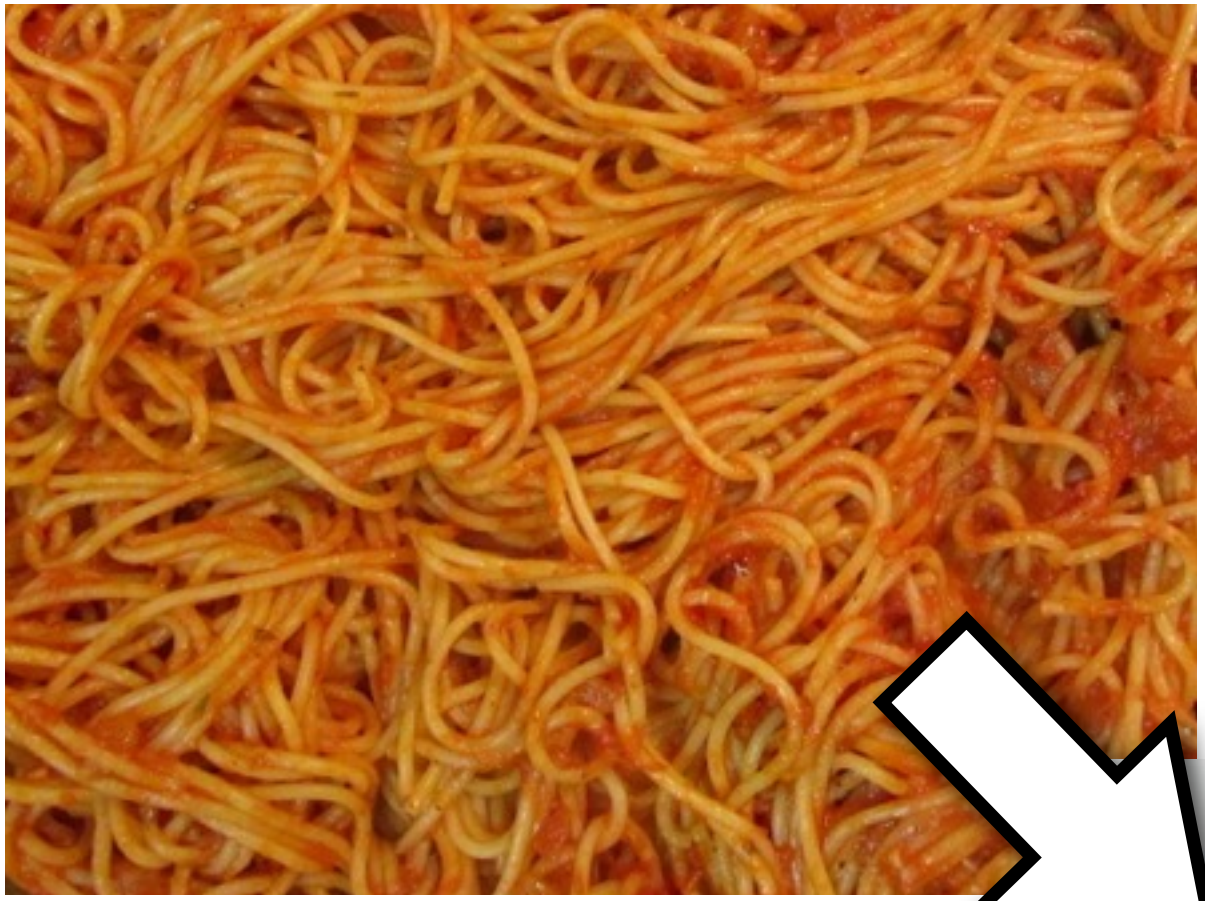
A Layered Architecture
Robustness & Efficiency

dennis.docter@spilgames.com
March 22nd, 2013



spilgames

spilgames

A legacy of growth

Social and casual gaming platform

Casual games

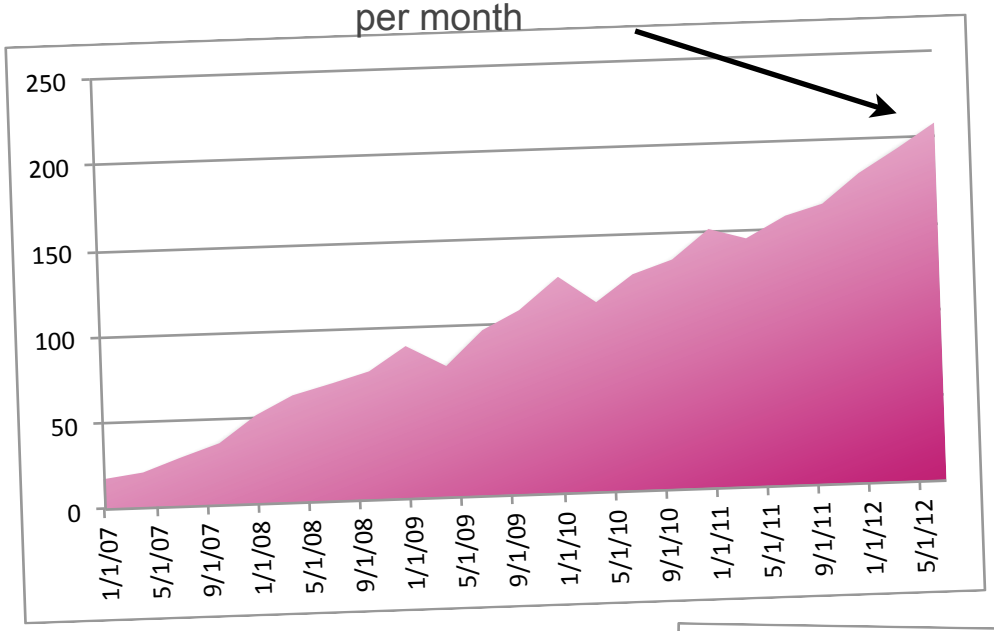
Multi player games

Social games

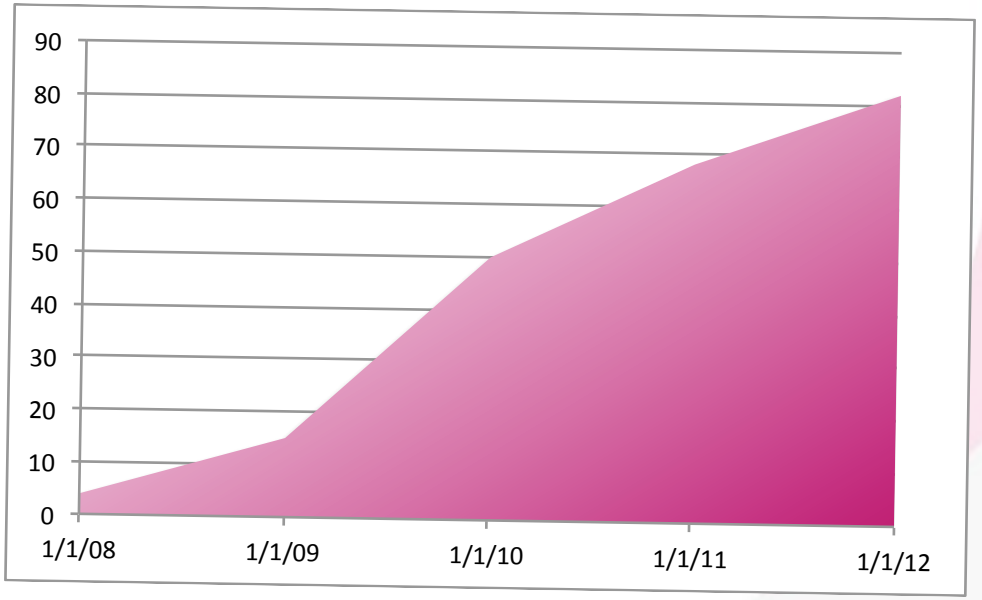
Native games

spilgames

200+ million unique users per month

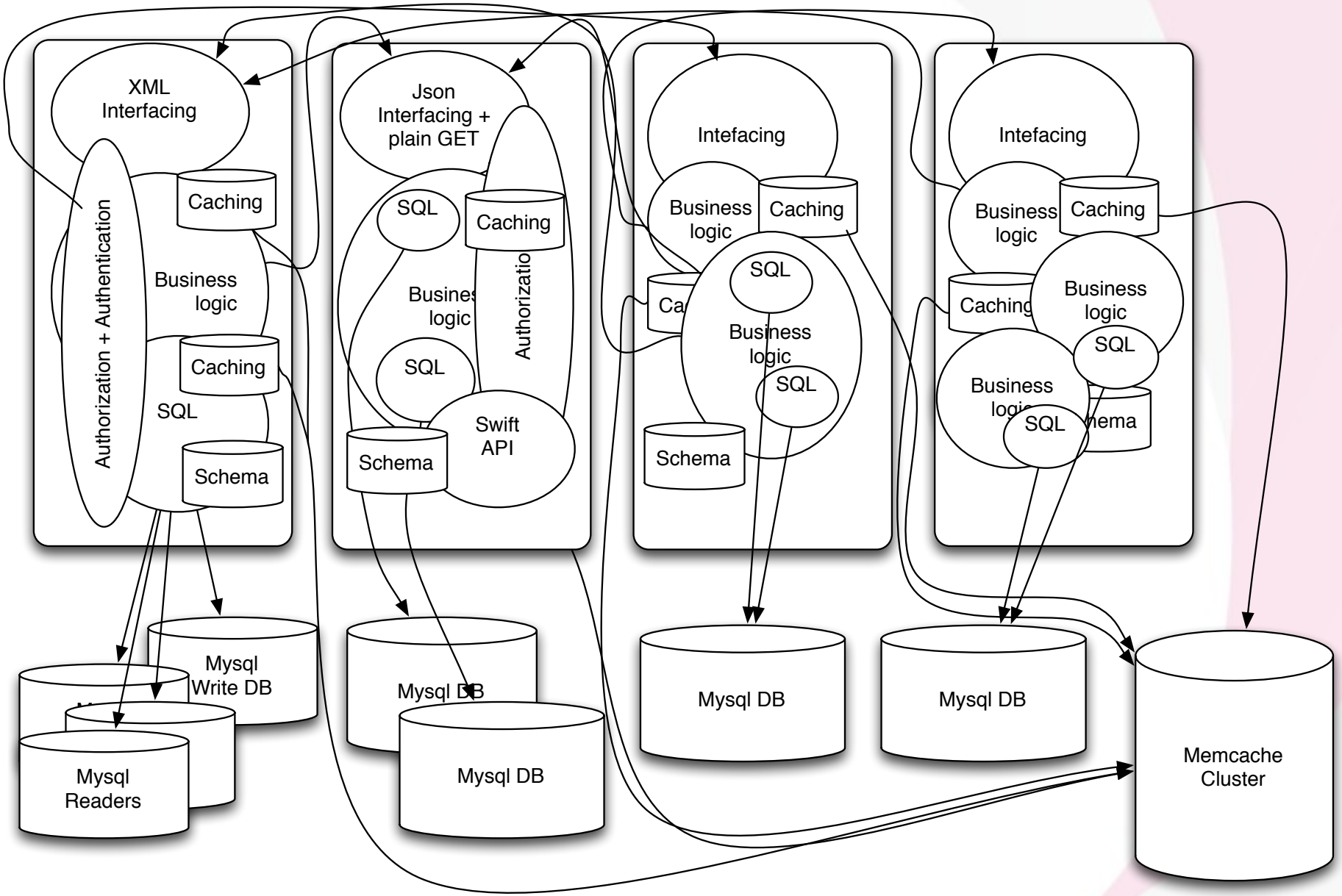


Traffic growth over 1600%



Tech dept. growth over 2000%

spilgames


XML
Interfacing
Authorization + Authentication
Caching
Business
logic
Caching
SQL
Schema
Json
Interfacing +
plain GET
SQL
Caching
Busines
logic
Authorizatio
SQL
Schema
Swift
API
Intefacing
Business
logic
Caching
SQL
Business
logic
SQL
Schema
Intefacing
Business
logic
Caching
Caching
Business
logic
SQL
Business
logic
SQL
Mysql
Write DB
Mysql
Readers
Mysql DB
Mysql DB
Mysql DB
Mysql DB
Memcache
Cluster

Problems

- Inconsistent design & interfaces
- Unclear responsibilities
- Technical debt
- Independent teams developing the same features.
- 7 layers of caching hell
- Scaling

spilgames

Problems

- Inconsistent design & interfaces
- Unclear responsibilities
- Technical debt
- Independent teams developing the same features.
- 7 layers of caching hell
- Scaling

spilgames

A new approach

New architecture expectations

- Flexible
- Predictable
- Scalable

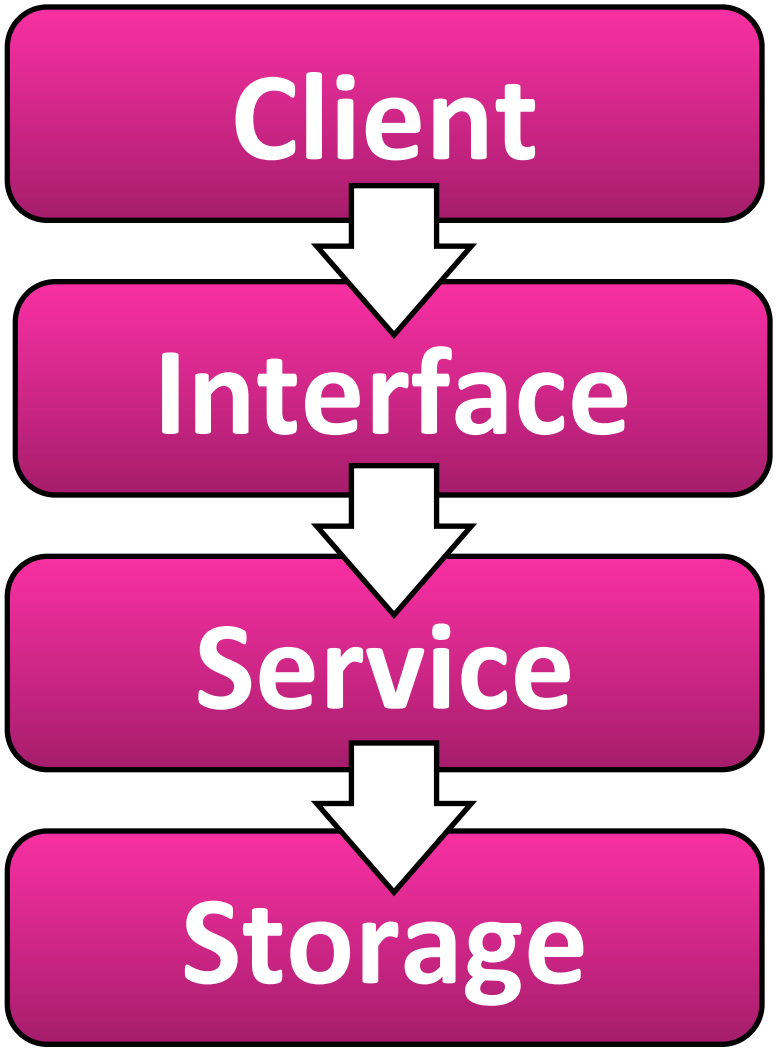
Buzzword™ Compliant

spilgames

Choices made

- Simple
- Clean separation
- No horizontal dependencies
- Fault tolerant
- Distributed

Layers



Responsibilities

- Integration
- State

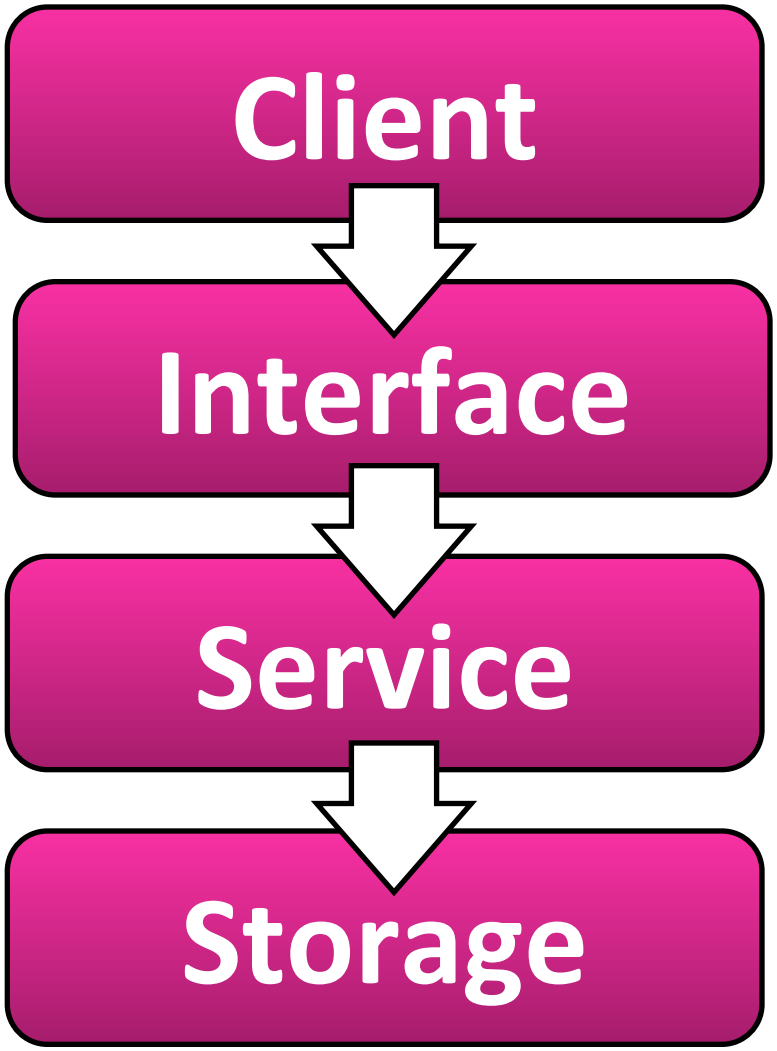
- Authentication
- Aggregation
- Validation

- Authorization
- Composition
- Transformation

- Caching
- Persistence
- Sharding

spilgames

Layers



Responsibilities

- Integration
- State

- Authentication
- Aggregation
- Validation

- Authorization
- Composition
- Transformation

Stateless

- Caching
- Persistence
- Sharding

spilgames

Building blocks

Application

- Small & Simple
- Limited scope
- Quick to deploy

Platform

- Abstracts complexity
- Low change rate
- Long lifetime

spilgames

Walkthrough

Interfaces

https post
/user/get/

```
{“auth”:
   {“token”: “UwAA_AWeff...”}
}
```

3rd Party Backend

User Interface

get([friends])

User Service

Application Interface

Highscore Service

list

get(me)

count

list

Friends Service

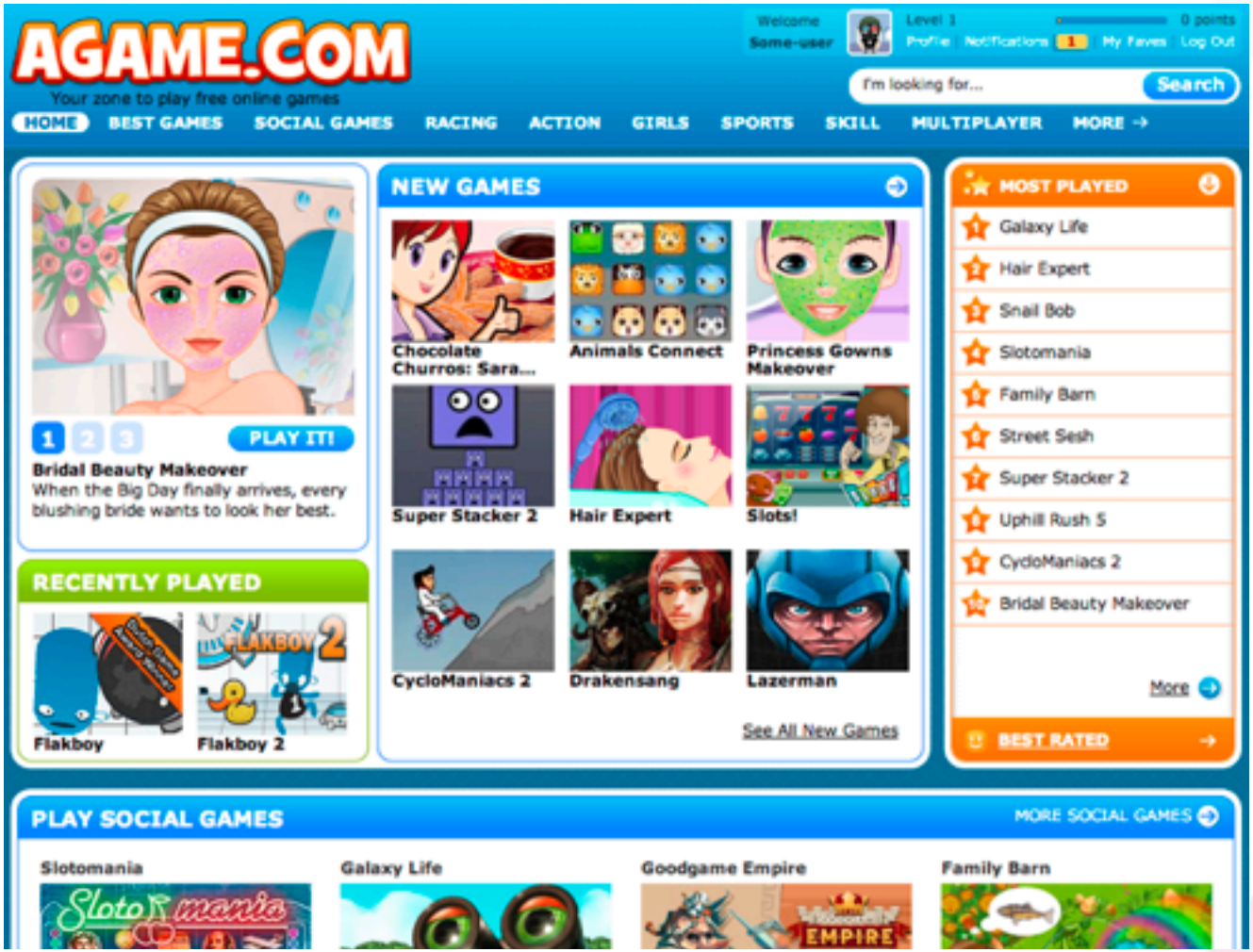
User Service

Awards Service

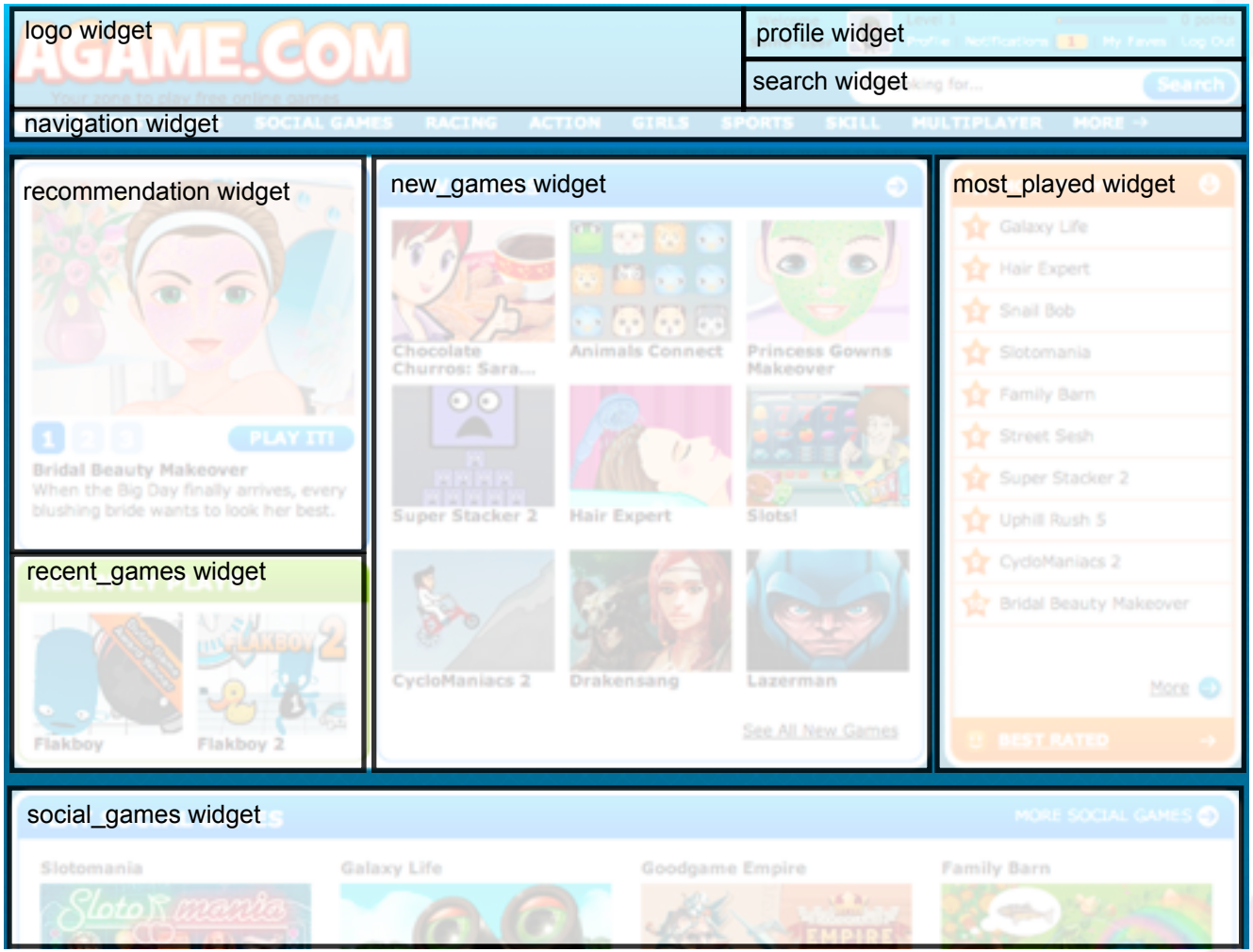
/application/highscores/

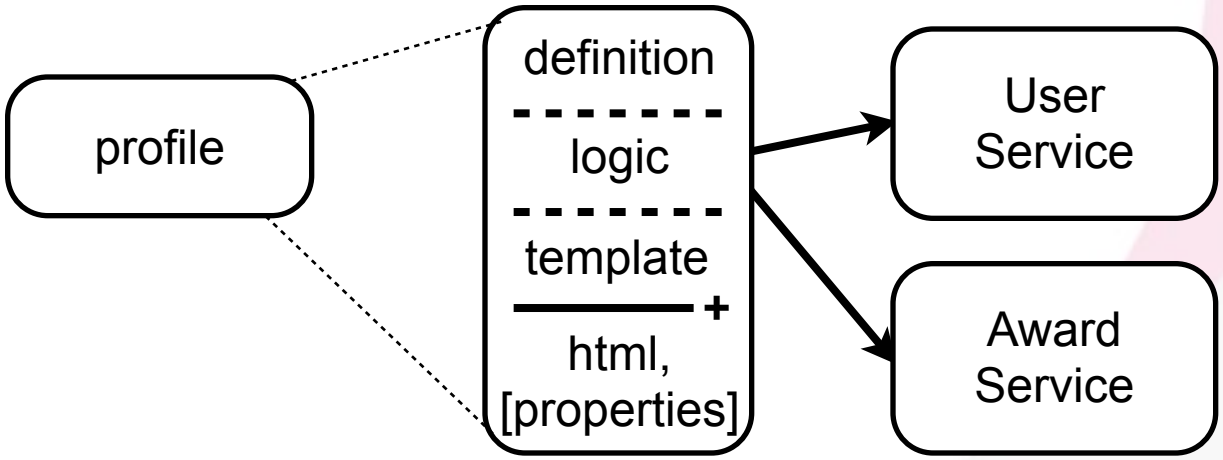
spilgames

Widgets

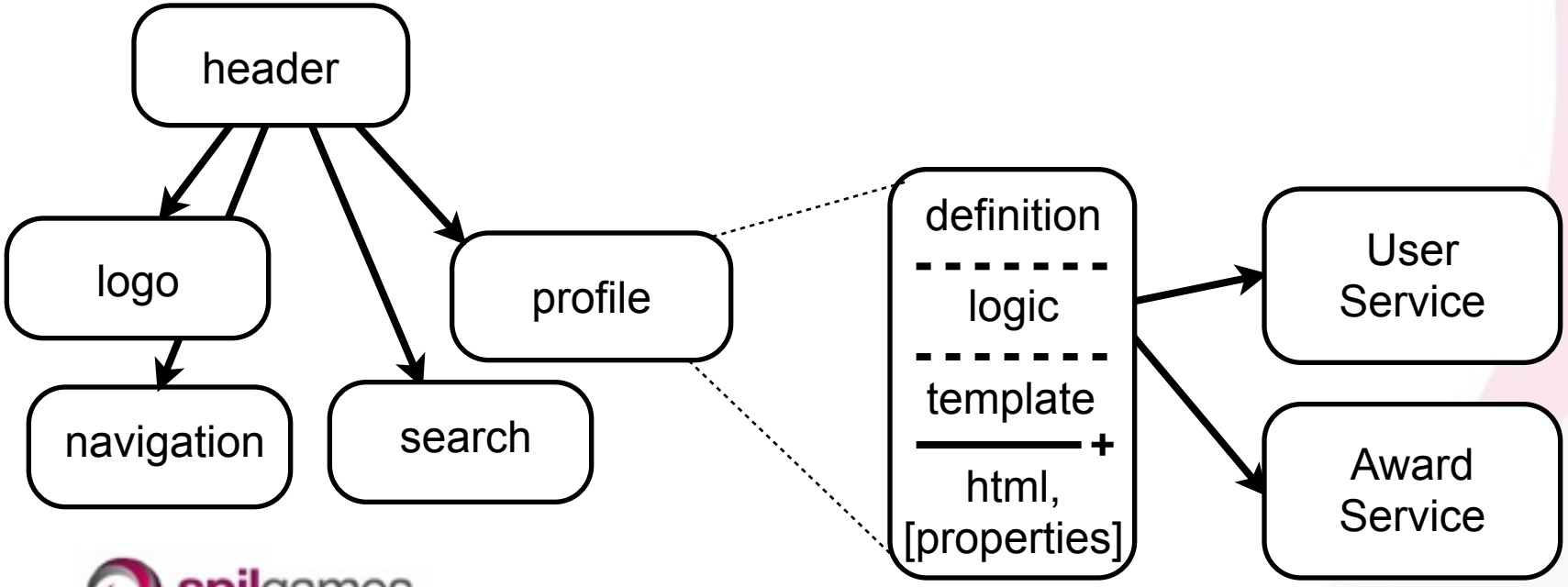

AGAME.COM
Your zone to play free online games
Welcome
Some-user
Level 1
0 points
Profile | Notifications 1 | My Faves | Log Out
I'm looking for...
Search
HOME
BEST GAMES
SOCIAL GAMES
RACING
ACTION
GIRLS
SPORTS
SKILL
MULTIPLAYER
MORE
1 2 3
PLAY IT!
Bridal Beauty Makeover
When the Big Day finally arrives, every blushing bride wants to look her best.
NEW GAMES
Chocolate Churros: Sara...
Animals Connect
Princess Gowns Makeover
Super Stacker 2
Hair Expert
Slots!
CycloManiacs 2
Drakensang
Lazerman
See All New Games
MOST PLAYED
Galaxy Life
Hair Expert
Snail Bob
Slotomania
Family Barn
Street Sesh
Super Stacker 2
Uphill Rush 5
CycloManiacs 2
Bridal Beauty Makeover
More
BEST RATED
RECENTLY PLAYED
Flakboy
Flakboy 2
PLAY SOCIAL GAMES
MORE SOCIAL GAMES
Slotomania
Galaxy Life
Goodgame Empire
Family Barn

Widgets

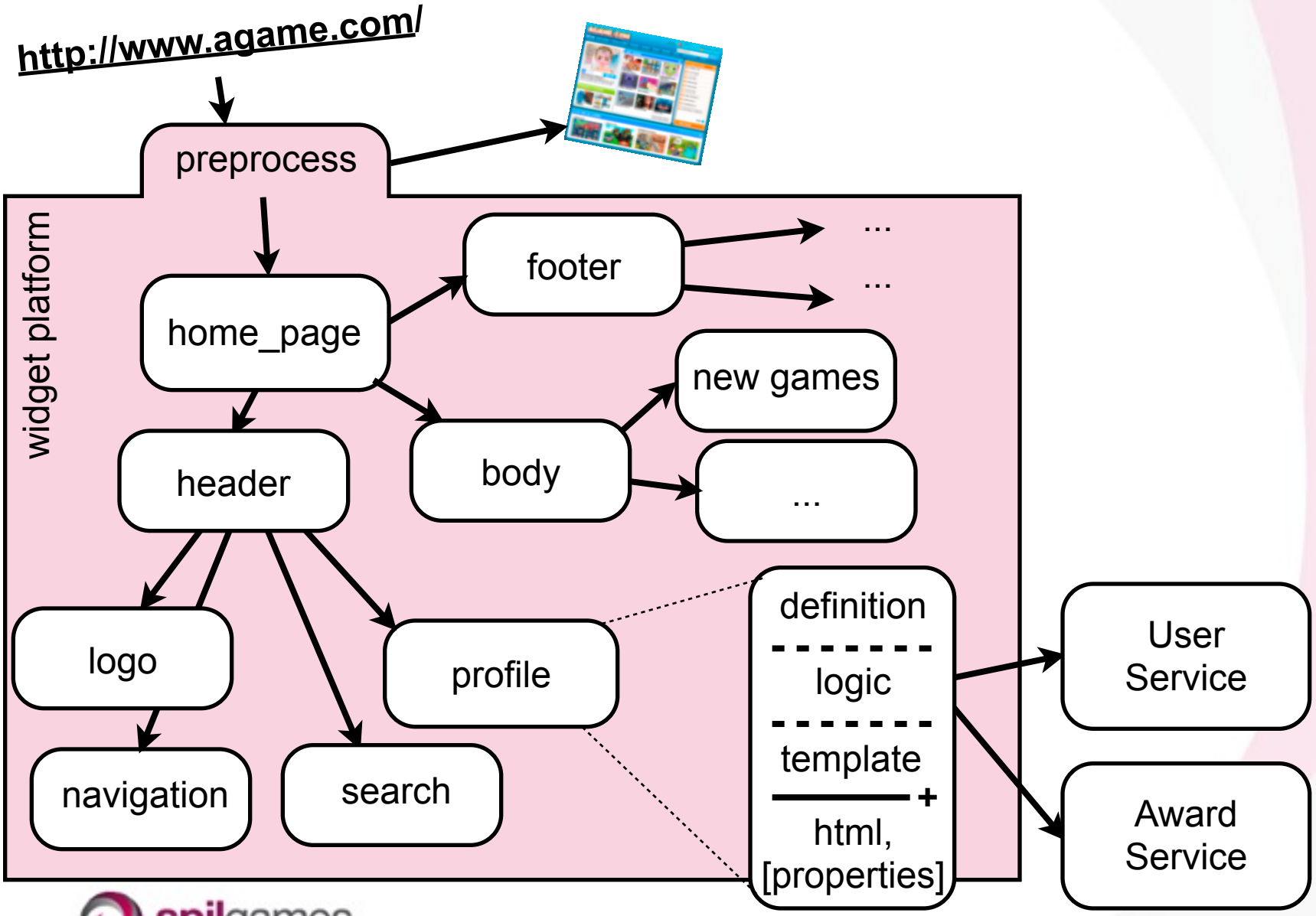

logo widget
profile widget
search widget
navigation widget
recommendation widget
new_games widget
most_played widget
recent_games widget
social_games widget


profile
definition
logic
template
+
html,
[properties]
User
Service
Award
Service

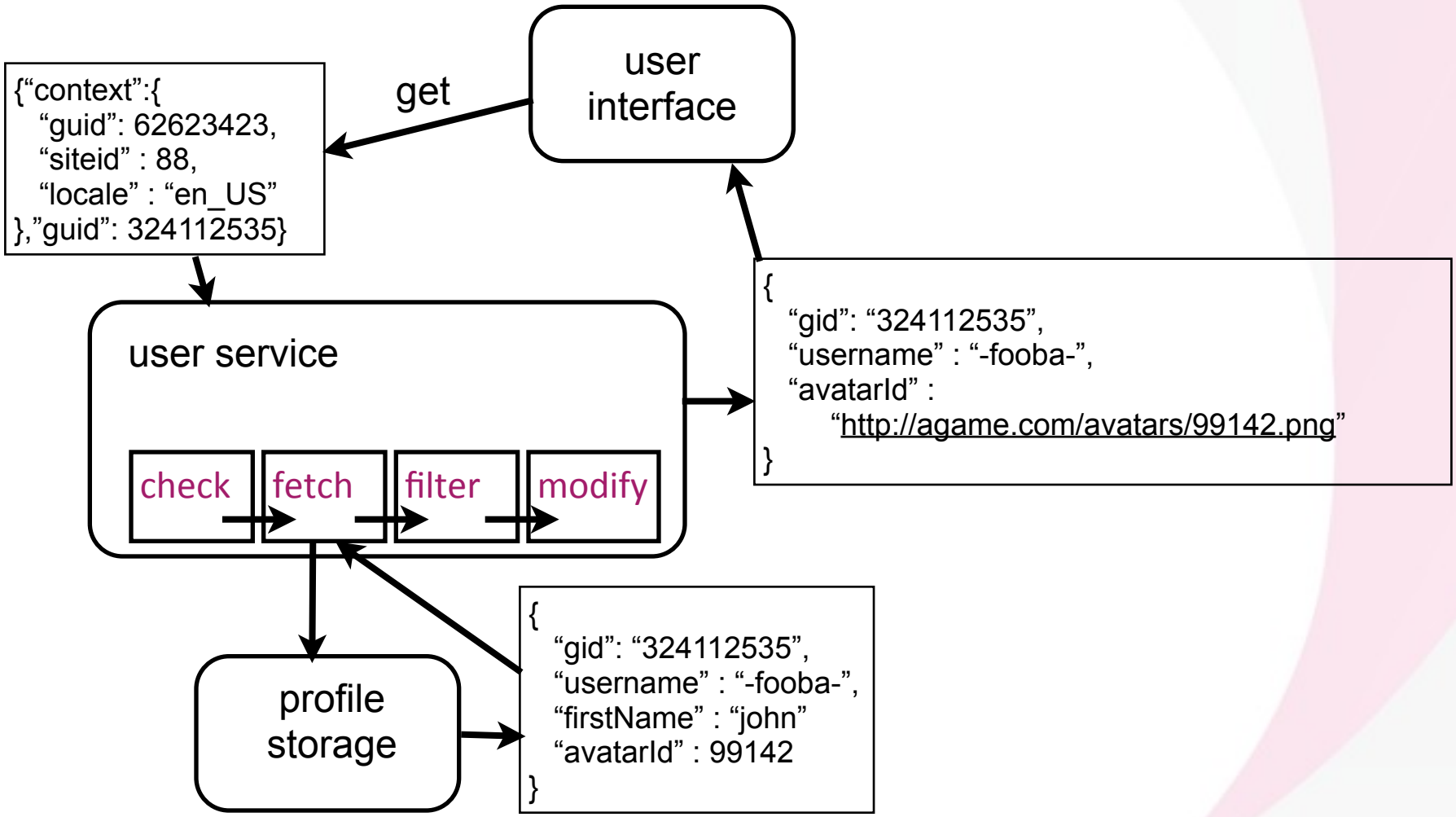
spilgames


header
logo
navigation
search
profile
definition
logic
template
html,
[properties]
User
Service
Award
Service

spilgames


http://www.agame.com/
preprocess
widget platform
footer
...
...
home_page
new games
header
body
...
logo
profile
definition
logic
template
+
html,
[properties]
User Service
Award Service
navigation
search

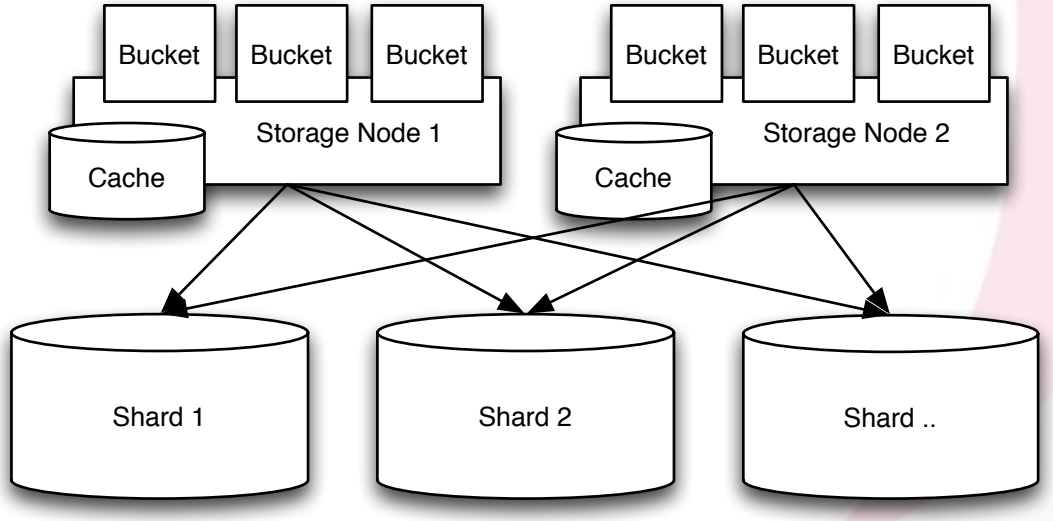
Services



spilgames

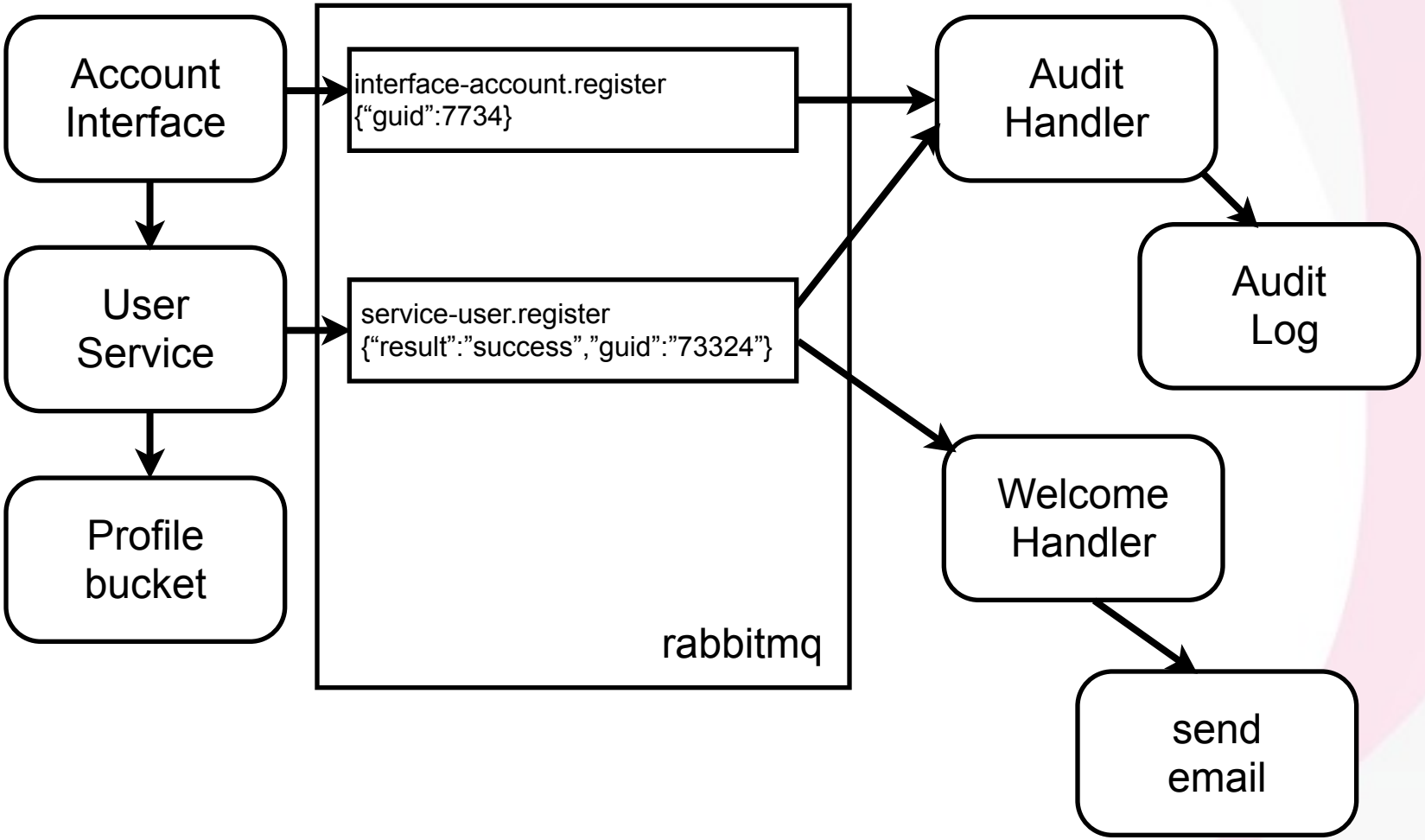
Storage platform

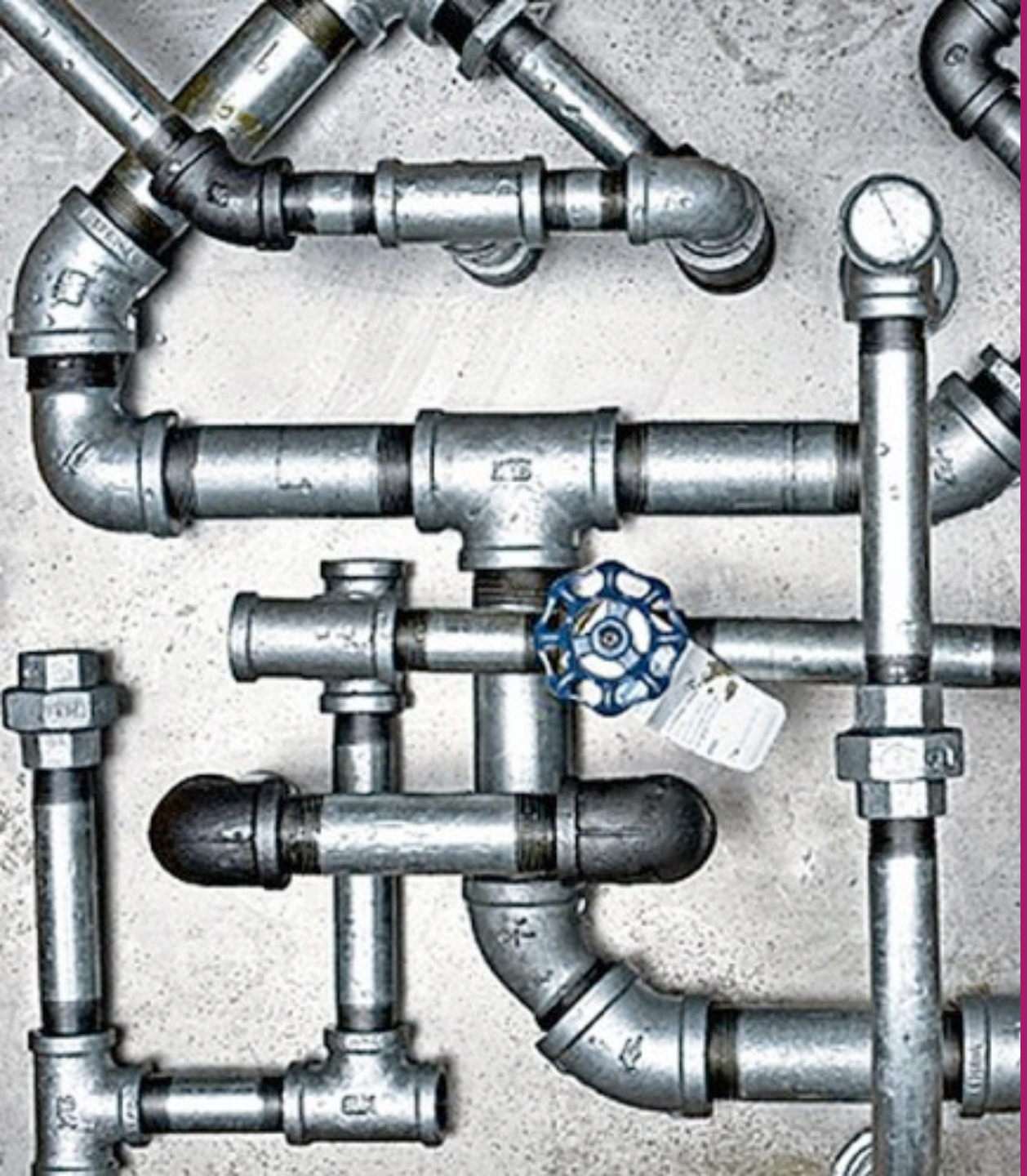
- API instead of SQL
- Buckets of data = Key-Value with Schema
- Abstracts storage



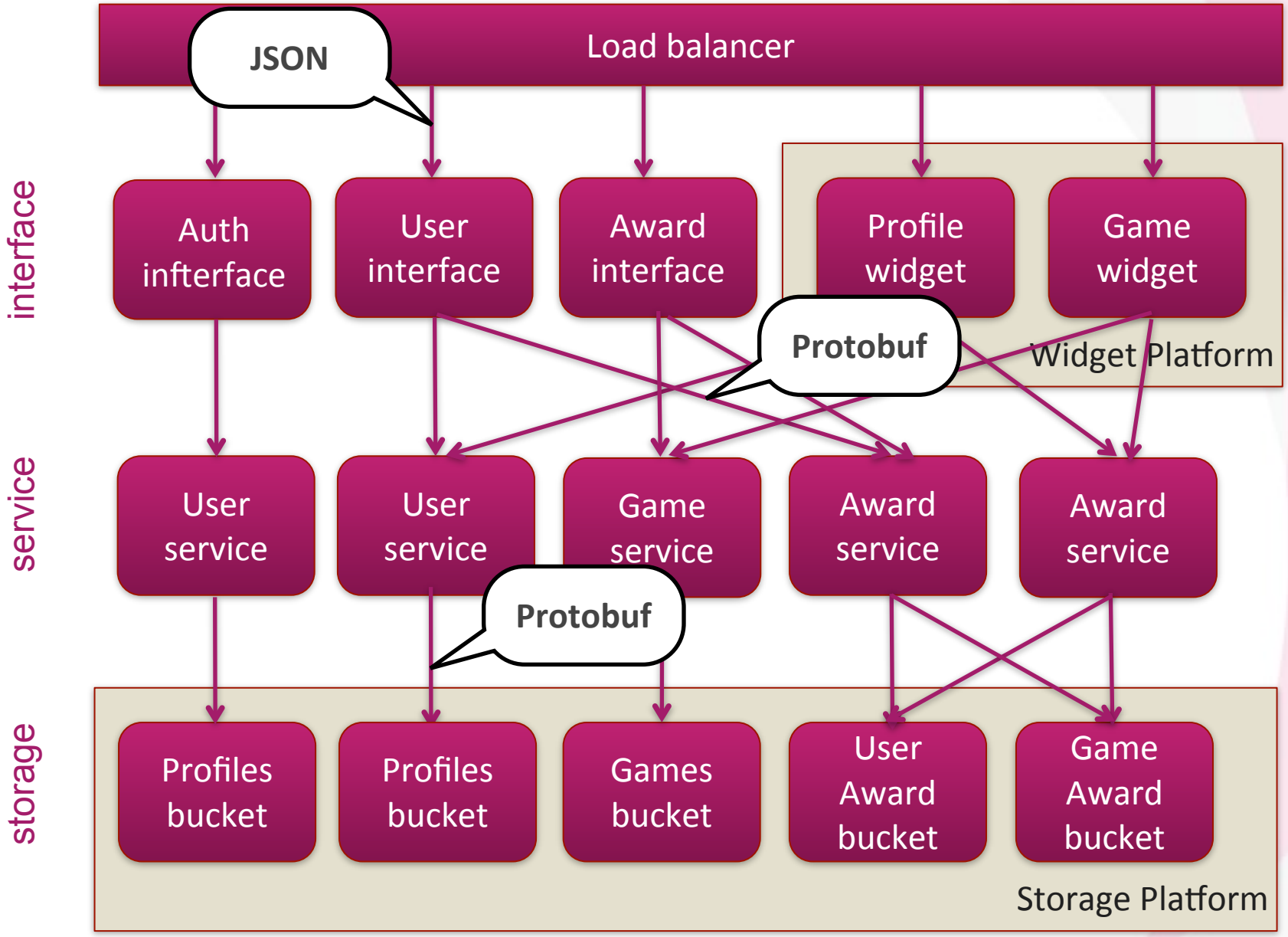
spilgames

Job handlers





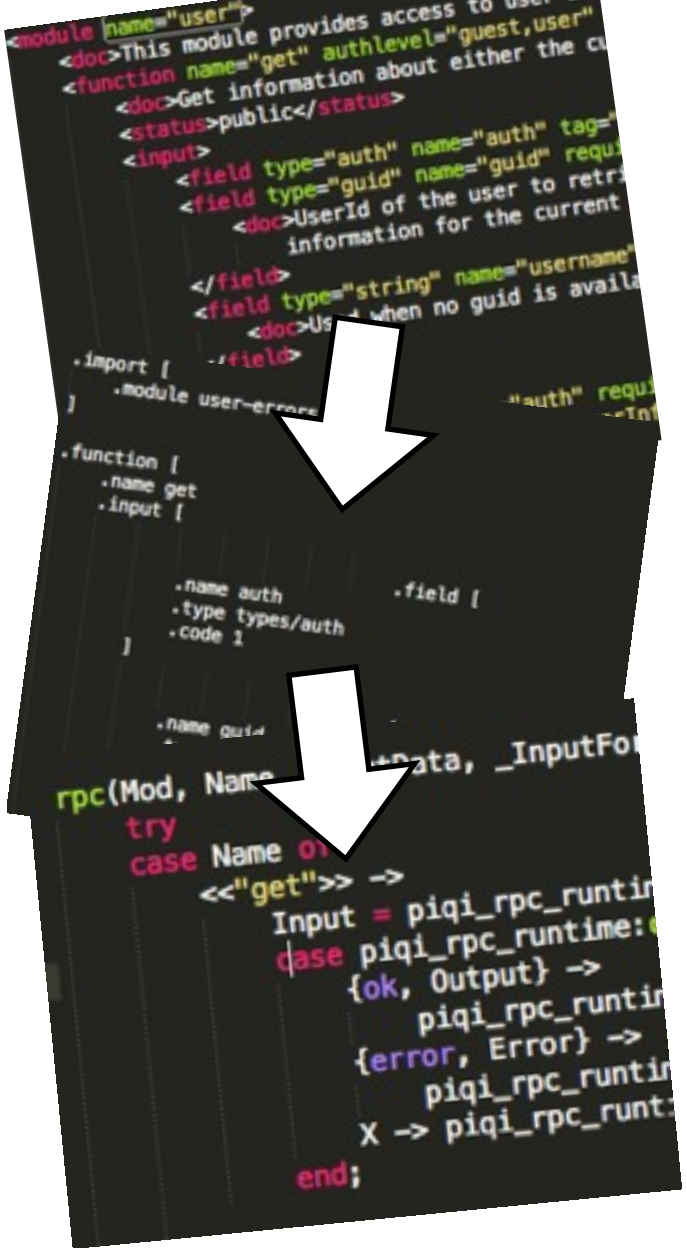
Fitting it together


Load balancer
JSON
interface
service
storage
Auth infterface
User interface
Award interface
Profile widget
Game widget
Widget Platform
Protobuf
User service
User service
Game service
Award service
Award service
Protobuf
Profiles bucket
Profiles bucket
Games bucket
User Award bucket
Game Award bucket
Storage Platform

How we do it

- Strictly define **EVERYTHING**
  - input/output format, urls,
  - rpc functions, types, validation
- Strict rules on changes
  - Everything optional
  - Unless you're really, really sure (and have proof!)
- Generate code and documentation
  - Interface derived from specification
  - Documentation derived from specification

spilgames



- XML: Definition
- XSLT: Transformation
- Piqi:
  - Protobuf & Json encoding/decoding
  - HTTP RPC

(http://piqi.org/)

spilgames

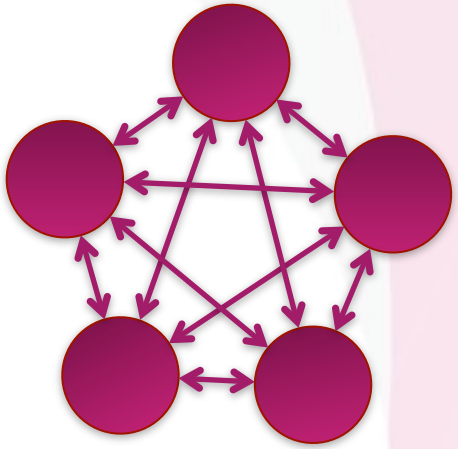
What we ended up with…

- Small applications running on their own nodes
- Scale up where necessary
- Many of them on the same machine
- Small cluster already contains hundreds of nodes

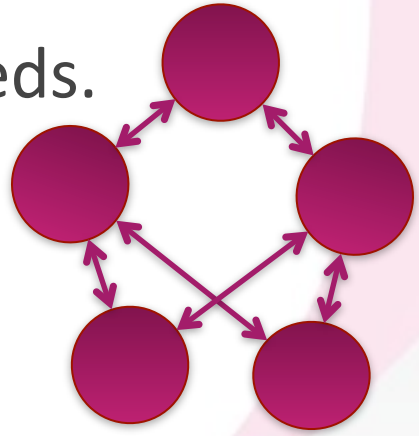
spilgames

Keeping it connected

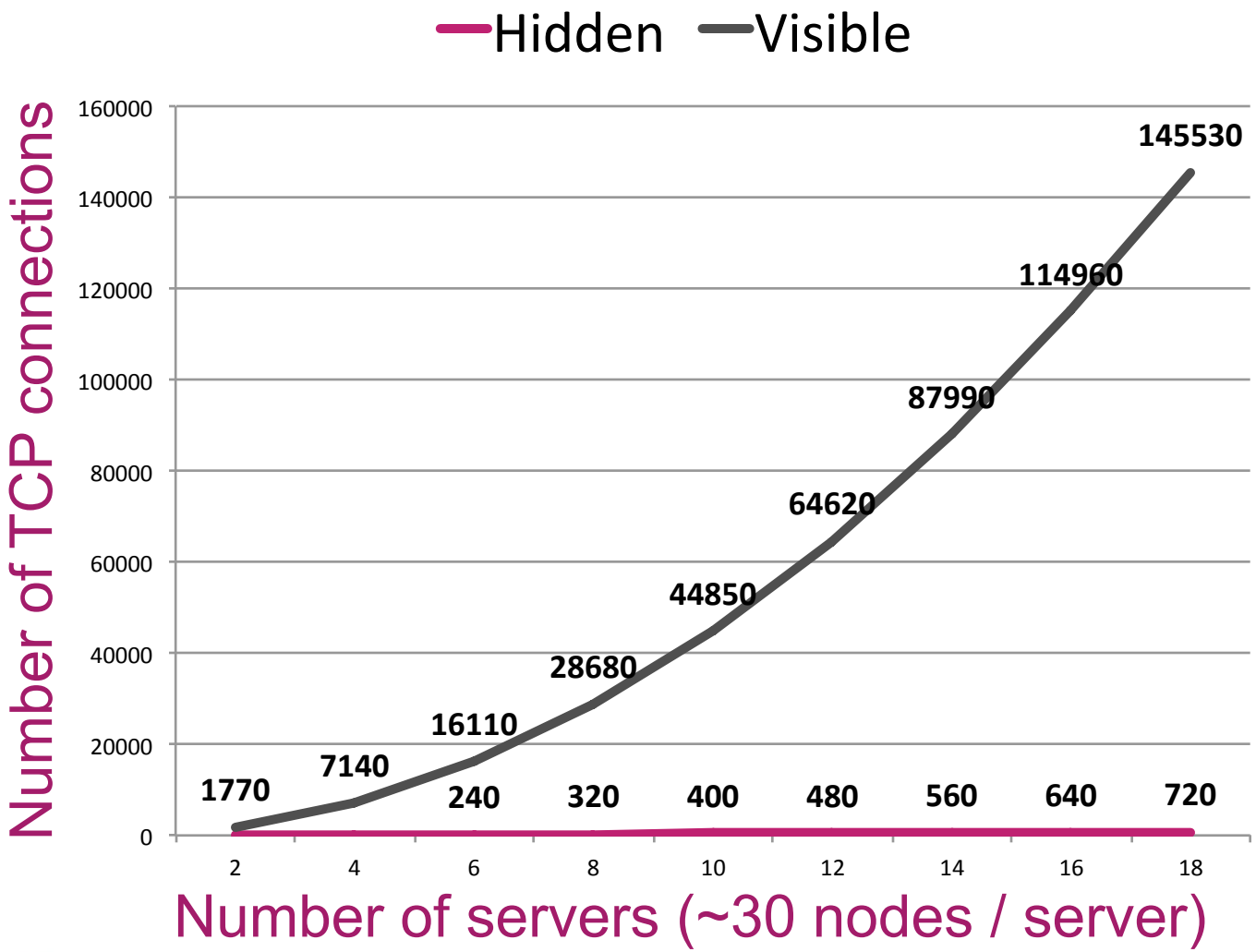
- Connecting all nodes is a bad idea:
  - Not needed
  - Number of open connections



- Solution: use hidden nodes
  - Each node specifies the nodes it needs.
  - How does it scale?



Hidden nodes vs. visible nodes



spilgames

Router

- Each application has it's own router
- Scans & Monitors
- Balances requests across nodes
- Parallelizes requests as much as possible
- Application can only talk to what it has **defined**

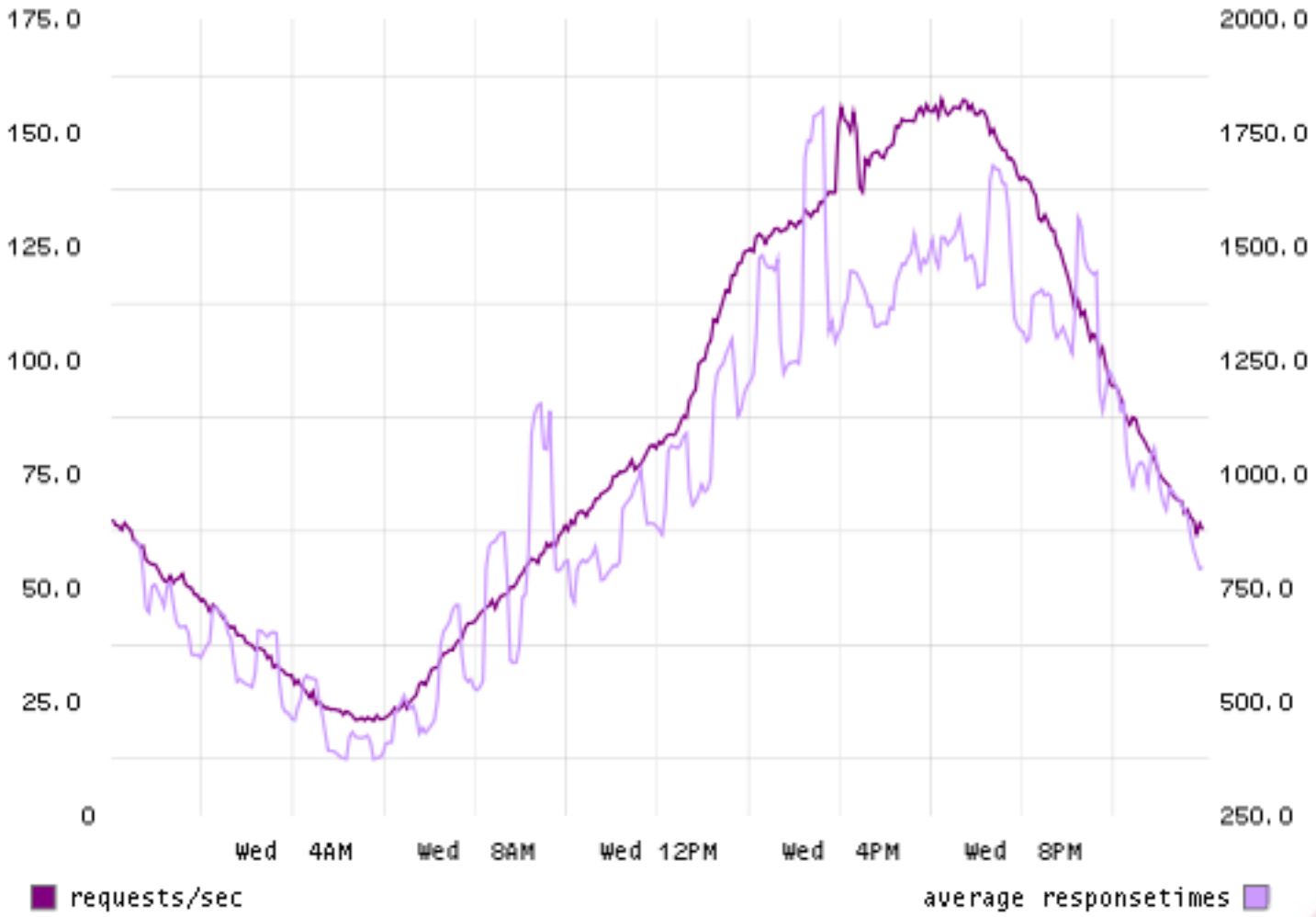
spilgames

But what about response times?

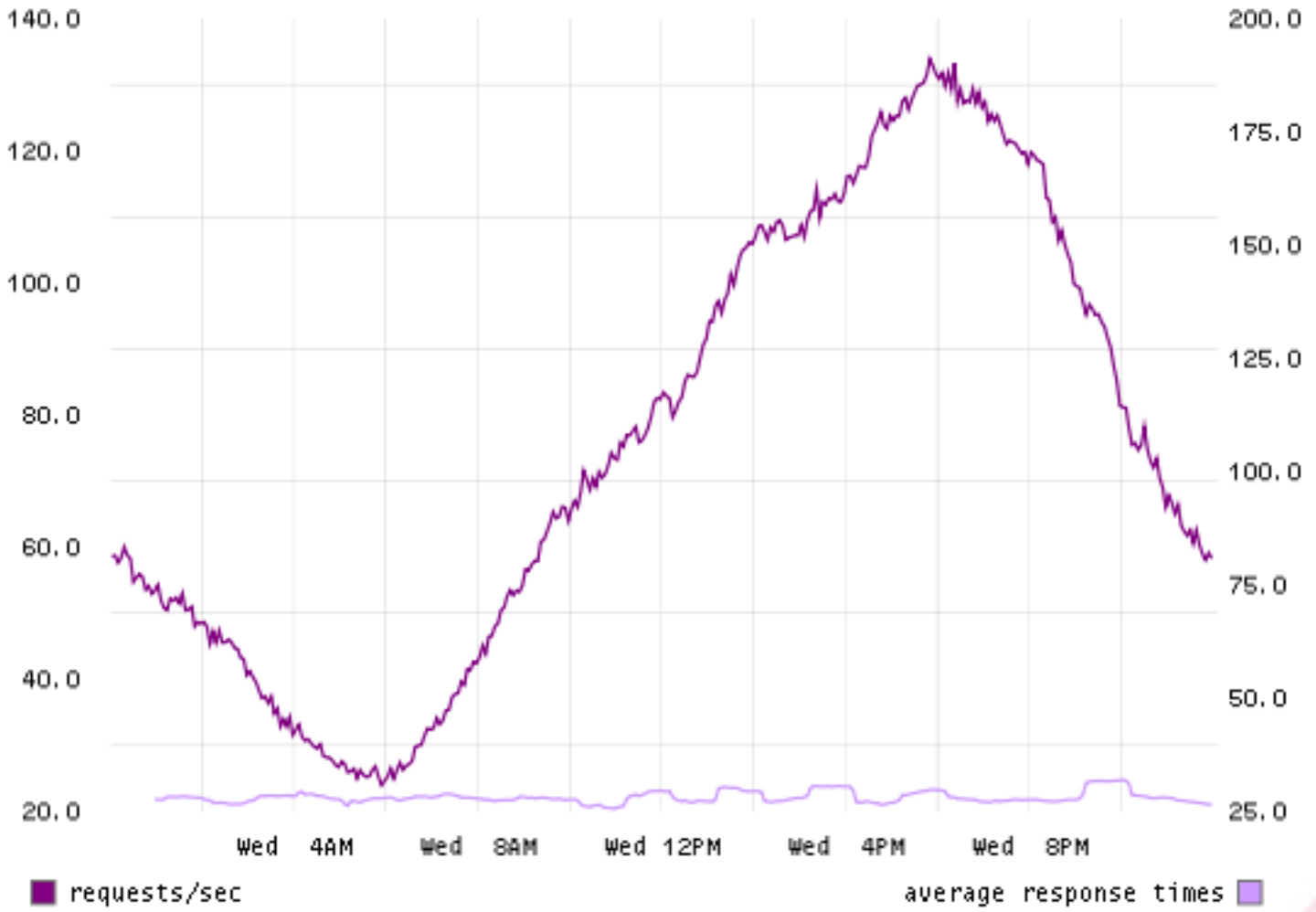
- Use proper transports (Erlang RPC vs. HTTP)
- Yes there is some transport overhead
- Typically < 1ms per layer (+network latency)
- But each layer has predictable response times
- As long as enough downward resources are available

spilgames

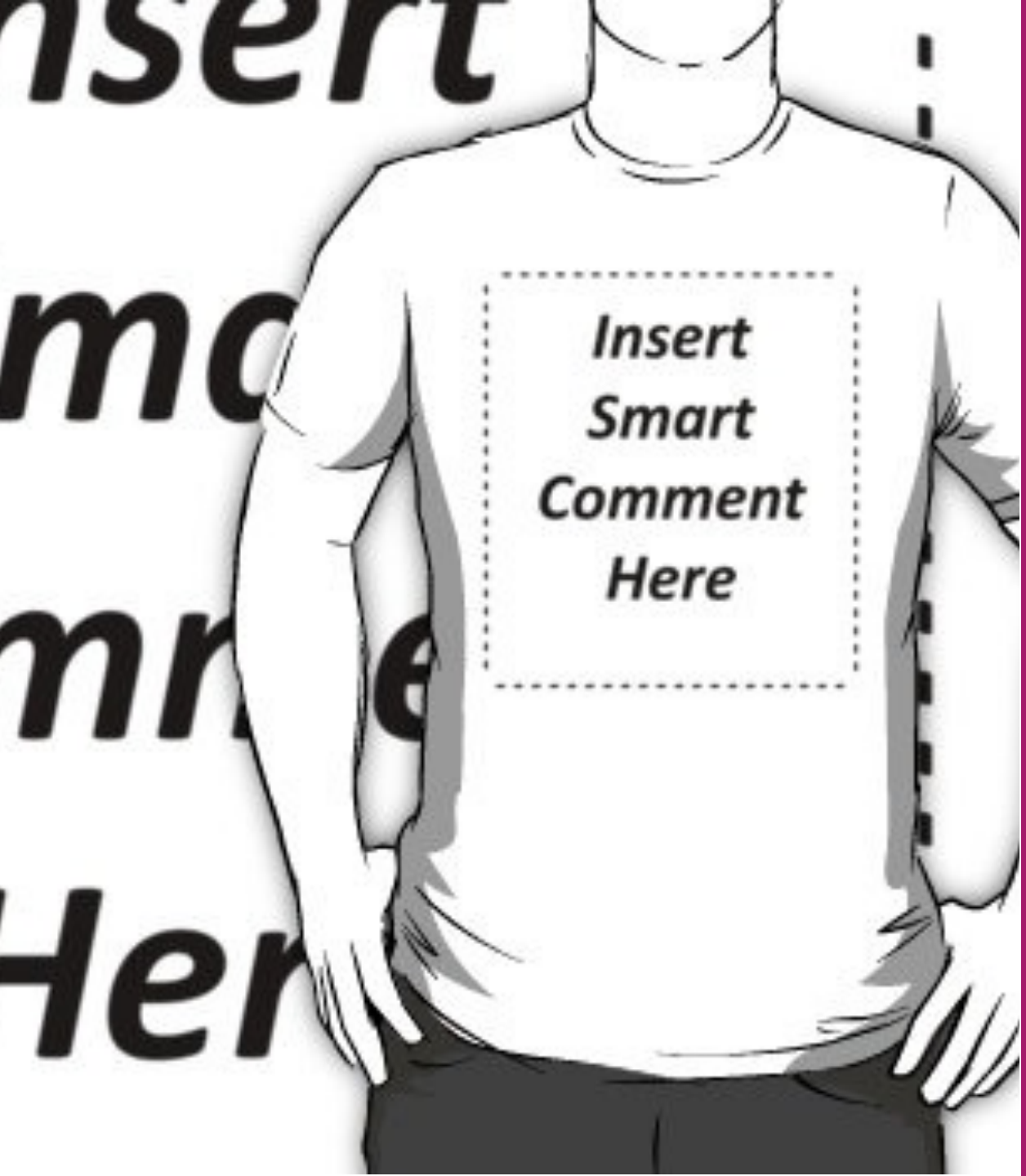
The old ...


175.0
150.0
125.0
100.0
75.0
50.0
25.0
0
2000.0
1750.0
1500.0
1250.0
1000.0
750.0
500.0
250.0
Wed 4AM
Wed 8AM
Wed 12PM
Wed 4PM
Wed 8PM
requests/sec
average responsetimes

.. and the new.



spilgames



Closing Words

Where are we now..

- Interface layer has been defined and largely implemented
- Implementing the service & storage layer step by step
  - Meanwhile we have adapters in place
- Several of the new API's are in use in production
- First portals on the new widget platform will be deployed in the coming months

spilgames

... And where we're going

- Deploying to multiple datacenters this year
- Bring elasticity to our deployments
- Decommission our "old architecture"
- Erlang game platform

spilgames

Lessons Learned

- An architectural vision is crucial
- Integrating into a mess is a mess
- Layers with clear responsibilities
- Strict communication
- Package small
- Change is the only constant
- Optimize for predictability first

spilgames

Questions

Thank you!