

How to deal with too much success



Today's subject

- About Spil Games
- How success became a problem
- How to deal with scalability
 - Throughput scalability
 - Scalability of data
 - Scalability of organization

About me



- Enrique Paz Perez
- Classic-metal-loving, motorcycle-riding, Erlang enthusiast



spilgames

**About Spil
Games**

Stand-alone games



Single Player: Uphill Rush 4



Mobile: Shopaholic

Social Games



Multiplayer: Galaxy Life



Real-time: 7-Sum-Up

AGAME.COM

Your zone to play free online games

JOIN

OR

LOG IN

I'm looking for...

Search

HOME

BEST GAMES

SOCIAL GAMES

RACING

ACTION

SPORTS

SKILL

ADVENTURE

MULTIPLAYER

MORE →

PLAY NOW!

Also available on the
App Store

CHECK THESE OUT!

ADVERTISEMENT



1 2 3

PLAY IT!

Disney/Pixar's Brave

Grab your bow and arrow and join this fiery heroine for some magical adventures in the Scottish highlands!

**NEW: Play Anywhere**

Now Agame.com's on your cell!

NEW GAMES

**Vintage Car Parking****Rbots****World Basketball Champi...****Goalkeeper Premier****Cookieland****Offroad Rage****Death Racers 2****ivivva Style Quiz****Blackwood & Bell Myster...**[See All New Games](#)

★ MOST PLAYED

- ★ Galaxy Life
- ★ Snail Bob
- ★ Family Barn
- ★ Offroad Rage
- ★ Sara's Cooking Class:...
- ★ Lose the Heat 3: High...
- ★ Uphill Rush 4
- ★ Slotomania
- ★ Uphill Rush 2
- ★ Death Racers 2

[More](#) →

★ BEST RATED

PLAY SOCIAL GAMES

MORE SOCIAL GAMES →

Family Barn

Goodgame Empire

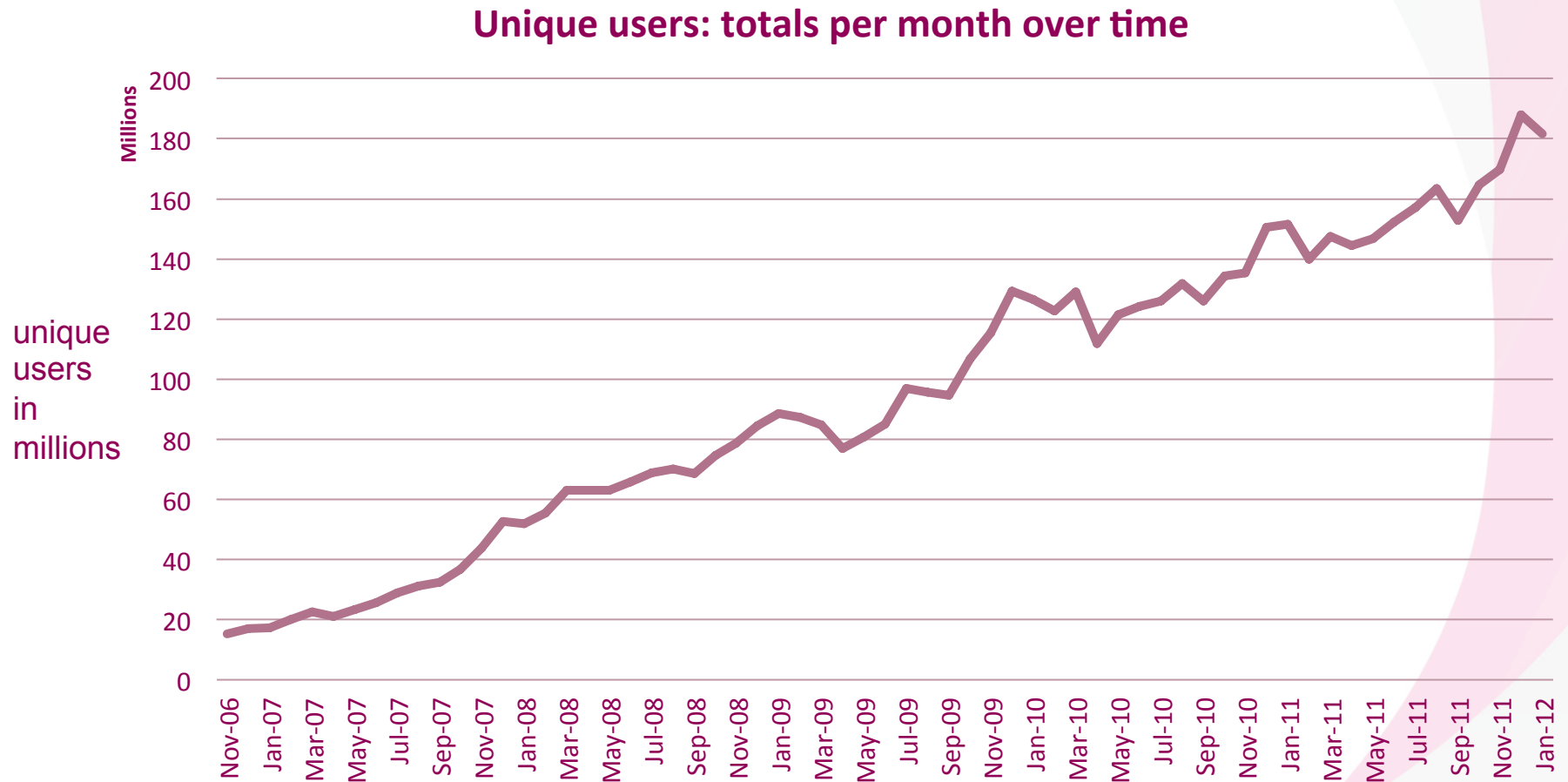
Slotomania

Galaxy Life

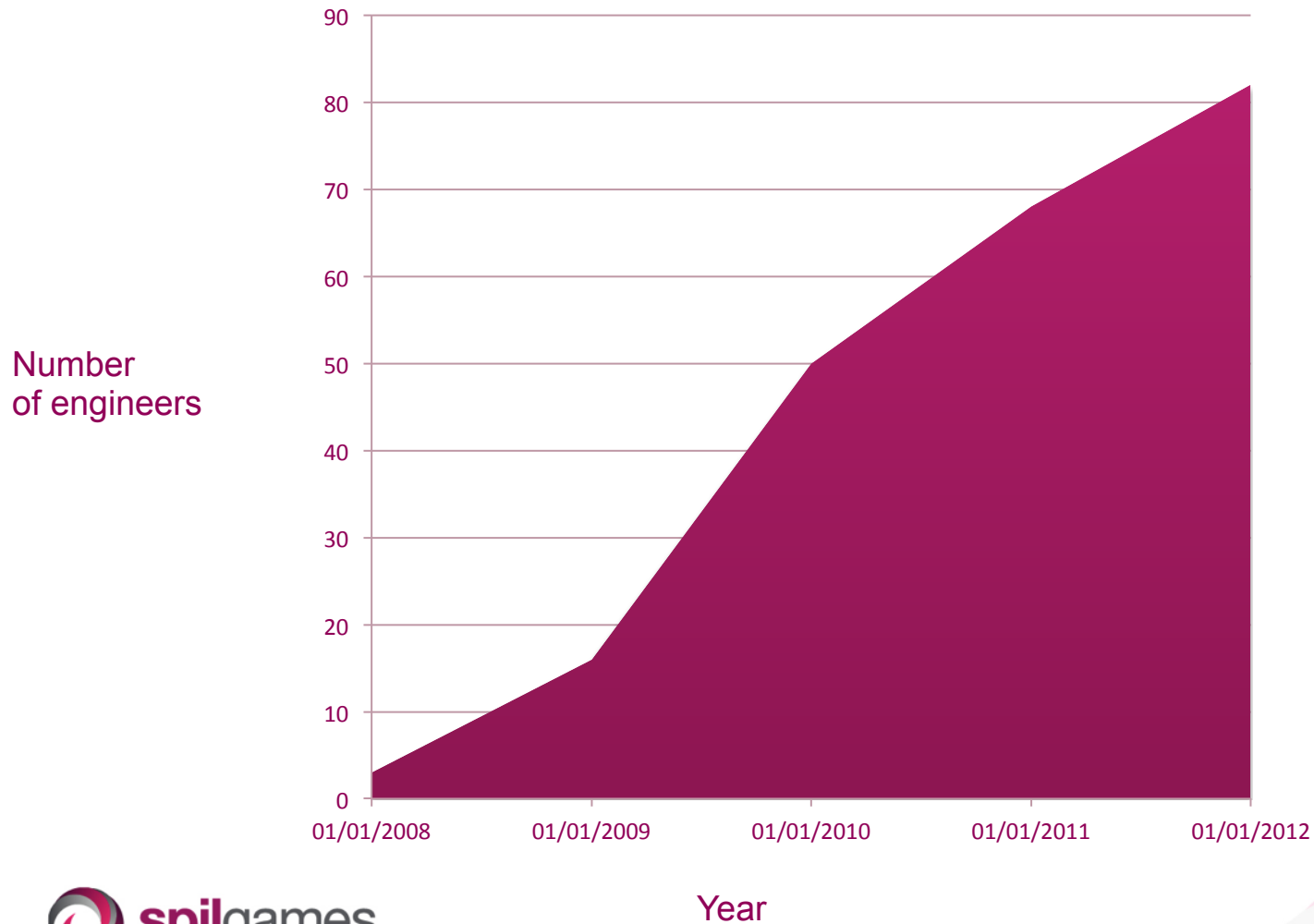
About Spil Games

- Social casual gaming platform
- Serving data to 190+ countries world-wide
- 180+ million unique users per month!
- Multiple platforms: desktop, mobile, iOS
- 300+ employees
- Offices in The Netherlands & China
- Revenue: advertising & EUM

Traffic growth over 1600%



Tech department growth over 2000%!





**Did we
have
control?**

Core technologies

NGINX™

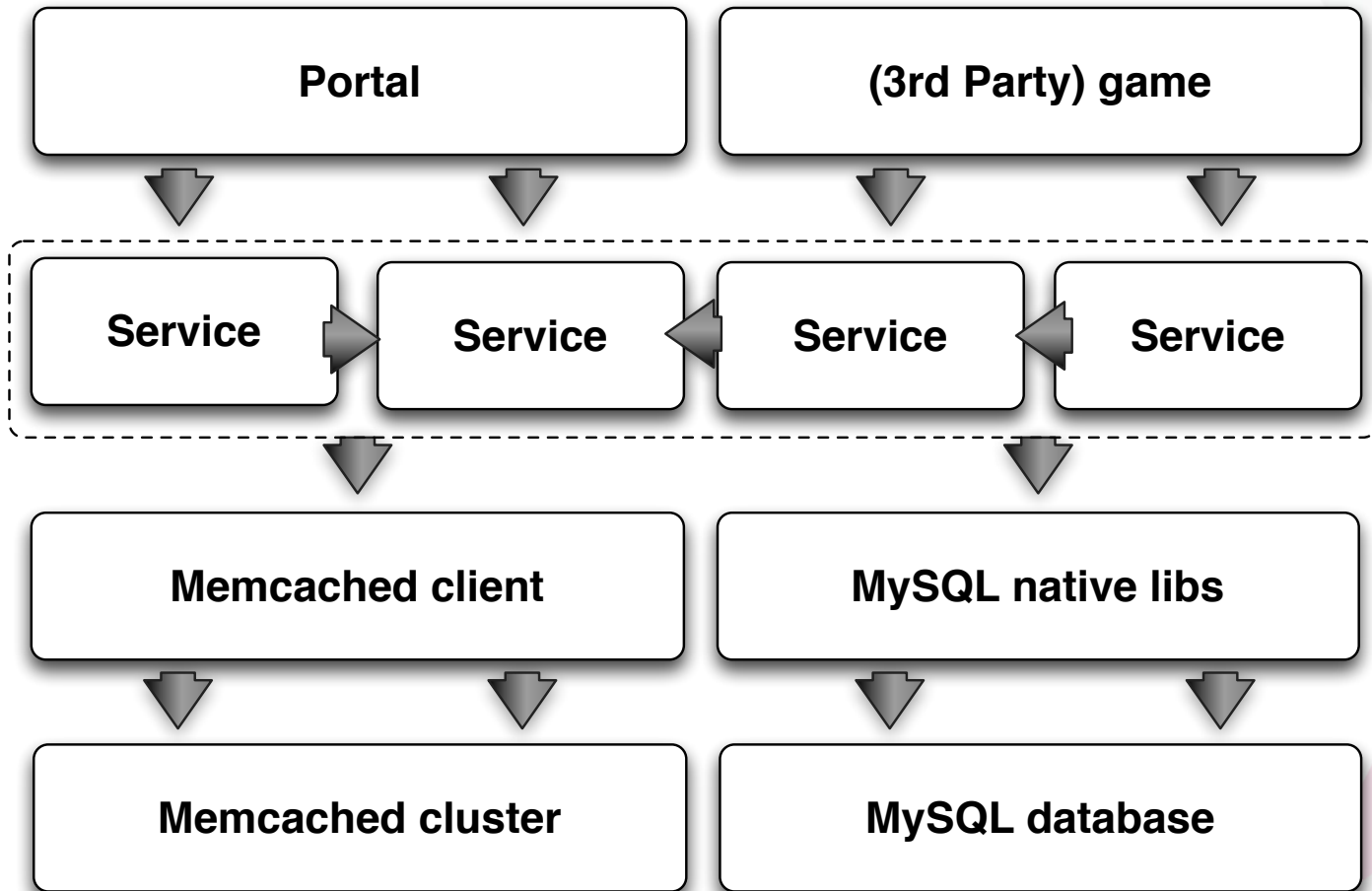
 python

MySQL® 

 php

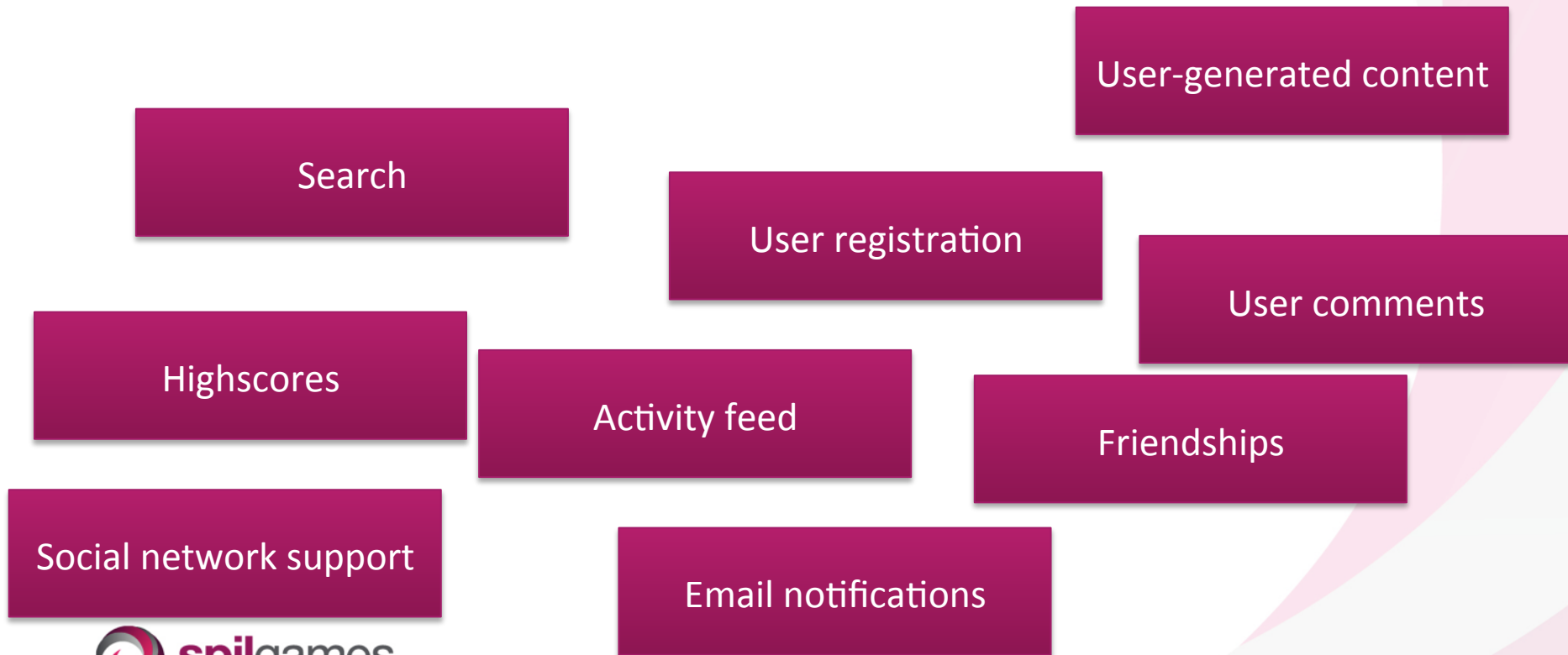
 m

Spil, old architecture



The road to success

- From a single-player static site (2006) ...
- ... to a multiplayer social gaming platform (2011)



Luxury problems!

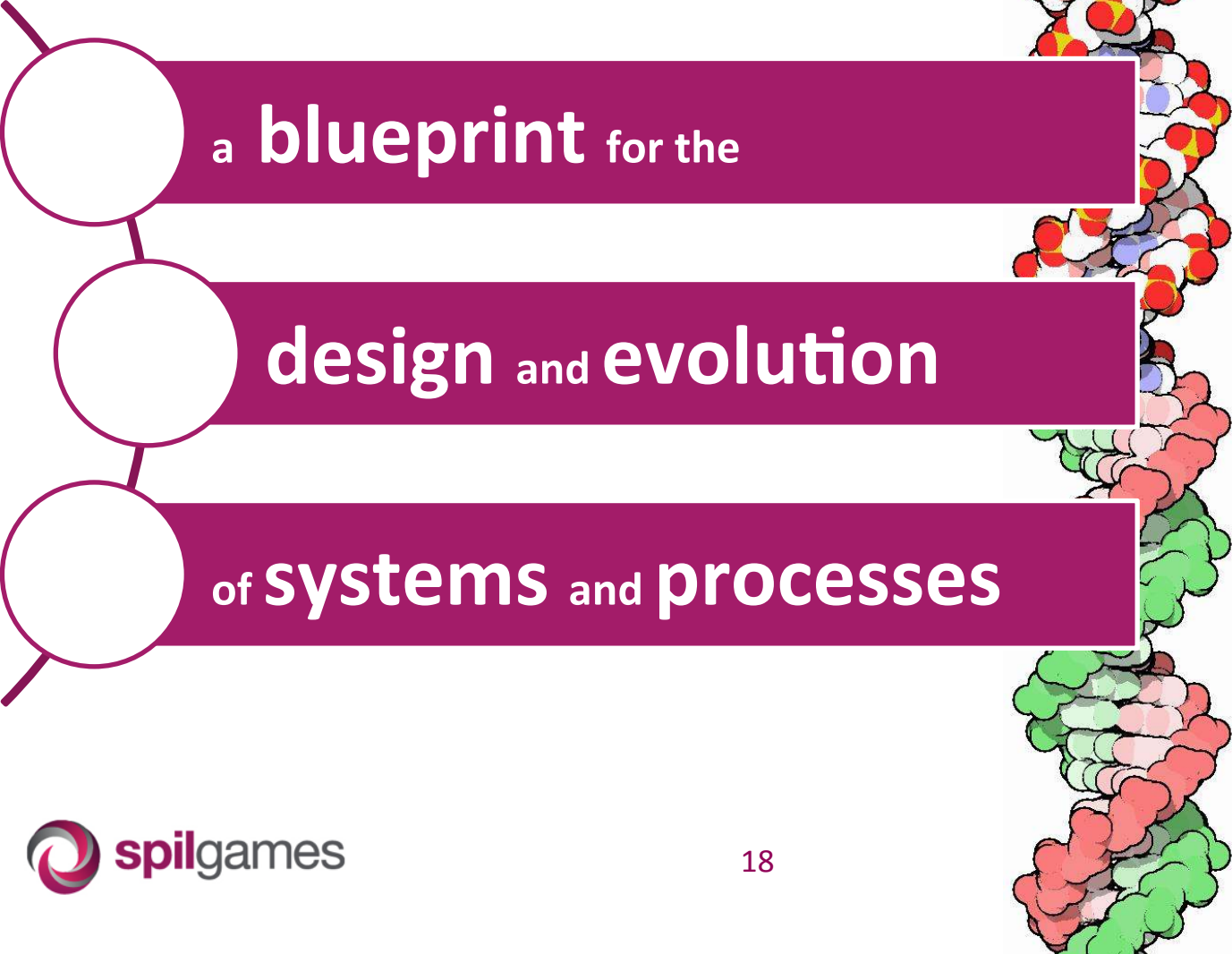
- Putting more people on a project makes it go slower (Brooks)
- No possibilities for sharding (native libs)
- Scaling by adding servers is not linear
- No clear architectural vision:
 - Too many horizontal dependencies
 - Lack of maintainability
 - Independent teams developing the same features
 - Very hard to predict effects of changes





A new approach

Architecture is like DNA...



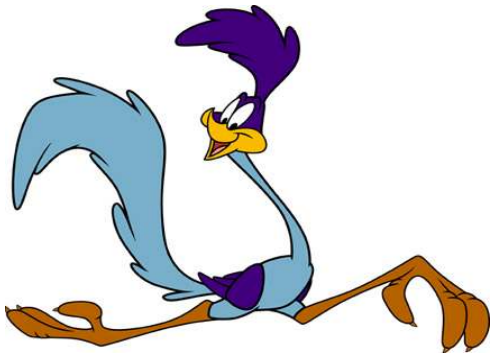
a **blueprint** for the

design and **evolution**

of **systems** and **processes**

New architecture expectations

- Scale the throughput of the traffic
- Scale the data storage using sharding
- Grow the company and people in a sustainable way



New architecture, new technology

- Like Apache, but better!
- Not Only MySQL but MySQL
- Fault-tolerant distributed applications

NGINX™



Why Erlang?

- Functional language
- High availability: designed for telecom solutions
- Excels at concurrency, distribution, fault tolerance
- Do more with less!
- Other companies using Erlang:

ERICSSON 

 WhatsApp

 **klarna**

 **DEMONWARE**

 **wooga**

amazon.com


basho

facebook

 **spilgames**

Presentation layer



Client-side API



Server API



Application Model



Storage platform



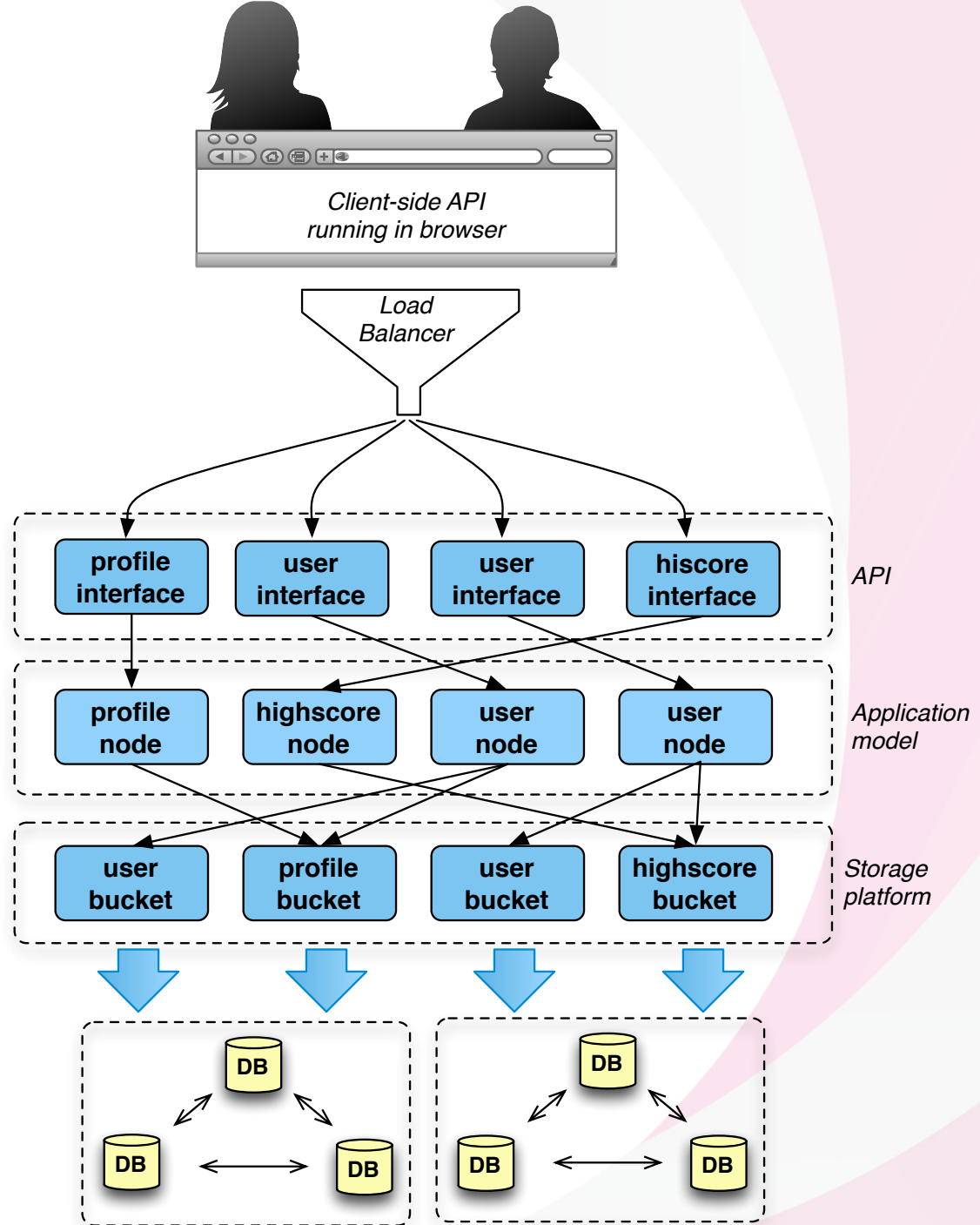
Physical storage

Throughput Scalability

Throughput Scalability

- Scale horizontally in every layer
- More efficient
- Inter-layer communication is redundant & fault-tolerant
- Avoiding overhead
- Concurrency & isolation by default

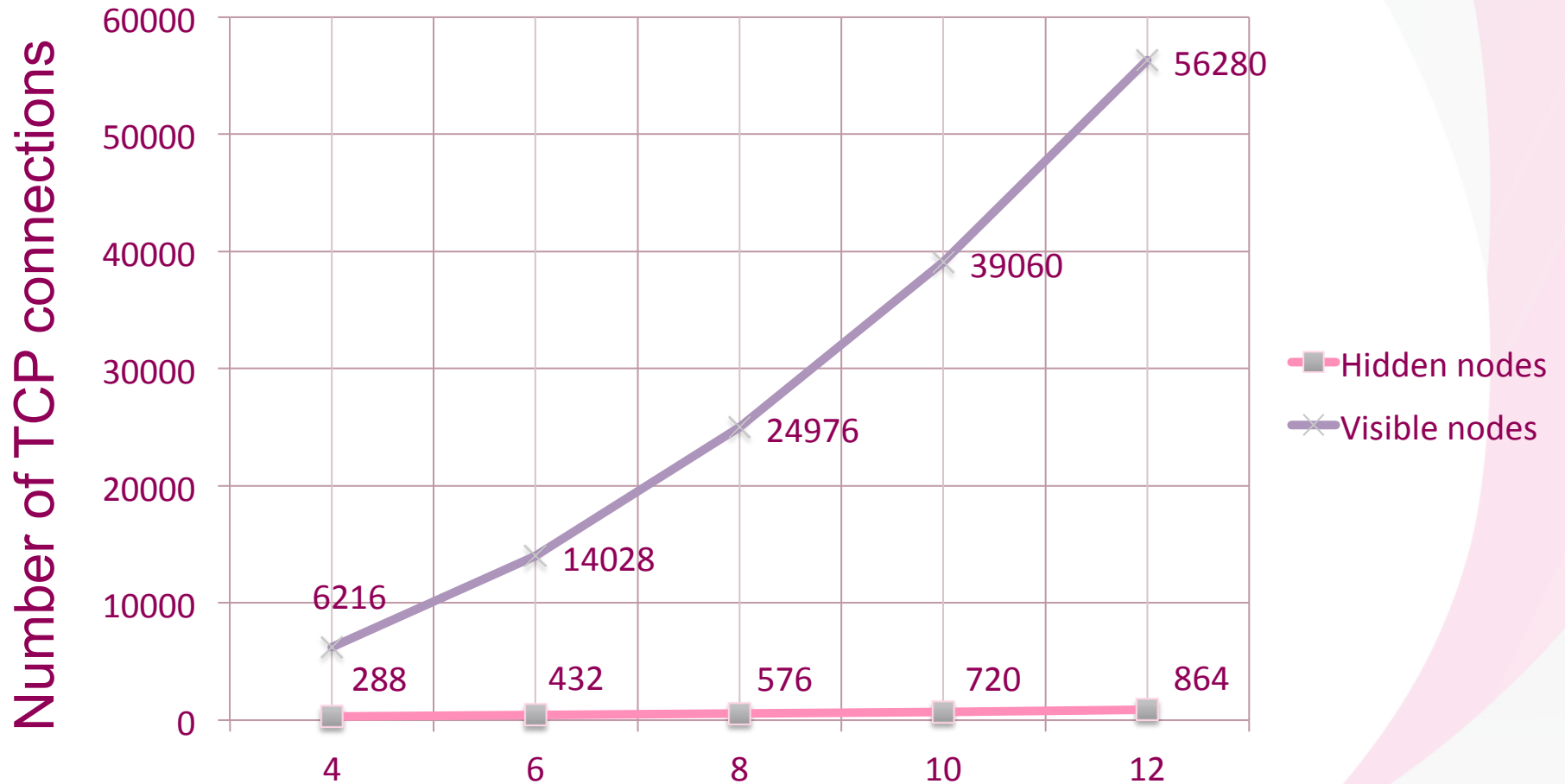
- Well-defined interface
- Scalable layers
- Efficient communication



Keeping the Erlang cluster connected

- Scale individual subsystems
- Vertical network connections
- Connecting all nodes is a bad idea:
 - Not needed
 - Number of open connections
- Solution: use hidden nodes!
 - Each node specifies the nodes it needs
 - How does it scale?

Hidden nodes vs. visible nodes

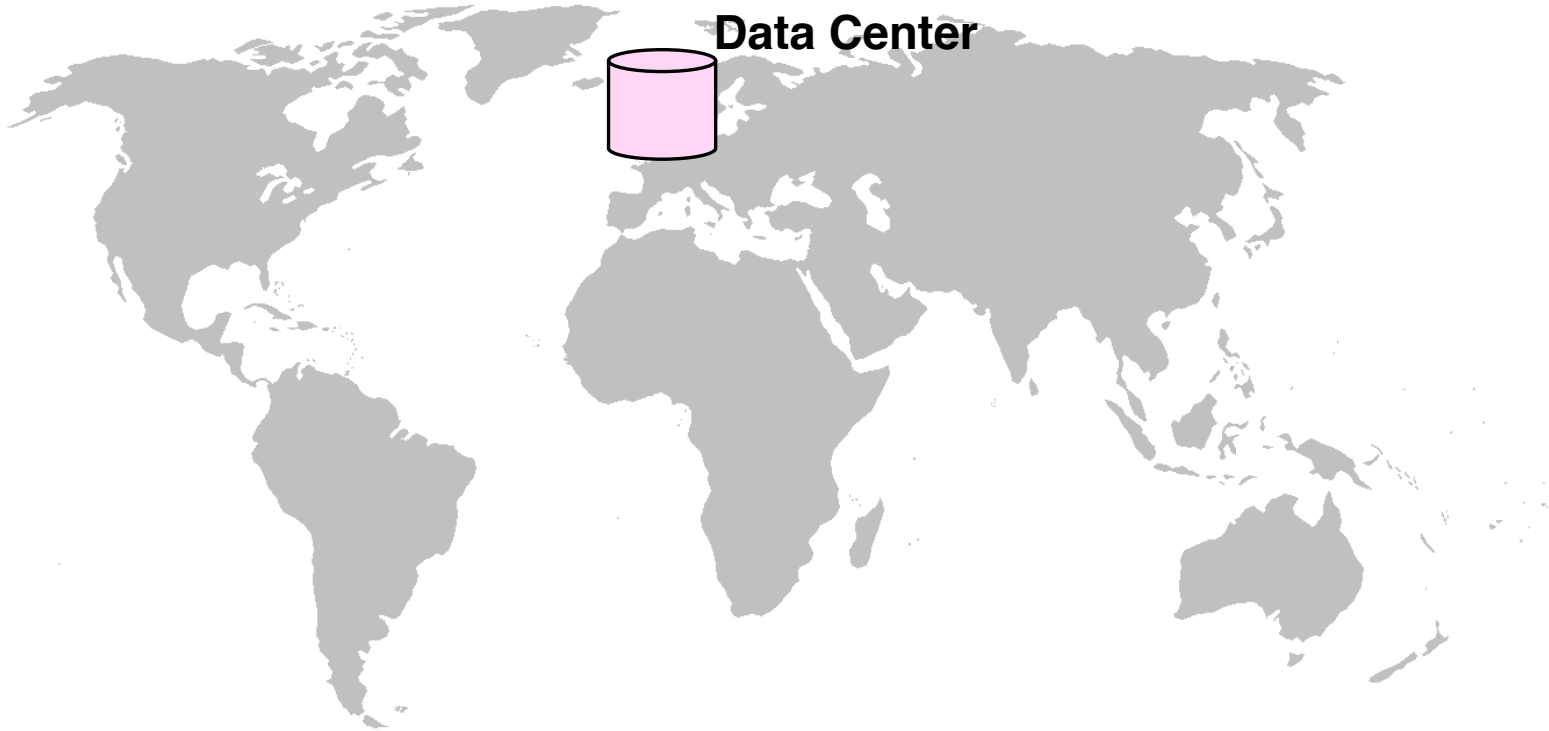


Number of servers (~30 nodes / server)

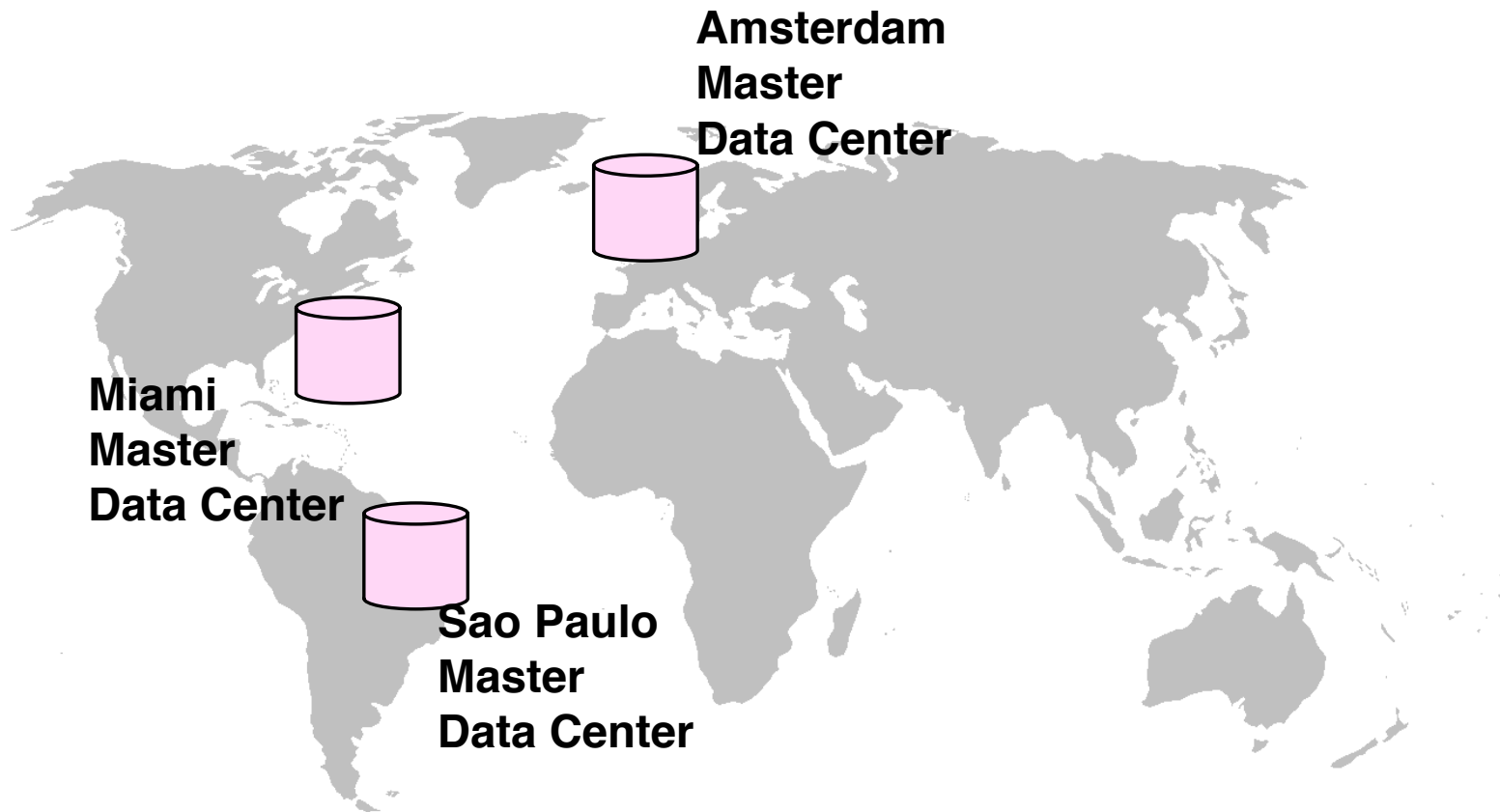
Data Scalability

From one data center...

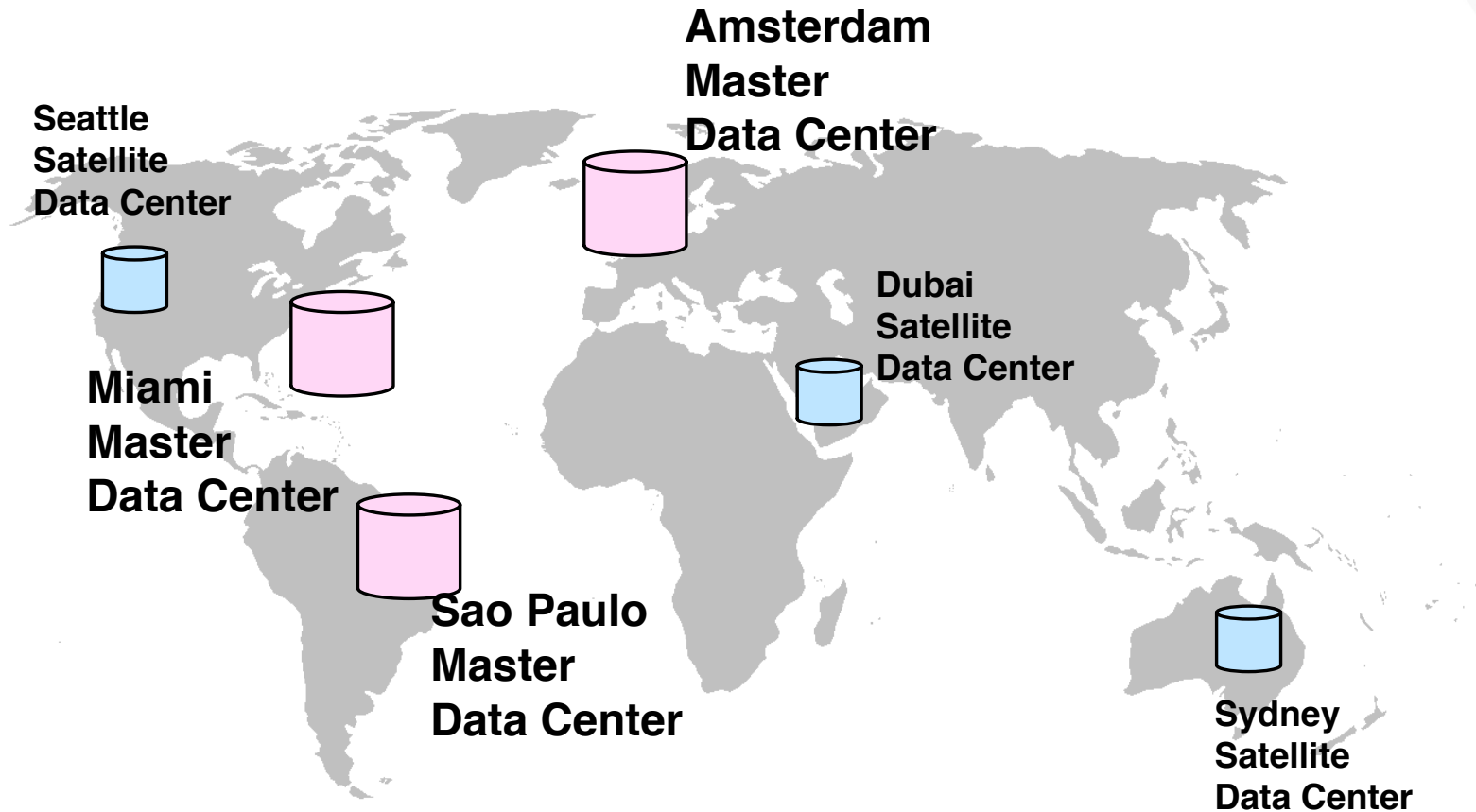
**Amsterdam
Master
Data Center**



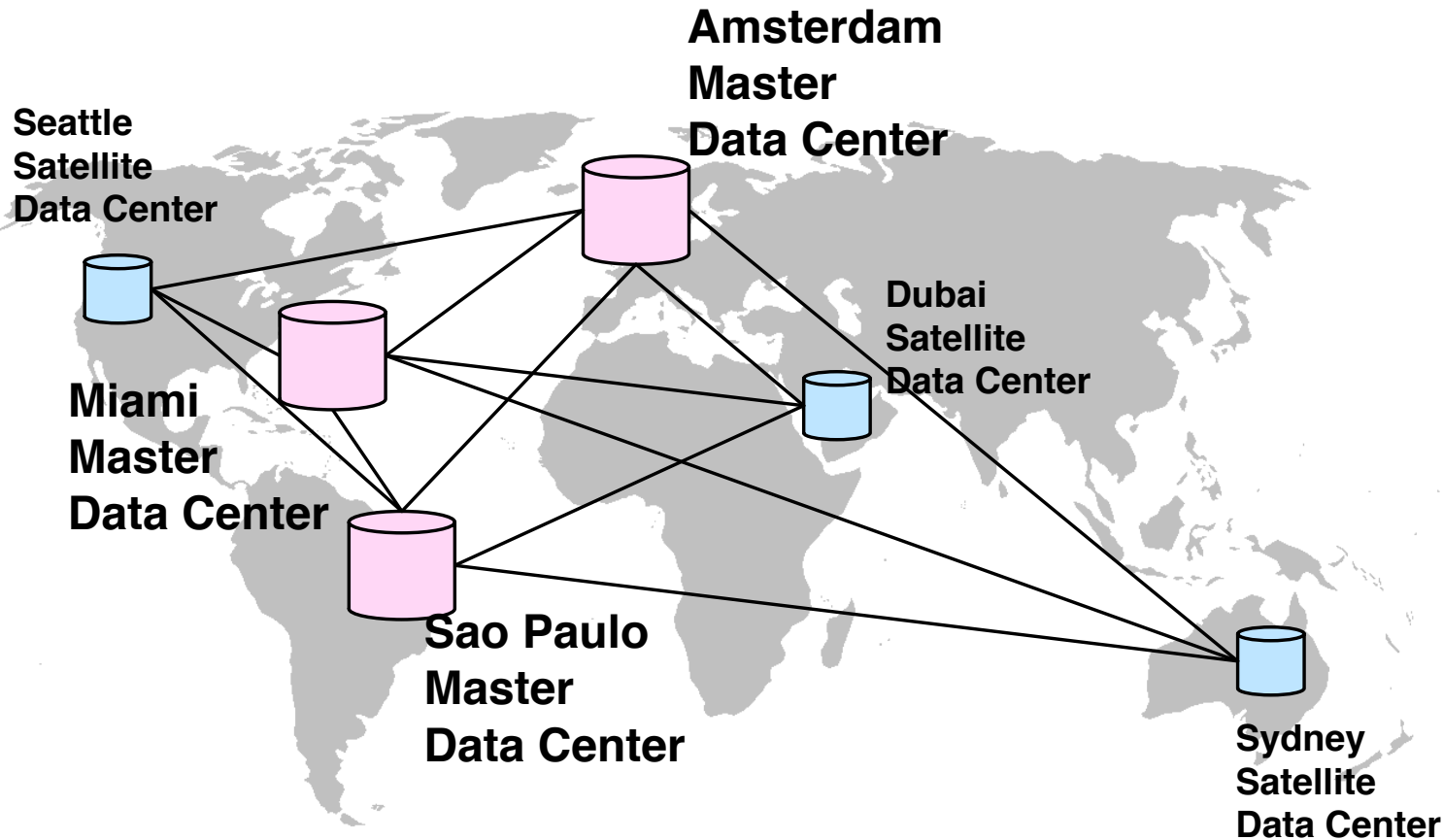
... to many more.



Adding *memory-only* data centers...

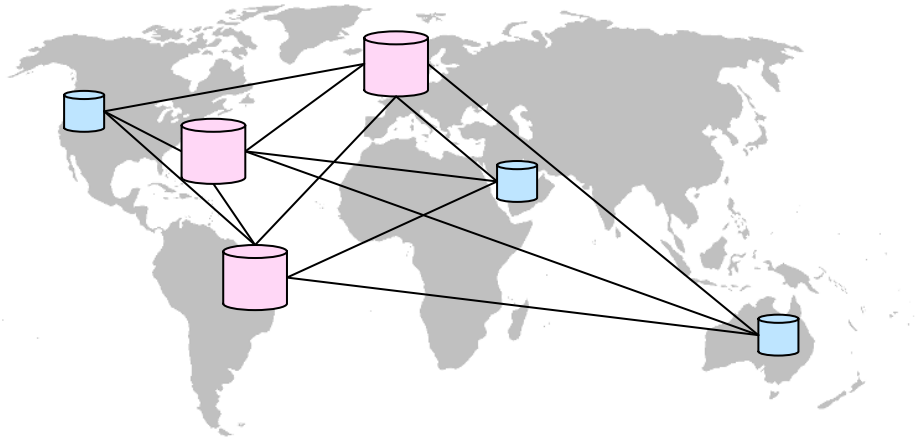


... which are all interconnected.



Data Scalability

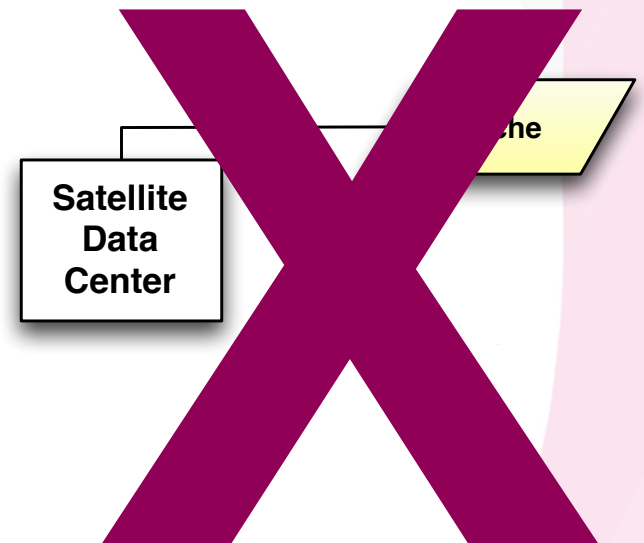
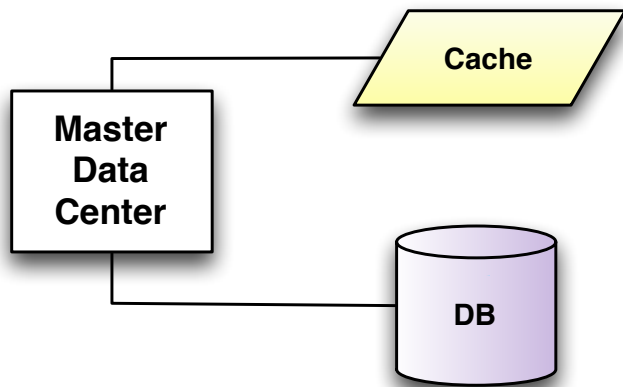
- Local sharding of data is user-based
- Data migrates automatically
- Data migrates on-demand
- Geographically distributed data centers



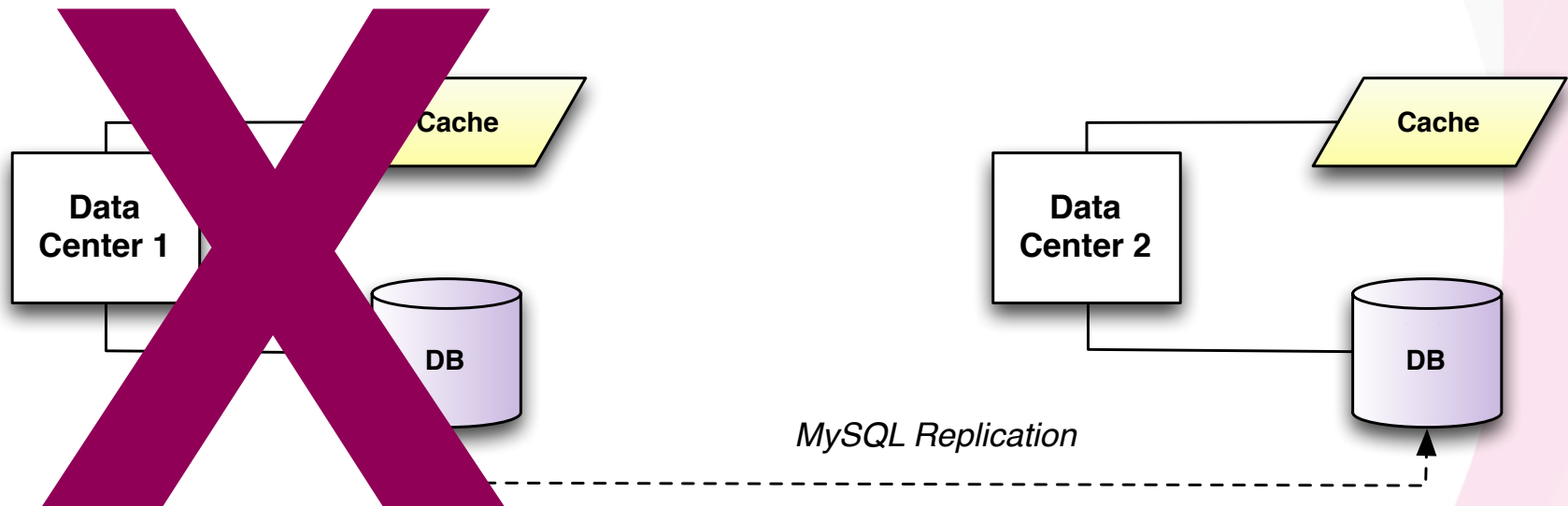
Properties of an operation

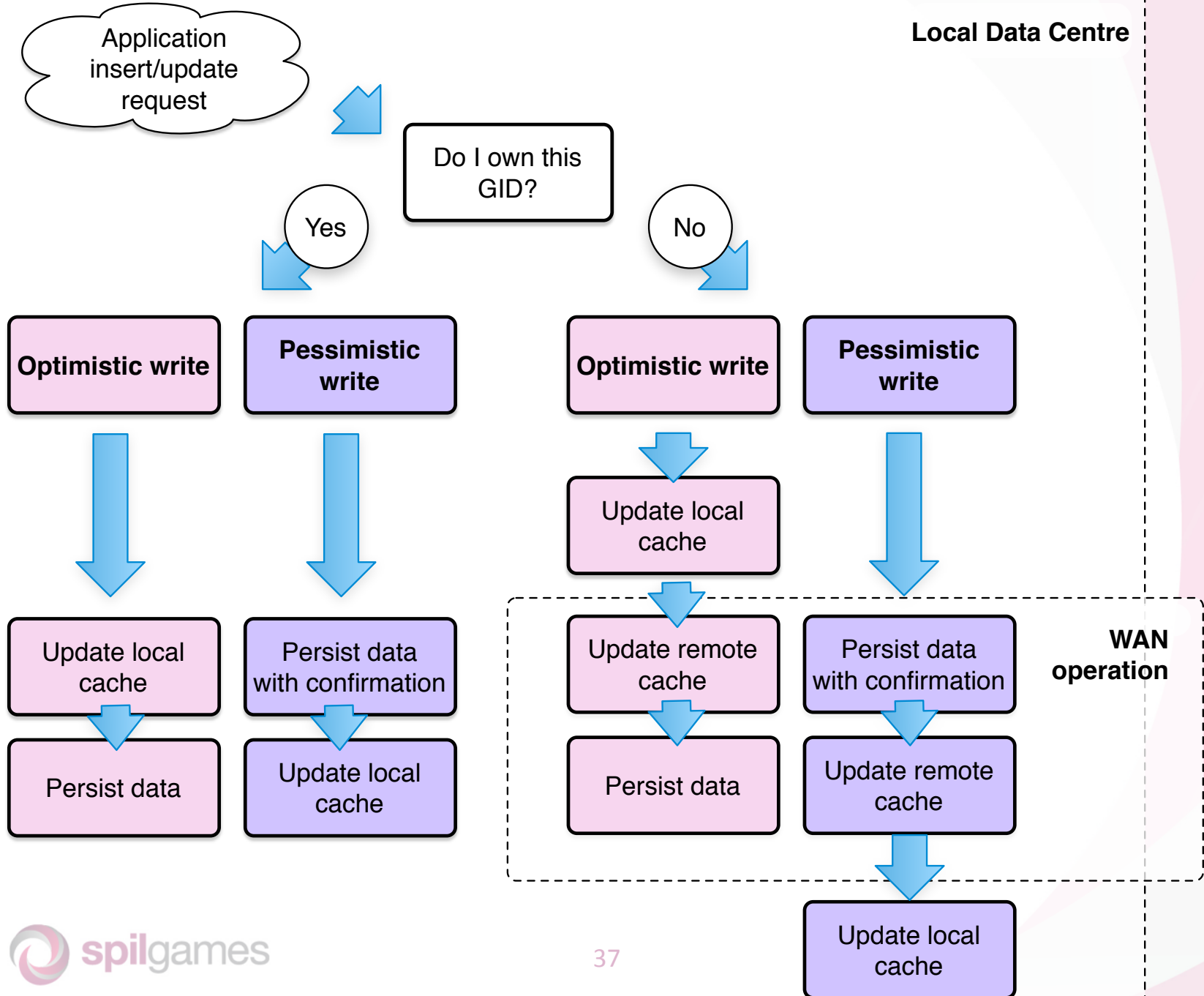
- Global identifiers: GIDs
- Every Bucket/GID operation is atomic
- Two types of storage:
 - Cache
 - Persisted
- Two types of operations:
 - Optimistic
 - Pessimistic

Disposable Data Centers



Dealing with catastrophies



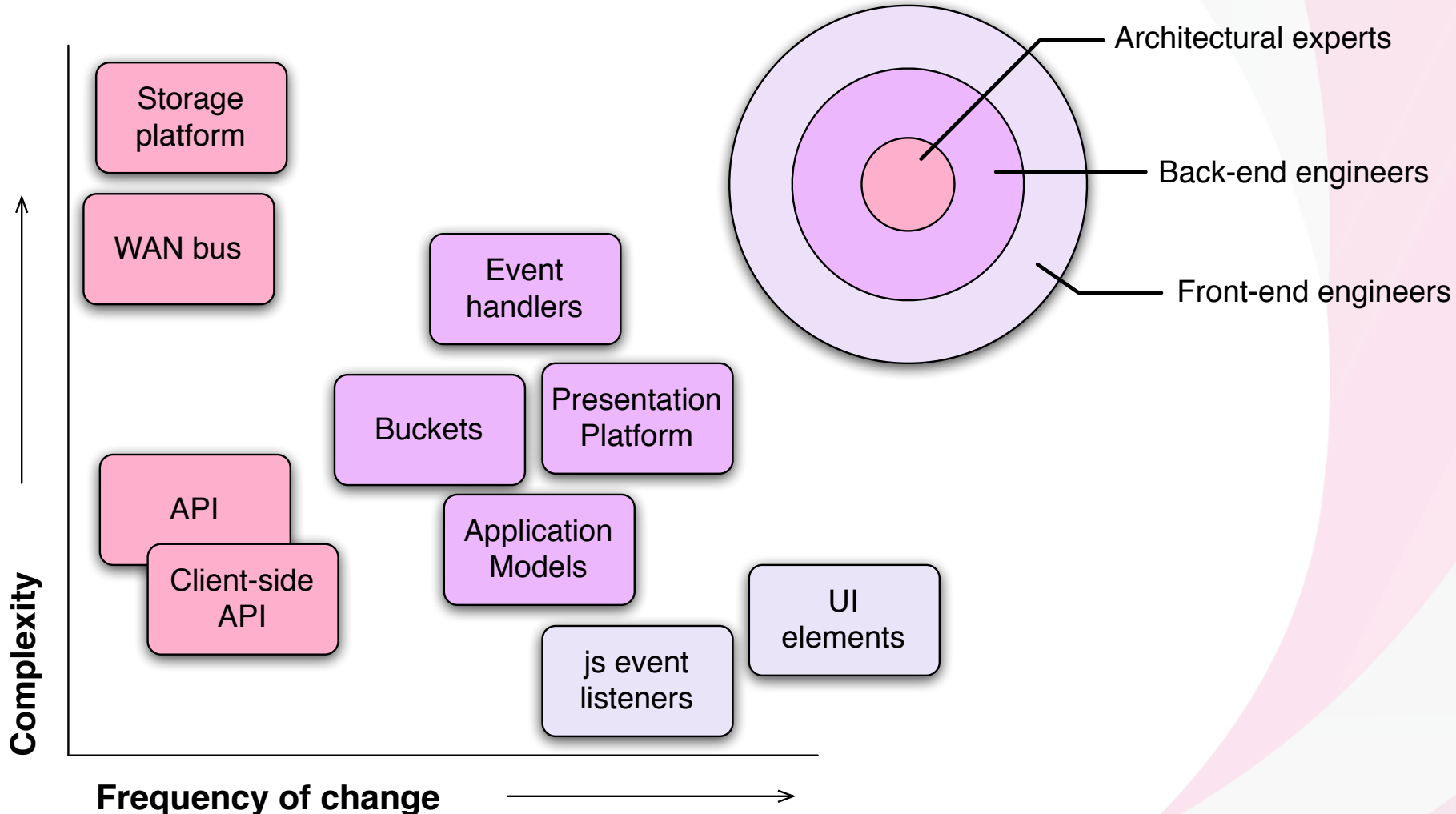


Organizational Scalability

What we want

- Teams need to *adapt* to a fast-changing market
- Every team needs to be able to *deliver*
- No need to learn *everything*

A flexible development workforce



Presentation layer



Client-side API



Server API



Application Model



Storage platform



Physical storage



Front-end engineers
Javascript/HTML/CSS

75%



**Back-end engineers &
Architectural experts**
Erlang

20%



Database specialists
MySQL

5%

**Organizational
Scalability**

$$c \frac{d\ell}{dt} = \frac{1}{\tau} \frac{1}{P} \frac{dP}{dt}$$

$$D^2 = \frac{1}{P^2} \frac{P_0 - P}{P} \sim \frac{1}{P^2} \quad (1a)$$

$$D^2 = \frac{k_0}{3} \frac{P}{P_0 - P} \sim \frac{1}{P^2} \quad (2a)$$

$$D^2 \sim 10^{-53}$$

$$\sim 10^{-26}$$

$$P \sim 10^8 \text{ g.} \sim$$

$$t \sim 10^{10} (10^{11}) \sim$$

Lessons Learned

What we've done...

- Set up tooling for development
- Moved functions behind one, solid API
- Tested inter-layer communication with live traffic
- Set up development and deployment processes
- Deployed full vertical slices to live

... And what we're doing

- Organizational challenge:
 - Development & hosting landscape is changing
 - Independent teams
 - Training current workforce
- Adding geographical storage capabilities
- Back-end and front-end are being released now
- Next data center in early 2013

Lessons Learned

- An architectural vision is crucial:
 - Provide focus for developers
 - Less risk of chaos
 - Control the information flow
- Use the right tools for the right task:
 - Don't be afraid to switch to a different (proven) technology...
 - ... Also don't be afraid to stick with proven technology
- Hide implementation in a layer ...
- ... expose through a strict API ...
- ... with efficient communication



Questions

Useful Links

PIQI & PIQI-RPC:

<https://github.com/alavrik/piqi>

<https://github.com/alavrik/piqi-rpc>

Erlang productivity by Jan Henry Nystrom

<http://www.slideshare.net/JanHenryNystrom/productivity-gains-in-erlang> by Jan Henry Nystrom

Architectural principles by Randy Shoup:

<http://singztechmusings.wordpress.com/2011/09/05/performance-engineering-slides-on-ebay-architecture-principles-and-high-performance-computing/>

Joe Armstrong replying to Steve Vinoski regarding **Erlang-RPC**:

<http://armstrongonsoftware.blogspot.nl/2008/05/road-we-didnt-go-down.html>