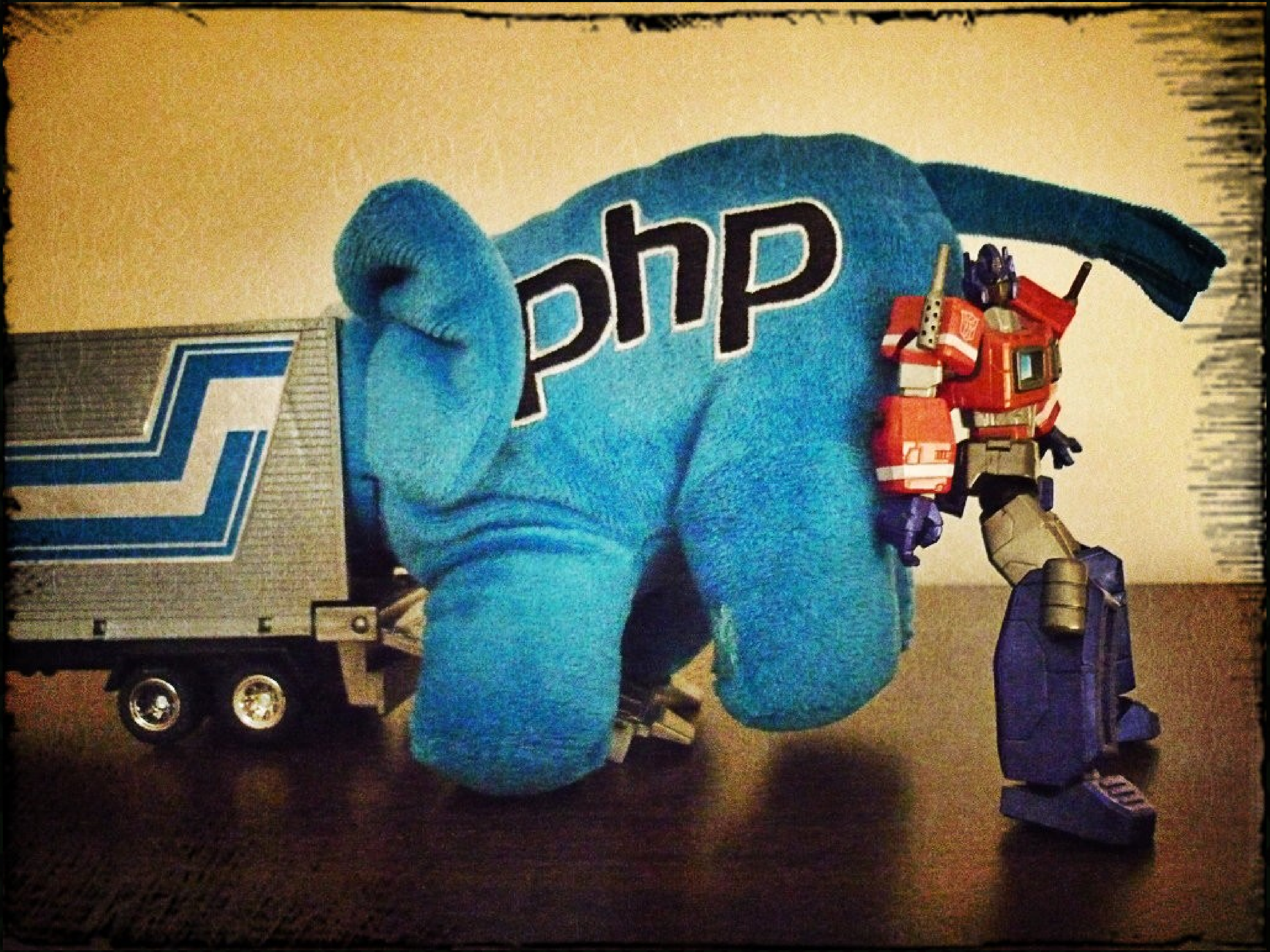
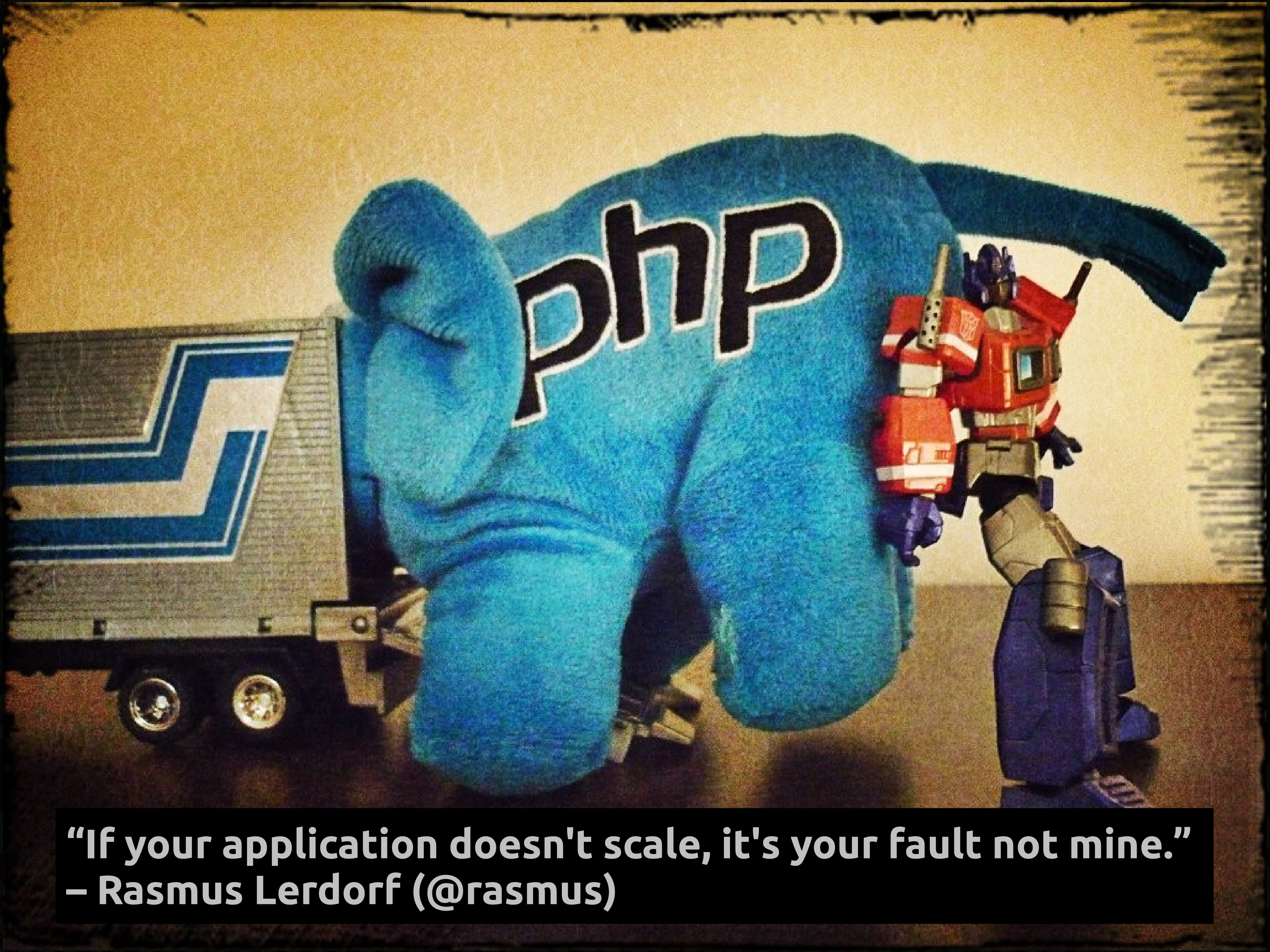
Sensible Scaling



Rowan Merewood



"If your application doesn't scale, it's your fault not mine."
– Rasmus Lerdorf (@rasmus)

Who?

Who?

- @rowan_m

Who?

- @rowan_m
- Software Engineer & Team Lead

Who?

- @rowan_m
- Software Engineer & Team Lead
- @ibuildings & @techportal

Why?

Why?

- I've built small apps

Why?

- I've built small apps
- and pretty large ones

Why?

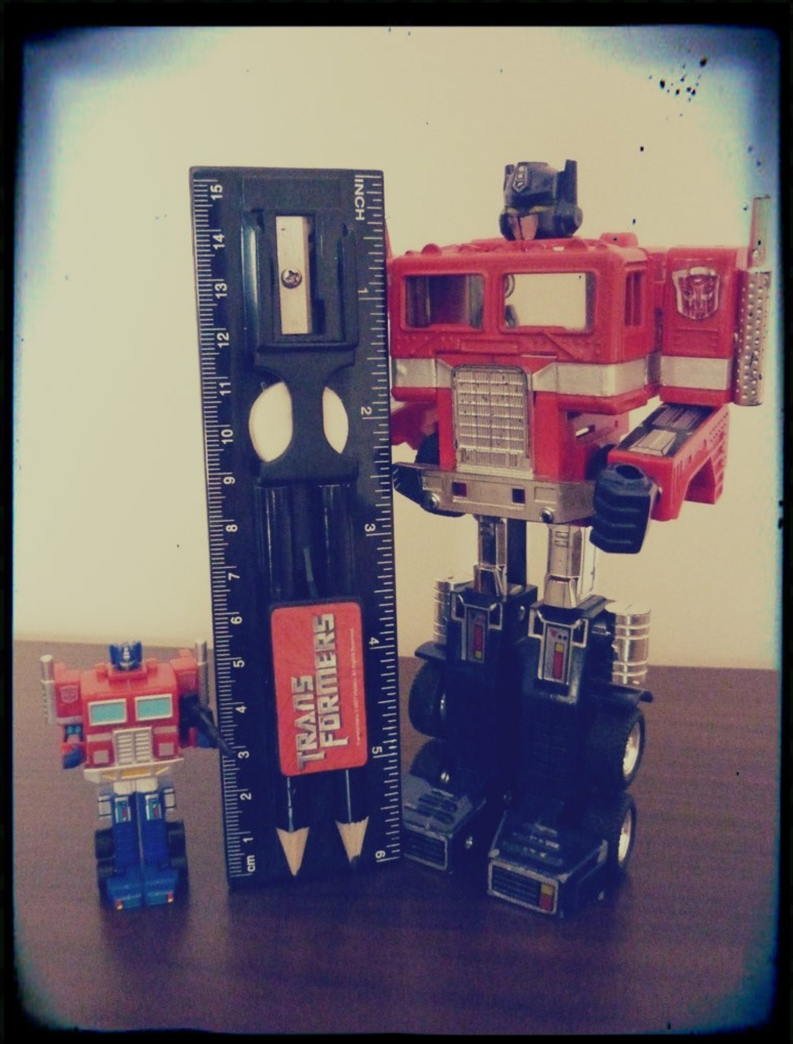
- I've built small apps
- and pretty large ones

- I've seen massive over-engineering

Why?

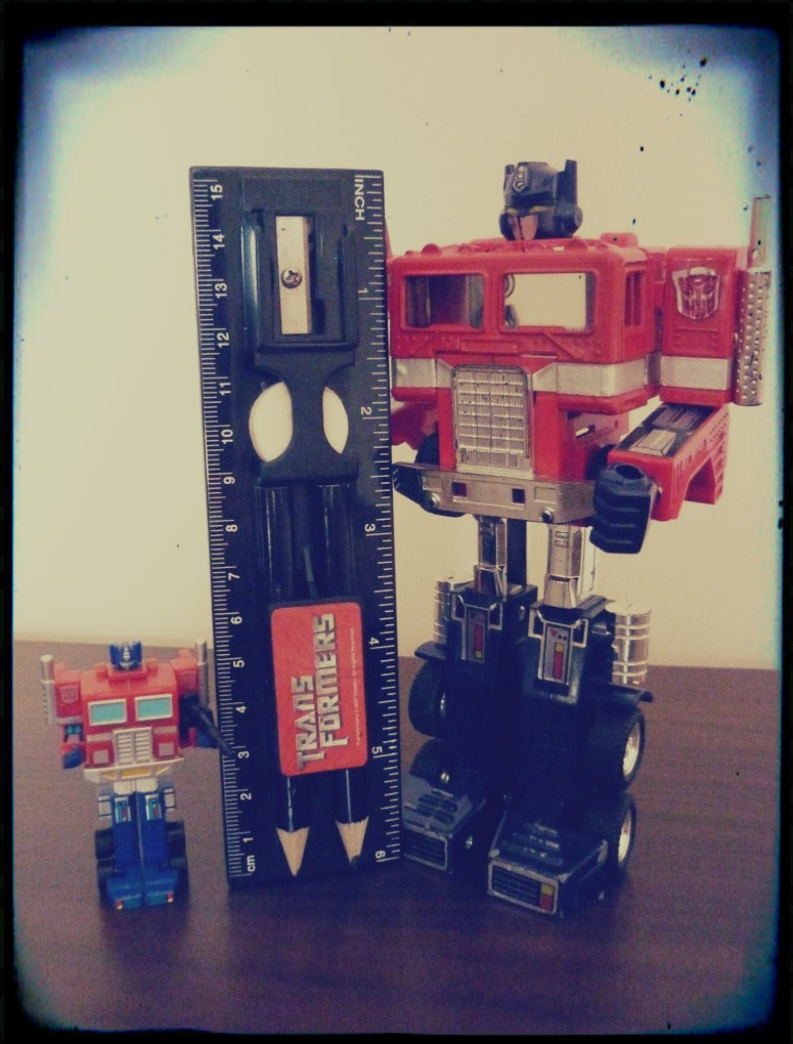
- I've built small apps
- and pretty large ones
- I've seen massive over-engineering
- and platforms that cannot scale

TDD Principles



TDD Principles

- Write **only** enough code to pass the test



TDD Principles

- Write **only** enough code to pass the test
- Do not over-engineer

TDD Principles

- Write **only** enough code to pass the test
- Do not over-engineer
- Don't give away work for free!

Build for now

Build for now

- Don't be afraid to throw code away

Build for now

- Don't be afraid to throw code away
- XP's spikes

Build for now

- Don't be afraid to throw code away
- XP's spikes
- Take advantage of current tech. without tying yourself to it

Who are your users?

Who are your users?

- Lots of short visits?

Who are your users?

- Lots of short visits?
- Anonymous or session-based?

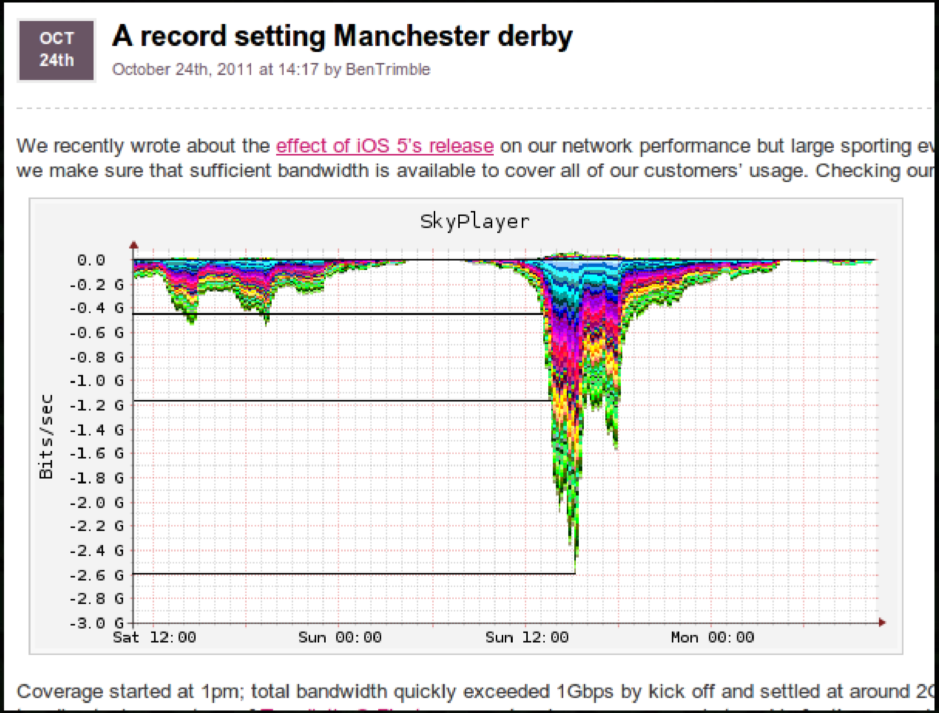
Who are your users?

- Lots of short visits?
- Anonymous or session-based?
- Long, complex interactions?

Who are your users?

- Lots of short visits?
- Anonymous or session-based?
- Long, complex interactions?
- Subscribers or free users?

Monitor and Analyse



OCT 24th

A record setting Manchester derby

October 24th, 2011 at 14:17 by BenTrimble

We recently wrote about the effect of iOS 5's release on our network performance but large sporting ev
we make sure that sufficient bandwidth is available to cover all of our customers' usage. Checking our

Coverage started at 1pm; total bandwidth quickly exceeded 1Gbps by kick off and settled at around 2(

http://community.plus.net/blog/2011/10/24/a-record-setting-manchester-derby/

Monitor and Analyse

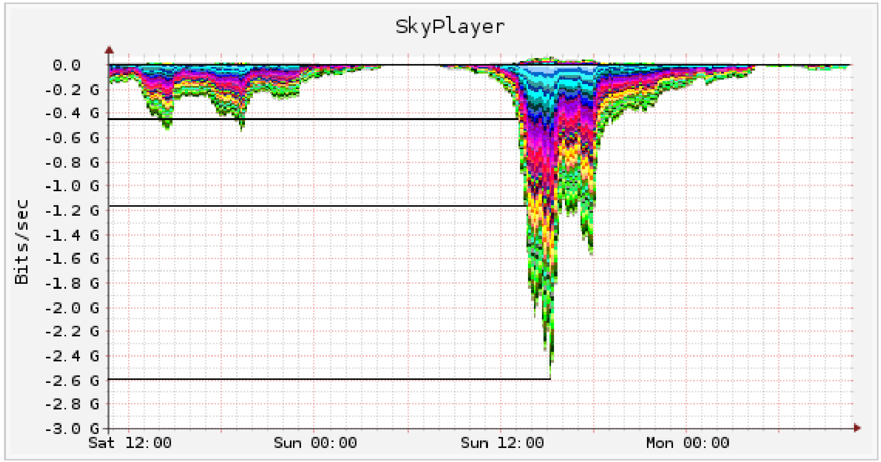
- Predict usage patterns

OCT 24th

A record setting Manchester derby

October 24th, 2011 at 14:17 by BenTrimble

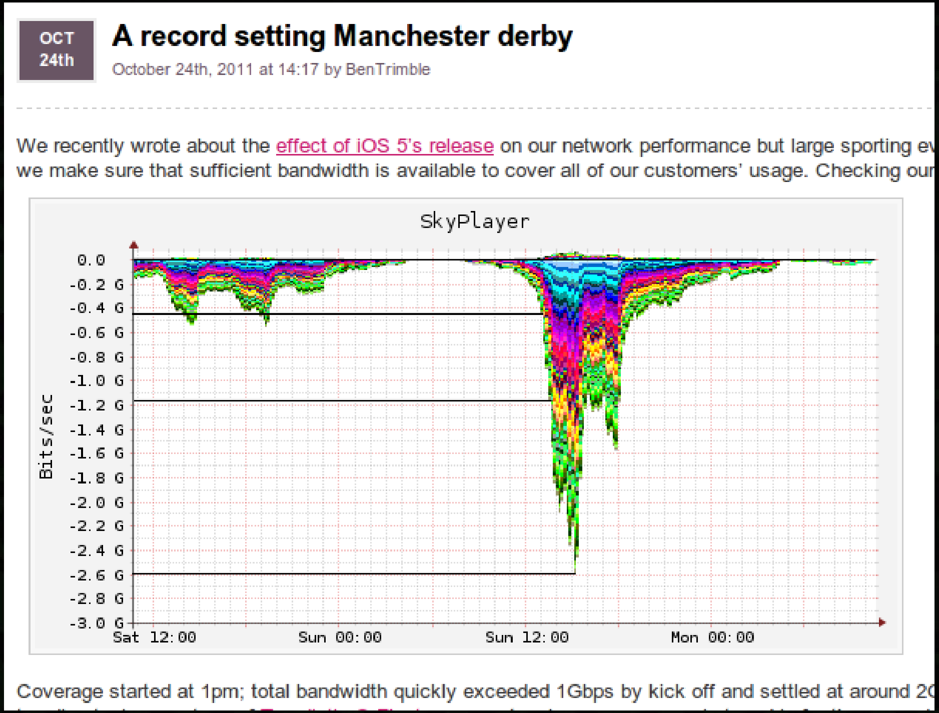
We recently wrote about the effect of iOS 5's release on our network performance but large sporting ev
we make sure that sufficient bandwidth is available to cover all of our customers' usage. Checking our



Coverage started at 1pm; total bandwidth quickly exceeded 1Gbps by kick off and settled at around 2(

http://community.plus.net/blog/2011/10/24/a-record-setting-manchester-derby/

Monitor and Analyse



OCT 24th

A record setting Manchester derby

October 24th, 2011 at 14:17 by BenTrimble

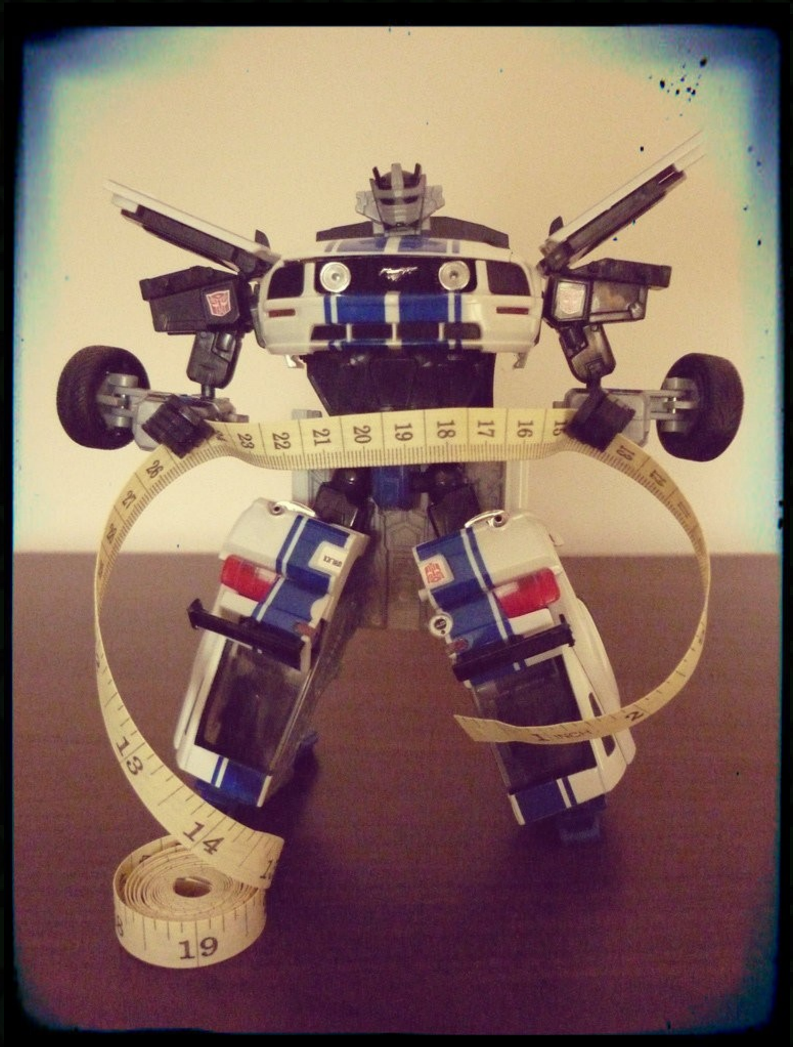
We recently wrote about the effect of iOS 5's release on our network performance but large sporting ev
we make sure that sufficient bandwidth is available to cover all of our customers' usage. Checking our

Coverage started at 1pm; total bandwidth quickly exceeded 1Gbps by kick off and settled at around 2(

- Predict usage patterns
- Plan for peaks and troughs

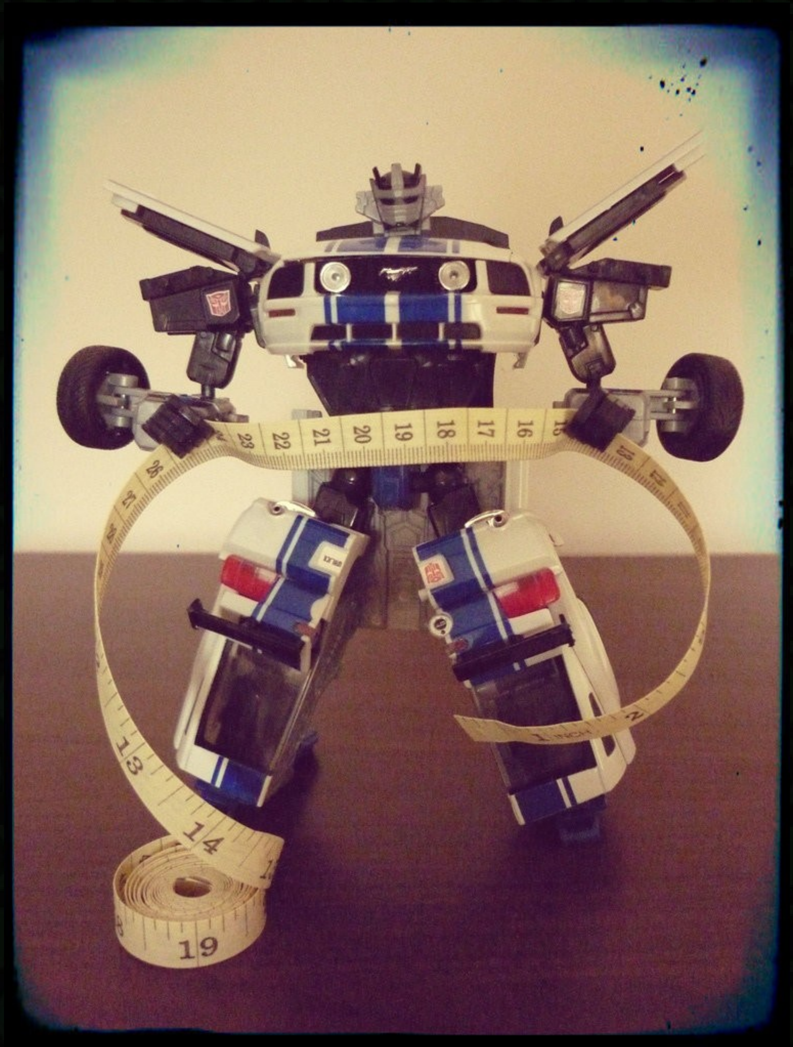
http://community.plus.net/blog/2011/10/24/a-record-setting-manchester-derby/

Profiling



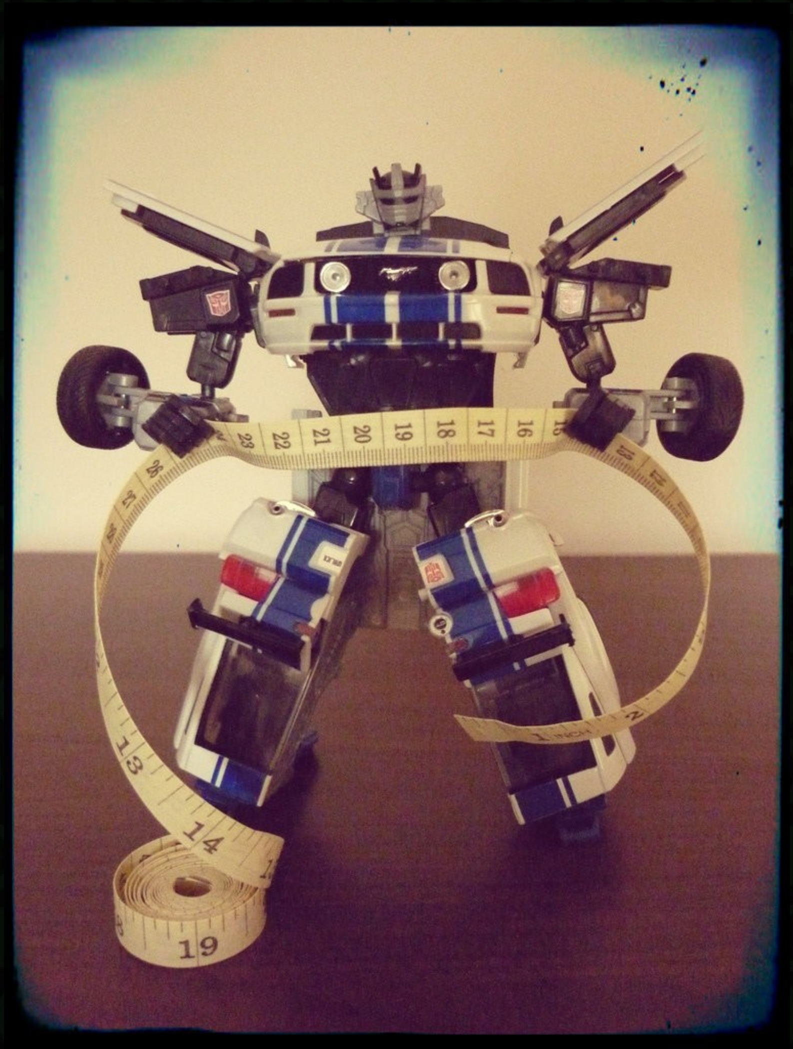
Profiling

- You do not know what is slow



Profiling

- You do not know what is slow
- Ensure you measure more than just code!



Profiling

Profiling

- PHP - Xdebug – Derick Rethans (@derickr)
  - Kcachegrind, Webgrind, etc.

Profiling

- PHP - Xdebug – Derick Rethans (@derickr)
  - Kcachegrind, Webgrind, etc.
- PHP – XHProf

Profiling

- PHP - Xdebug – Derick Rethans (@derickr)
  - Kcachegrind, Webgrind, etc.
- PHP – XHProf
- JavaScript – Firebug

Profiling

- PHP - Xdebug – Derick Rethans (@derickr)
  - Kcachegrind, Webgrind, etc.
- PHP – XHProf
- JavaScript – Firebug
- JavaScript – Venkman

Profiling

- PHP - Xdebug – Derick Rethans (@derickr)
  - Kcachegrind, Webgrind, etc.
- PHP – XHProf
- JavaScript – Firebug
- JavaScript – Venkman

- General page display - YSlow

Profiling

- CPU Load – `top` & `/proc/loadavg`

Profiling

- CPU Load – `top` & `/proc/loadavg`
- Disk IO – `iotop`

Profiling

- CPU Load – `top` & `/proc/loadavg`
- Disk IO – `iotop`
- Memory – `free`

Profiling

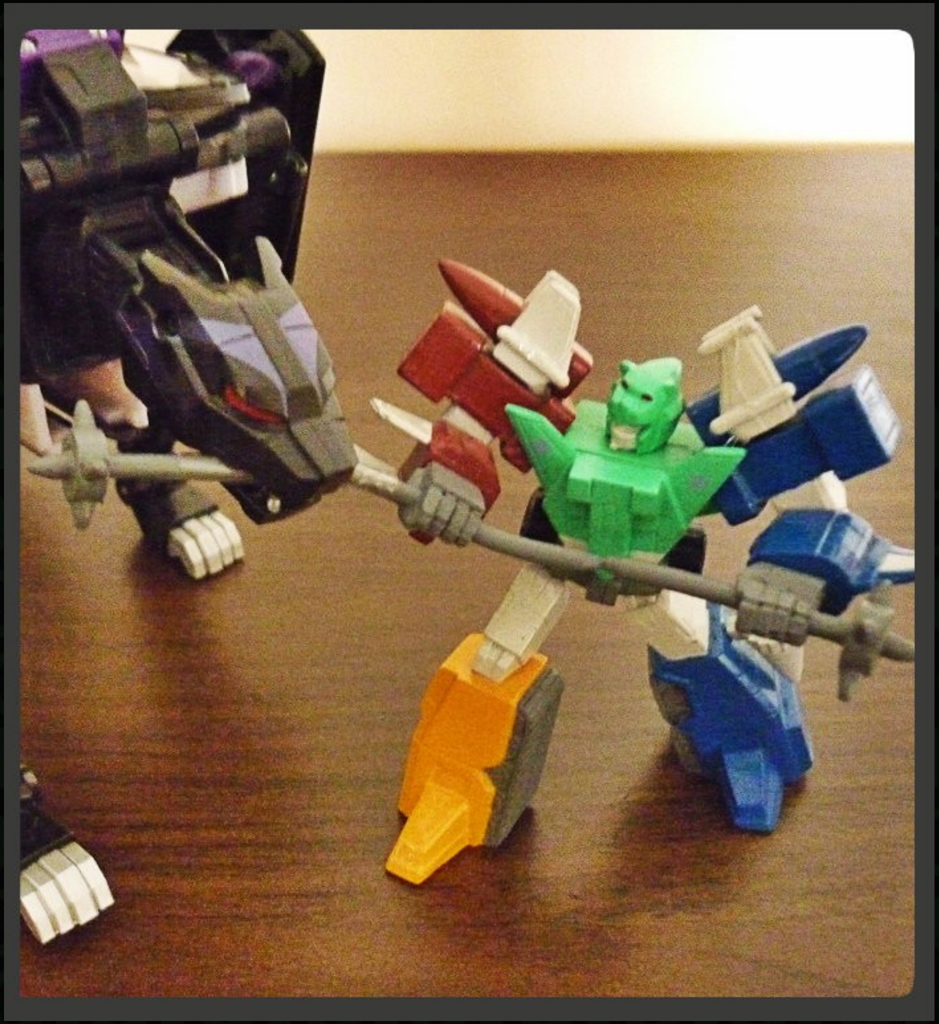
- CPU Load – `top` & `/proc/loadavg`
- Disk IO – `iotop`
- Memory – `free`
- Network – `netstat`

Profiling

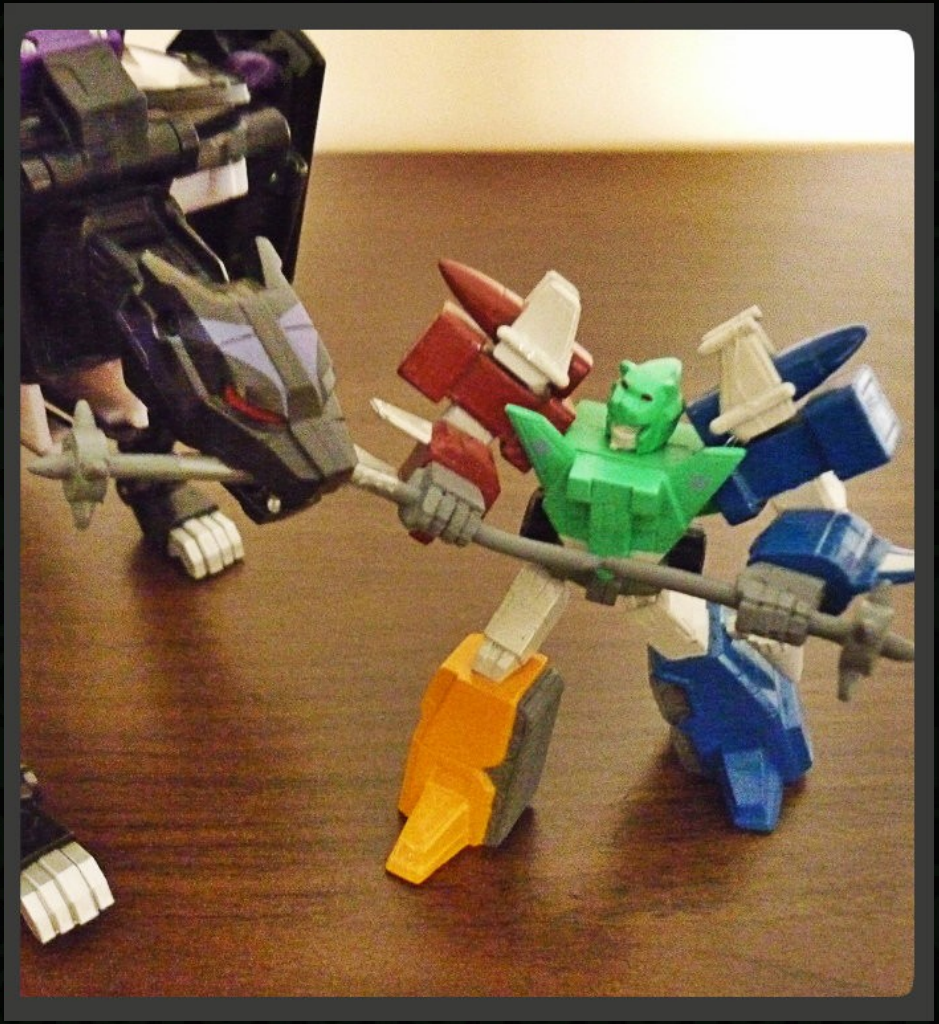
- CPU Load – `top` & `/proc/loadavg`
- Disk IO – `iotop`
- Memory – `free`
- Network – `netstat`

- Monitoring – `watch` & `time`

Profiling under load

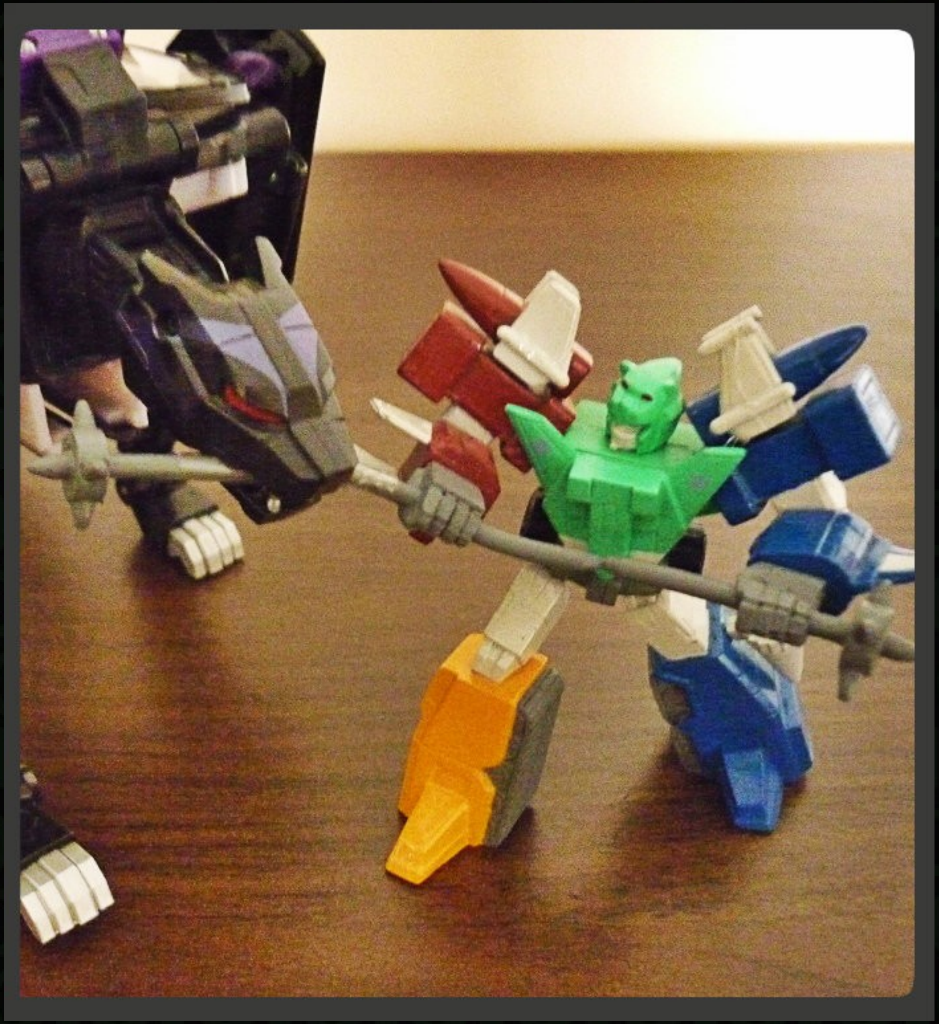


Profiling under load



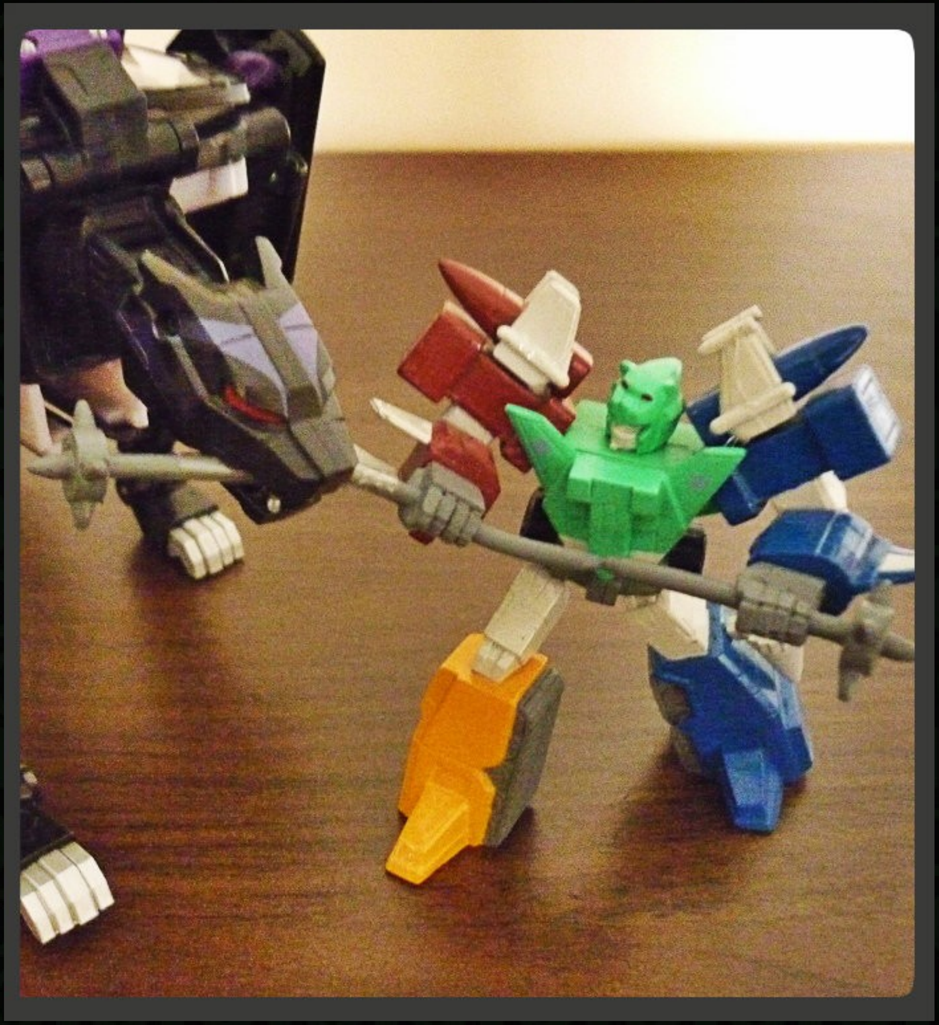
- Code bottlenecks != Platform bottlenecks

Profiling under load



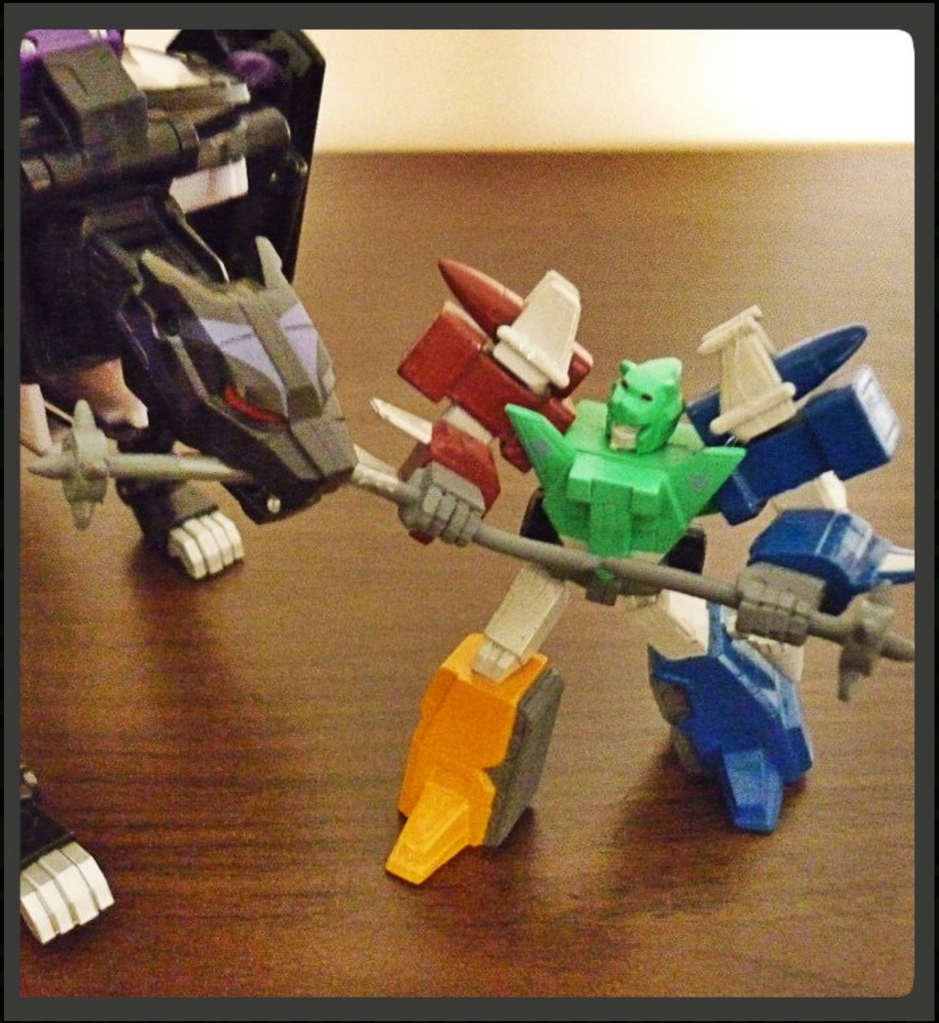
- Code bottlenecks != Platform bottlenecks
- Test in production, if possible

Profiling under load



- Code bottlenecks != Platform bottlenecks
- Test in production, if possible
- Use multiple VMs (Vagrant can help automate this)

Profiling under load



- Code bottlenecks != Platform bottlenecks
- Test in production, if possible
- Use multiple VMs (Vagrant can help automate this)
- Use siege & ab

Finding bottlenecks

Finding bottlenecks

Right	Wrong

Finding bottlenecks

Right **Wrong**

- Database

Finding bottlenecks

Right	Wrong
• Database • Disk IO	

Finding bottlenecks

Right

- Database
- Disk IO
- External services

Wrong

Finding bottlenecks

Right	Wrong
Database	
Disk IO	
External services	
Application work	

Finding bottlenecks

Right	Wrong
• Database	• Autoloaders
• Disk IO	
• External services	
• Application work	

Finding bottlenecks

Right

- Database
- Disk IO
- External services
- Application work

Wrong

- Autoloaders
- Config files

Finding bottlenecks

Right

- Database
- Disk IO
- External services
- Application work

Wrong

- Autoloaders
- Config files
- Logging

Finding bottlenecks

Right

- Database
- Disk IO
- External services
- Application work

Wrong

- Autoloaders
- Config files
- Logging
- Object instantiation

Finding bottlenecks

Right	Wrong
Database	Autoloaders
Disk IO	Config files
External services	Logging
Application work	Object instantiation
	Frameworks

Finding bottlenecks

Right	Wrong
• Database	• Autoloaders
• Disk IO	• Config files
• External services	• Logging
• Application work	• Object instantiation
	• Frameworks

Ask Jo (@juokas)
about Doctrine!

Tell users it's slow

Tell users it's slow

- User experience != Raw speed

Tell users it's slow

- User experience != Raw speed
- Keep the UI responsive

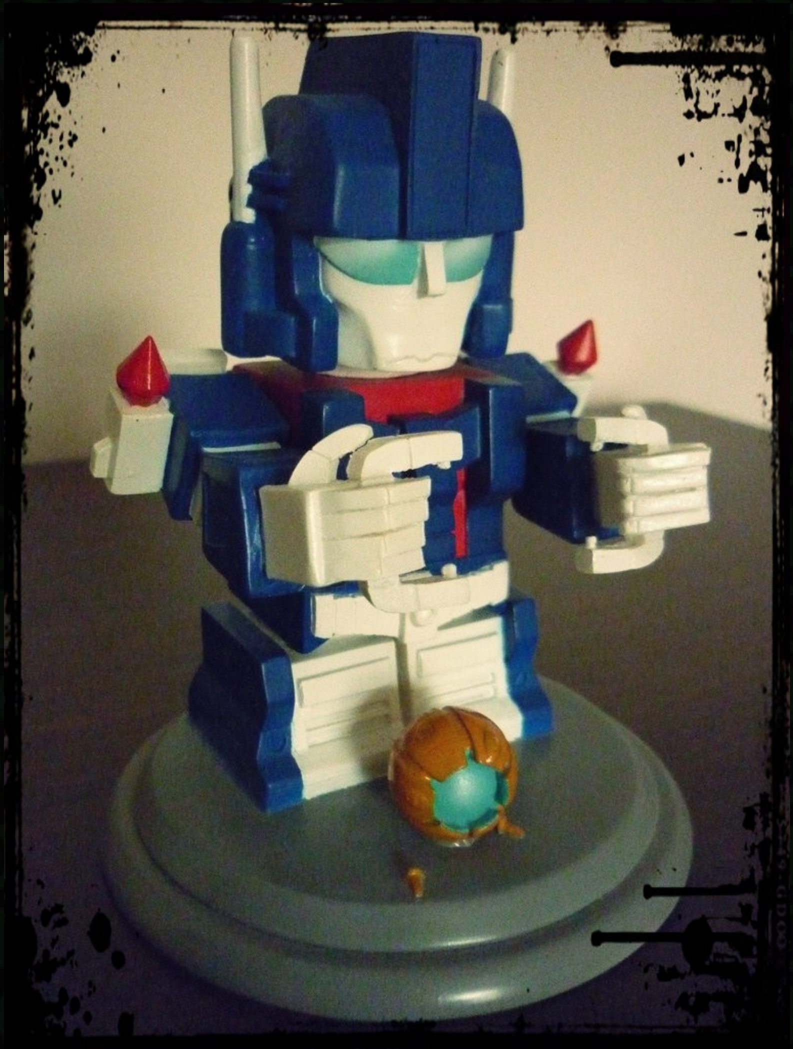
Tell users it's slow

- User experience != Raw speed
- Keep the UI responsive
- Set expectations

Tell users it's slow

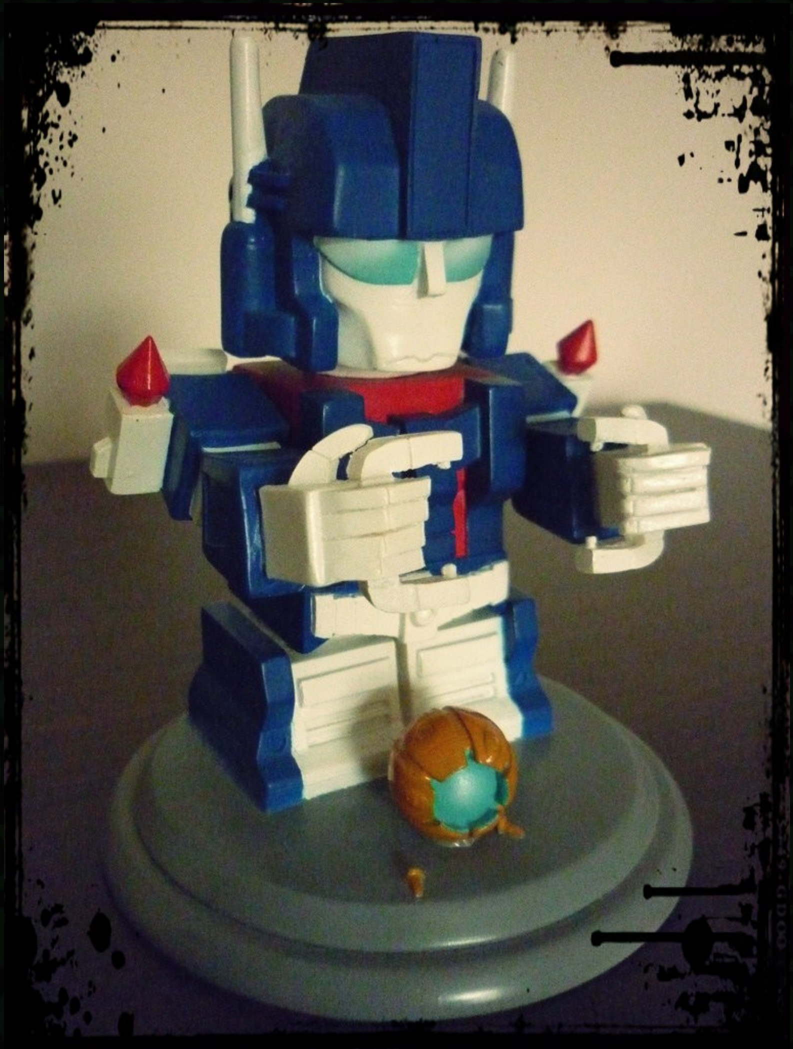
- User experience != Raw speed
- Keep the UI responsive
- Set expectations
- Adapt to changes (@blongden)

Fail gracefully



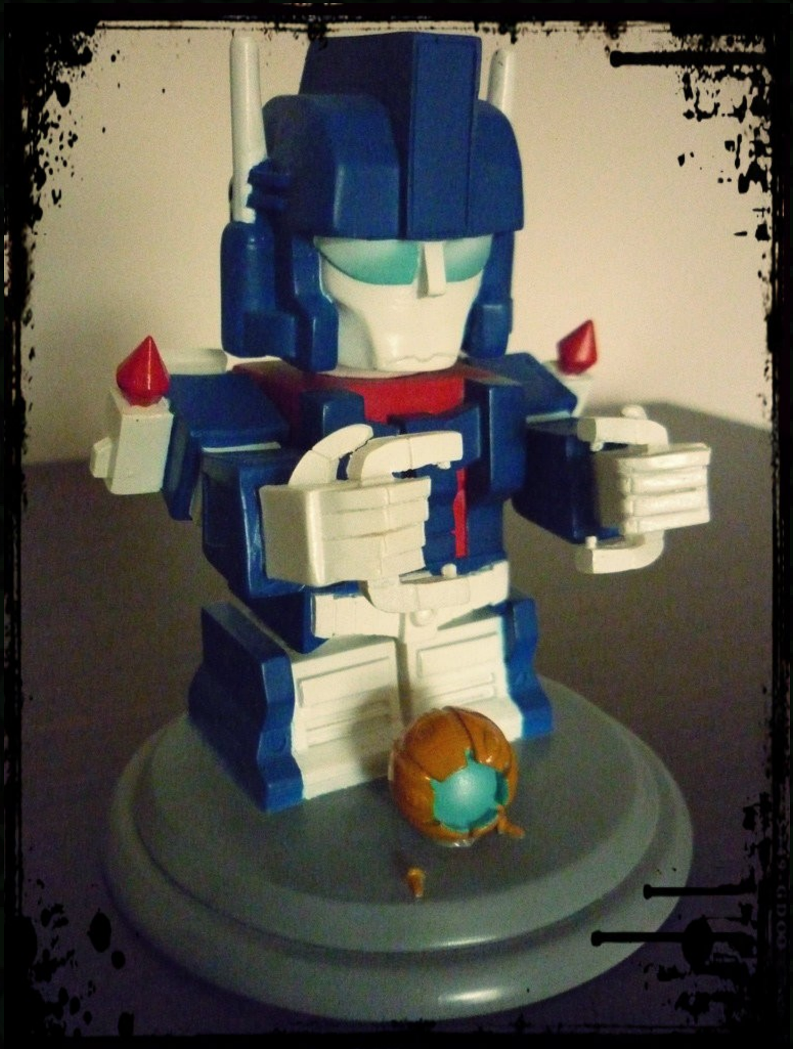
Fail gracefully

- Actively choose to time-out



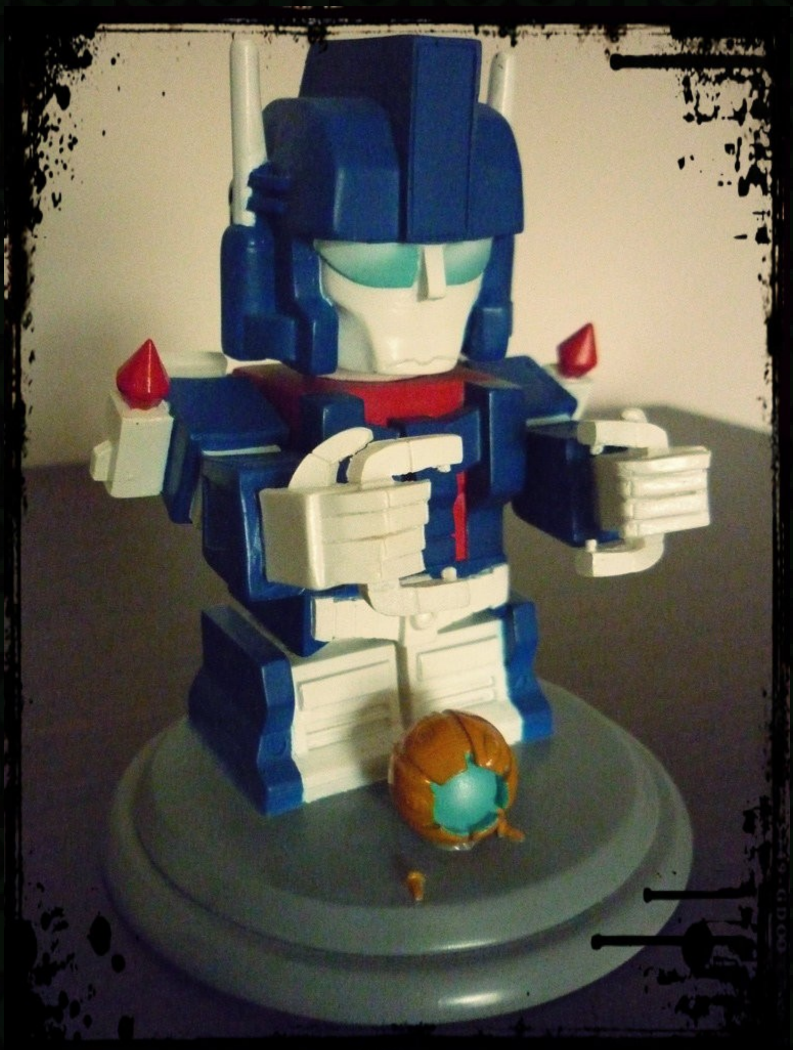
Fail gracefully

- Actively choose to time-out
- Prioritise core/profitable functionality



Fail gracefully

- Actively choose to time-out
- Prioritise core/profitable functionality
- Have a BIG **RED** BUTTON



Separate functionality

Separate functionality

Admin

Batch

Separate functionality

Admin **Batch**

- Different users == diff. requirements

Separate functionality

Admin

- Different users == diff. requirements

- No cache

Batch

Separate functionality

Admin

- Different users == diff. requirements

- No cache
- “Master” storage

Batch

Separate functionality

Admin

- Different users == diff. requirements
- No cache
- “Master” storage
- Security

Batch

Separate functionality

Admin

- Different users == diff. requirements
- No cache
- “Master” storage
- Security

Batch

- Not time critical

Separate functionality

Admin

- Different users == diff. requirements
- No cache
- “Master” storage
- Security

Batch

- Not time critical
- CPU/RAM hungry

Separate functionality

Admin

- Different users == diff. requirements
- No cache
- “Master” storage
- Security

Batch

- Not time critical
- CPU/RAM hungry
- `nice` / `ionice`

Separate functionality

Admin

- Different users == diff. requirements
- No cache
- “Master” storage
- Security

Batch

- Not time critical
- CPU/RAM hungry
- `nice` / `ionice`
- Network rate limiting

Separate functionality

Admin

- Different users == diff. requirements
- No cache
- “Master” storage
- Security

Batch

- Not time critical
- CPU/RAM hungry
- `nice` / `ionice`
- Network rate limiting
- Read only?

Configure hardware correctly

Configure hardware correctly

- What is required for 1 request?

Configure hardware correctly

- What is required for 1 request?
- How many concurrent requests?

Configure hardware correctly

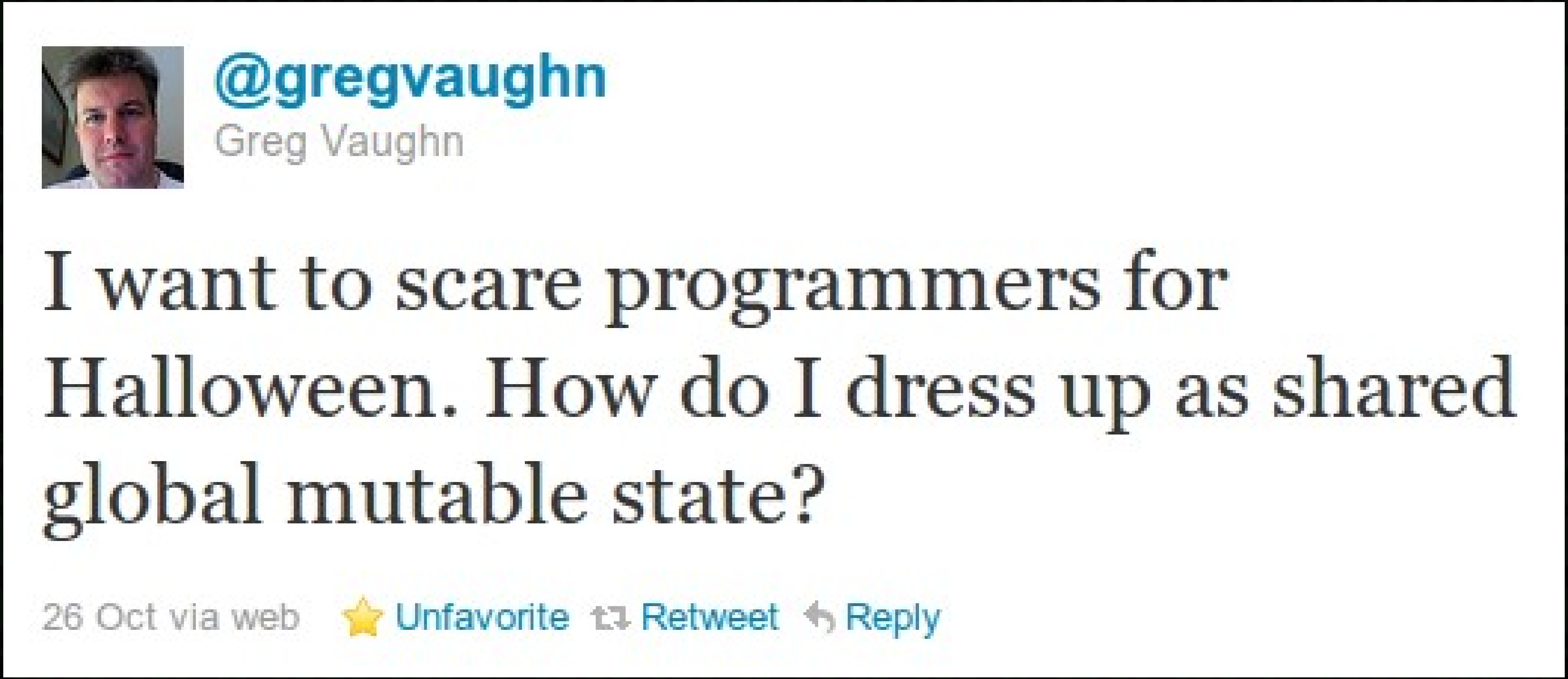
- What is required for 1 request?
- How many concurrent requests?
- Is it a linear scale?

Configure hardware correctly

- What is required for 1 request?
- How many concurrent requests?
- Is it a linear scale?
- Ask your hosting company

Aim for services

Aim for services


@gregvaughn
Greg Vaughn

I want to scare programmers for Halloween. How do I dress up as shared global mutable state?

26 Oct via web

Obey the “Law of Demeter”

Obey the “Law of Demeter”

- Talk to your friends, don't talk to strangers

Obey the “Law of Demeter”

- Talk to your friends, don't talk to strangers
- Promotes loose coupling

Obey the “Law of Demeter”

Good

Bad

Obey the “Law of Demeter”

Good

- `$person->requestPayment($amount);`

Bad

Obey the “Law of Demeter”

Good

- `$person->requestPayment($amount);`

Bad

- $wallet = $person->getWallet();
- $wallet->getMoney($amount);

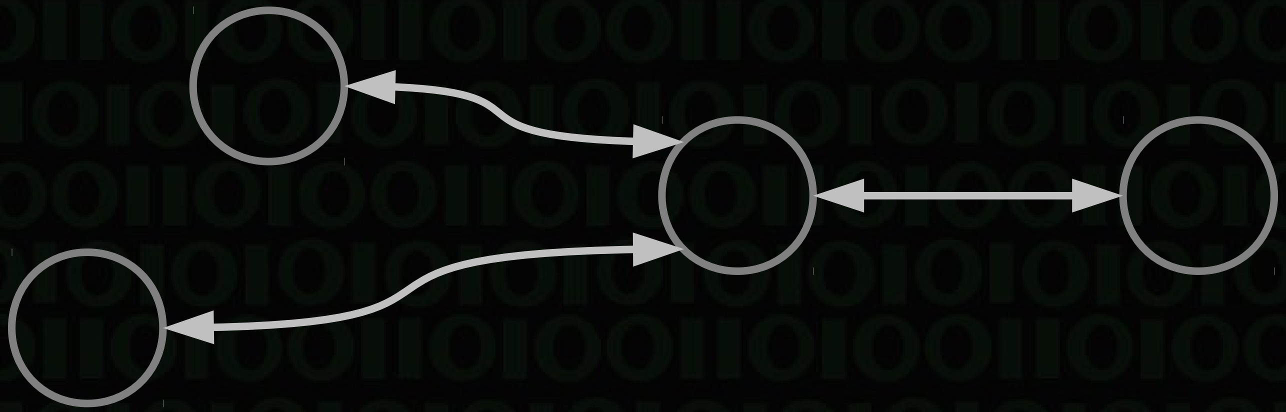
Obey the “Law of Demeter”

Good

Bad

Obey the "Law of Demeter"

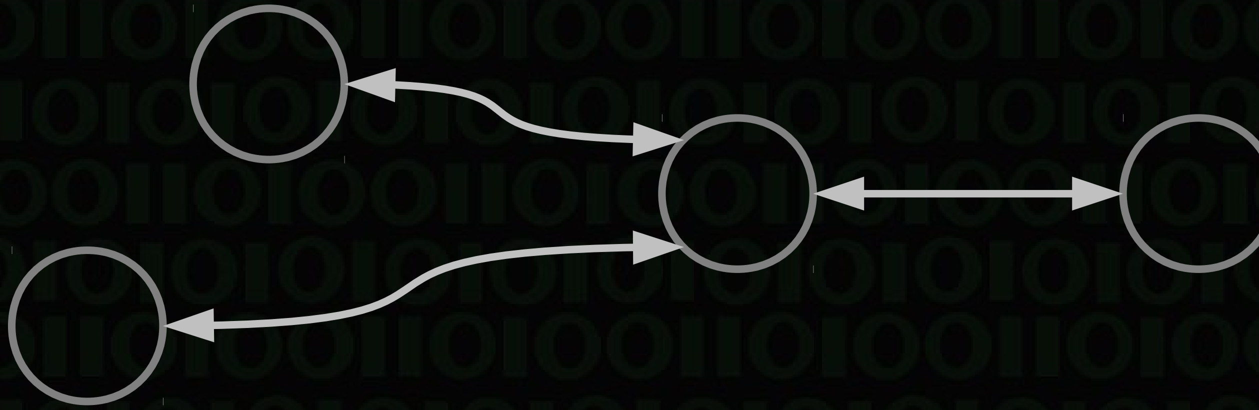
Good



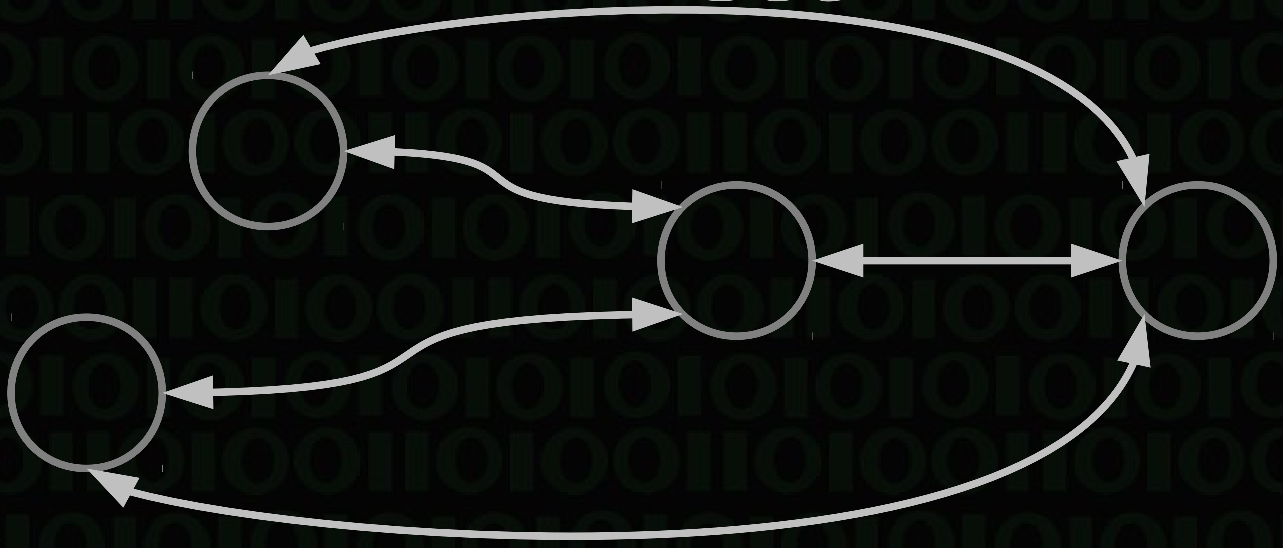
Bad

Obey the “Law of Demeter”

Good



Bad



Create immutable objects

Create immutable objects

- Objects that cannot change after creation

Create immutable objects

- Objects that cannot change after creation
- Copy On Write (COW - 🐮 ← unicode cow)

Create immutable objects

- Objects that cannot change after creation
- Copy On Write (COW - 🐮 ← unicode cow)
- More memory-hungry, but easy to roll back

Create immutable objects

- Objects that cannot change after creation
- Copy On Write (COW - 🐮 ← unicode cow)
- More memory-hungry, but easy to roll back
- Value objects should be immutable

Create immutable objects

- Objects that cannot change after creation
- Copy On Write (COW - 🐮 ← unicode cow)
- More memory-hungry, but easy to roll back
- Value objects should be immutable

- Think of them like a response from a service

Identify “single server” factors

Identify “single server” factors

- File uploads / user content

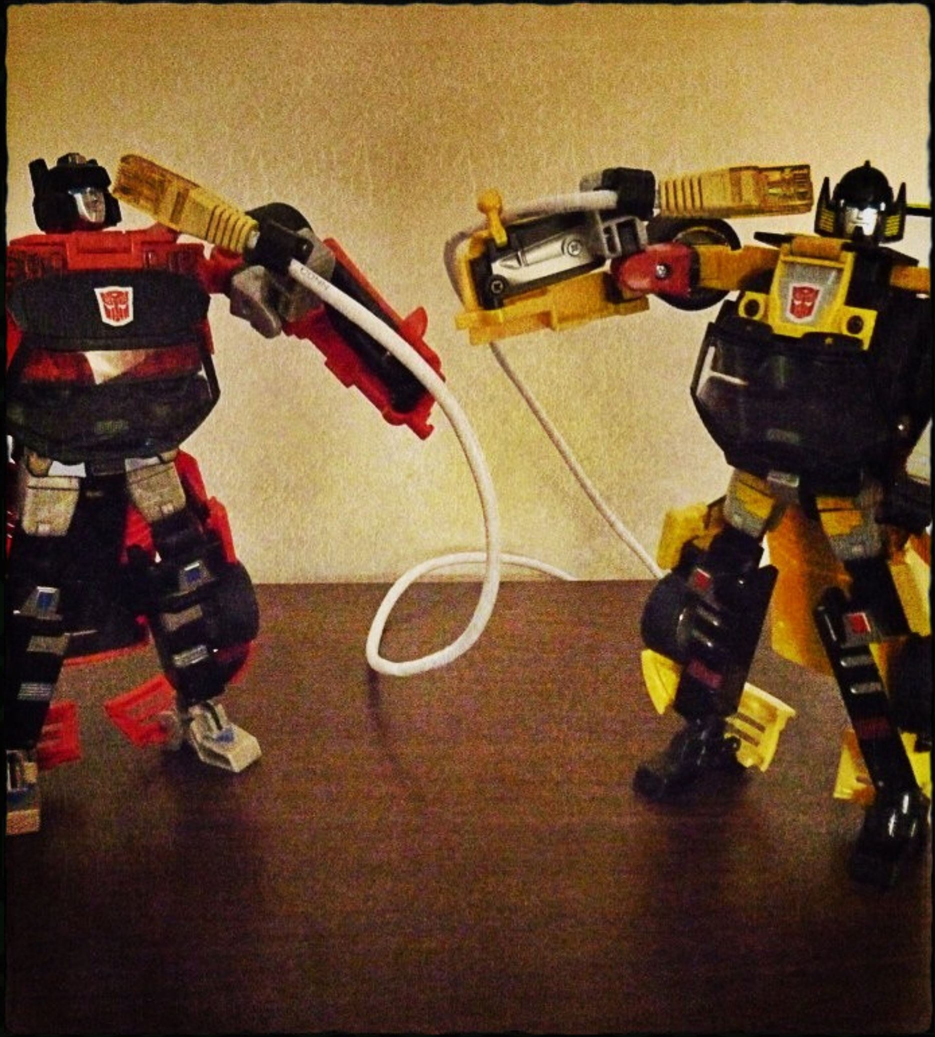
Identify “single server” factors

- File uploads / user content
- IP restrictions / server access

Identify “single server” factors

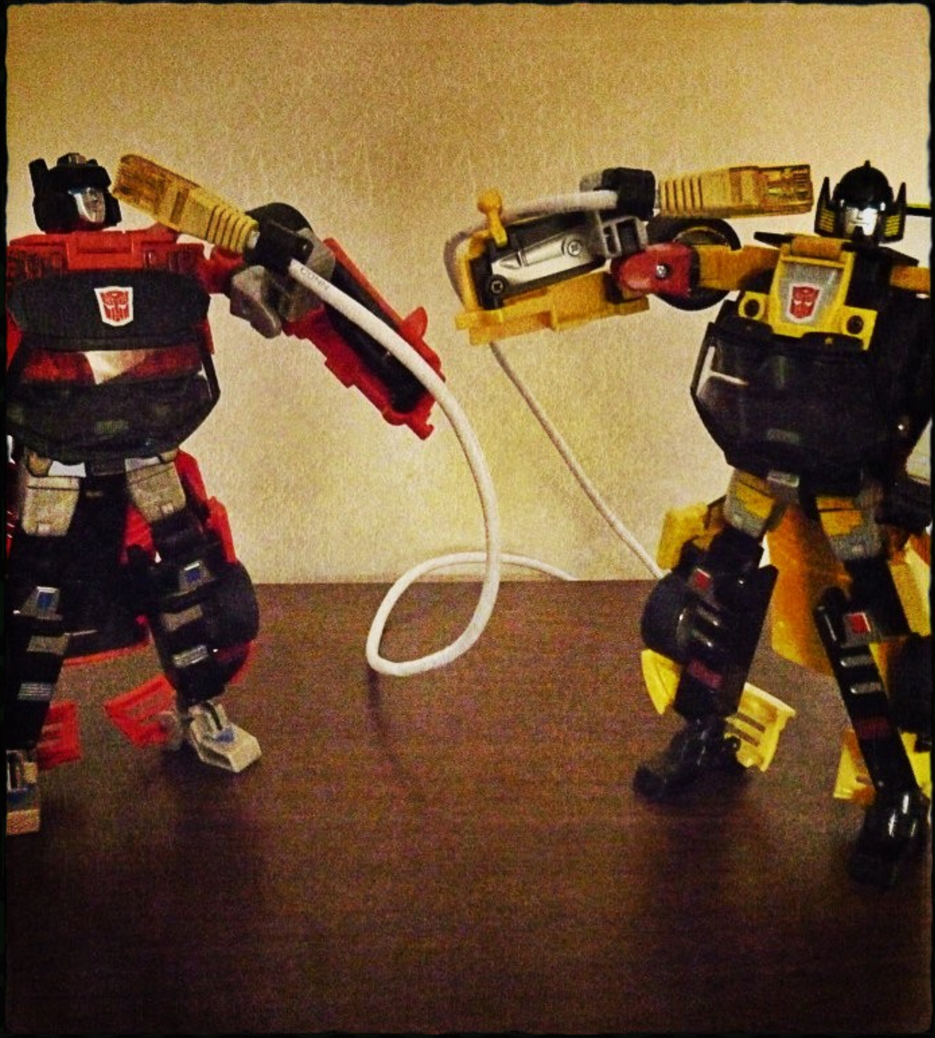
- File uploads / user content
- IP restrictions / server access
- Licensing?

Create services



Create services

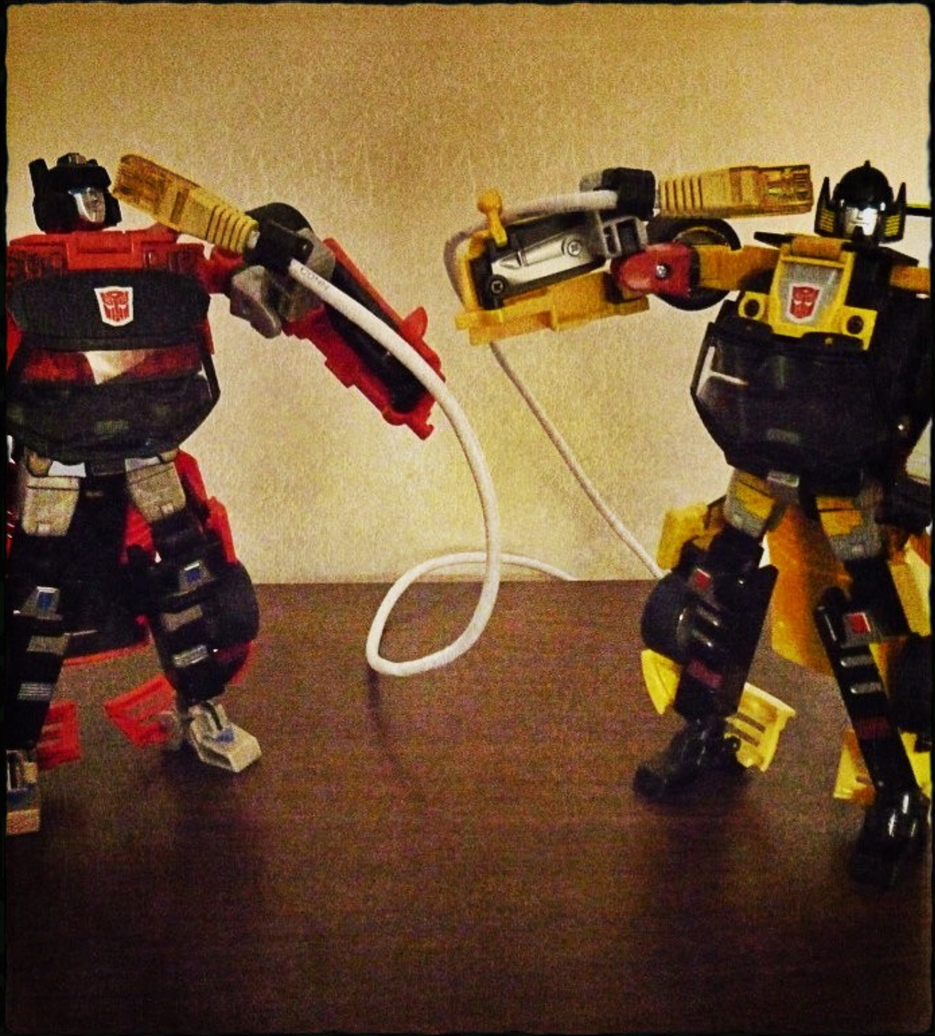
Your API should be:



Create services

Your API should be:

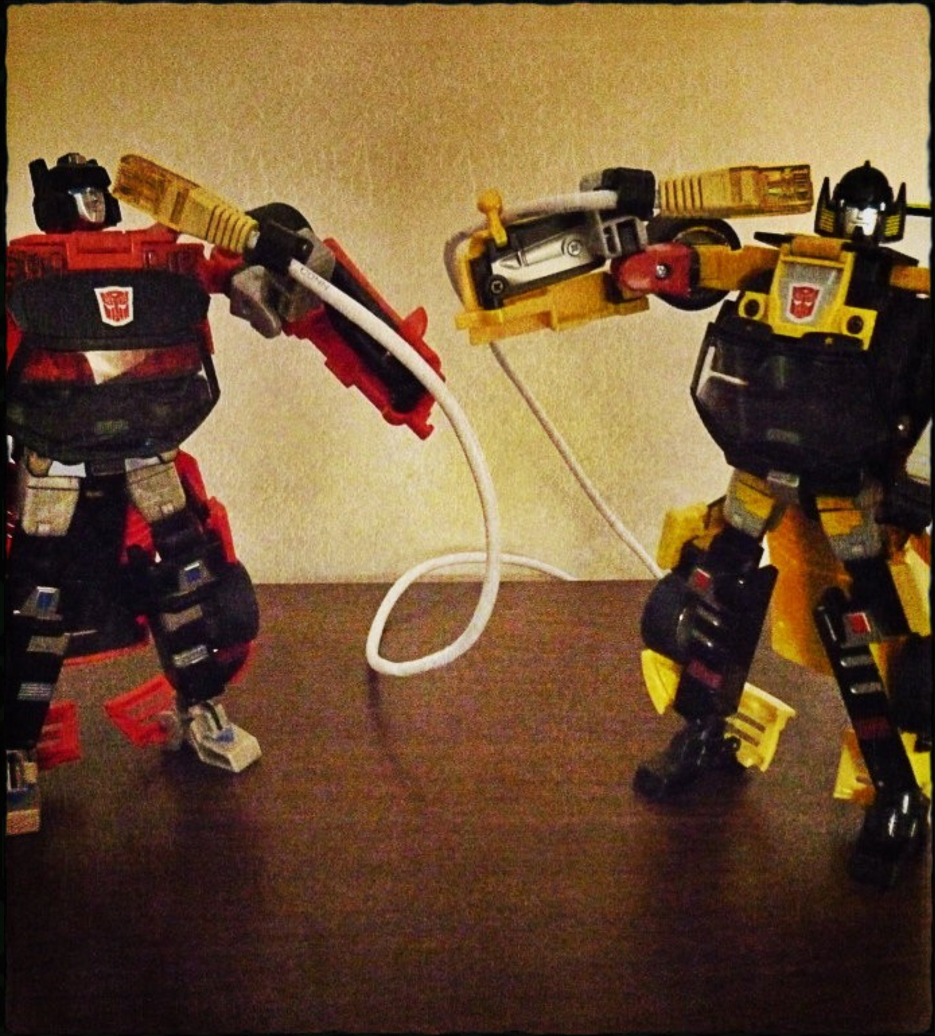
- Independent of application state



Create services

Your API should be:

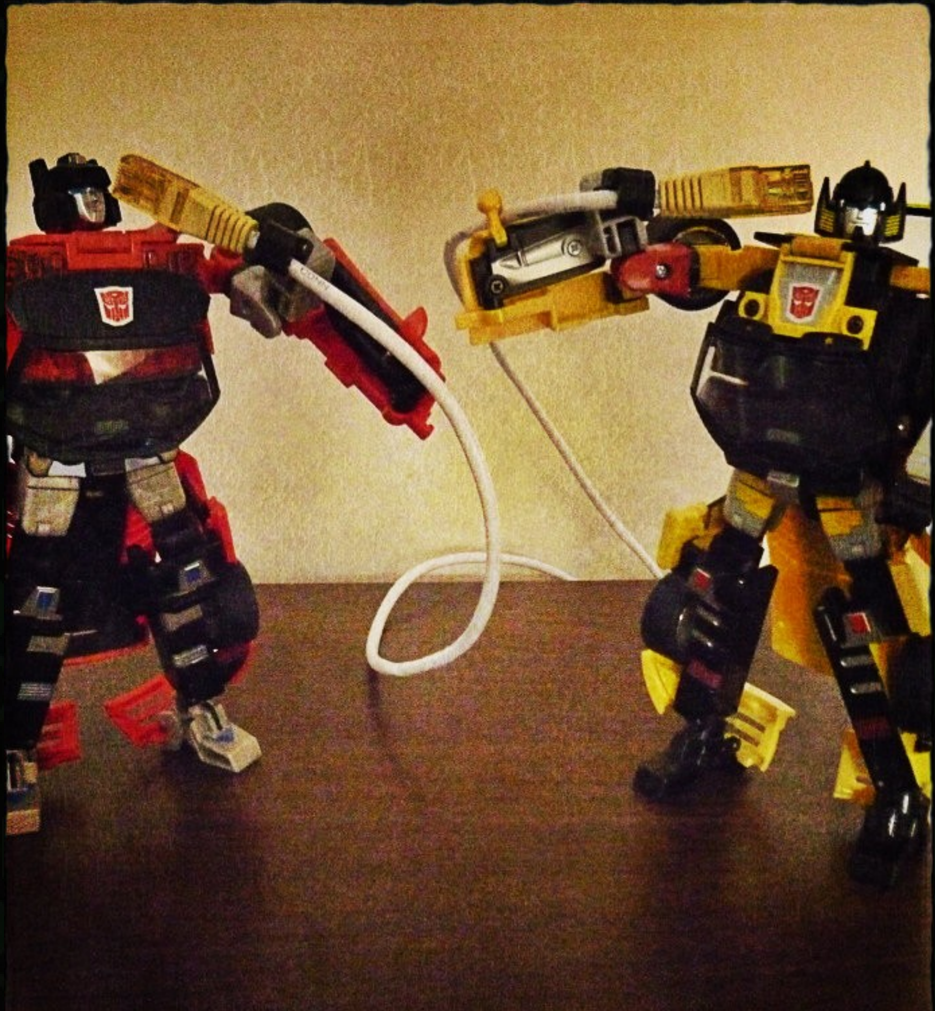
- Independent of application state
- Loosely coupled



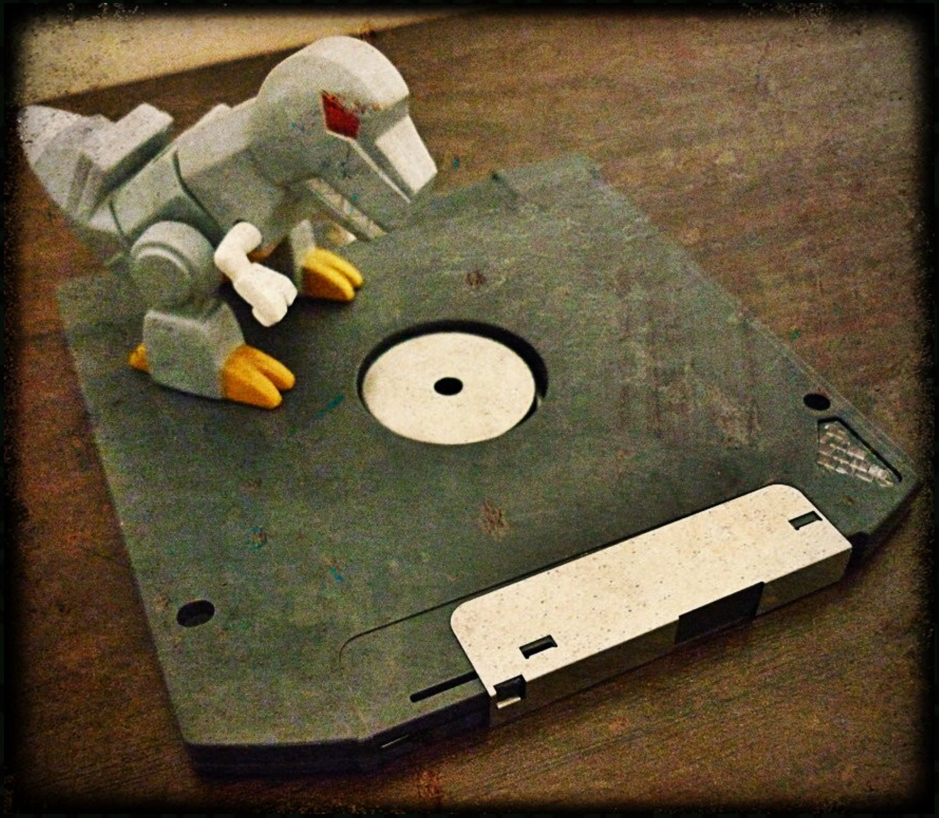
Create services

Your API should be:

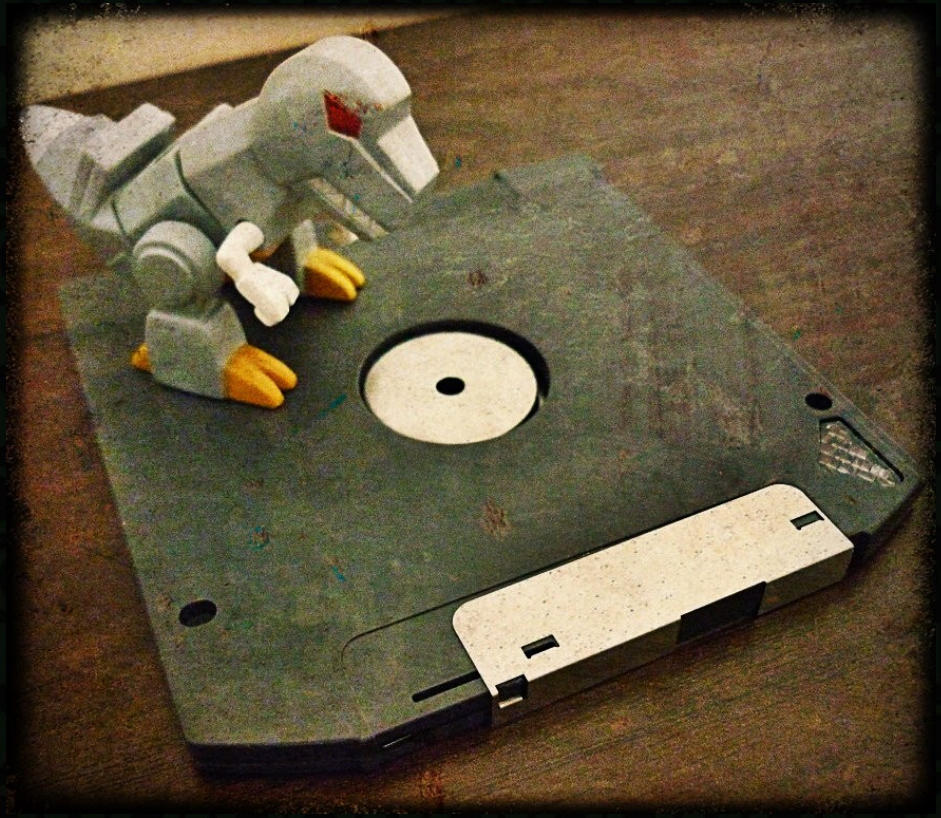
- Independent of application state
- Loosely coupled
- Accepting/returning immutable objects



Use caches

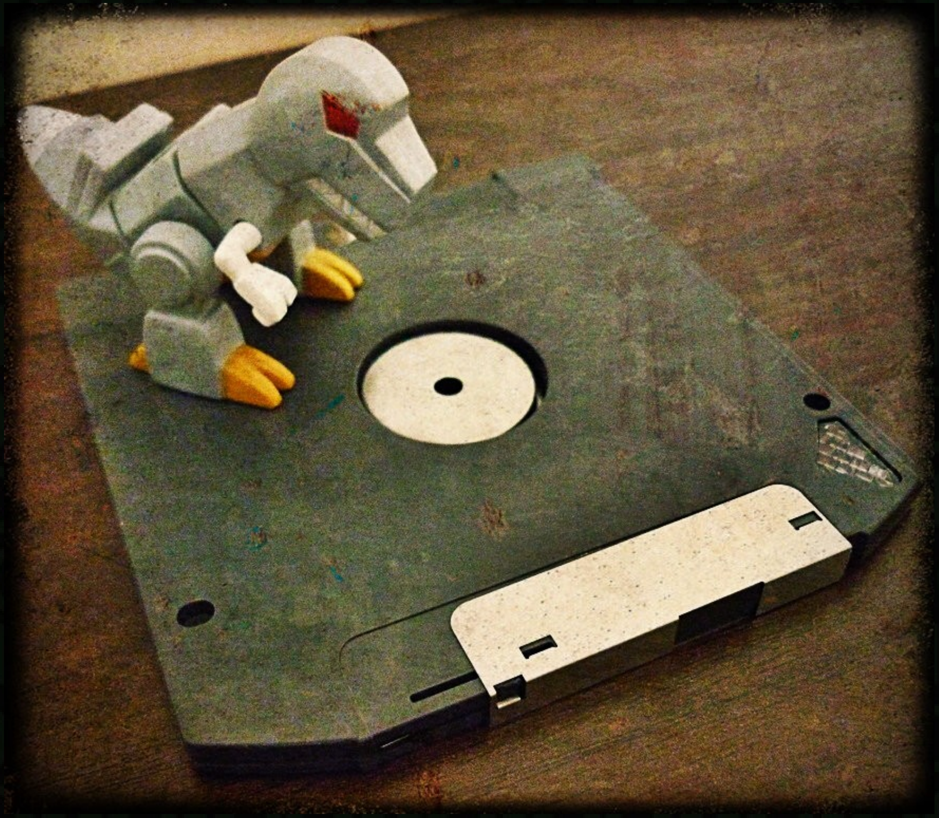


Use caches



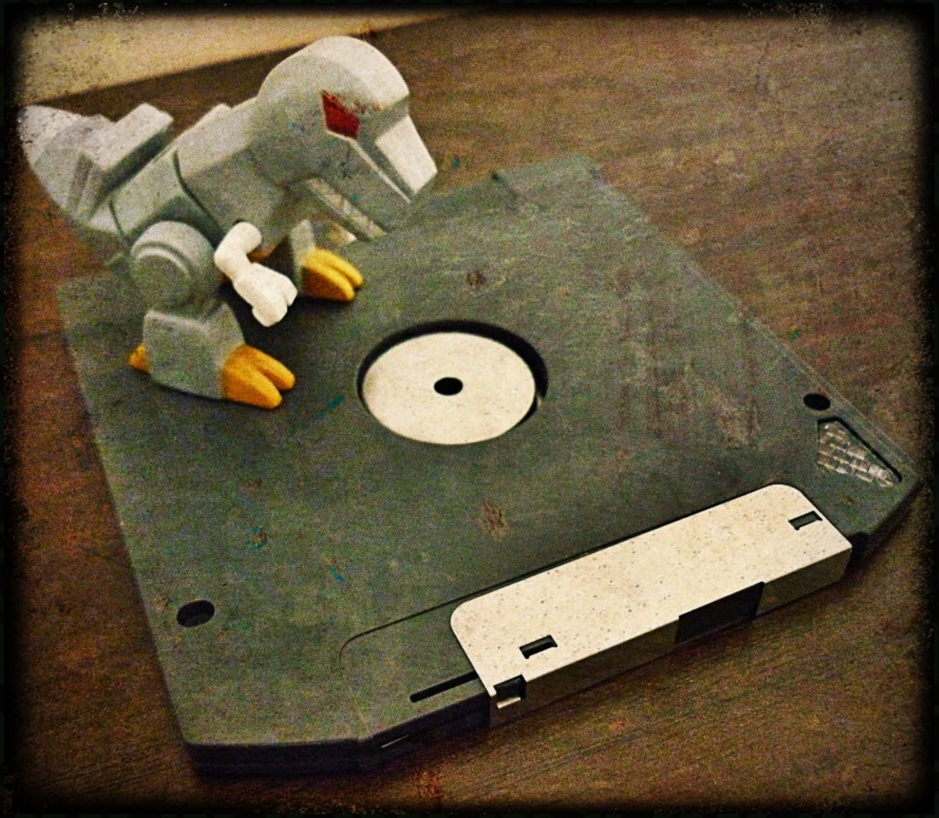
- REST-ful services over HTTP let you cache easily

Use caches



- REST-ful services over HTTP let you cache easily
- Internal caching is trickier, but “out-of-the-box” with most frameworks

Use caches



- REST-ful services over HTTP let you cache easily
- Internal caching is trickier, but “out-of-the-box” with most frameworks

Proxy other people's services through your own cache!

Use queues

Use queues

- Full blown job server: Gearman

Use queues

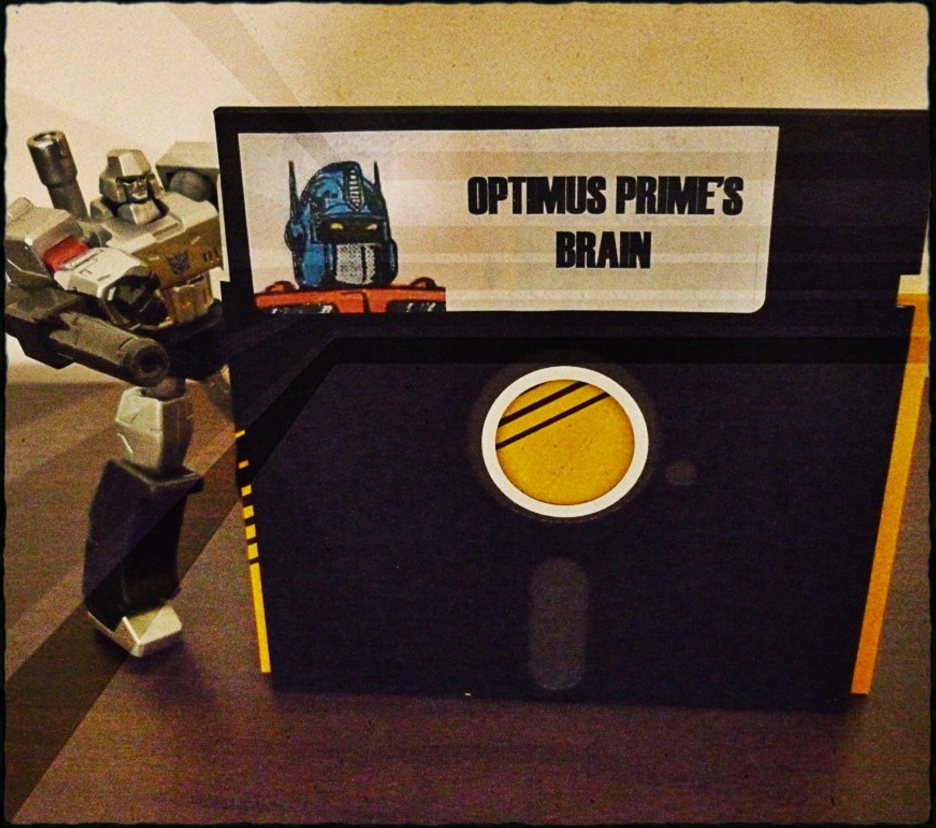
- Full blown job server: Gearman
- AMQP implementation: RabbitMQ

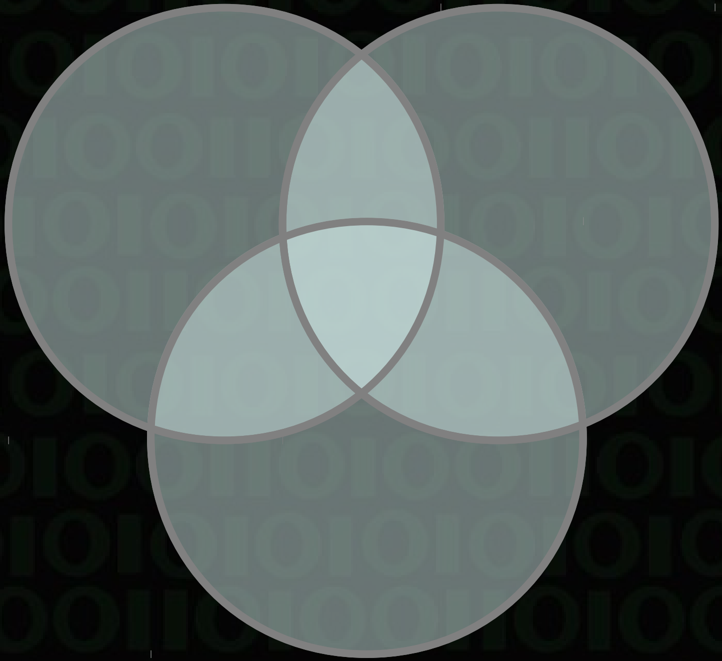
Use queues

- Full blown job server: Gearman
- AMQP implementation: RabbitMQ
- Light-weight sockets: 0MQ

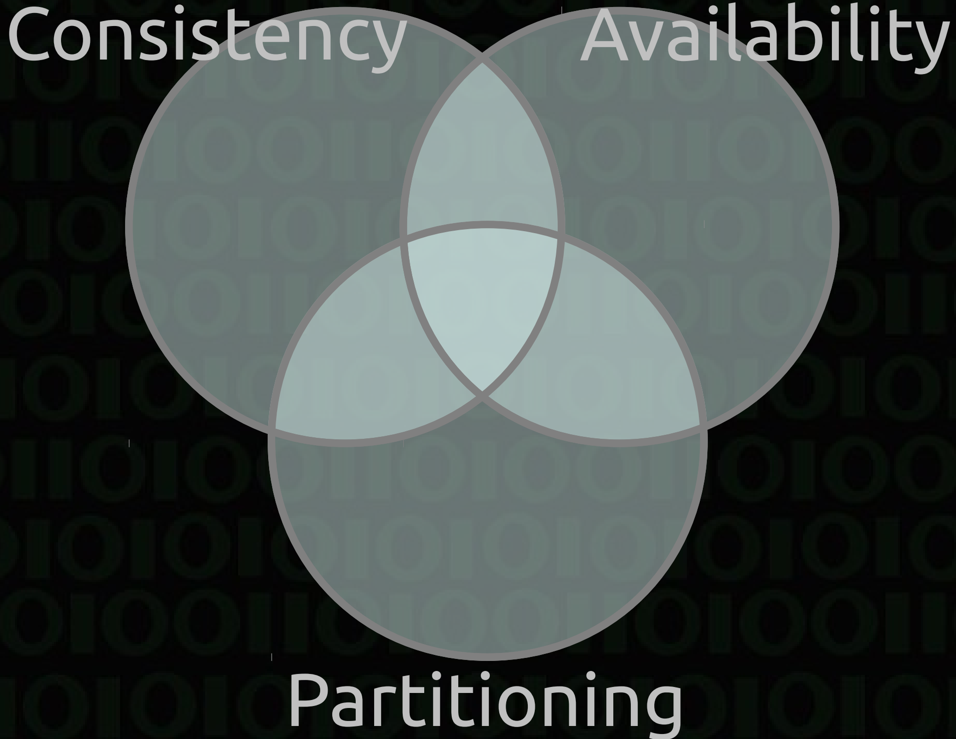
Use the right storage

Use the right storage

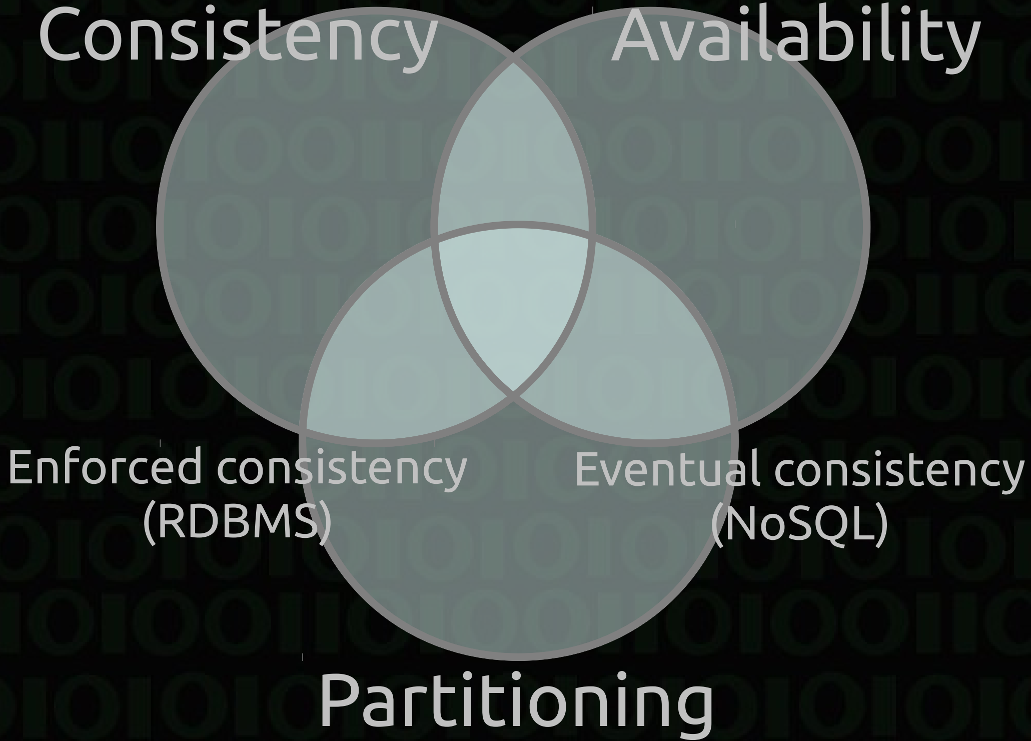




Use the right storage



Use the right storage



Scale your storage

Scale your storage

- Master/slave replication

Scale your storage

- Master/slave replication
- Table partitioning

Scale your storage

- Master/slave replication
- Table partitioning
- Row partitioning

Scale your storage

- Master/slave replication
- Table partitioning
- Row partitioning
- rsync

Scale your storage

- Master/slave replication
- Table partitioning
- Row partitioning

- rsync
- NFS

Scale your storage

- Master/slave replication
- Table partitioning
- Row partitioning

- rsync
- NFS
- GlusterFS

Now you can use the cloud

Now you can use the cloud

- Full service – orchestra.io

Now you can use the cloud

- Full service orchestra.io
- Infrastructure mgmt. Scalr

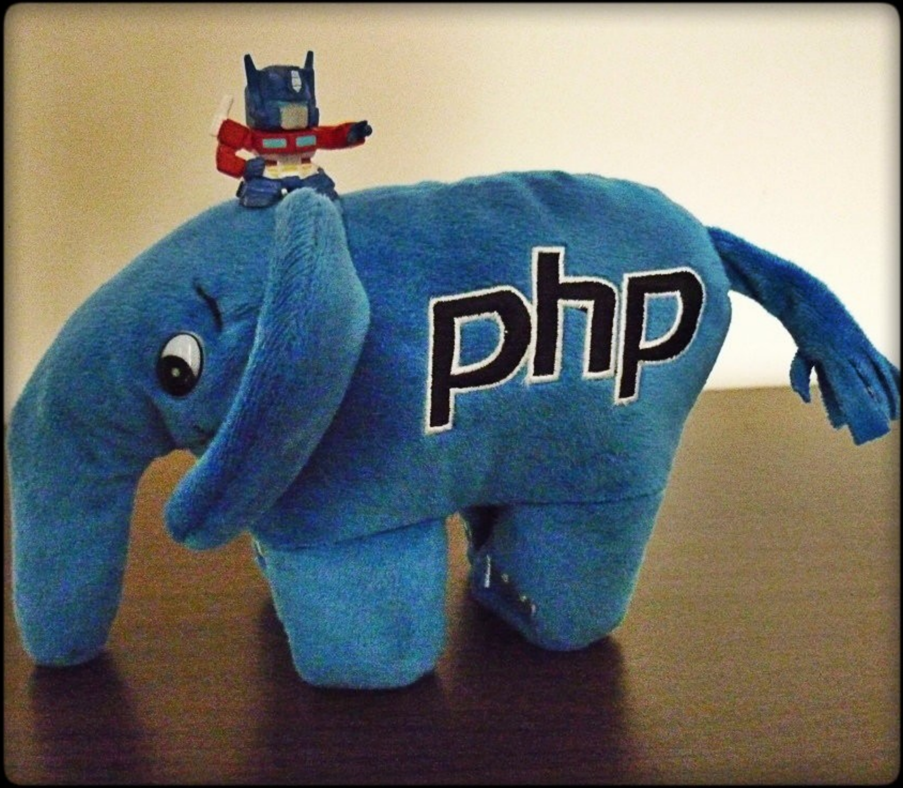
Now you can use the cloud

- Full service
  orchestra.io
- Infrastructure mgmt.
  Scalr
- Manually
  $your_code_here

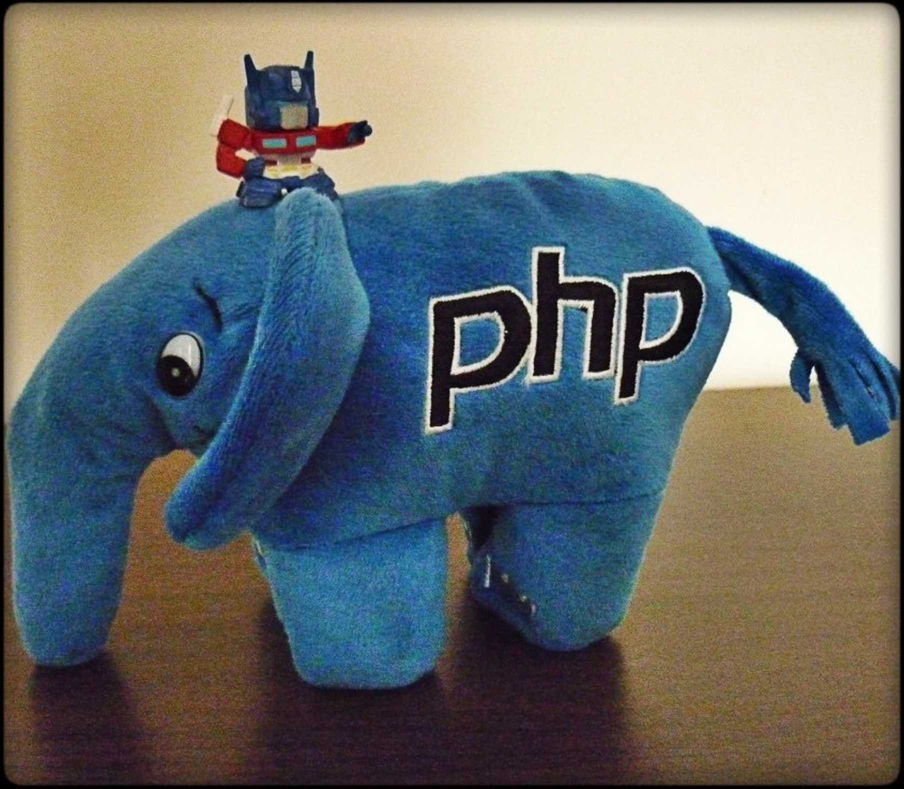
Now you can use the cloud

- Full service
  orchestra.io
- Infrastructure mgmt.
  Scalr
- Manually
  $your_code_here
- Automate!
  Chef & Puppet

Review

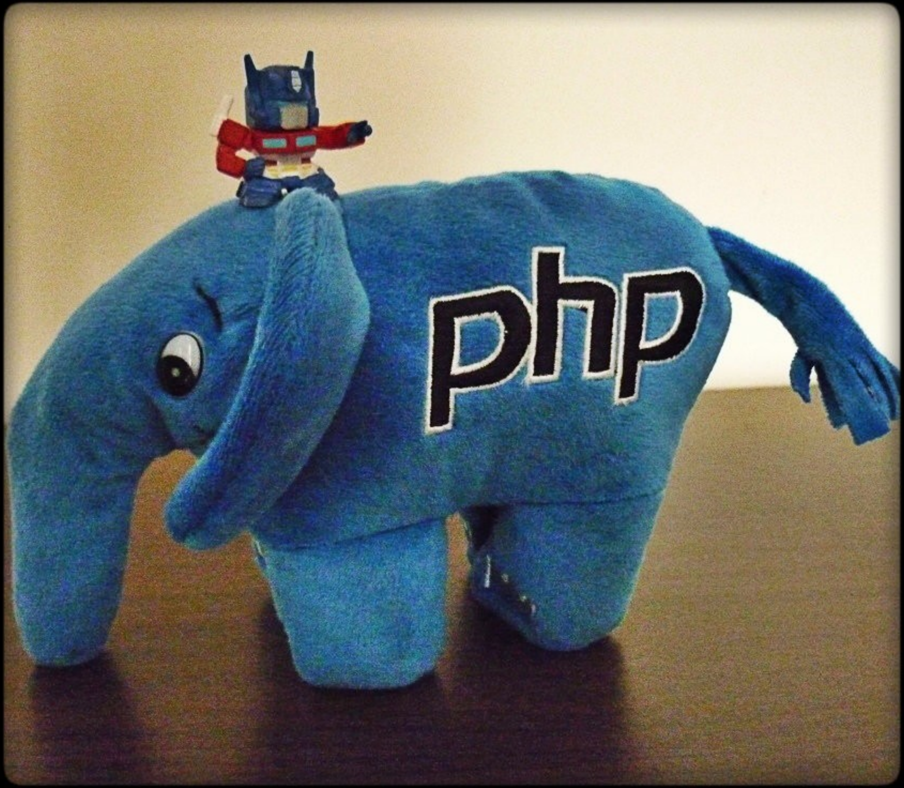


Review



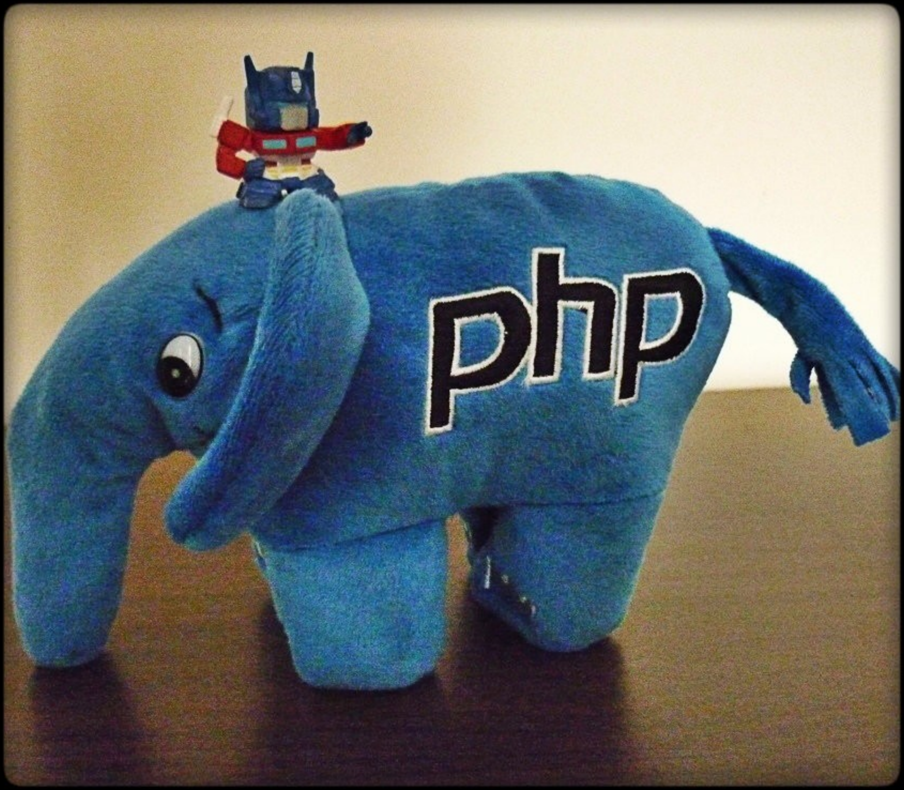
- Start with only what you need

Review



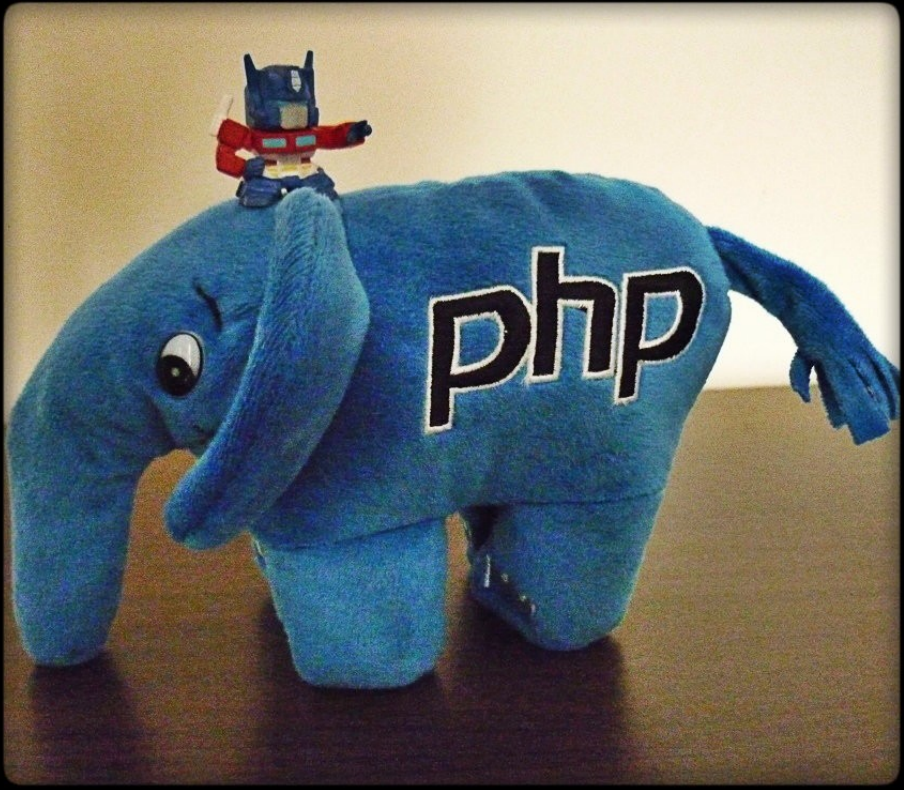
- Start with only what you need
- Identify the problems

Review



- Start with only what you need
- Identify the problems
- Pull the problem out to a service

Review



- Start with only what you need
- Identify the problems
- Pull the problem out to a service
- Distribute

Thank you!

- Feedback:

  http://joind.in/6324

- Photos & Transformers:
  Nina Merewood
- Legal?
  Takara & Hasbro
- ElePHPant smuggling:
  Johannes Schlüter (@phperror)
- Vintage photo nonsense:
  http://pixlr.com/o-matic/