to the rescue
- Database replication (but this is hard!)
- Transactions and strong consistency make this harder
- At "mega-scale" these techniques have problems too