



RED HAT® JBOSS®
MIDDLEWARE

Microservice Design Patterns for Java Applications

Arun Gupta, Red Hat

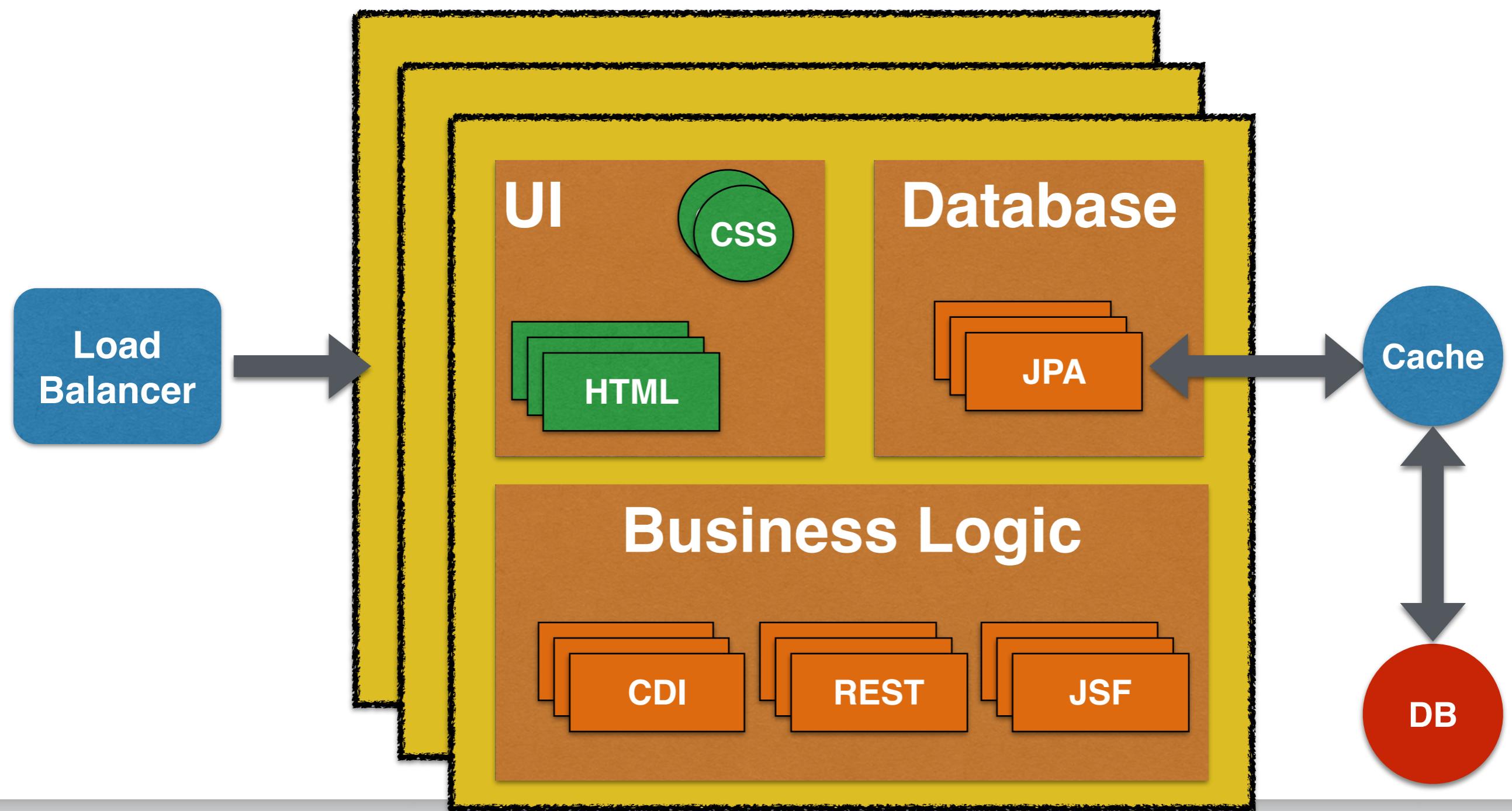


Arun Gupta

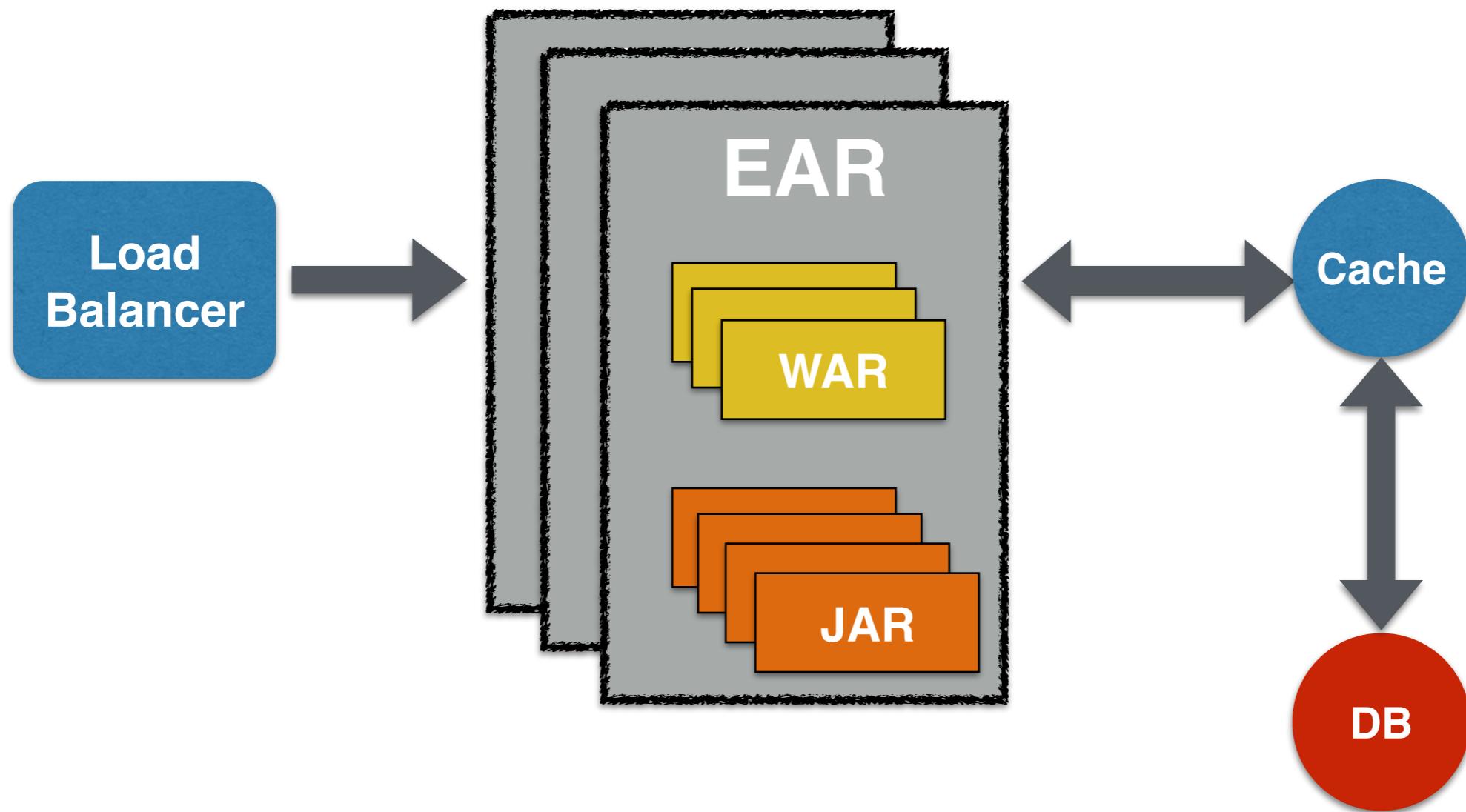
Director, Technical Marketing
& Developer Advocacy

@arungupta
blog.arungupta.me
arungupta@redhat.com

Monolith Application



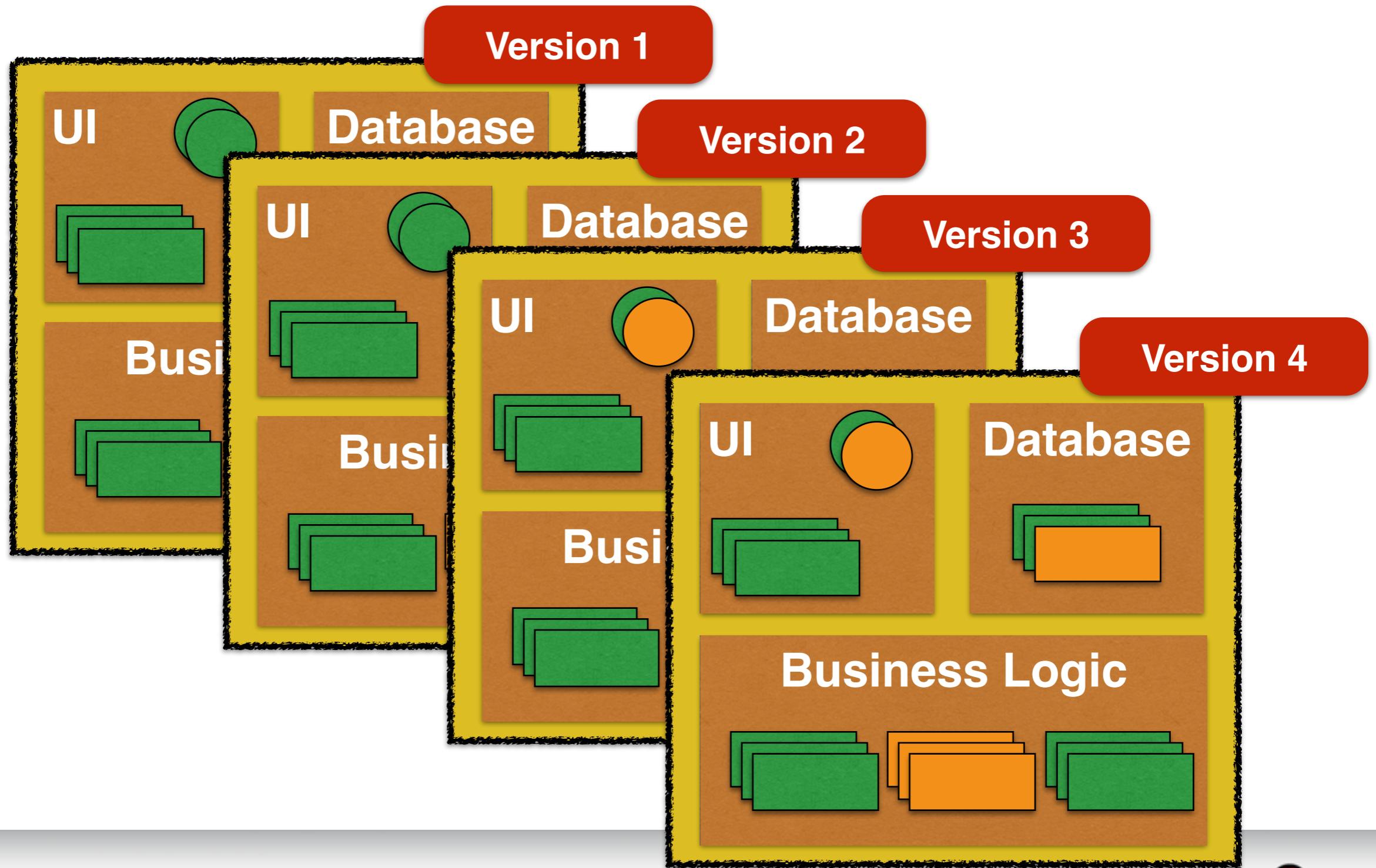
Monolith Application



Advantages of Monolith Application

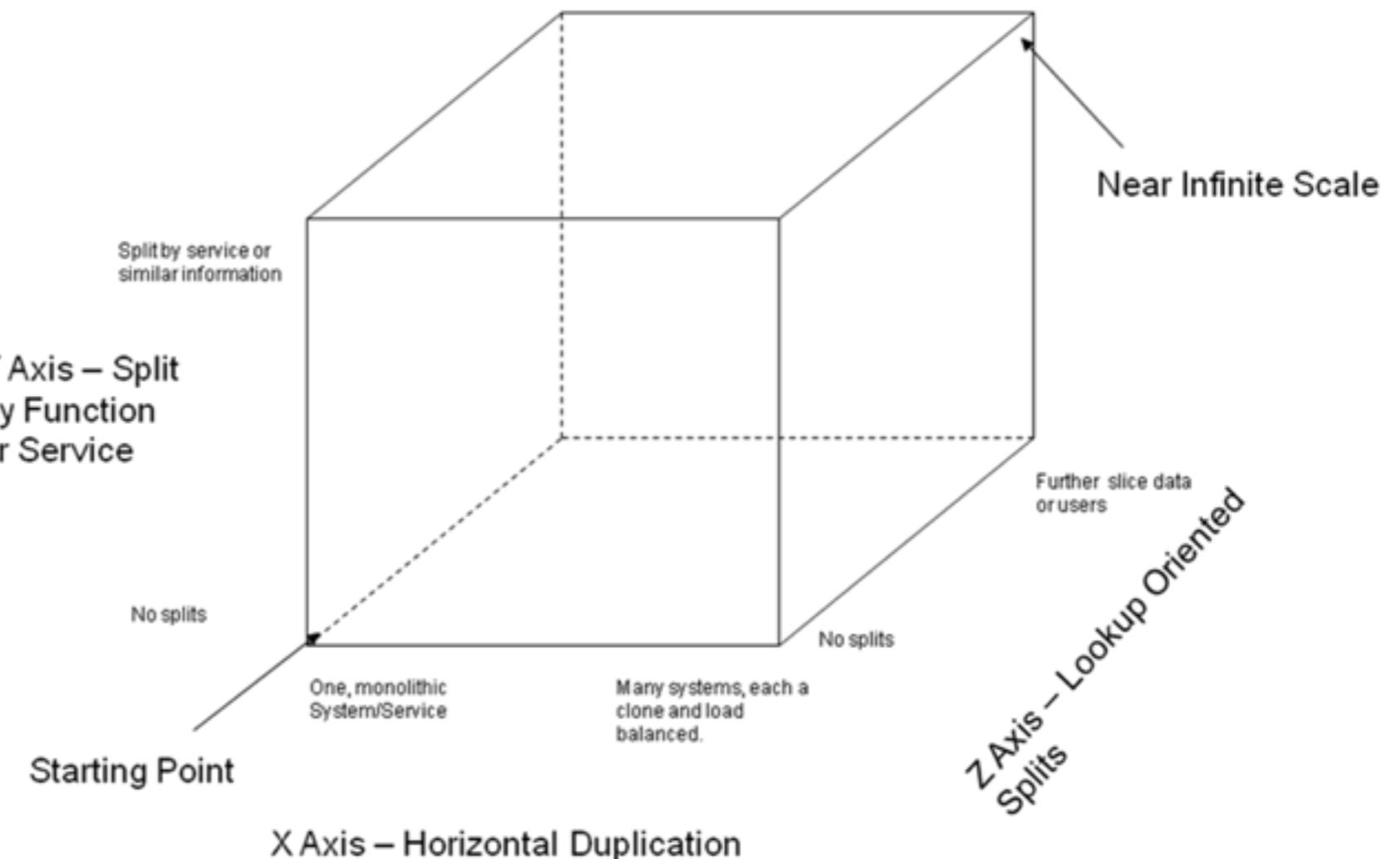
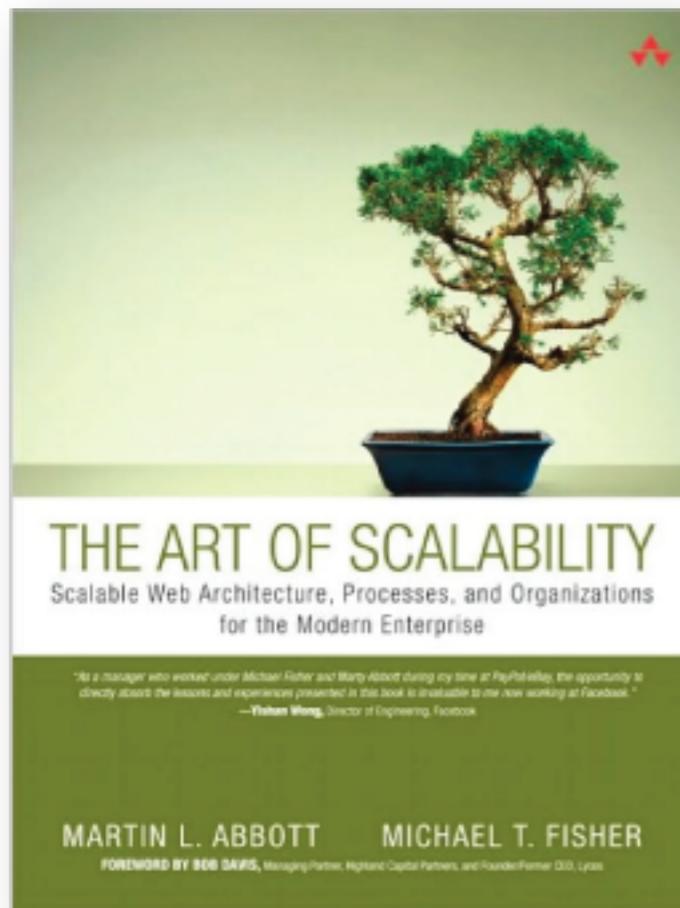
- Typically packaged in a single `.ear`
- Easy to test (all required services are up)
- Simple to develop

Monolith Application



Disadvantages of Monolith Application

- Difficult to deploy and maintain
- Obstacle to frequent deployments
- Makes it difficult to try out new technologies/ framework



<http://akfpartners.com/techblog/2008/05/08/splitting-applications-or-services-for-scale/>



An ***architectural approach***, that emphasizes the ***decomposition of applications*** into ***single-purpose, loosely coupled*** services managed by ***cross-functional teams***, for delivering and maintaining ***complex software systems*** with the velocity and quality required by today's ***digital business***.

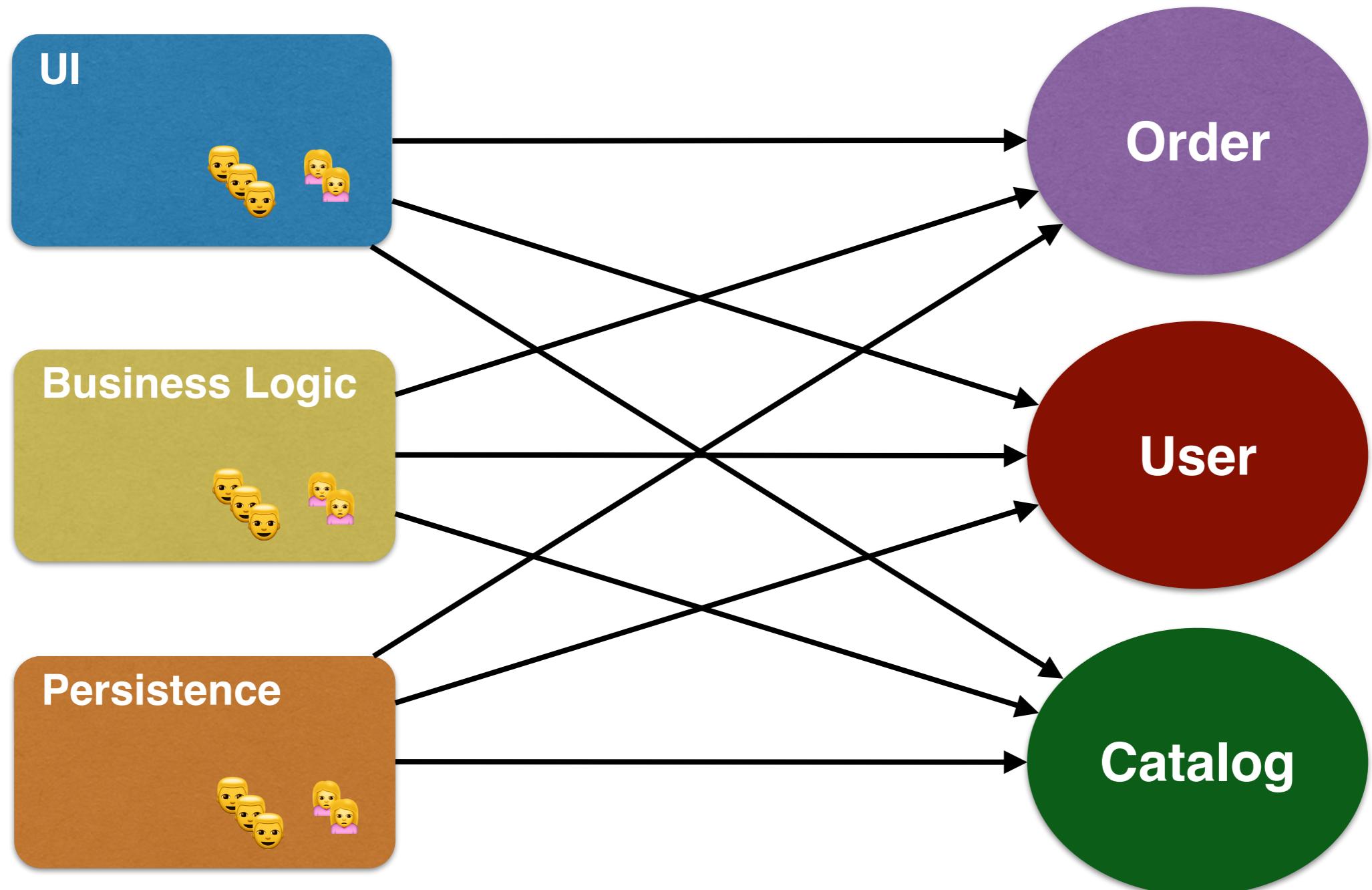


<http://martinfowler.com/articles/microservices.html>

I DONT ALWAYS
BUILD EVERYTHING



imgflip.com



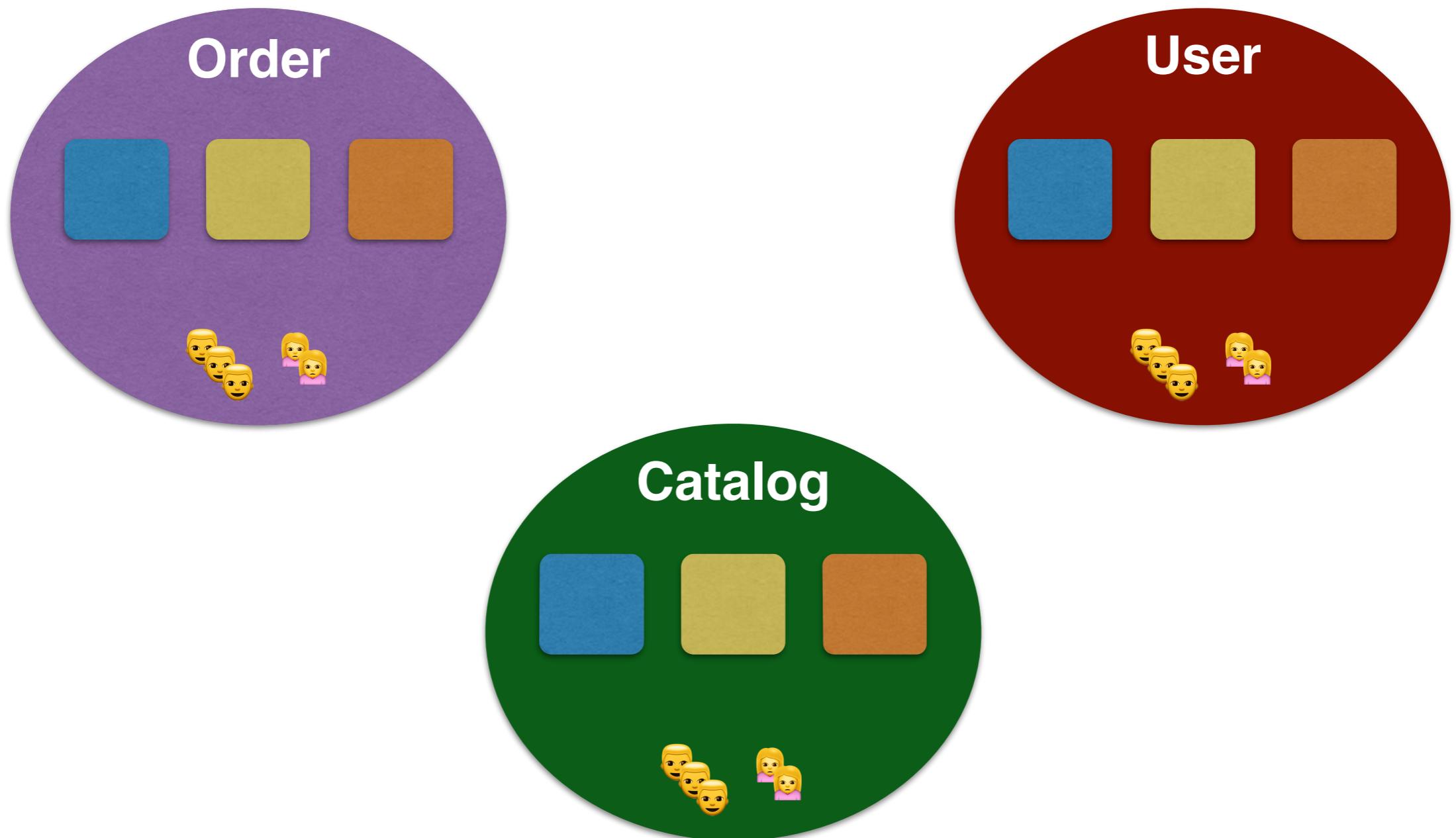


“Any ***organization*** that designs a system
(defined more broadly here than just information
systems) will inevitably produce a design
whose structure is a ***copy of the***
organization's communication structure.”

–Melvin Conway

http://www.melconway.com/Home/Committees_Paper.html

Teams around business capability



Single Responsibility Principle

DO
1
THING

Explicitly Published interface



Independently replaceable and upgradeable





With great
power, comes great
responsibility



“you build it, you run it!”

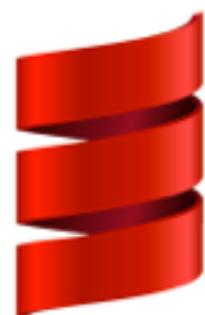
Designed for failure



Fault tolerance is a requirement, not a feature



Characteristics



Scala



ORACLE
D A T A B A S E



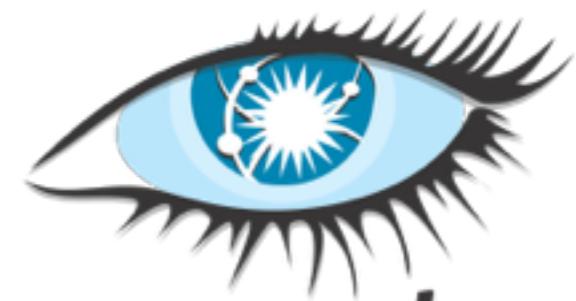
PostgreSQL



Couchbase



redis



cassandra



100% automated

Sign Off | Home | Locations | Contact Us | Open Account

WELLS FARGO 

Accounts Bill Pay Transfers Brokerage Account Services Messages & Alerts

Bill Pay Overview Payments Payees eBills Reports Notices User Profile

Bill Pay Overview [Help](#) [Unviewed Notices \(2\)](#) [Unpaid eBills \(3\)](#) [Pending Payments \(6\)](#)

Make Payment

Note: Delivery time for payment varies by payee. See number of business days in Send On column.

Payee Add a Payee	Pending Payment	Last Paid	Amount	Send On
AMERICAN EXPRESS	\$2,053.50 06/30/2004	\$1,349.93 05/24/2004	\$ <input type="text"/>	mm/dd/yyyy 3 Business Days
BANK OF AMERICA  Receiving eBills View eBill	\$198.80 06/30/2004	\$92.17 05/25/2004	\$ <input type="text"/>	mm/dd/yyyy 3 Business Days
BANK ONE / FIRST	\$55.00 06/25/2004	\$55.00 05/22/2004†	\$ <input type="text"/>	mm/dd/yyyy 3 Business Days
CHARLES SCHWAB			\$ <input type="text"/>	mm/dd/yyyy 5 Business Days
CITIBANK VISA  Pending activation	\$63.50 06/30/2004*	\$198.80 05/25/2004*	\$ <input type="text"/>	mm/dd/yyyy 5 Business Days
DIRECT TV  Activate eBills		\$63.50 05/25/2004	\$ <input type="text"/>	mm/dd/yyyy 3 Business Days
SBC-PACIFIC BELL		\$45.80 05/25/2004	\$ <input type="text"/>	mm/dd/yyyy 5 Business Days
SPRINT PCS  Activate eBills	\$49.78 06/30/2004	\$63.50 06/26/2004	\$ <input type="text"/>	mm/dd/yyyy 5 Business Days
SFPUC-WATER DE			\$ <input type="text"/>	mm/dd/yyyy 3 Business Days
WF HOME MORTGAGE		\$1,349.93 05/25/2004	\$ <input type="text"/>	mm/dd/yyyy 5 Business Days

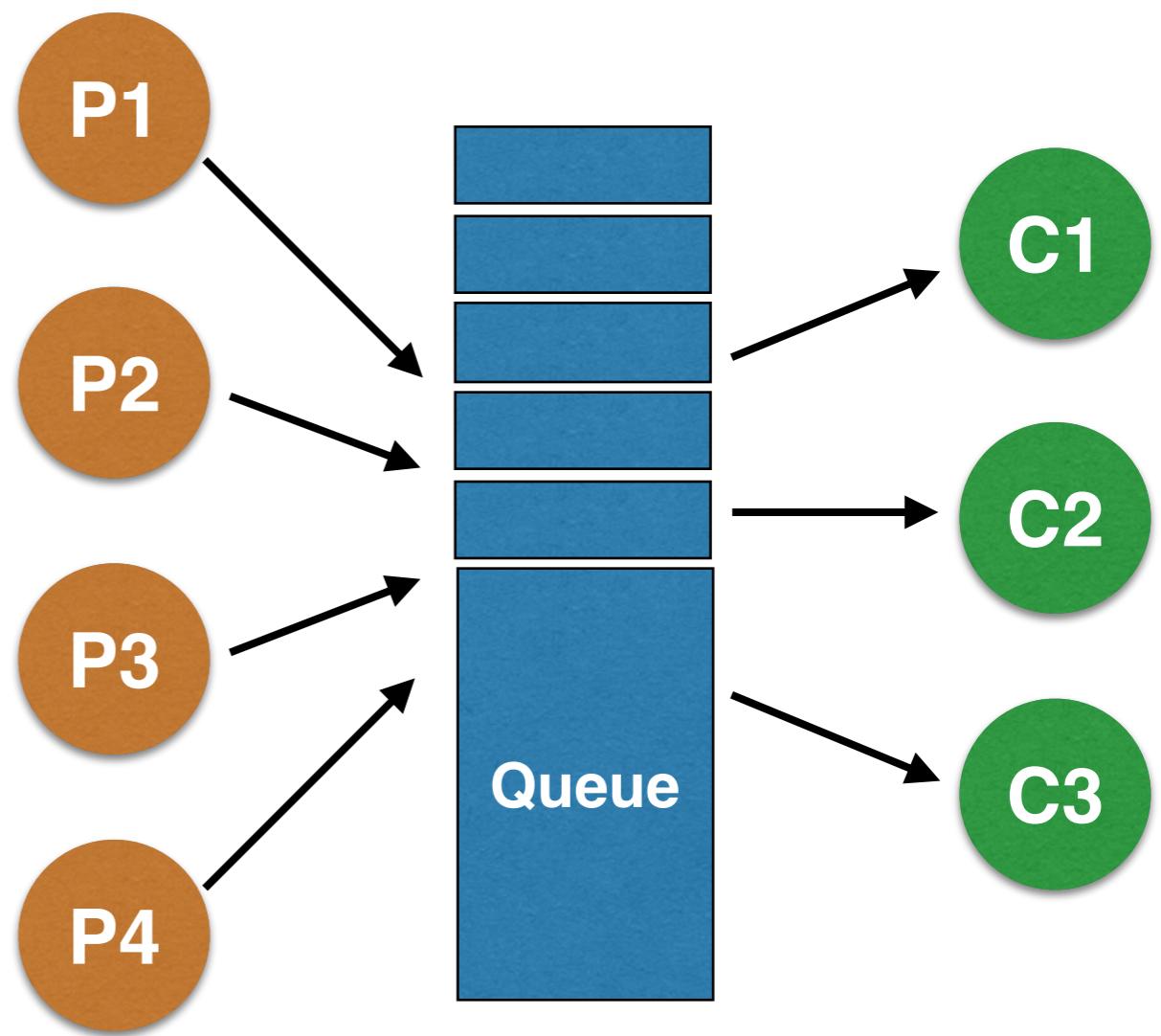
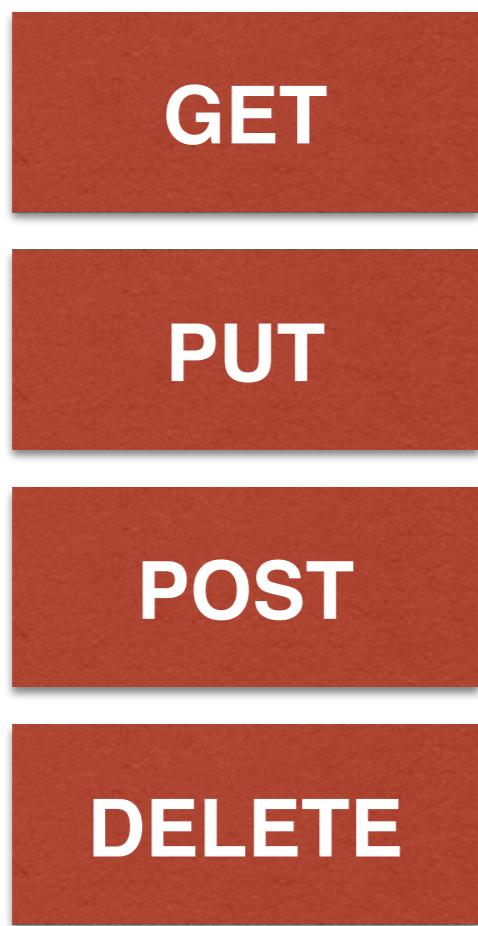
Make Payment

[Home](#) | [Locations](#) | [Contact Us](#) | [Open Account](#) | [Sign Off](#)
© 2001-2005 Wells Fargo. All rights reserved.

Sync or Async Messaging



REST vs Pub/Sub



“Smart endpoints Dumb pipes”



SOA

- SOA 2.0
- Hipster SOA
- SOA done right
- SOA++

SOA 2.0?



Arun Gupta
@arungupta

Microservices = SOA -ESB -SOAP -
Centralized governance/persistence -
Vendors +REST/HTTP +CI/CD +DevOps
+True Polyglot +Containers +PaaS WDYT?

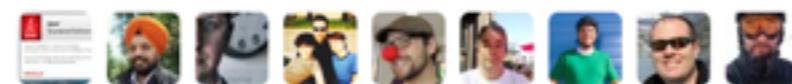


RETWEETS

72

FAVORITES

63



5:07 PM - 27 May 2015

- Conway's Law
- Service Discovery
- Immutable VM

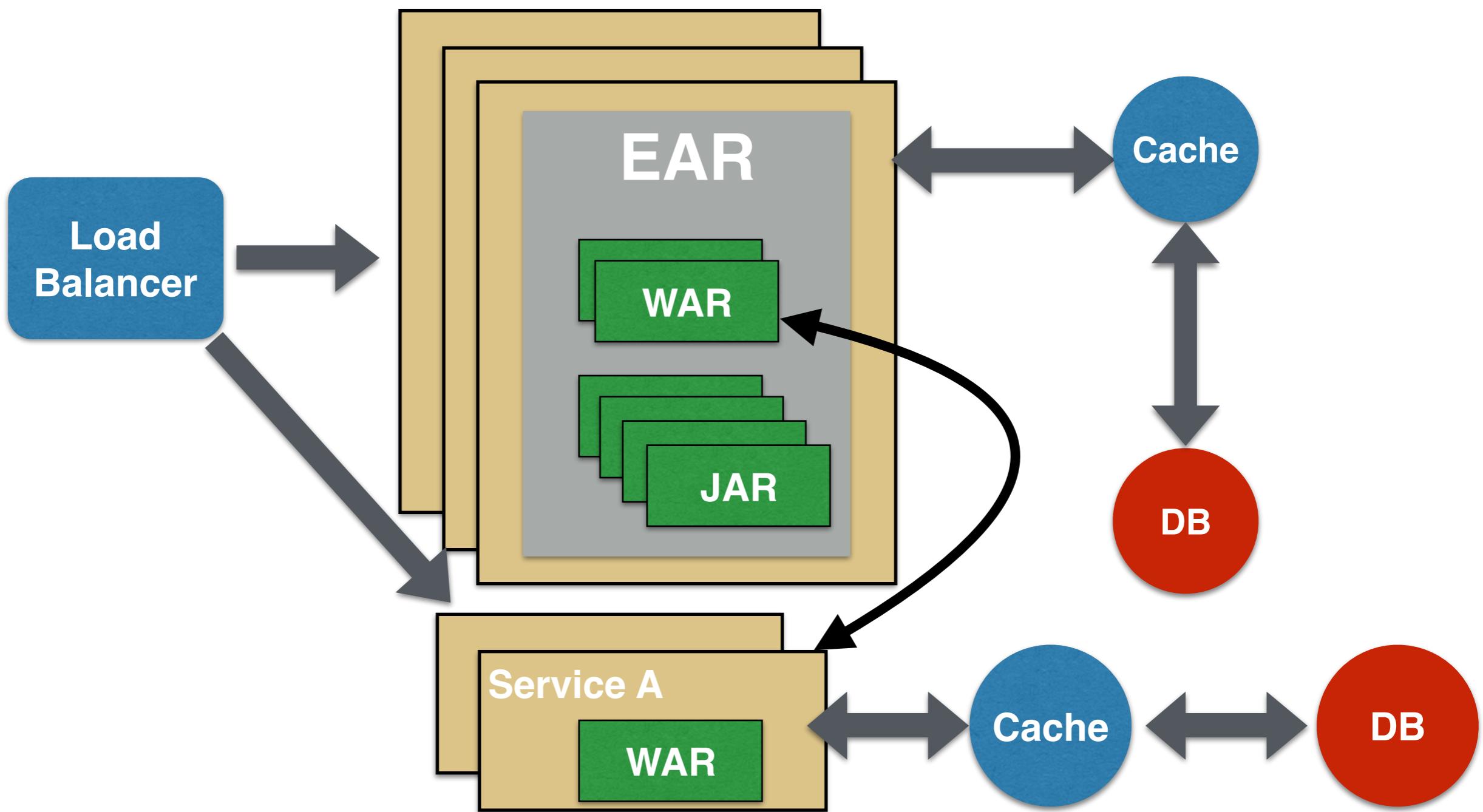
Strategies for decomposing



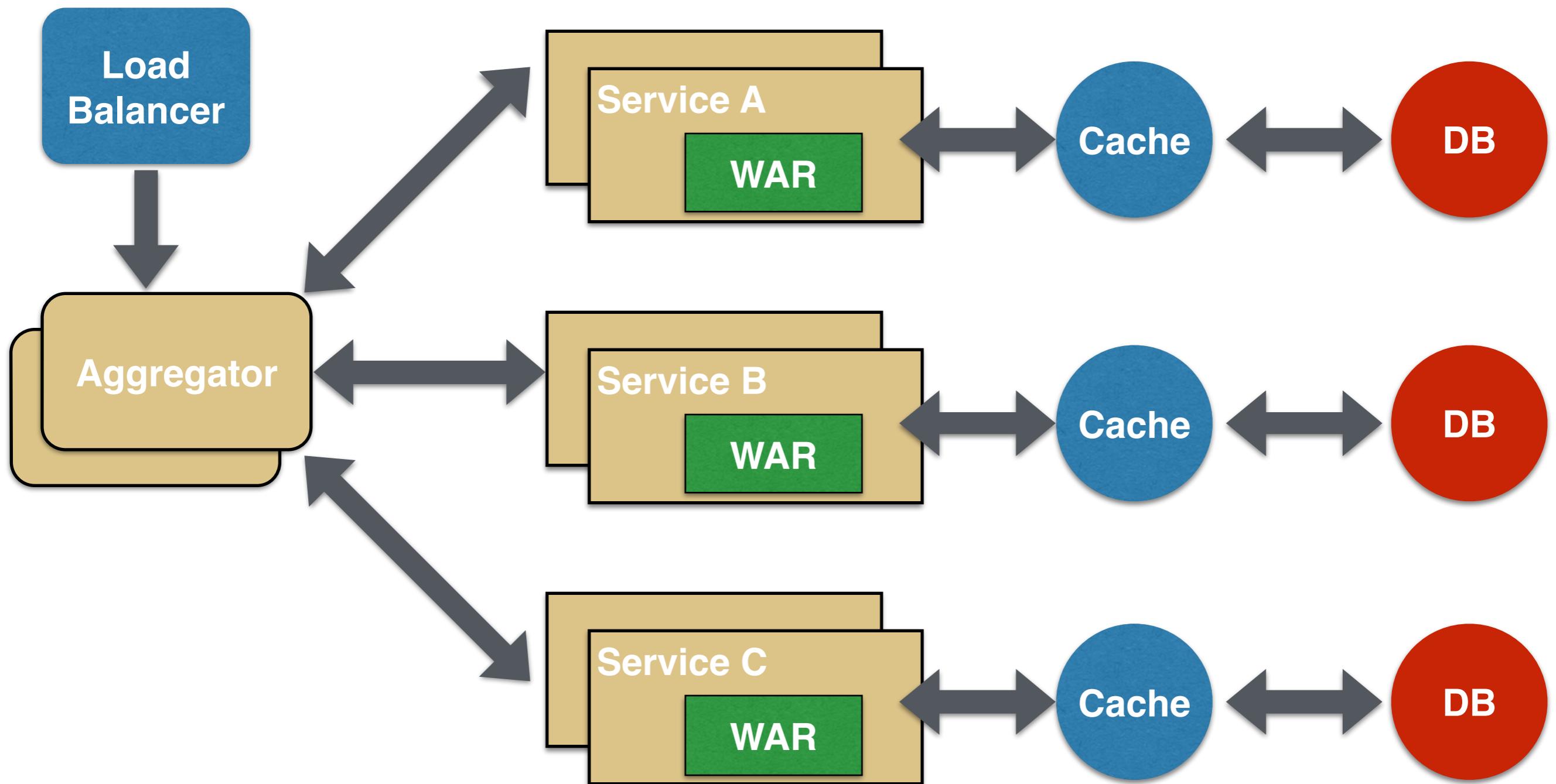
Strategies for decomposing

- Verb or usecase - e.g. Checkout UI
- Noun - e.g. Catalog product service
- Single Responsible Principle - e.g. Unix utilities

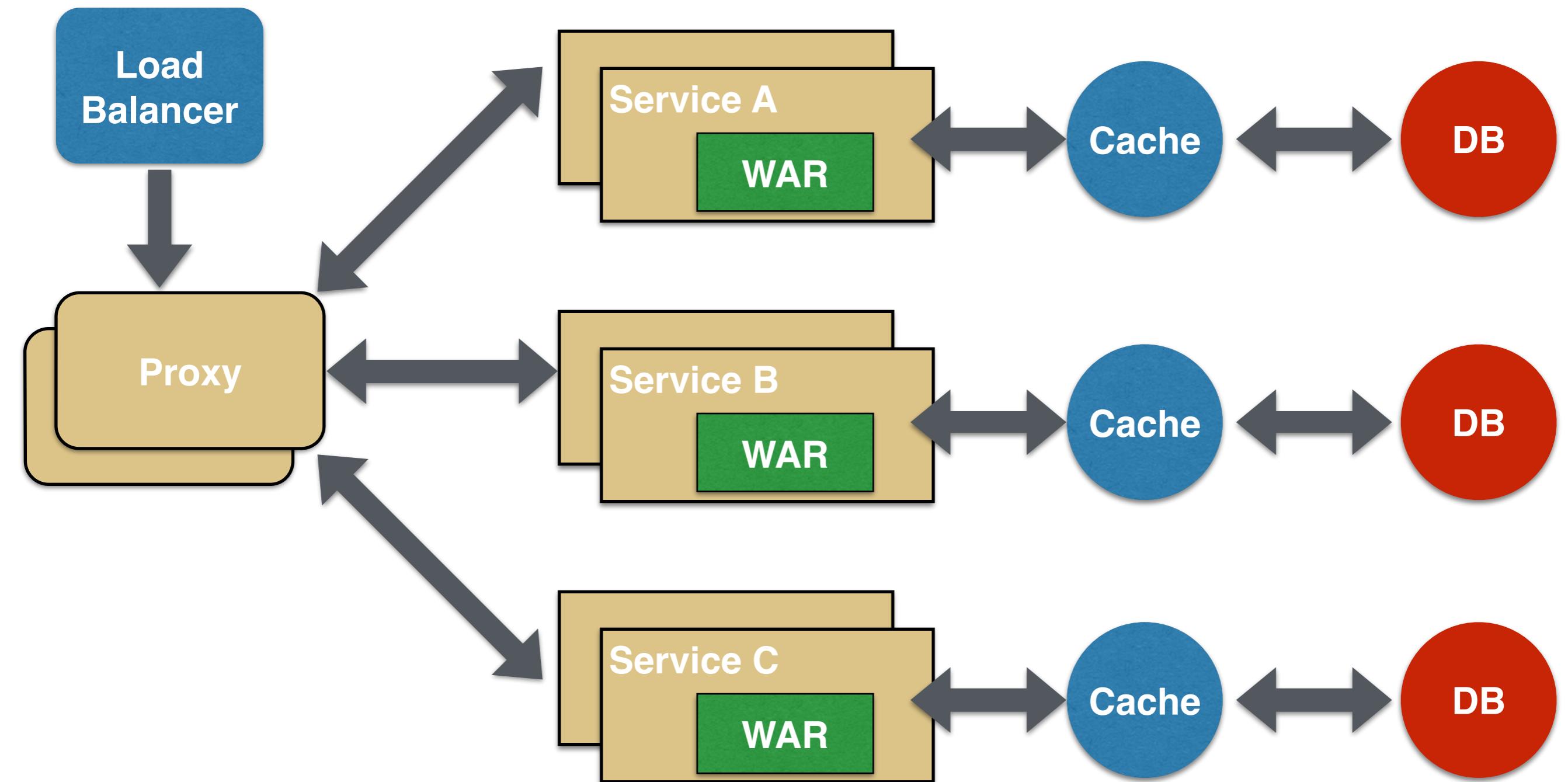
Towards microservices



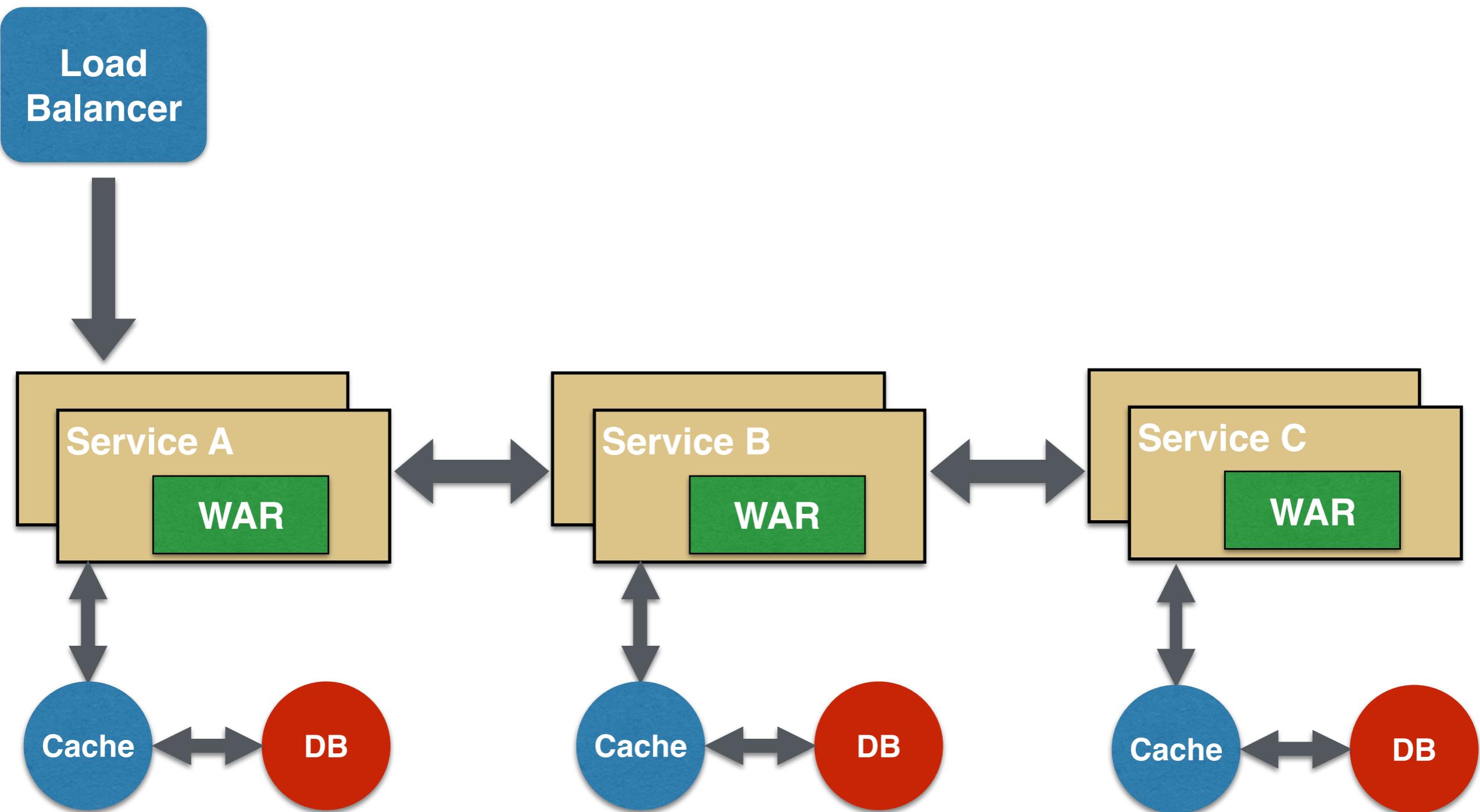
Aggregator Pattern #1



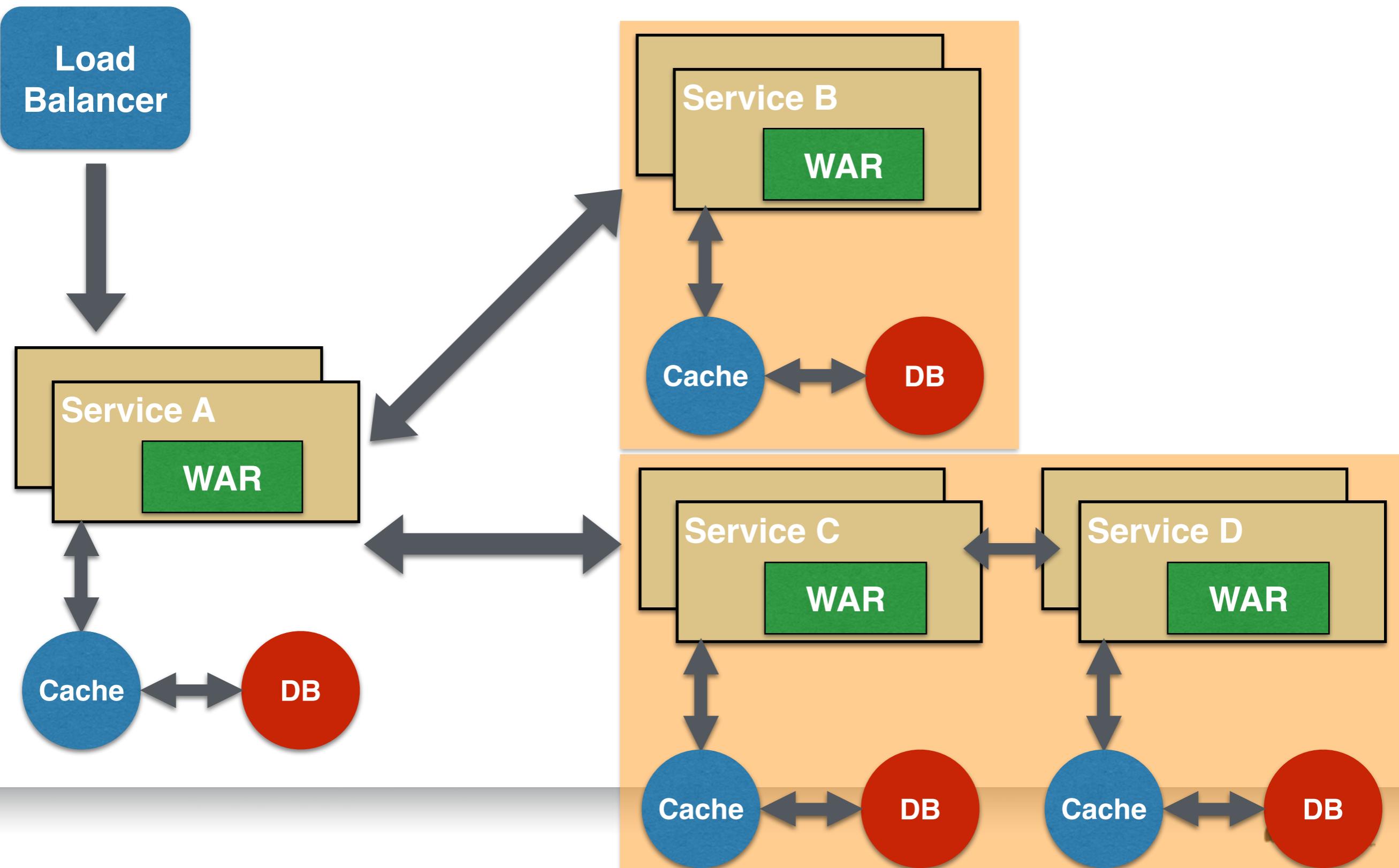
Proxy Pattern #2



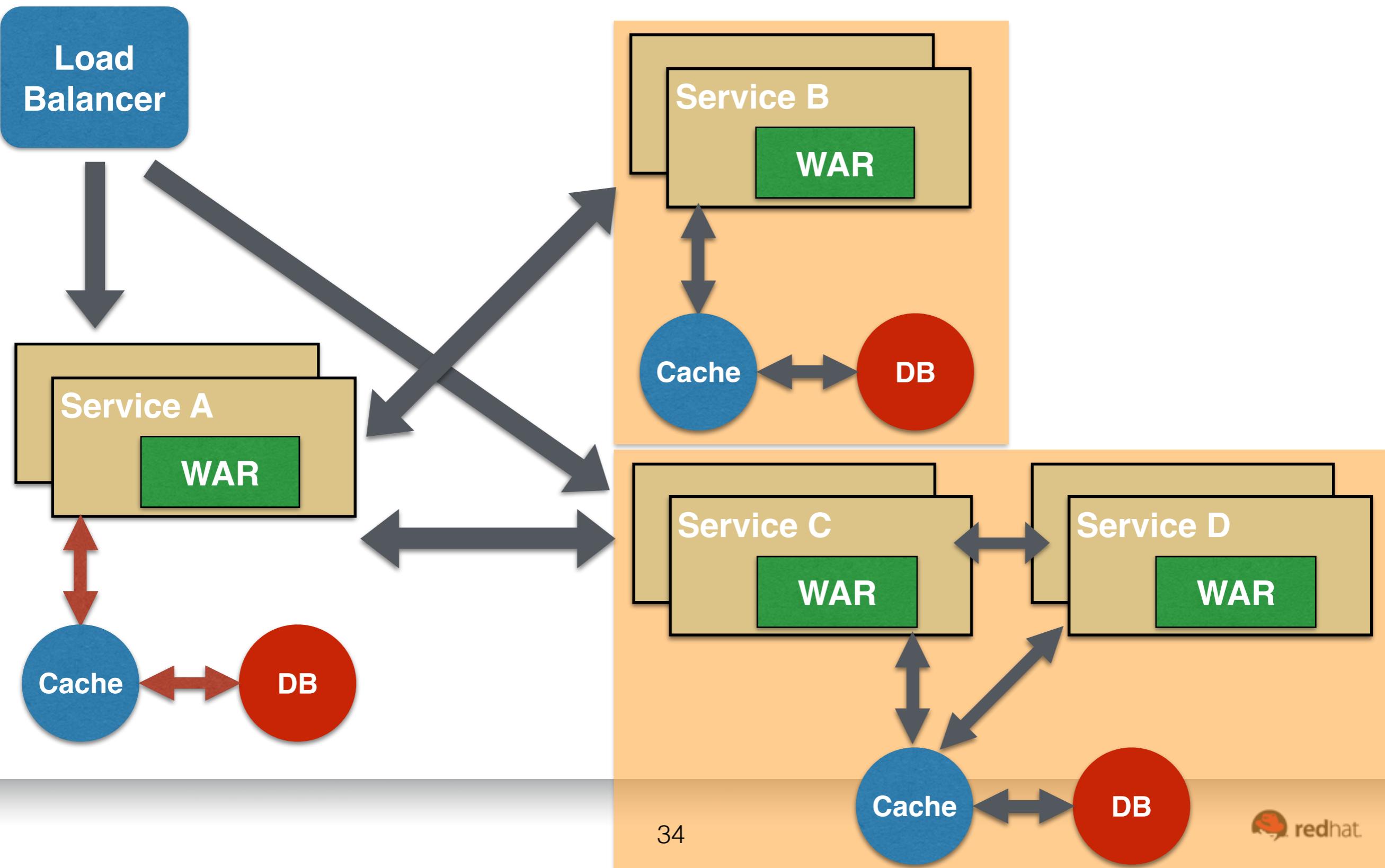
Chained Pattern #3



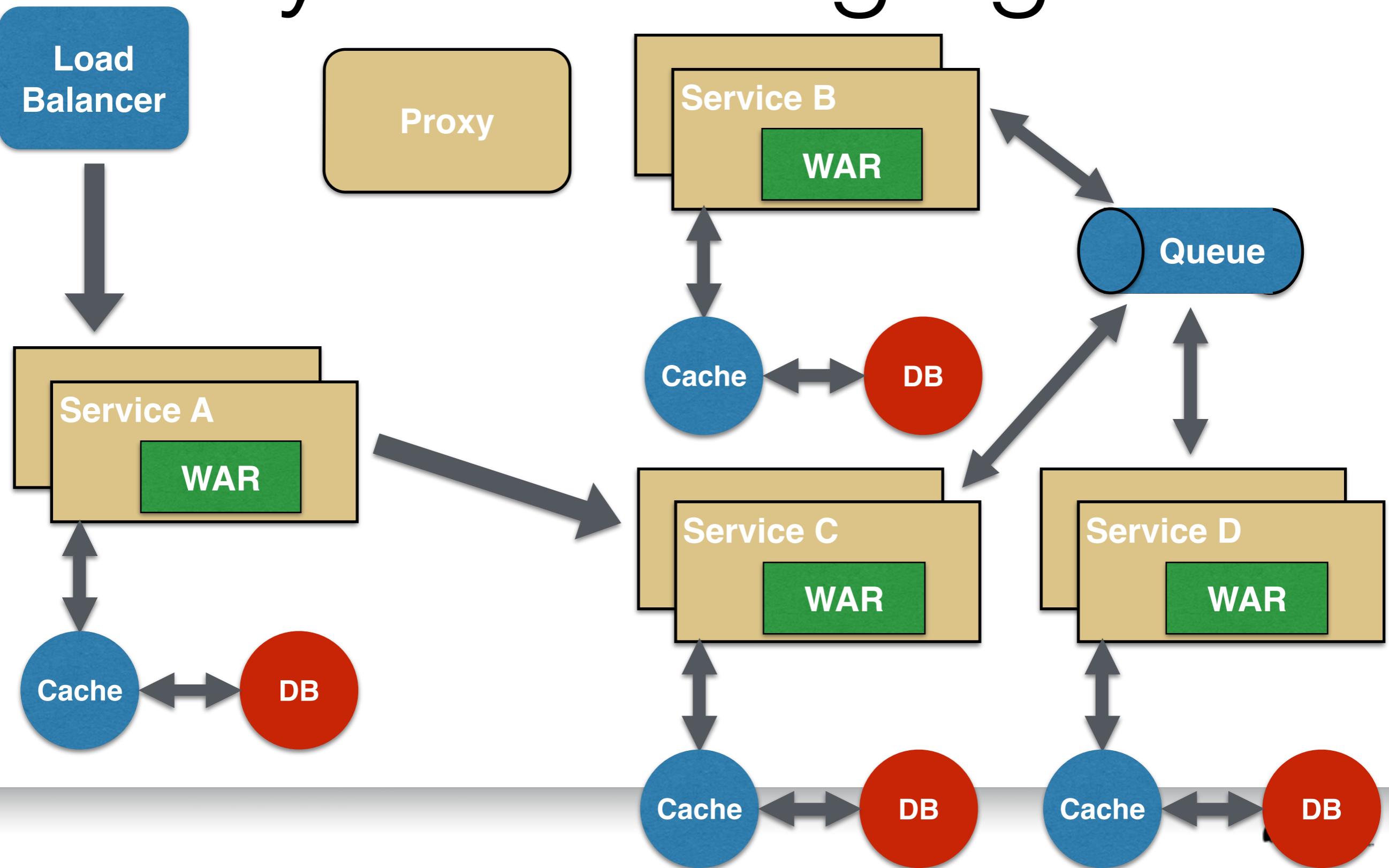
Branch Pattern #4



Shared Resources #5



Async Messaging #5



SAY MICROSERVICE



ONE MORE TIME

memegenerator.net

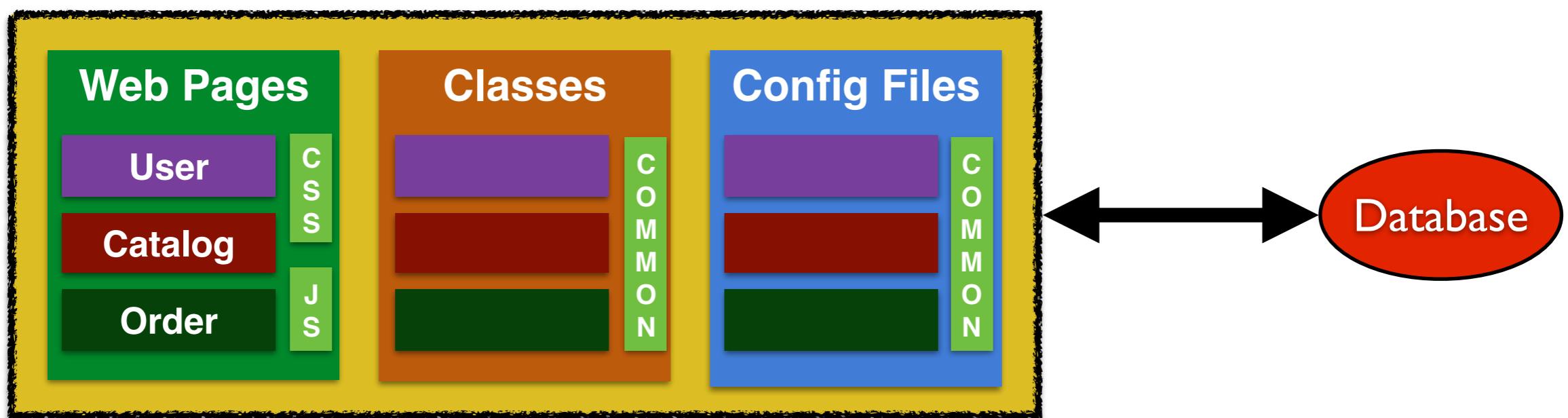
Advantages of microservices

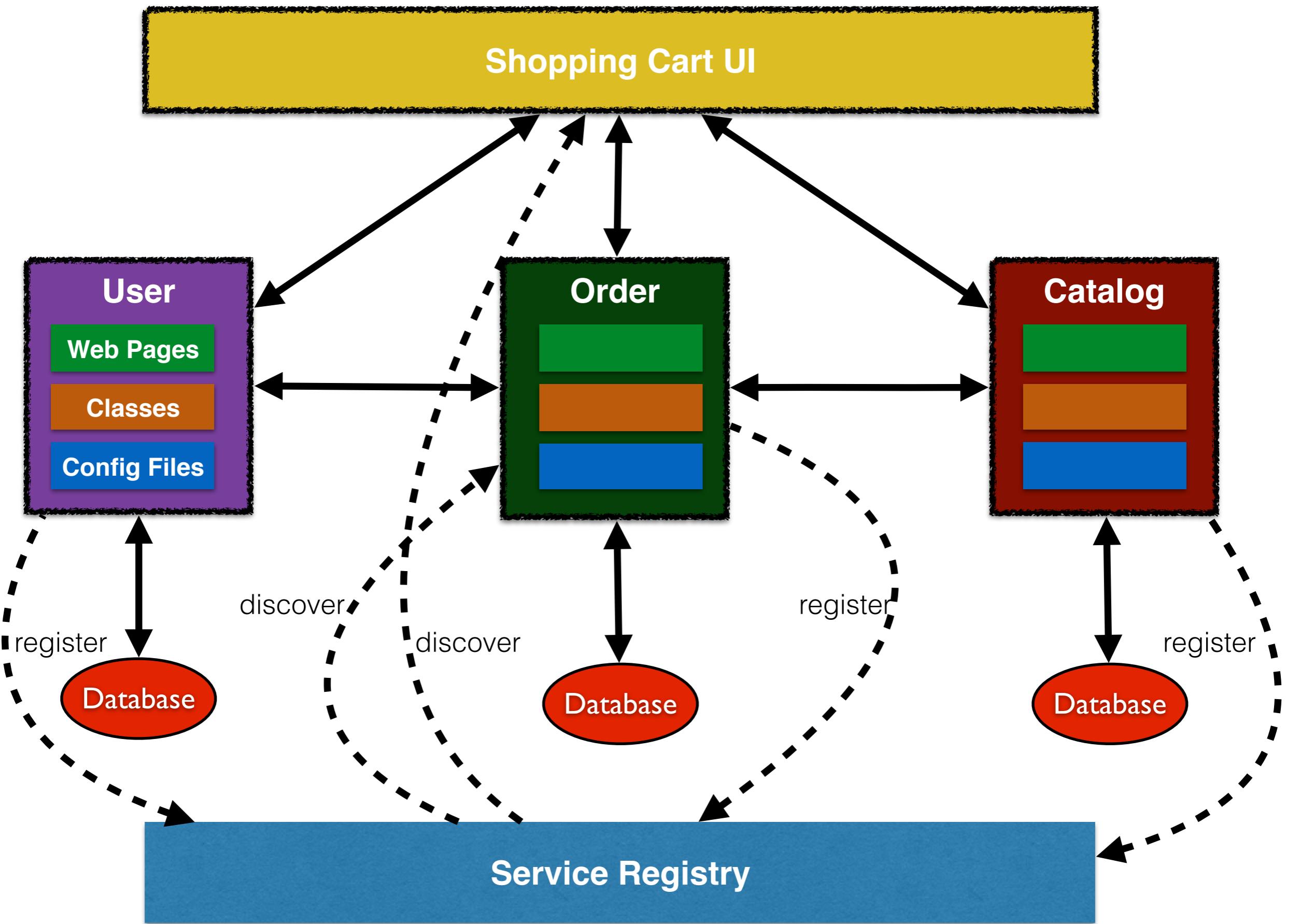
- Easier to develop, understand, maintain
- Starts faster than a monolith, speeds up deployments
- Local change can be easily deployed, great enabler of CD
- Each service can scale on X- and Z-axis
- Improves fault isolation
- Eliminates any long-term commitment to a technology stack
- Freedom of choice of technology, tools, frameworks



“If you can't build a [well-structured] monolith, what makes you think microservices are the answer?”

http://www.codingthearchitecture.com/2014/07/06/distributed_big_balls_of_mud.html





Monolith vs Microservice

	Monolith	Microservice
Archives	1	5 (Contracts, Order, User, Catalog, Web)
Web pages	8	8
Config Files	4 (persistence.xml, web.xml, load.sql, template.xhtml)	3 per archive
Classes	12	26 (Service registration/discovery, Application)
Archive Size	24 KB	~52 KB total

<http://blog.arungupta.me/monolithic-microservices-refactoring-javaee-applications/>

Design Considerations

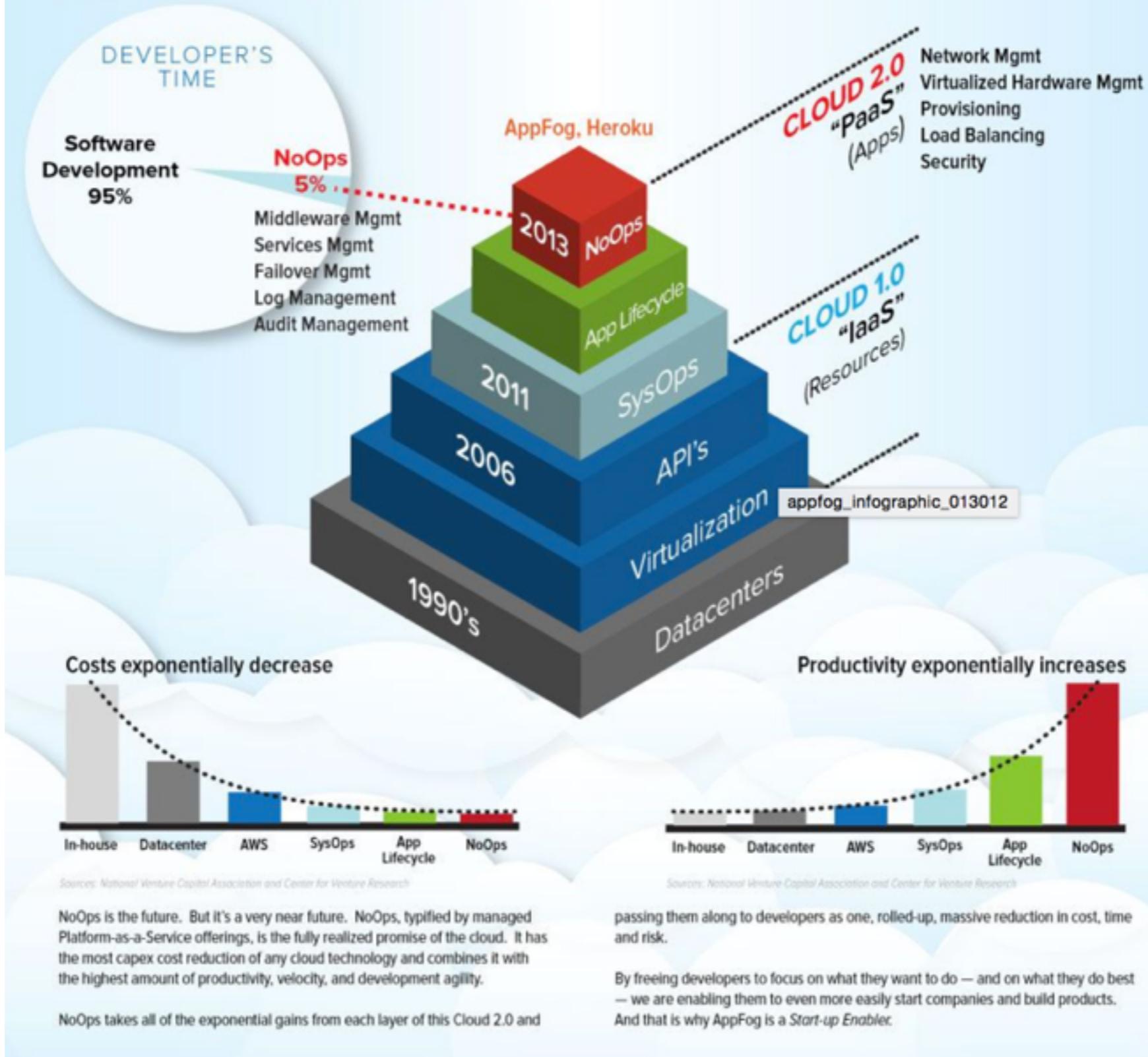
- UI and Full stack
 - Client-side composition (JavaScript?)
 - Server-side HTML generation (JSF?)
 - One service, one UI
- REST Services
- Compensating transactions instead of 2PC

NoOps

- Service replication (k8s, fabric8, etcd, ZK, ...)
- Dependency resolution (Nexus, ...)
- Failover (Circuit Breaker)
- Resiliency (Circuit Breaker)
- Service monitoring, alerts and events (New Relic, Log stash, ...)

2013: A bright NoOps future

So where does this all lead? The end-game is NoOps. Where building and running an app is purely a developer process — and where developers are not having to spend time doing Ops work.



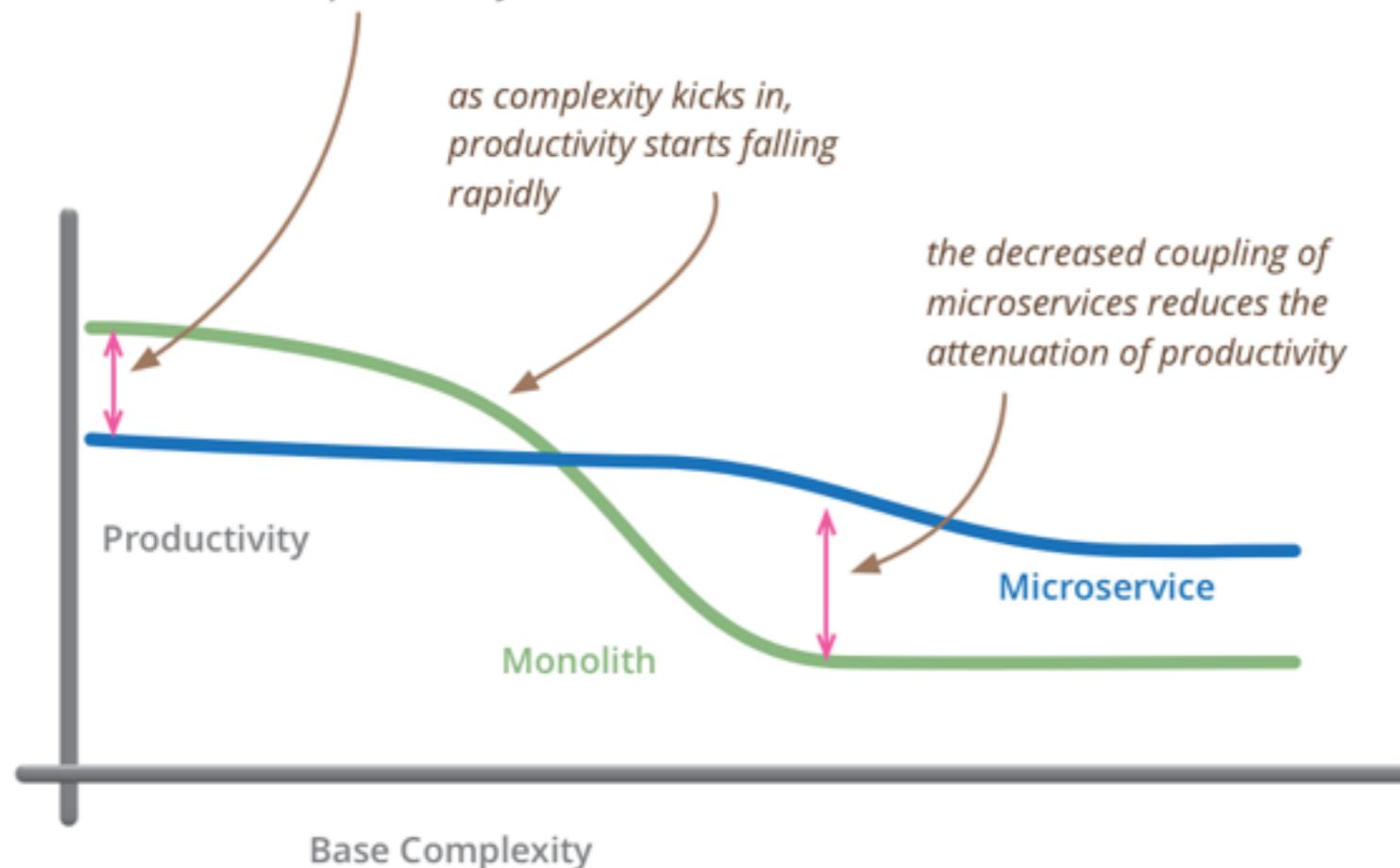
<https://gigaom.com/2012/01/31/why-2013-is-the-year-of-noops-for-programmers-infographic/>

Drawbacks of microservices

- Additional complexity of distributed systems
- Significant operational complexity, need high-level of automation
- Rollout plan to coordinate deployments
- Slower ROI, to begin with

Microservice Premium

for less-complex systems, the extra baggage required to manage microservices reduces productivity



but remember the skill of the team will outweigh any monolith/microservice choice

“don’t even consider microservices unless you have a system that’s too complex to manage as a monolith”

<http://martinfowler.com/bliki/MicroservicePremium.html>

Containers

- Faster deployments
- Isolation
- Portability - “it works on my machine”
- Snapshotting
- Security sandbox
- Limit resource usage
- Simplified dependency
- Sharing

Docker: Pros and Cons

- PROS
 - Extreme application portability
 - Very easy to create and work with derivative
 - Fast boot on containers
- CONS
 - Host-centric solution, not aware of anything
 - No higher-level provisioning
 - No usage tracking/reporting

Kubernetes



- Open source orchestration system for Docker containers
- Provide declarative primitives for the “desired state”
 - Self-healing
 - Auto-restarting
 - Schedule across hosts
 - Replicating

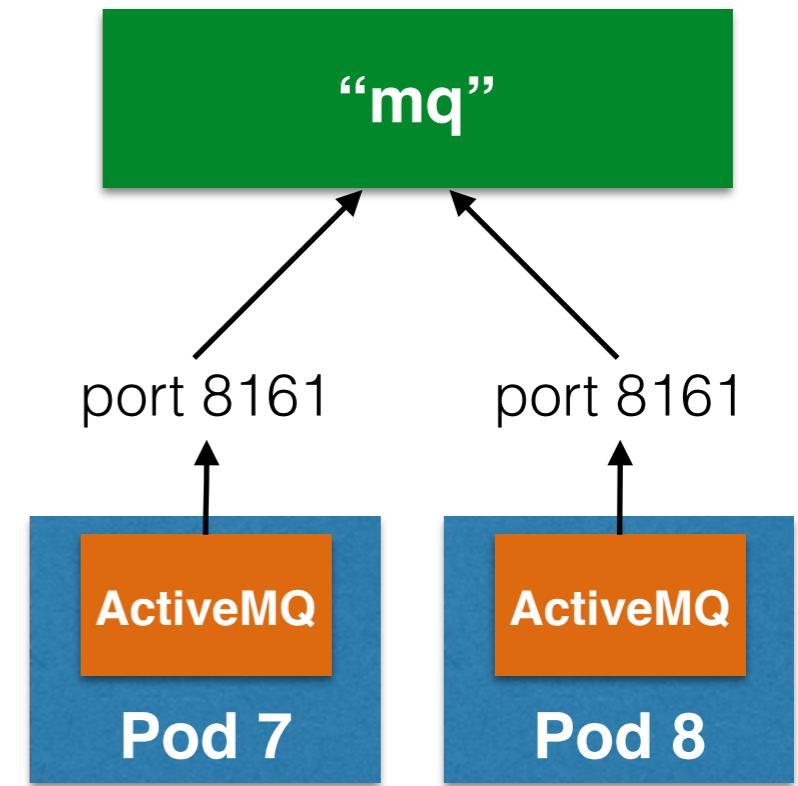
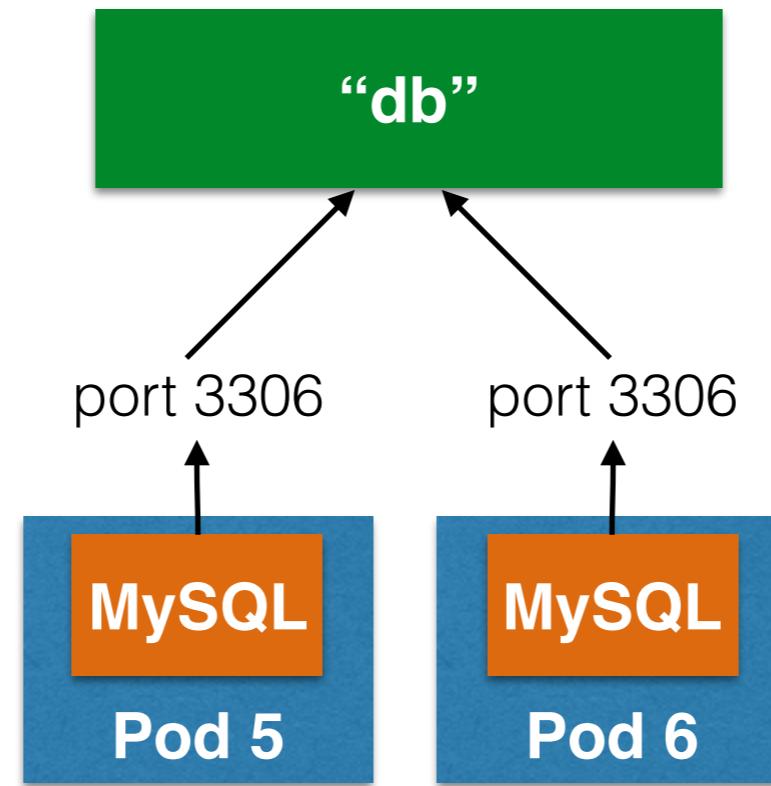
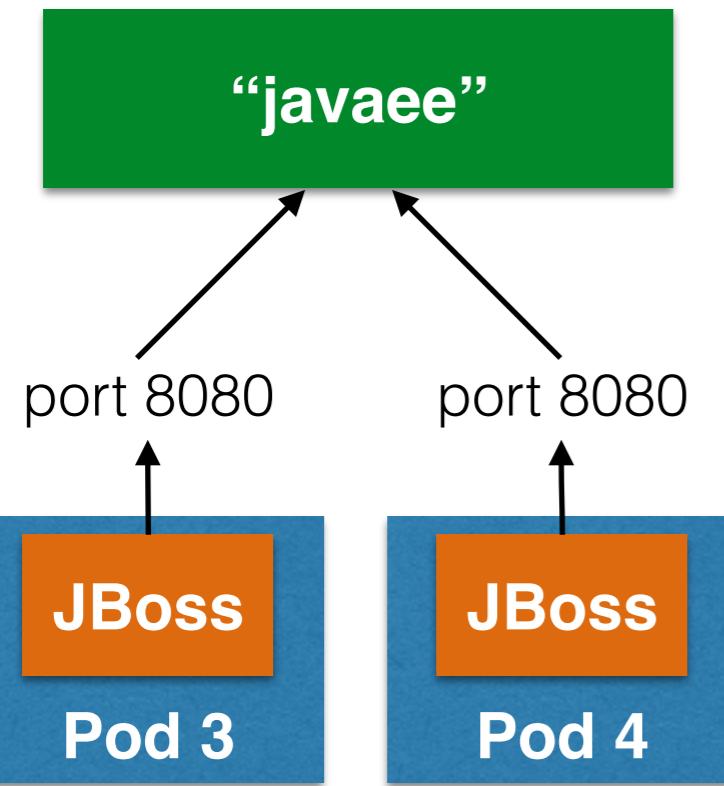
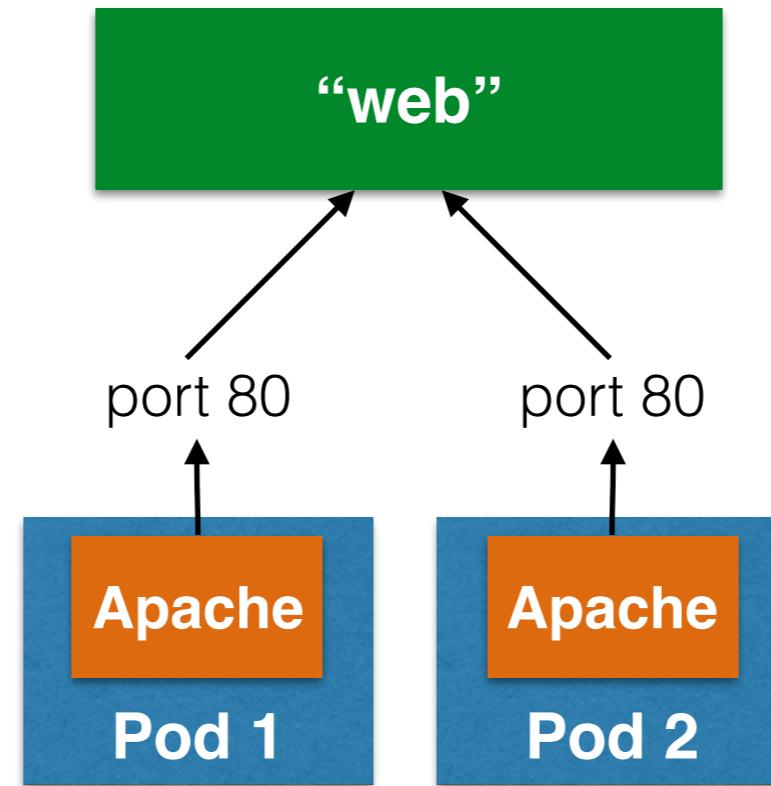
Kubernetes: Pros and Cons

- PROS
 - Manage related Docker containers as a unit
 - Container communication across hosts
 - Availability and scalability through automated deployment and monitoring of pods and their replicas, across hosts

Kubernetes: Pros and Cons

- CONS
 - Lifecycle of applications - build, deploy, manage, promote
 - Port existing source code to run in Kubernetes
 - DevOps: Dev -> Test -> Production
 - No multi-tenancy
 - On-premise (available on GCE)
 - Assumes inter-pod networking as part of infrastructure
 - Requires explicit load balancer

OpenShift
Application





Applications

xPaaS
VERT.X

RED HAT® JBOSS®
MIDDLEWARE
node.js™

PaaS



Containers & Orchestration



Container Host



IaaS



jboss.org



VERT.X



FEEDHENRY™

Infinispan



Qhawtio

