Transactional Middleware Reconsidered

Phil Bernstein

Sergey Bykov, Alan Geller, Gabriel Kliot, Jorgen Thelin

Microsoft Corporation

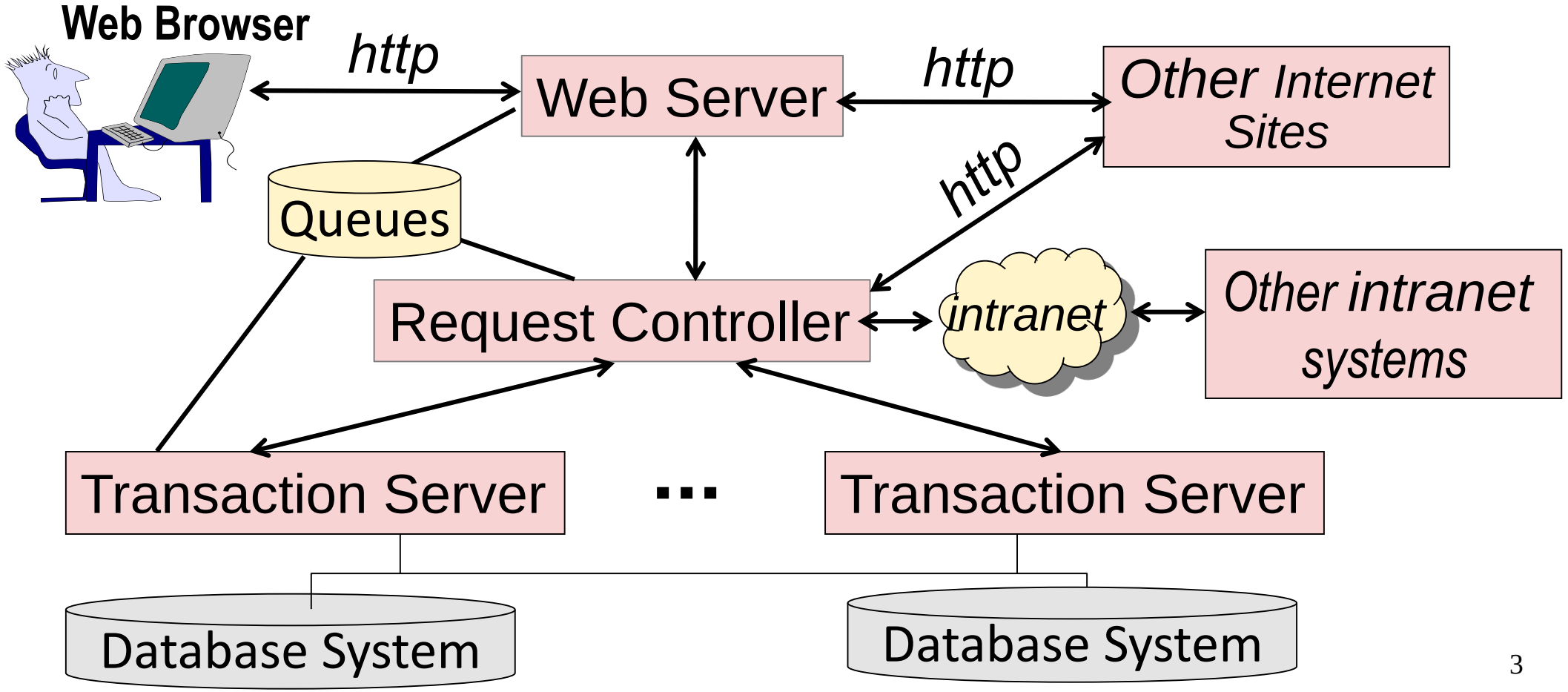
CIDR 2013, 1/7/13

Transactional Database Applications

- Are hard to build
- Have a highly regular structure

Transactional Application Structure

An application system must coordinate the flow of requests between message sources and apps that run requests as transactions.



Application Server Architecture

- Transactional middleware simplifies app development
  - Defines common app system structure
  - Adds missing platform features
- 1970s – 1980s: RPC, multithreaded processes, session pooling, forms, terminal management, automated recovery
- 1990s: OO programming, stateful communications, web browsers, 2-phase commit, queuing.
- Since 2000 (J2EE, .NET Framework): SOA & web services, XML, object-relational mapping

Today's Problem: Cloud Apps that Scale Out

- Cloud apps are very hard for mainstream developers to build
- What's hard?
  - Ensuring scalability, elasticity, and load balancing
  - Parallel programming, multi-threading
  - Composing independent services
  - Error-handling across services
  - Fault tolerance
- This problem is getting surprisingly little attention

Orleans, an actor-oriented programming model

- Makes it easy to develop cloud apps that scale “by default”

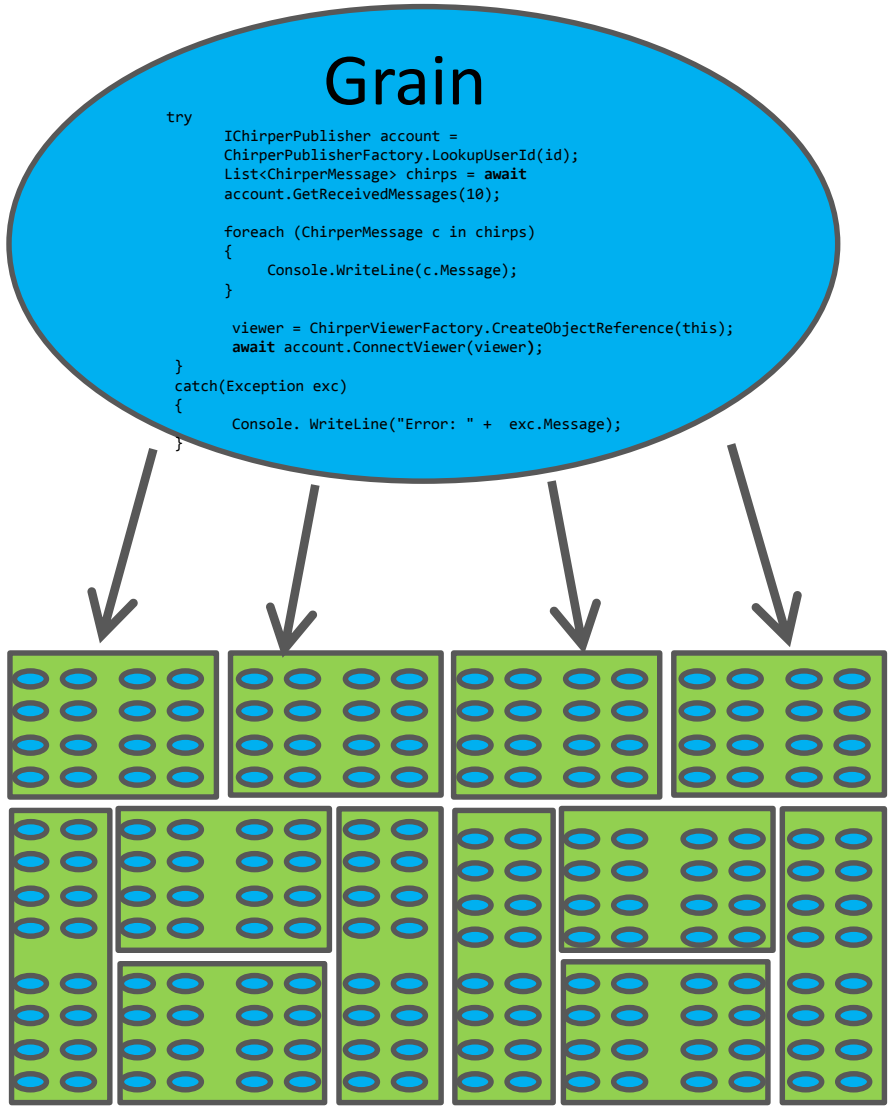
Orleans Programming Model

Application

Programming Model

Orleans Runtime

.NET + Azure



- Distributed, replicated actors, called “grains”
  - E.g., account, user, profile,
- Grains are single-threaded, created when called
- Communication is async and location transparent
- Apps scale through orders of magnitude without rewriting
- Transparent load balancing and fault tolerance

References

- Orleans: Cloud Computing for Everyone
  - Sergey Bykov, Alan Geller, Gabriel Kliot, James Larus, Ravi Pandya, Jorgen Theln
  - ACM Symposium on Cloud Computing, 2011