

Model-Driven Clouds

Matthew Fowler

NT/e

Agenda

1. Introduction
2. Grid/Cloud Background
3. JavaSpaces and GigaSpaces
4. GigaSystemBuilder
5. CloudSave

NT/e

- NT/e
 - New Technology / *enterprise*
 - Formed 1996
- 1998 - Server-side Java. BEA Partners
- 2007-8: £2.5m turnover, £100k profit
- JeeWiz Version 1.0: March 2002
- JeeWiz Version 5: 2008, engine open-sourced
- Key customers: DeutschePost/SOPERA, UBS, BEA, GigaSpaces

Products

Cloud
Save

| | |
|---|-----------------------------------|
|  | <i>J2EE, Struts</i> |
| | <i>Hibernate, Spring, JSF</i> |

GIGASYSTEM
BUILDER

Jeewiz!
Engine

GIGASYSTEM
BUILDER

JeeWiz Engine

- Add-on: work with any IDE
- Simple to start: Java, XML, Velocity, library
- Scripting/interpretive - undeclared attributes
- "Generate 95%; hand-code the rest"
- Large-scale integrated automation
 - Pluggable technology layers - "ecosystem" support
 - Simple model-to-model expansion using XML
 - Multi-tier generation
 - Integrated UI generation from single model

Applications

- Enterprise applications - JeeWiz commercial
- Eclipse Swordfish (SOA Runtime)
- Definition of DSLs
 - Meta-model - XML-Java generation
 - XML filter for customising
 - Makes starting a family easy
- XML-XML transformations
- Scripting - e.g. codebase-specific *diff*

JeeWiz - J2EE and Lite

- Original: Struts, J2EE mid-tier and DB
- Latest: Spring, Hibernate, JSF, CSS, Trinidad
- Modelled in XML, Rational Rsm/Rsa, Rose etc.
- > 1,000 templates - M2M & M2T
- Generation %
 - M2M expands original model by 30-50
 - 97-95% generation

The hierarchy of needs

- Levels
 - Isomorphic format transformation
 - Expression of intent - know-how
 - Best practice
 - Inter-layer management and coordination
 - Architecture and other variations
- External "abstraction-raisers" like Spring
 - make "know-how" harder to get a handle on
 - May also affect others, but less

Where do Grids End & Clouds Begin?

- Grids: lots of interconnected computers
 - Historically for technical calculations
- Clouds: on-demand grids, by the hundreds

Grid/Clouds - The Long-Term View

- Applications are getting bigger
 - 1994: 20 million requests (r/w) per day
 - 2009: 1 billion PCs
 - 4 billion mobile phones
 - 600 billion RFID tags in use
 - 30 million ad clicks/day on Google
 - 30 million mobile users active on Facebook
 - 30 million photo uploads/day on Facebook
- Single-core CPU performance is almost static
 - but main memory is still on Moore's law

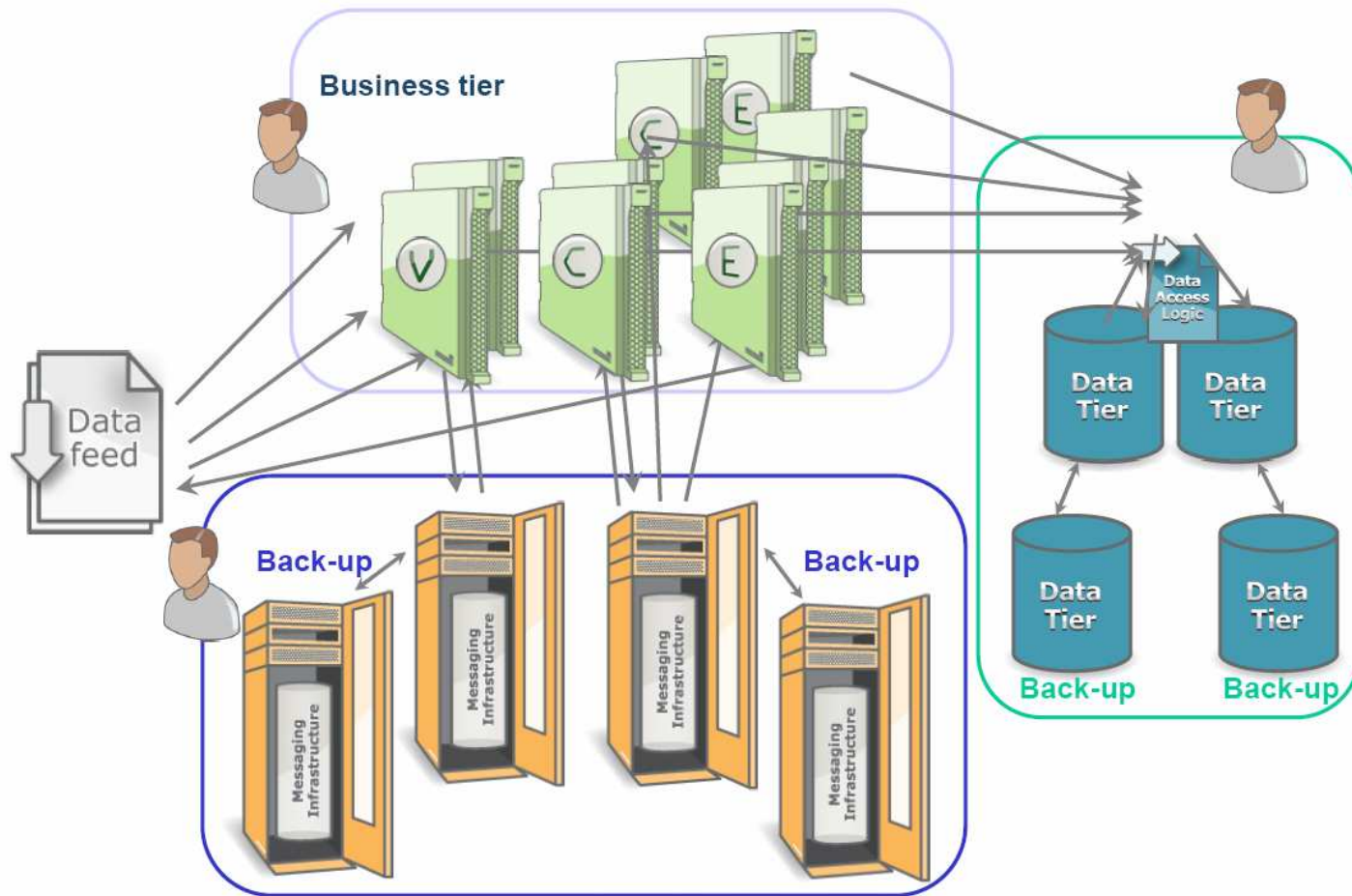
More numbers

- Windows Live (huh?)
 - 130,000 servers
 - 3.2 million page views ... per second
 - 200,000 emails/txts ... per second
- Windows Azure Chicago centre - 350k servers
- Google - >> 500,000 servers
- The "50,000 server club"
 - mostly companies you've never heard of!

Applications Growing Up

- X per Y
 - 1980: many applications per computer
 - 2010: many computers per application
- Architecture evolution
 - Database centric
 - Cache centric - "compute grid", "data grid"
 - In-Memory Data Grid

Computing with Tears?



Problems with Tier-based Apps

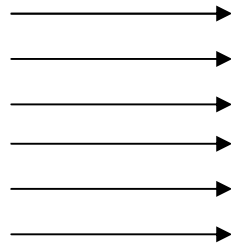
- Doesn't scale linearly
 - Growth*10 \Rightarrow rearchitect
- Failover is complicated
 - Often tiers impact each other
 - Redundancy can dramatically increase
 - network bandwidth
 - latency
- Network hops waste time
- Disks are slow

Other Cloud Computing Strands

- Data centre automation
- Instant deployment via clouds
 - Great for proving volume capability
 - "I want 200 servers, for a day"
"That'll be \$500 then"
- Virtualisation
 - Improve utilization from 10-15% to 50%

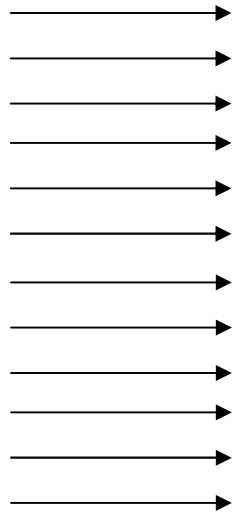
Scaling 0 - lots of cores

processing

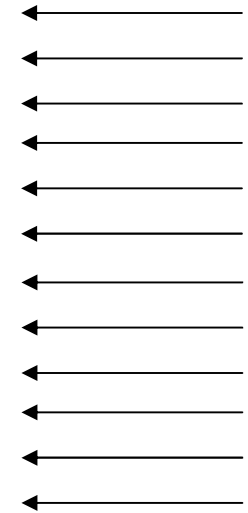


Scaling 1 - scale out - bigger

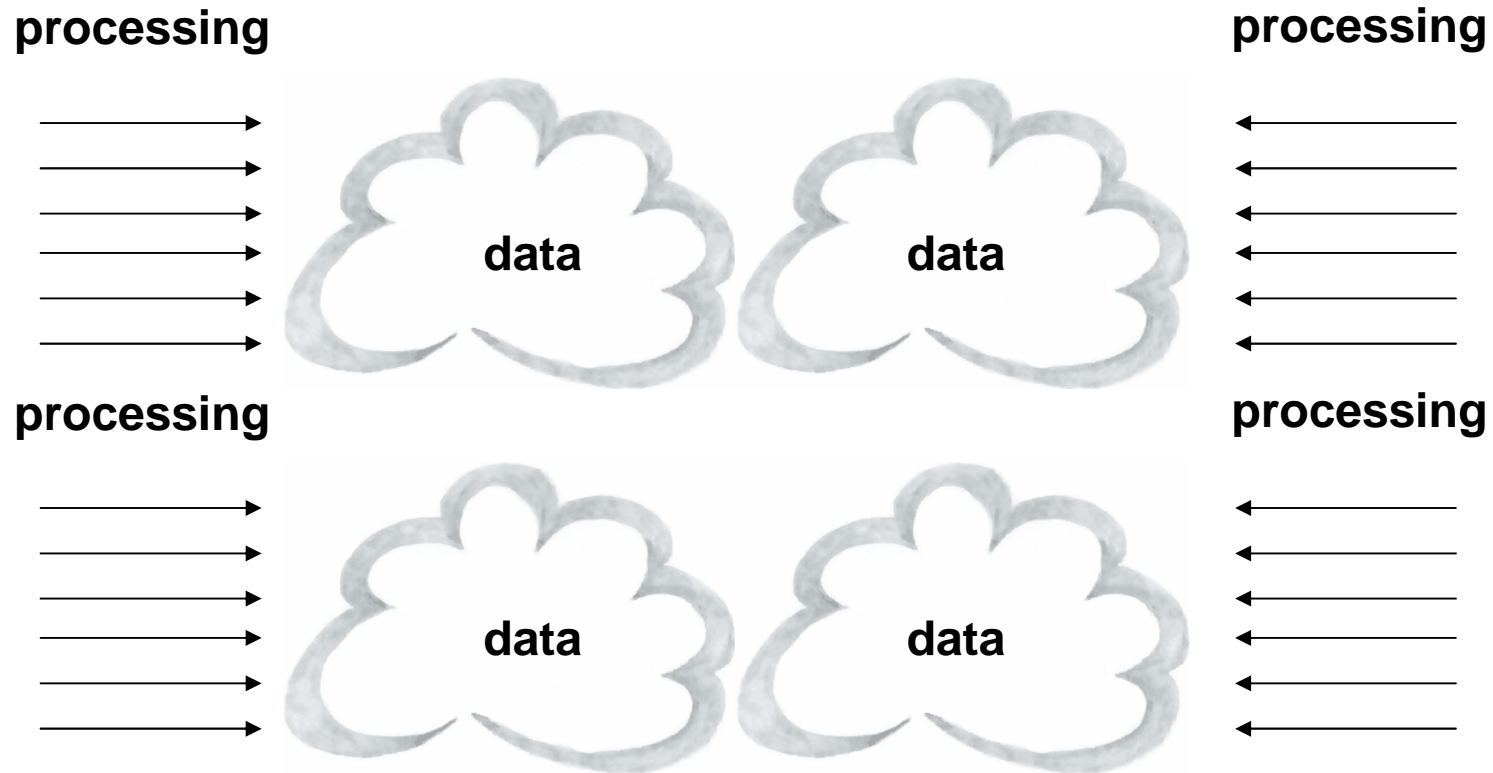
processing



processing



Scaling 2 - scale up - more nodes (needs partitioning)



JavaSpaces and GigaSpaces

- Distributed "Spaces" - data buckets, + CPU
 - Content-searchable (SQL)
 - automatic backup \Rightarrow failure protection
 - two main memory backups more reliable than disk
- Partitioning
 - routing on content (sharding) \Rightarrow linear scalability
 - handle very large datasets/high CPU requirements
- SLA-based scale out
 - start with small instances
 - scale out to large machines, temporarily

Architecture

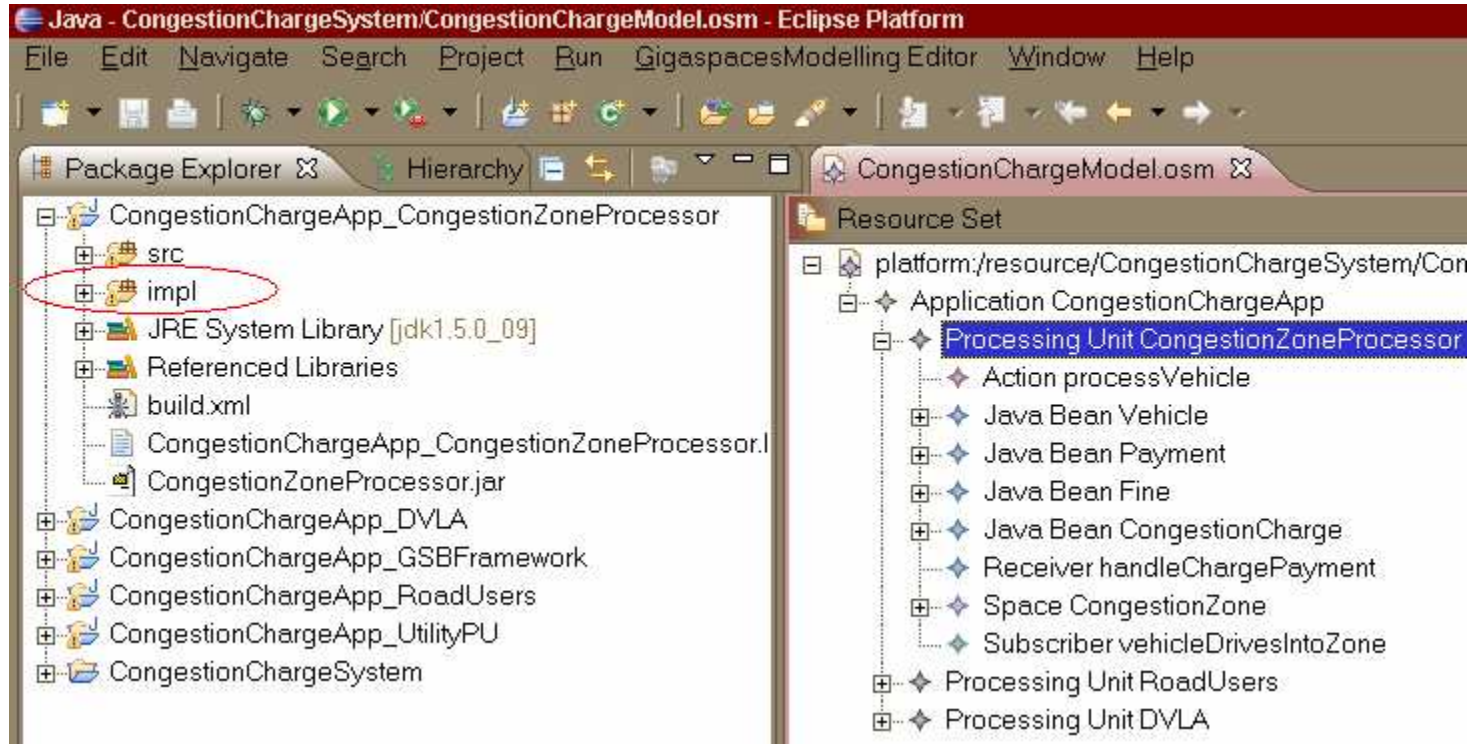
- Turn architecture inside-out to get 10x speed-up
 - Don't move the data - SOA-like integration layer
 - Business object functions in the space with data
- Event-based processing
 - Call out based on object state in space
 - Scalable, resilient message queues
- Service-based facade
 - Scalable, resilient SOA
 - SOA Spaces (not DB) as the integration point

3. GigaSystemBuilder Goals

- Fast demos
- Faster Proof Of Concepts (4 days)
- Faster to deployment (\$)
- Increase the developer base, remove the hurdles
- Expand GigaSystem ecosystem delivery capacity

GigaSystemBuilder Tour

- www.nte.co.uk/java/gsb.html



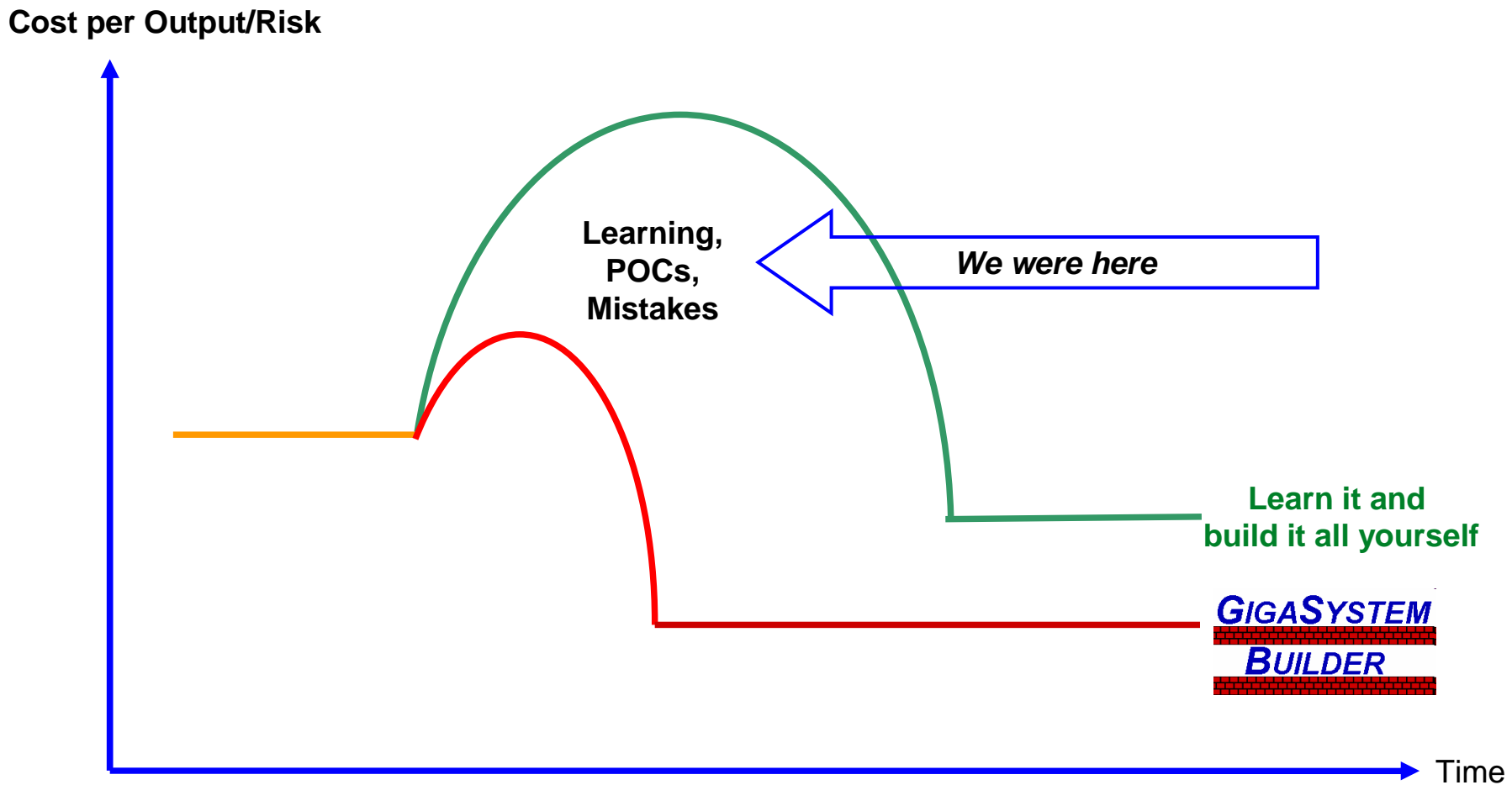
Development - The Easy Bits

- DSL mechanics
 - Meta-model definition
 - Filter into Eclipse Modelling environment
 - Generation of classes, services and config

GSB Development - Difficulties

- The correct abstractions
 - V0 - Thrown away
 - V1 - Event-driven development
 - V2 - WS/SOA integration, deployment automation
- Starting distributed applications
 - Multi-threading, workflow definition
 - Essential for rich interacting space applications
- Alignment of best practices/patterns → J2EE

GSB Development - The Reality



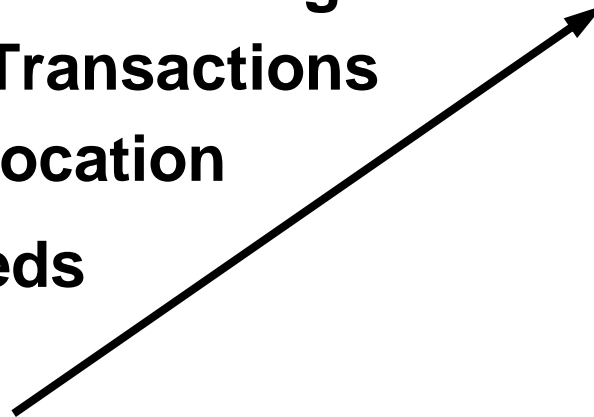
**Cloud
Save**



**Big Data
Transactions**

Optimal Co-location

Commit at Grid Speeds





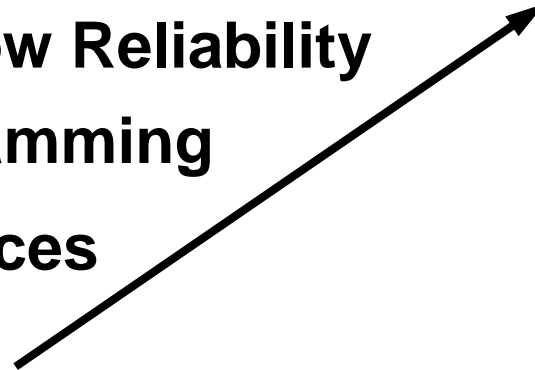
 **Cloud**
of
Unknowing

~~Distributed Transactions~~

Low Reliability

Complicated Programming

Unintended Consequences



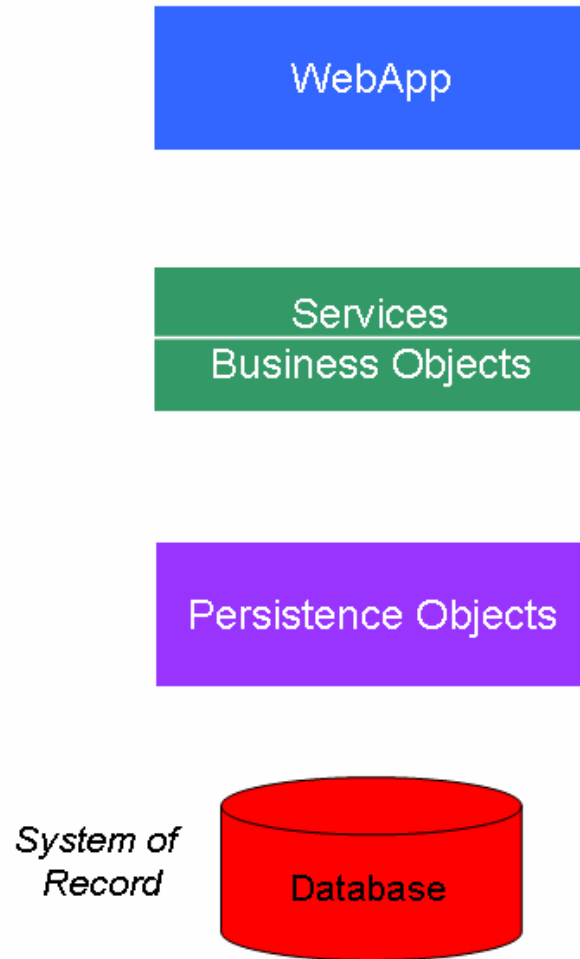


CloudSave

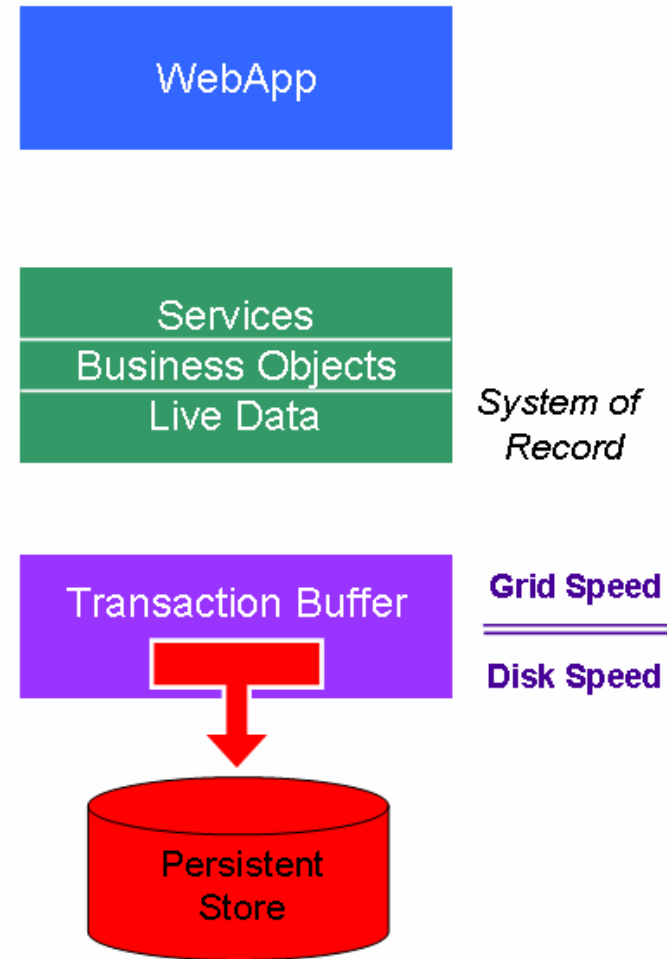


- ♥ As safe as a database, as fast as the Cloud
- ♥ Distributed Transactions
- ♥ Optimal Placement - Modelled
- ♥ Reduce Unintended Consequences
- ♥ Simple API - better than Hibernate/JPA

c 2000



c 2010



In-Memory Data Bases - Are You Crazy?

- What's it worth:
 - Loss of sales - down by 7% from 5 - 6 seconds
 - Business justification for \$100m/year co:
 - \$7m/year for performance
 - cost of Amazon 8Gb AMI: \$2,400/yr
 - *2 for software costs? = \$5,000/yr say
 - PAYS FOR 1,400 SERVERS, 1.1TB in-memory
- ⇒ Reduce overheads by space-based architecture
- ⇒ Use in-memory DB

As Safe As A Database?

- We say
 - I accept the order before writing to disk
- But 'n' backup machines are as safe as you like
 - '1' should be enough, two for HA
- Still write to disk
 - Power failure etc.
 - Poor track record of large-scale infrastructures

End