Introduction to Software Architecture and Documentation

Michael Hilton

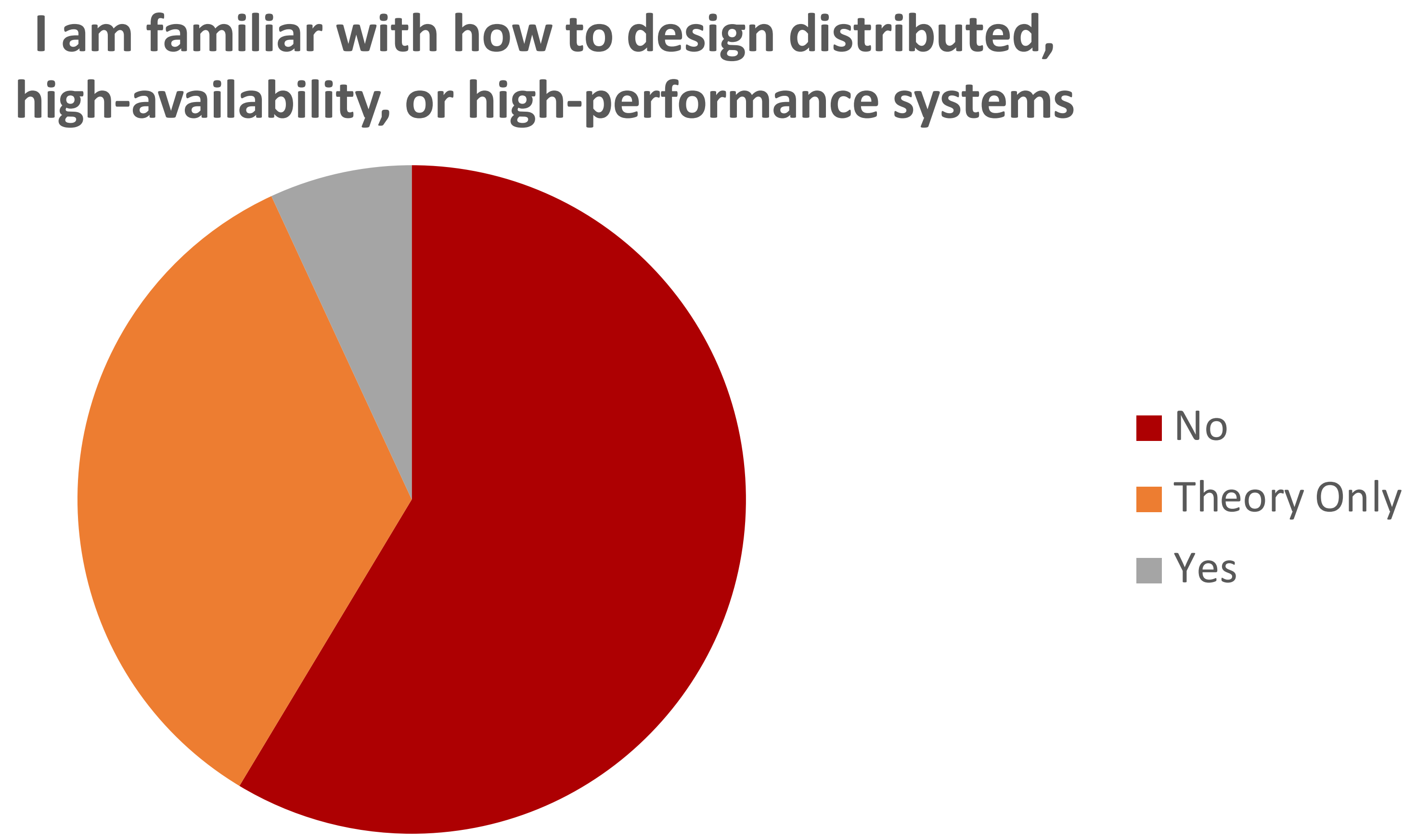
Administrativa

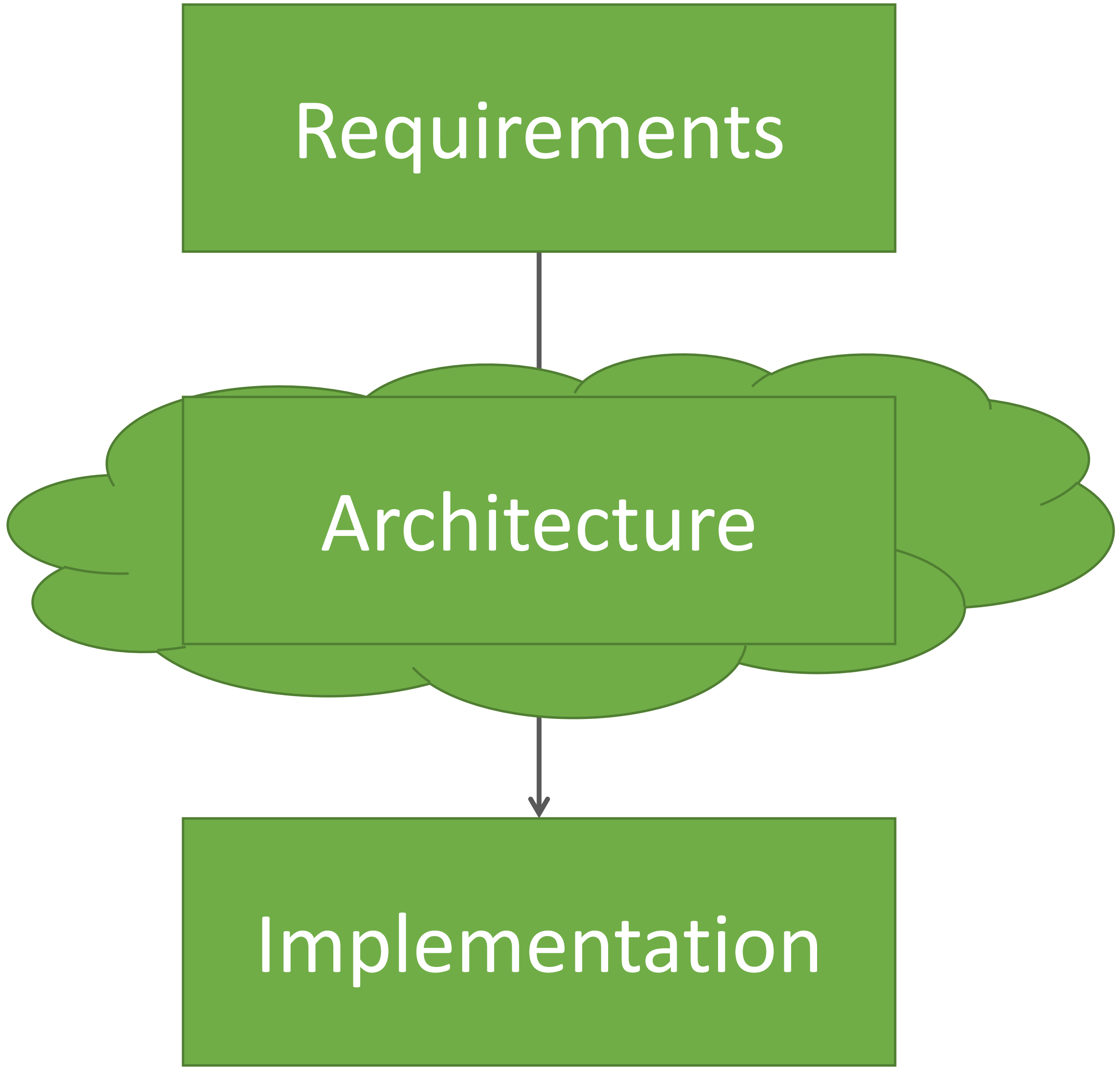
- Homework 3 due ~~tonight~~ Tomorrow

Learning Goals

- Understand the abstraction level of architectural reasoning
- Approach software architecture with quality attributes in mind
- Distinguish software architecture from (object-oriented) software design
- Use notation and views to describe the architecture suitable to the purpose
- Document architectures clearly, without ambiguity

About You


I am familiar with how to design distributed, high-availability, or high-performance systems
No
Theory Only
Yes


Requirements
Architecture
Implementation

Quality Requirements, now what?

- "should be highly available"
- "should answer quickly, accuracy is less relevant"
- "needs to be extensible"
- "should efficiently use hardware resources"

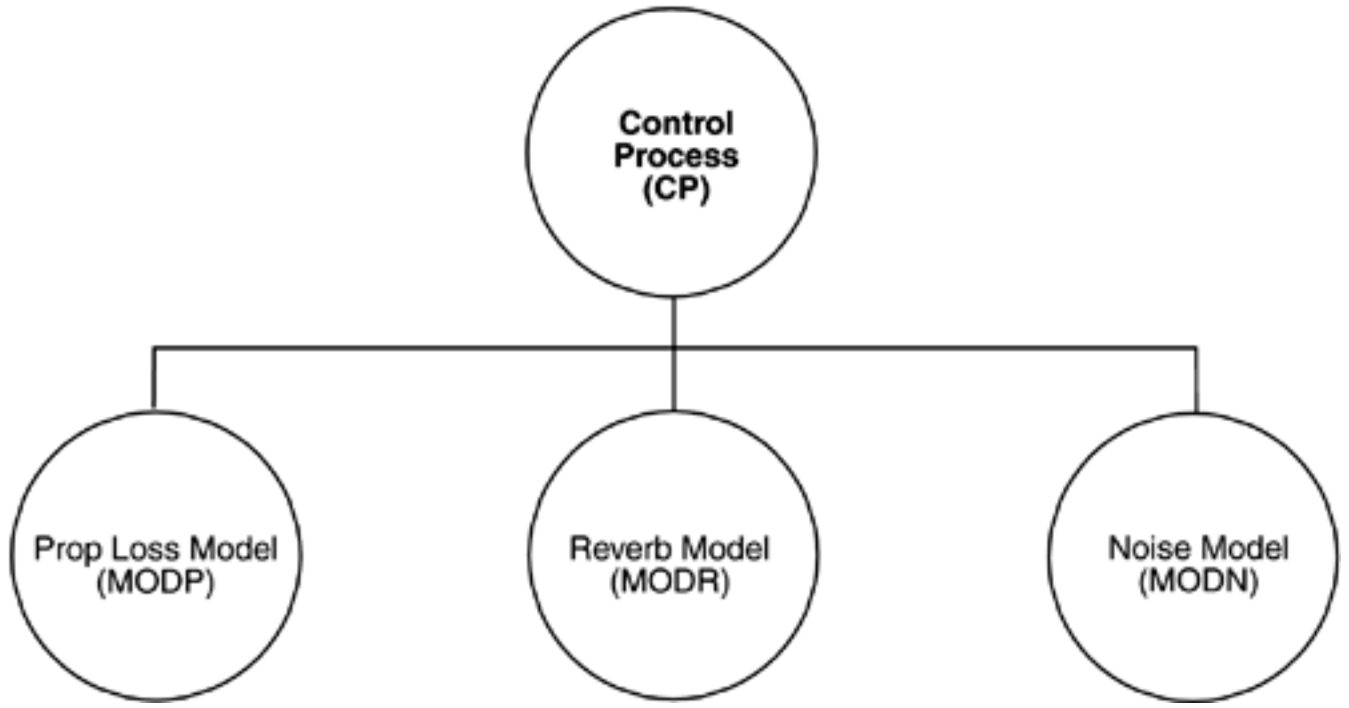
SOFTWARE ARCHITECTURE

Software Architecture

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

[Bass et al. 2003]

Note: this definition is ambivalent to whether the architecture is known, or whether it's any good!



From Bass et al. Software Architecture in Practice, 2nd ed.

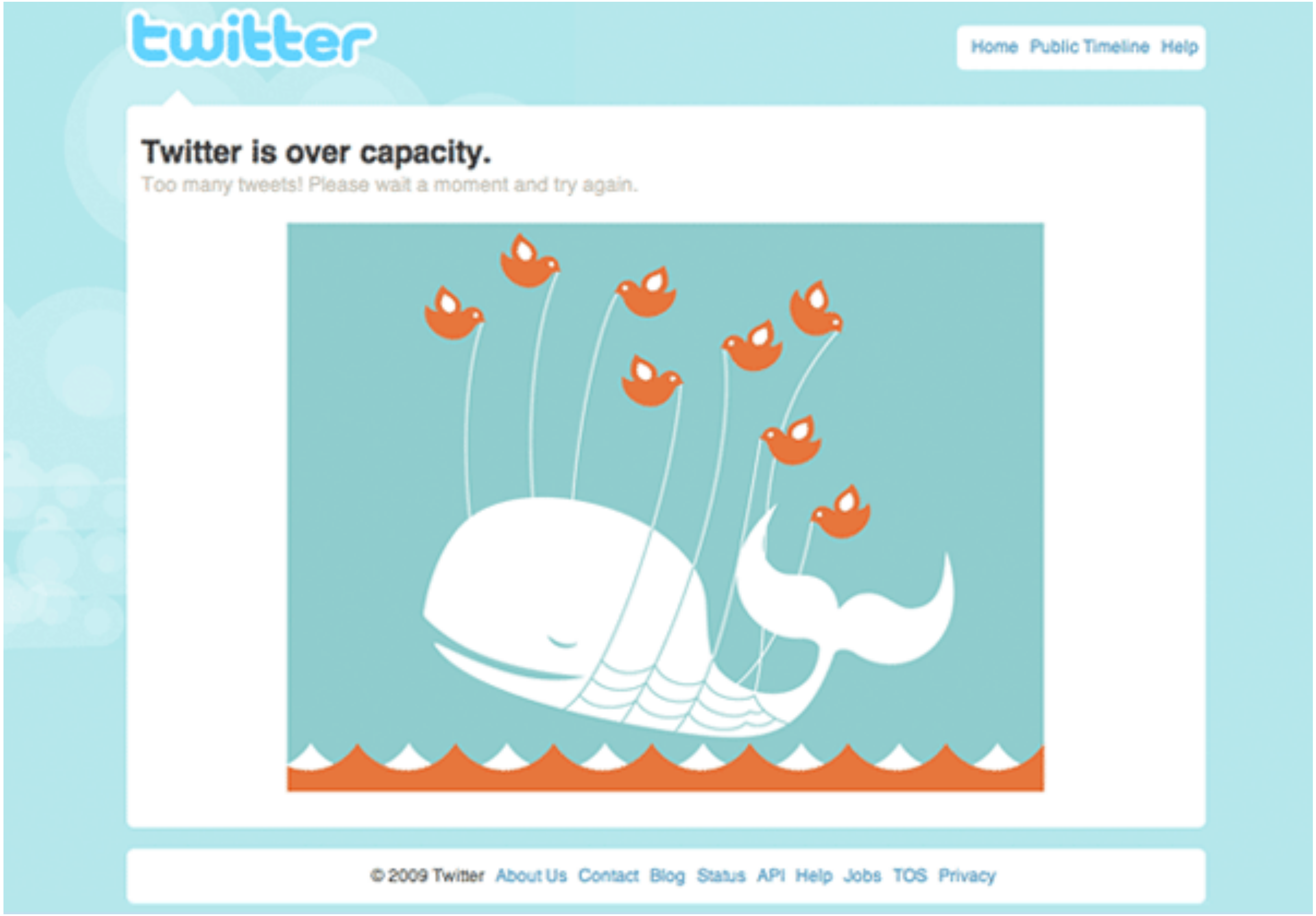
Why Architecture? [BCK03]

- Represents earliest design decisions.
- Aids in communication with stakeholders
  - Shows them "how" at a level they can understand, raising questions about whether it meets their needs
- Defines constraints on implementation
  - Design decisions form "load-bearing walls" of application
- Dictates organizational structure
  - Teams work on different components
- Inhibits or enables quality attributes
  - Similar to design patterns
- Supports predicting cost, quality, and schedule
  - Typically by predicting information for each component
- Aids in software evolution
  - Reason about cost, design, and effect of changes
- Aids in prototyping
  - Can implement architectural skeleton early

Beyond functional correctness

- Quality matters, eg.,
  - Performance
  - Availability
  - Modifiability, portability
  - Scalability
  - Security
  - Testability
  - Usability
  - Cost to build, cost to operate

CASE STUDY: ARCHITECTURE AND QUALITY AT TWITTER



Home Public Timeline Help

Twitter is over capacity.

Too many tweets! Please wait a moment and try again.

© 2009 Twitter About Us Contact Blog Status API Help Jobs TOS Privacy

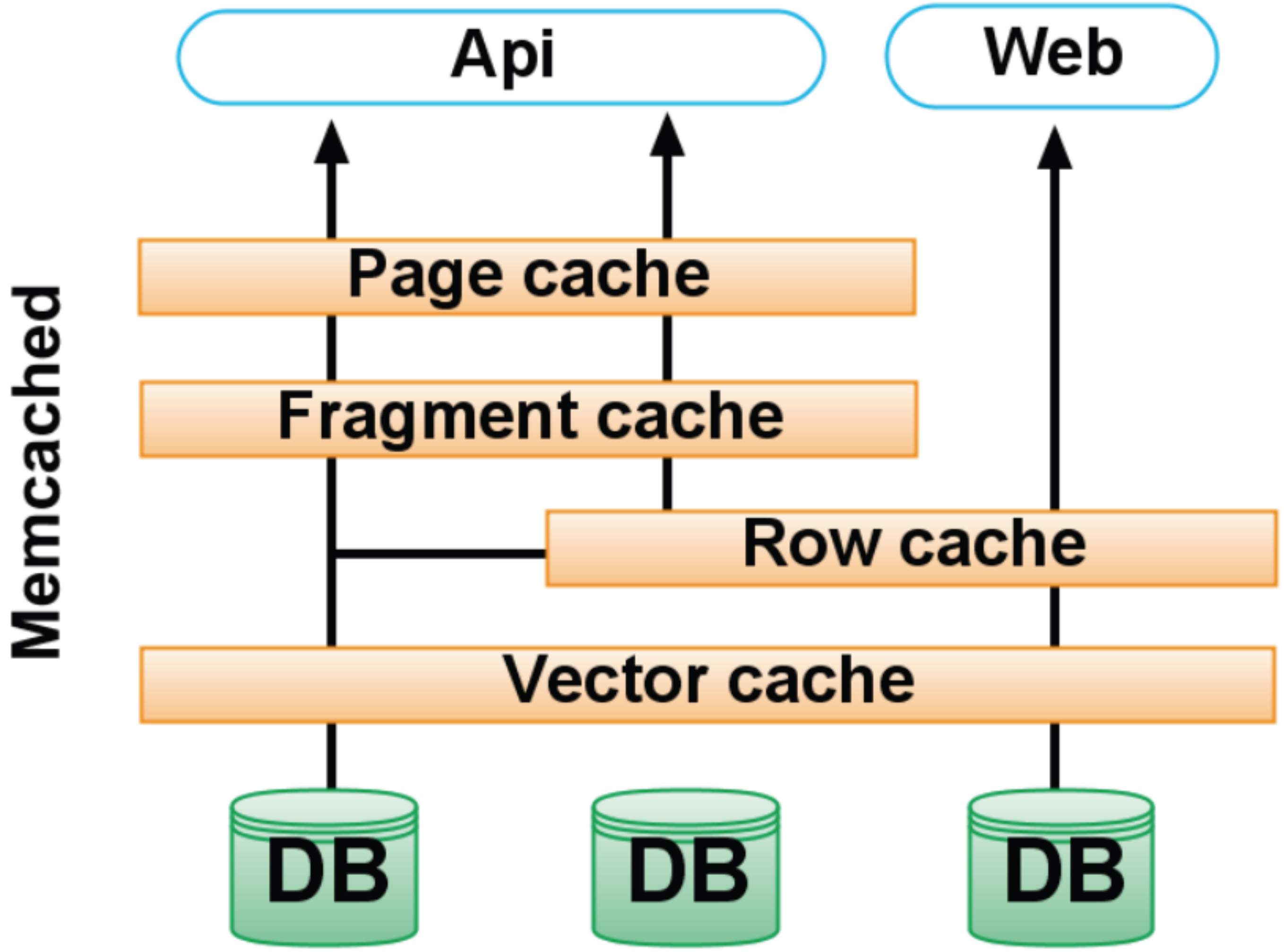
SOUTH
AFRICA
2010

FIFA
WORLD CUP

Inspecting the State of Engineering

- Running one of the world's largest Ruby on Rails installations
- 200 engineers
- Monolithic: managing raw database, memcache, rendering the site, and presenting the public APIs in one codebase
- Increasingly difficult to understand system; organizationally challenging to manage and parallelize engineering teams
- Reached the limit of throughput on our storage systems (MySQL); read and write hot spots throughout our databases
- Throwing machines at the problem; low throughput per machine (CPU + RAM limit, network not saturated)
- Optimization corner: trading off code readability vs performance

Caching


Api
Web
Memcached
Page cache
Fragment cache
Row cache
Vector cache
DB
DB
DB

Twitter's Quality Requirements/Redesign goals??

- Improve median latency; lower outliers
- Reduce number of machines 10x
- Isolate failures
- "We wanted cleaner boundaries with "related" logic being in one place"
  - encapsulation and modularity at the systems level (rather than at the class, module, or package level)
- Quicker release of new features
  - "run small and empowered engineering teams that could make local decisions and ship user-facing changes, independent of other teams"

performance

reliability

maintainability

modifiability

JVM vs Ruby VM

- Rails servers capabile of 200-300 requests / sec / host
- Experience with Scala on the JVM; level of trust
- Rewrite for JVM allowed 10-20k requests / sec / host

Programming Model

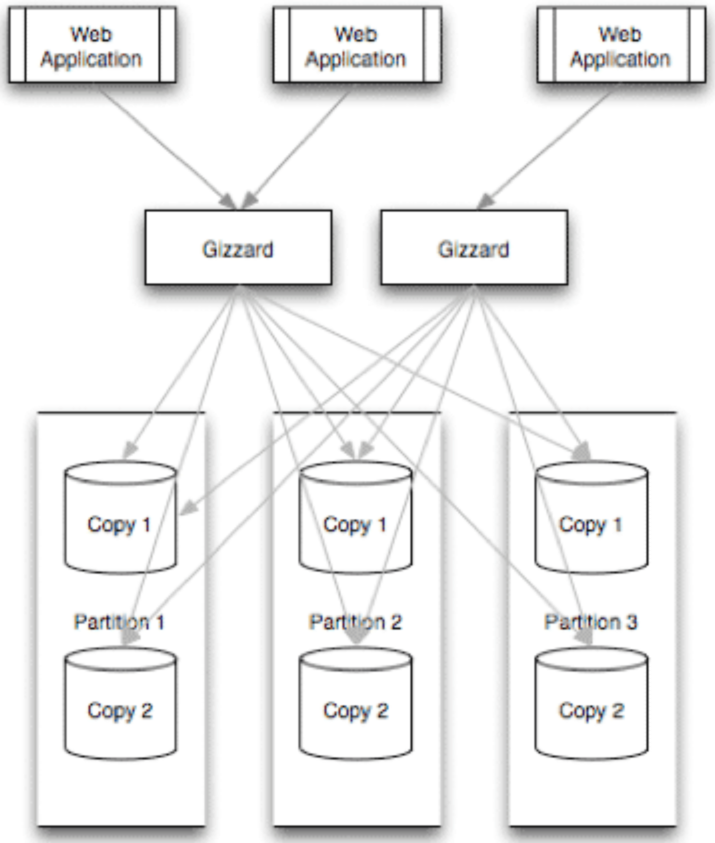
- Ruby model: Concurrency at process level; request queued to be handled by one process
- Twitter response aggregated from several services – additive response times
- *"As we started to decompose the system into services, each team took slightly different approaches. For example, the failure semantics from clients to services didn't interact well: we had no consistent back-pressure mechanism for servers to signal back to clients and we experienced "thundering herds" from clients aggressively retrying latent services."*
- Goal: Single and uniform way of thinking about concurrency
  - Implemented in a library for RPC (Finagle), connection pooling, failover strategies and load balancing

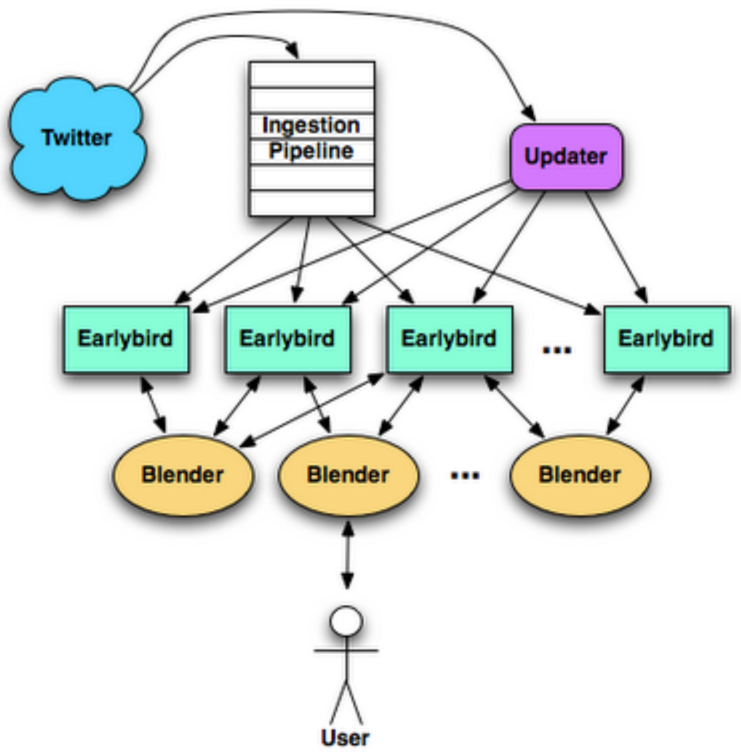
Independent Systems

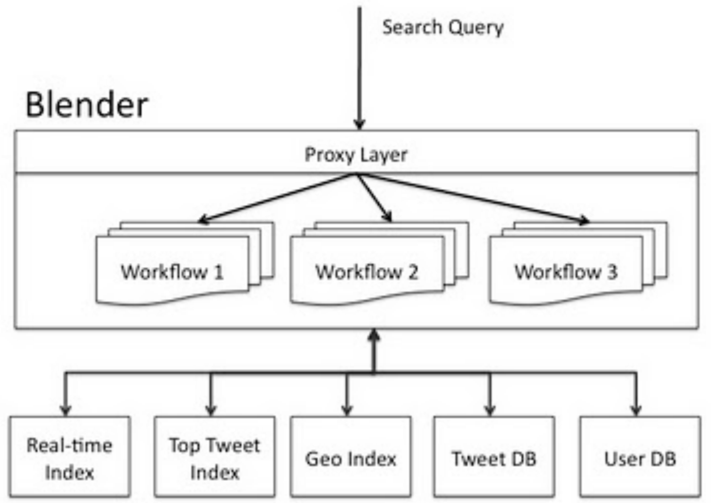
- " In our monolithic world, we either needed experts who understood the entire codebase or clear owners at the module or class level. Sadly, the codebase was getting too large to have global experts and, in practice, having clear owners at the module or class level wasn't working. Our codebase was becoming harder to maintain, and teams constantly spent time going on "archeology digs" to understand certain functionality. Or we'd organize "whale hunting expeditions" to try to understand large scale failures that occurred."
- From monolithic system to multiple services
    - Agree on RPC interfaces, develop system internals independently
    - Self-contained teams

Storage

- Single-master MySQL database bottleneck despite more modular code
- Temporal clustering
  - Short-term solution
  - Skewed load balance
  - One machine + replications every 3 weeks
- Move to distributed database (Glizzard on MySQL) with "roughly sortable" ids
- Stability over features – using older MySQL version



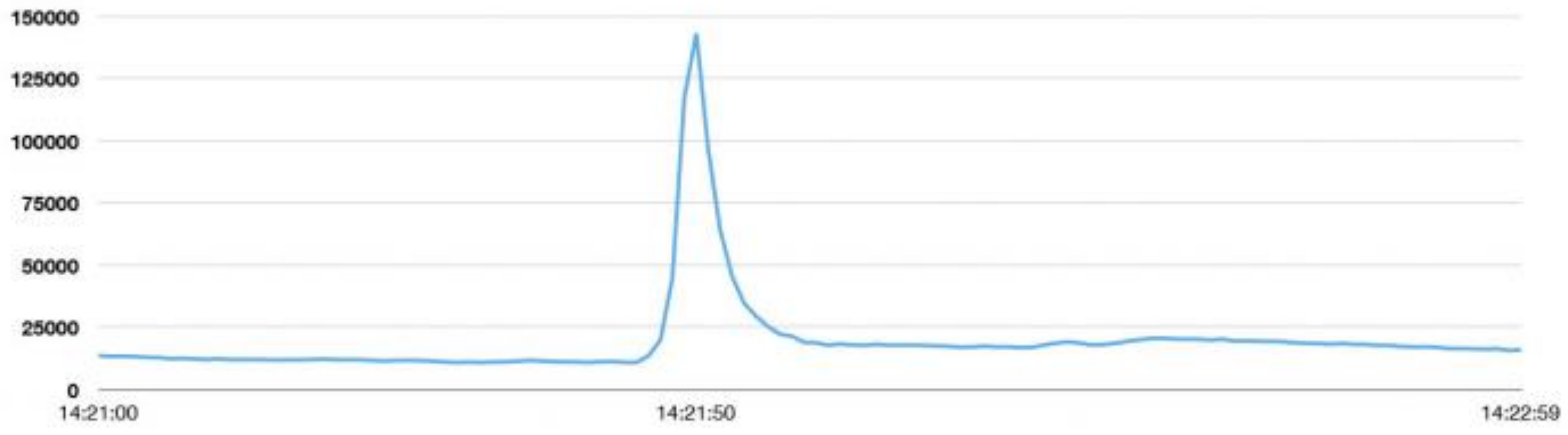

Twitter
Ingestion Pipeline
Updater
Earlybird
Earlybird
Earlybird
...
Earlybird
Blender
Blender
...
Blender
User


Search Query
Blender
Proxy Layer
Workflow 1
Workflow 2
Workflow 3
Real-time Index
Top Tweet Index
Geo Index
Tweet DB
User DB

Data-Driven Decisions

- Many small independent services, number growing
- Own dynamic analysis tool on top of RPC framework
- Framework to configure large numbers of machines
    - Including facility to expose feature to parts of users only

血湧き
肉躍る
漫画映画。
天空の城
ラピュタ



On Saturday, August 3 in Japan, people watched an airing of Castle in the Sky, and at one moment they took to Twitter so much that we hit a one-second peak of 143,199 Tweets per second.

Key Insights: Twitter Case Study

- Architectural decisions affect entire systems, not only individual modules
- Abstract, different abstractions for different scenarios
- Reason about quality attributes early
- Make architectural decisions explicit
- Question: *Did the original architect make poor decisions?*