Your First Million Players

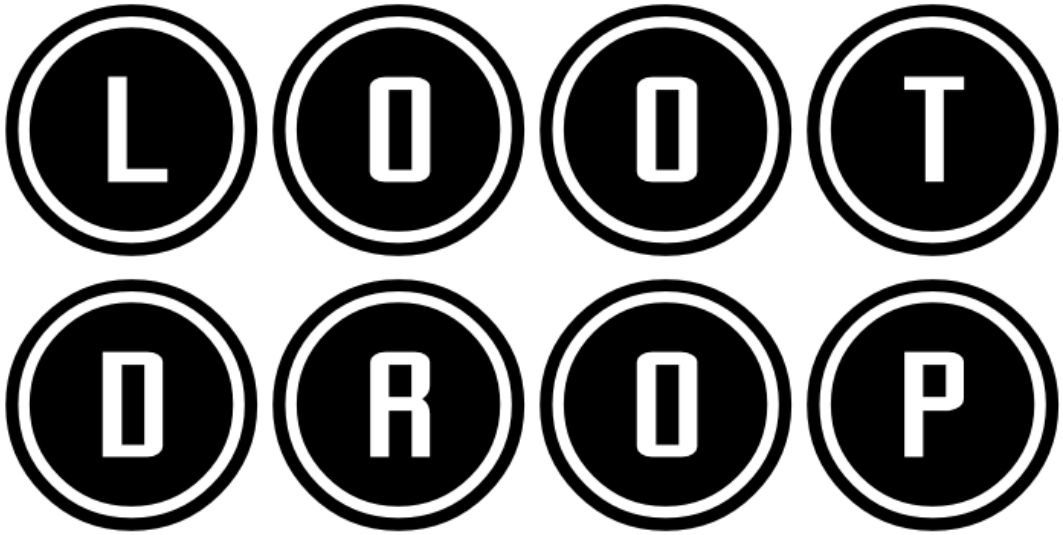
Scalable Architecture for Digital Games

Who are we?

Andrew Kane

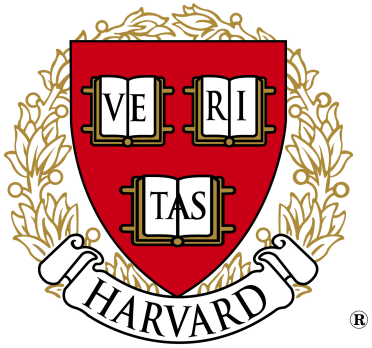
andrewmkane.com

@codemastermm



Who are we?

Dev Purkayastha
devpurkayastha.com
@devp



Who are we?

Jordan Toor
@gammasts

THUUZ™

What is "scalability"?

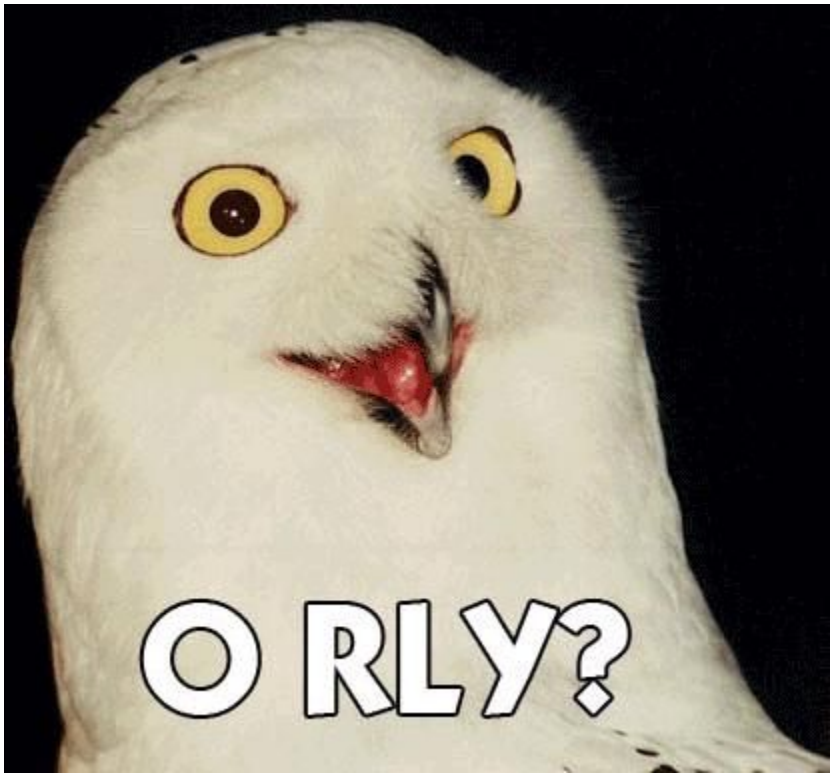
"Ability of a system ... to handle a growing amount of work ... or its ability to be enlarged to accommodate that growth"

http://en.wikipedia.org/wiki/Scalability

How does one scale?

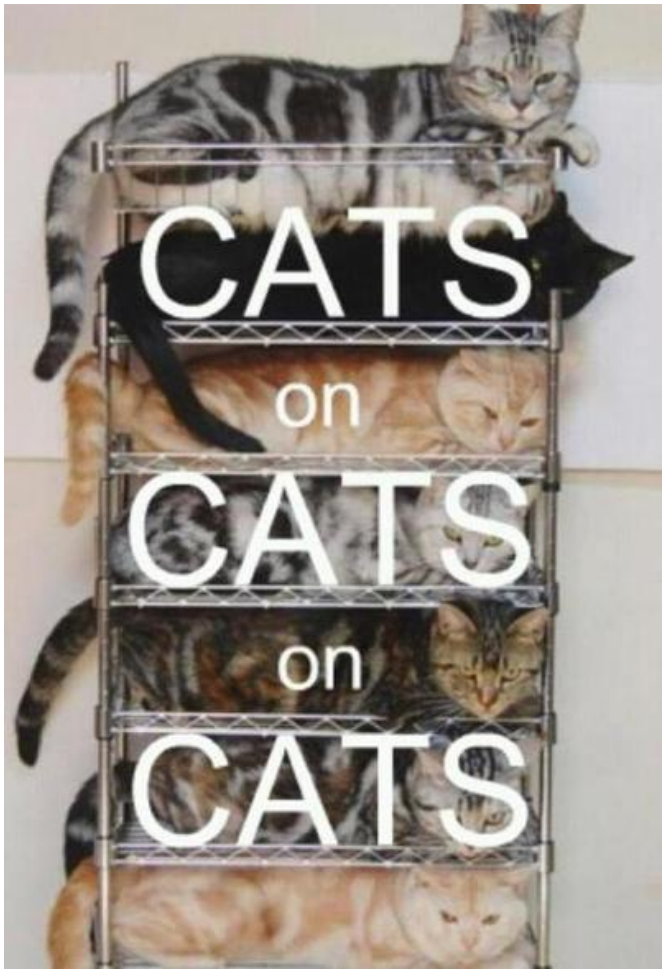
"it depends"

- Brian Ballsun-Stanton (Data Architect, University of New South Wales)

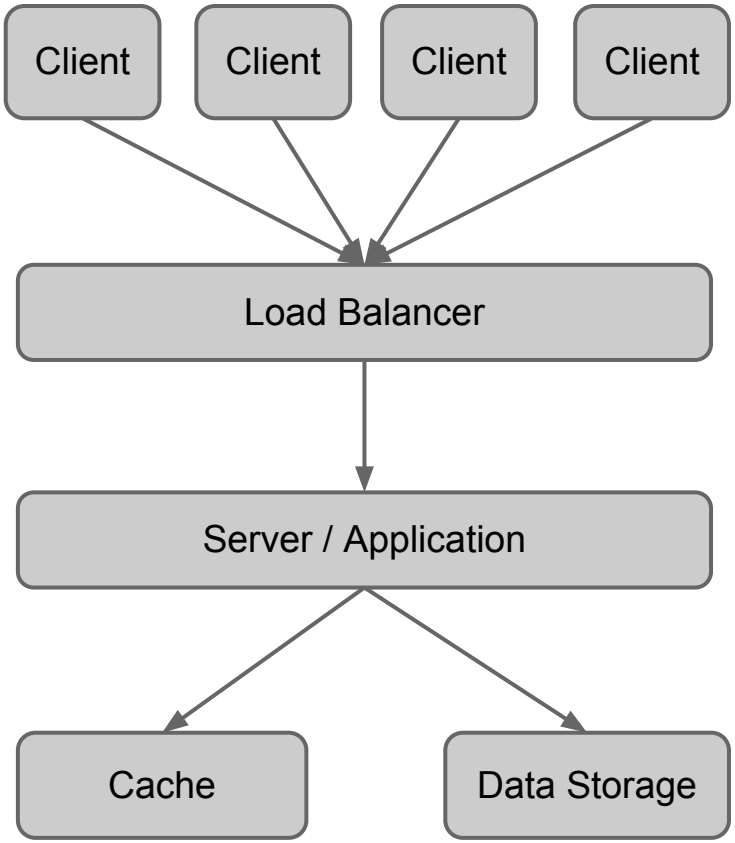


Yes, really. "It depends".

Functional Tiers of a Game



Functional Tiers of a Game



Client Tier

- Buffering server communication
- Only grab what you need
- Cache on the client when possible

Static data

- Store consistent data in clients' memory
- Provide generated data in client builds
- Cache mostly consistent data queries
- Content Delivery Network (CDN)

Consistent Data Stored Client-side

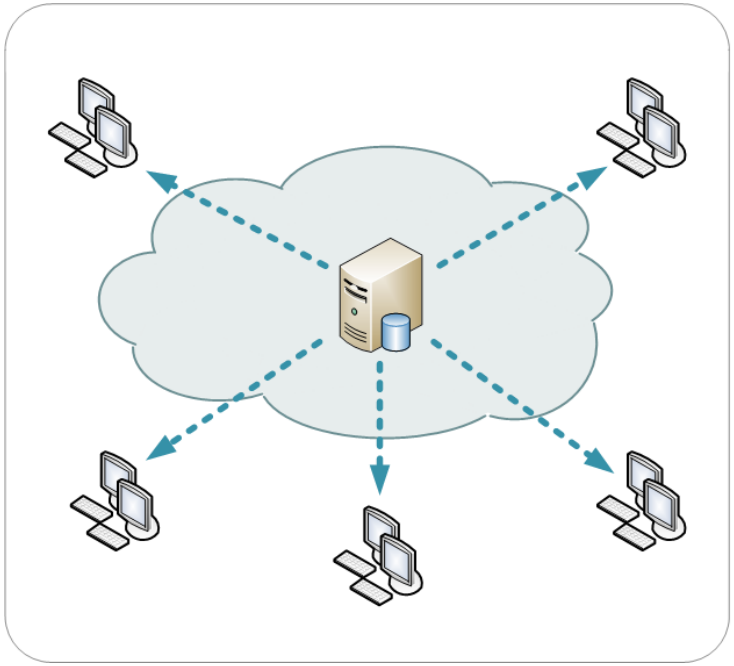
Store data that the client will be able to reuse.

Examples:

- item information
- localization of text

Content Delivery Network (CDN)

- Almost unlimited scaling (for static content)
- Offset scaling issues to service providers
- Data is closer to the client as well



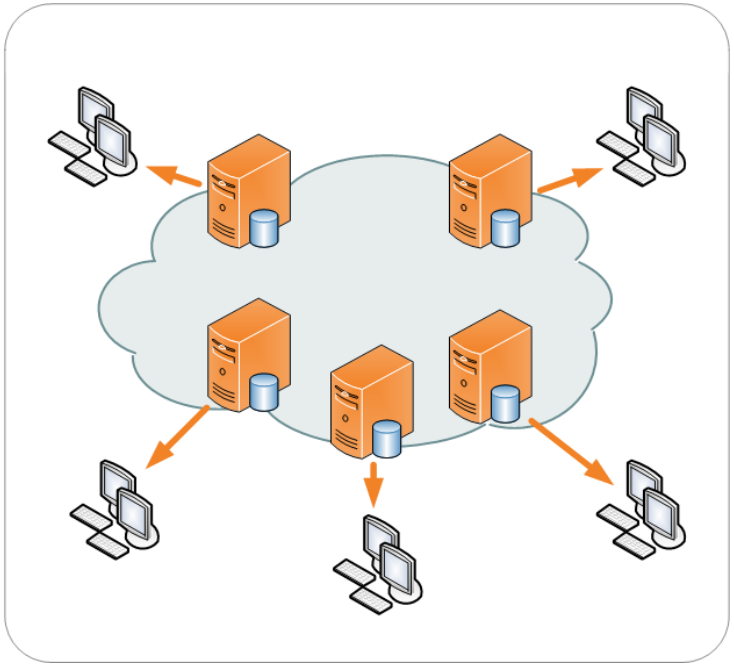


image via Kanoha - wikipedia

Connectivity Tier

- Send only what you need
- Slim down the packet:
  - compress data (gzip, etc.)
  - msgpack
  - Protocol Buffers
- Separate networks
- For mobile: make fewer requests

Interchange Format Comparison

XML: "create_account"

```
<?xml version="1.0" encoding="UTF-8" ?>
	<api>create_account</api>
	<account>
		<username>user123</username>
		<password>popsicles</password>
		<email>user123@example.com</email>
	</account>
	<user_details>
		<name type="first>Joe</name>
		<name type="last">Schmo</name>
	</user_details>
</xml>
```

JSON: "create_account"

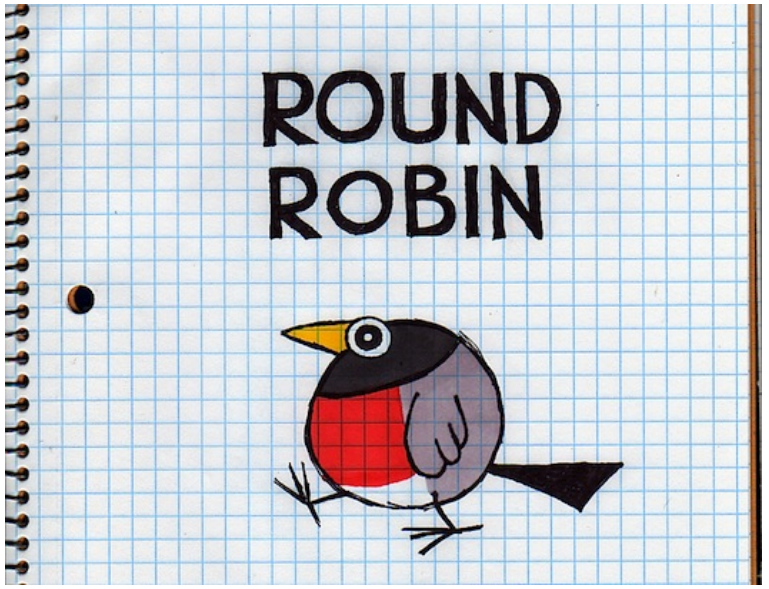
```
{
	"api": "create_account",
	"username": "user123",
	"password": "popsicles",
	"email": "user123@example.com",
	"first_name": "Joe",
	"last_name": "Schmo"
}
```

Protocol Buffer: "create account"

```
0a 07 75 73 65 72 31 32  33 12 09 70 6f 70 73 69   |..user12  3..popsi|
63 6c 65 73 1a 13 75 73  65 72 31 32 33 40 65 78   |cles..us  er123@ex|
61 6d 70 6c 65 2e 63 6f  6d 22 03 4a 6f 65 2a 05   |ample.co  m".Joe*.|
53 63 68 6d 6f                                     |Schmo             |
```

Balancing

- Balance workload between servers
- Data Priority
- Content-aware delegation
- Security
  - Firewall
  - DoS protection



Server/Application Tier

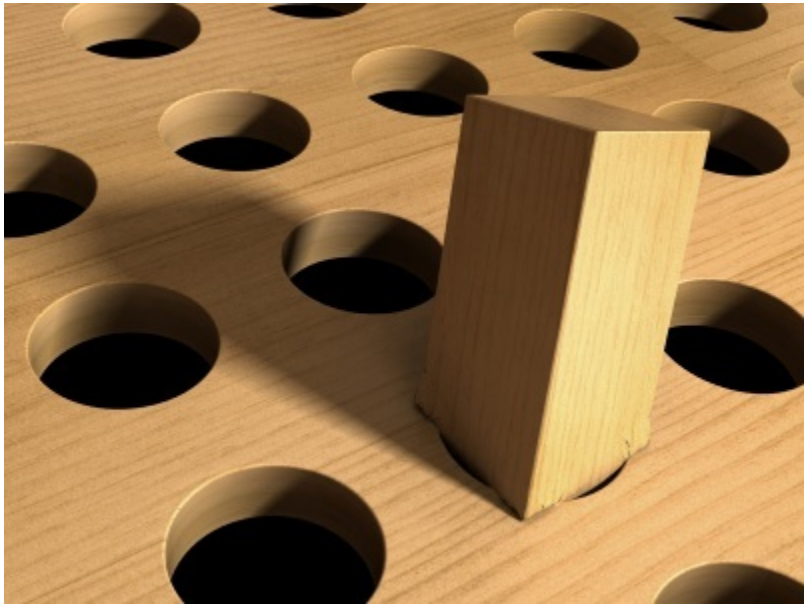
- Modularize!
- Support your balancing strategy
  - Usually stateless
- Proper application server
  - Example: Apache HTTP vs nginx for HTTP
  - Example: Glassfish vs Tomcat vs Jetty vs ...
- Language choice probably doesn't matter
  - Unless it does

Data(base) Tier

- Cache seldom changed data queries
- Proper DBMS for data storage & access
- Optimize database queries
  - Indices may be good or bad
  - Normalization can also good or bad

Choosing your DBMS

- Use DBMSs that store your data logically
  - Tabular (MySQL, PostgreSQL, MSSQL, etc.)
  - Document (CouchDB, MongoDB, etc.)
  - Key-Value (Membase, Redis, Riak, etc.)



DBMS Choice Example

Data Sharding

document or tabular store:

```
player { id, server_id }
```

key-value store:

```
player_id => server_id
```

Item Information

key-value store:

```
item_id => json information
```

tabular store:

```
item { item_id, name, desc,
effect1, effect1, effect3,
status1, status2 ... }
```

document store:

```
item { item_id, name, desc,
effects: [ ... ],
status: [ ... ] }
```

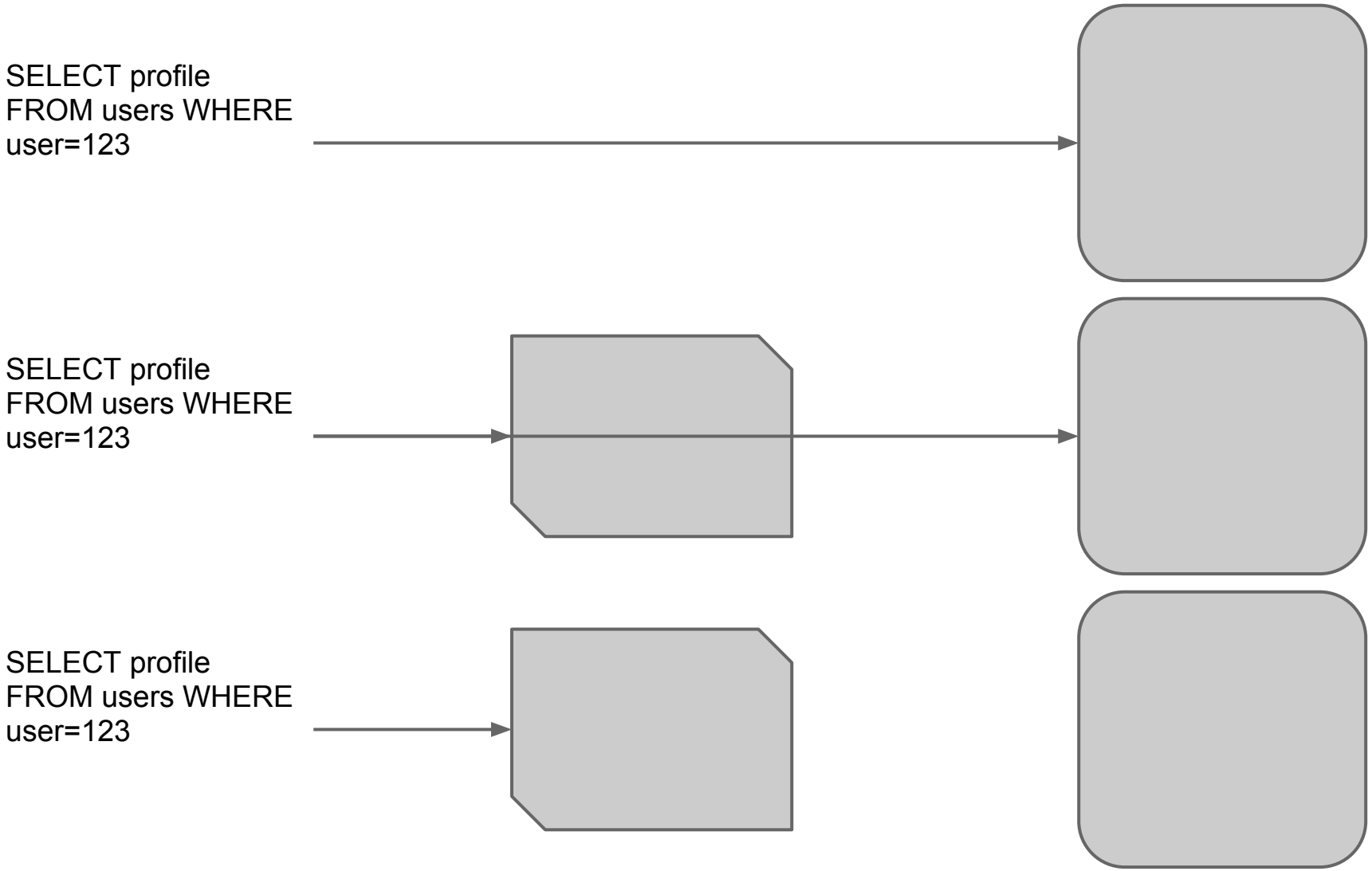
Caching Data Queries

Cache calls to services that provide near consistent data results.

Examples:

- GeoLocation
- Enemy & Monster information
- Account information

Example!


SELECT profile
FROM users WHERE
user=123
SELECT profile
FROM users WHERE
user=123
SELECT profile
FROM users WHERE
user=123

Virtualization and Clouds

- New servers in an instant!
- IaaS (AWS, Heroku, etc.)
  - Easier
  - Less control
- But will instant servers help you scale?

Scaling Timeline

Early stage:

"We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil." - Donald Knuth

(It depends.)

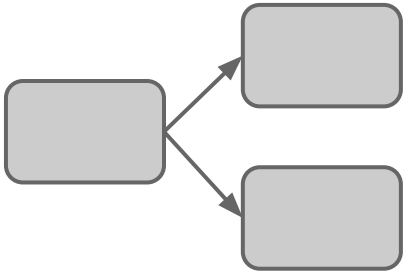
Scaling Timeline

Before you launch:

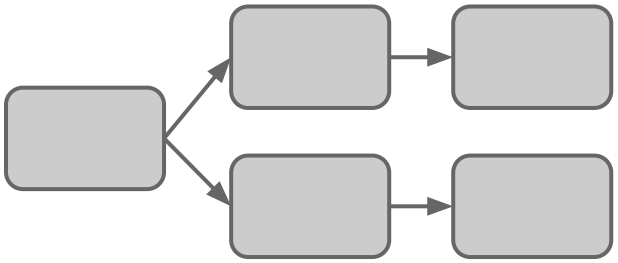
- monitoring and metrics
  - You can't fight what you can't see.
- velvet rope for new users
  - You will be overwhelmed.

Scaling Timeline

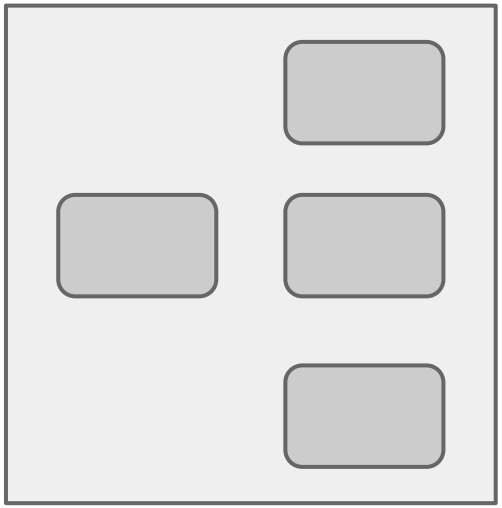
Single App/DB Node



App Node, Cache Node, Database Node



Load Balancer
Multiple App Nodes
Multiple Cache Nodes
Multiple Database Nodes



Splitting off Services (Account Service, Player Service, etc.)

Questions? Contact Info? Puppies?

Feel free to contact us later!

Andrew: @codemastermm andrew@andrewmkane.com
Dev: @devp devp@modmetaphor.com
Jordan: @gammasts gammasts@gmail.com