Rails Application Deployment

July 2007 @ Philly on Rails

CHARIOT SOLUTIONS

What Shall We Deploy Tonight?

mephisto

– Blogging/publishing system
– Standard Rails application
– Ships with gems in vendor directory
– Easy rake task for database setup
– http://mephistoblog.com

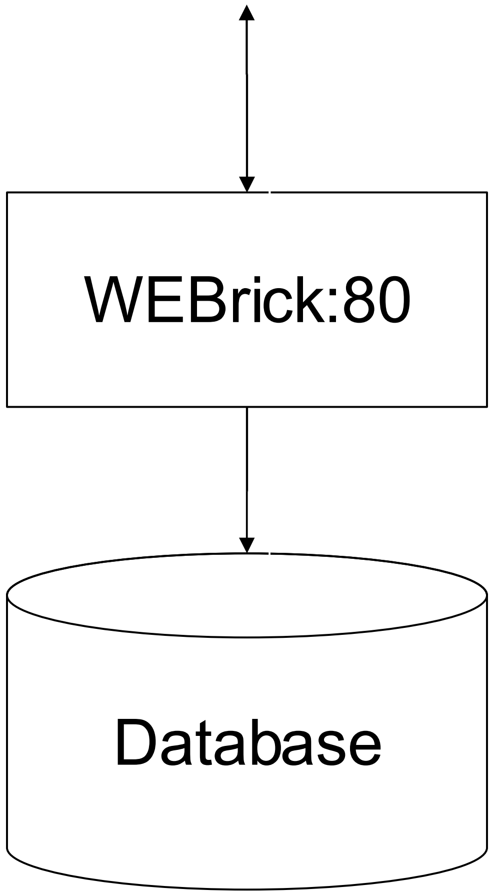
Installing and Configuring Mephisto

```
> gem install rails
> mysql # create mephisto database/user
> tar xzf mephisto-0.7.3.tar.gz
> cd mephisto-0.7.3
> vi config/database.yml # configure db
> rake db:bootstrap RAILS_ENV=production
```

CHARIOT SOLUTIONS

Basic, Single-Server Deployment

<1> WEBrick Standalone



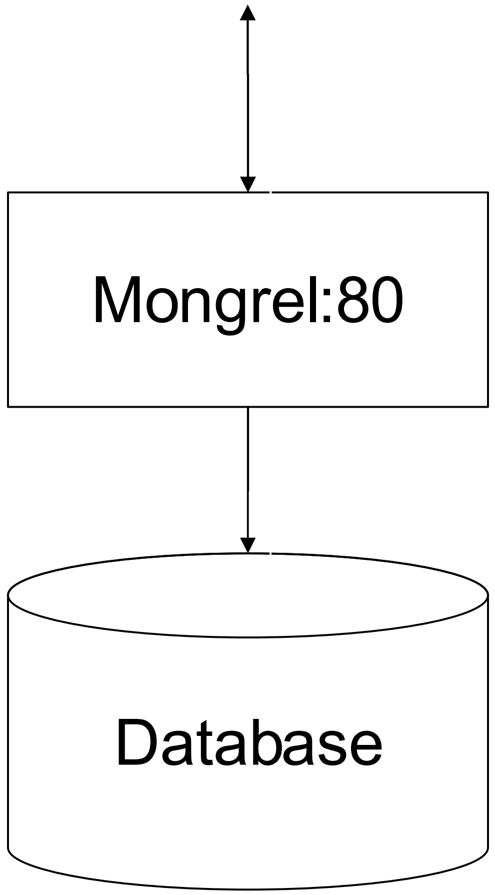
<1> WEBrick Standalone

```
> cd mephisto-0.7.3
> script/server -e production -p 80

# Browse to http://localhost/admin
```

<1> WEBrick Standalone

- Default, built-in HTTP server
- Runs fine in development
- Fine for demos
- Not suitable for production
- Very easy to replace with mongrel

<2> Mongrel Standalone



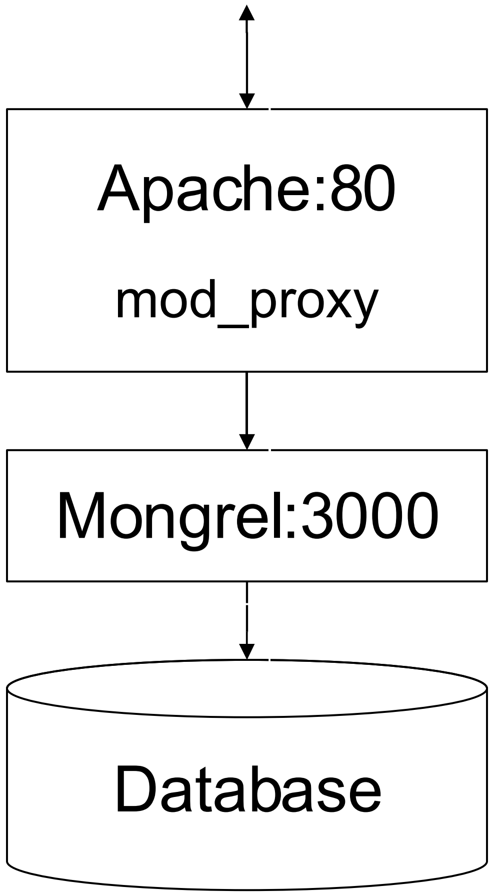
<2> Mongrel Standalone

```
> gem install mongrel
> script/server -e production -p 80


# Browse to http://localhost/admin
> Ctrl-C
> mongrel_rails start -d -e production \
                           -p 80
> mongrel_rails stop
```

<2> Mongrel Standalone

- Very easy to install and run
- Standalone is fine for demos
- But Mongrel isn't good at all things
  - handling lots of simultaneous connections
  - serving static files
  - monitoring/manageability
- Put something in front of it to do those

<3> Mongrel Behind Apache



<3> Mongrel Behind Apache

```
# Editing /etc/sysconfig/apache2 on SuSE


...
APACHE_MODULES="actions alias auth_basic
authn_file authz_host authz_groupfile
authz_default authz_user authn_dbm
autoindex cgi dir env expires include
log_config mime negotiation setenvif ssl
suexec userdir php5 proxy proxy_http"
...
```

<3> Mongrel Behind Apache

```
# New conf file in conf.d/ or vhosts.d/
<VirtualHost *:80>
  ServerName whatever.com
  ProxyPass / http://localhost:3000/
  ProxyPassReverse / http://localhost:3000
  ProxyPreserveHost on
  ErrorLog /tmp/mephisto_proxy_errors_log
  CustomLog /tmp/mephisto_proxy_access_log
</VirtualHost>
```

<3> Mongrel Behind Apache

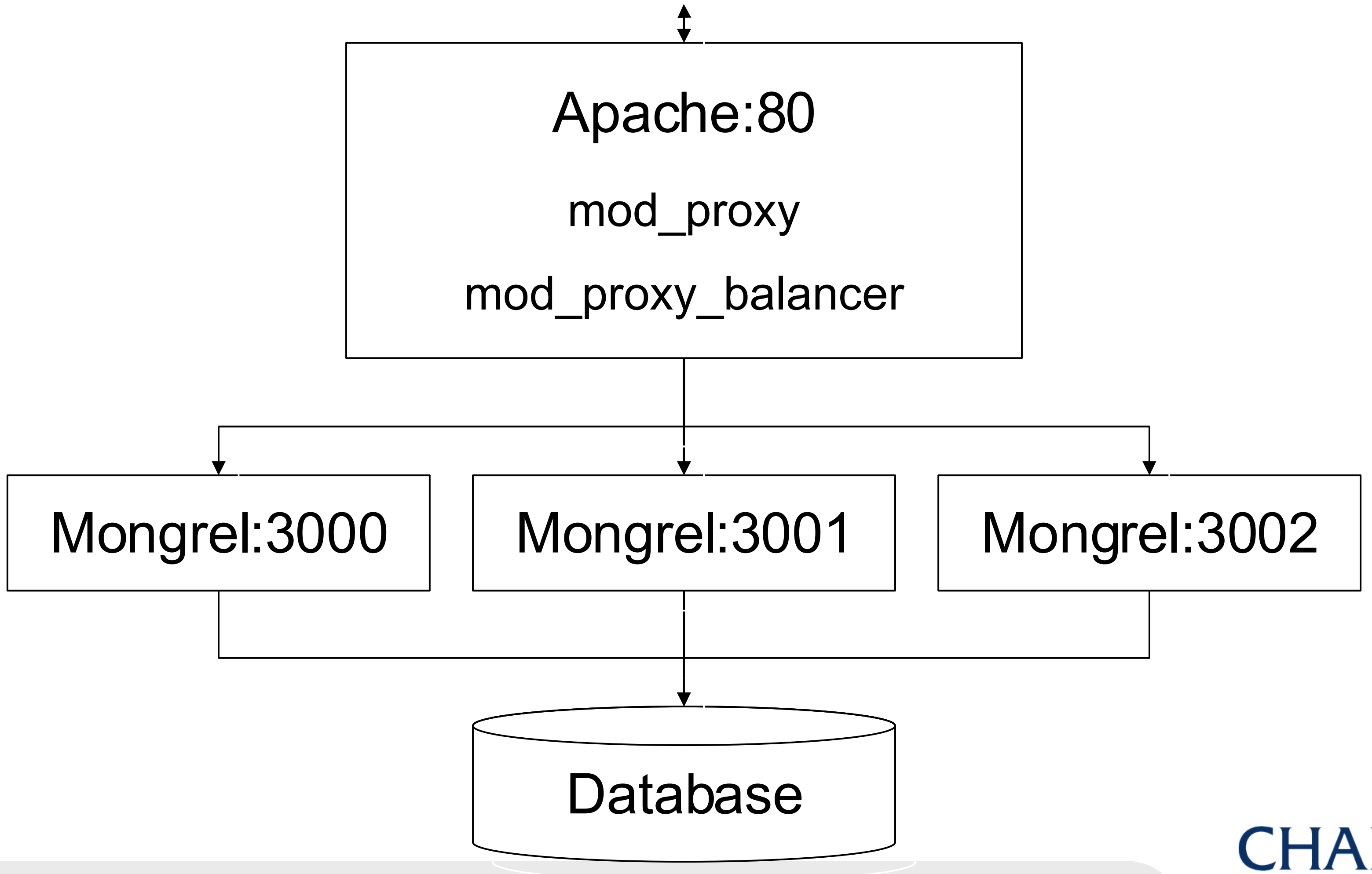
```
> mongrel_rails start -d -e production \
                         -p 3000
> rcapache2 start

# Browse to http://localhost/admin
```

CHARIOT SOLUTIONS

<3> Mongrel Behind Apache

- Great first step
- Works fine for small sites
- Upgrades/crashes => downtime
- You'll want more mongrel instances for redundancy and performance

<4> Many Mongrels Behind Apache



<4> Many Mongrels Behind Apache

```
# Editing /etc/sysconfig/apache2 on SuSE
...
APACHE_MODULES="actions alias auth_basic
authn_file authz_host authz_groupfile
authz_default authz_user authn_dbm
autoindex cgi dir env expires include
log_config mime negotiation setenvif ssl
suexec userdir php5 proxy proxy_http
proxy_balancer"
...
```

<4> Many Mongrels Behind Apache

```
# Editing my file in conf.d/ or vhosts.d/

<Proxy balancer://mephistos/>
  BalancerMember http://127.0.0.1:3000
  BalancerMember http://127.0.0.1:3001
  BalancerMember http://127.0.0.1:3002
</Proxy>

... # continued on next slide
```

<4> Many Mongrels Behind Apache

```
# continued from previous slide
...
<VirtualHost *:80>
  ServerName whatever.com
  ProxyPass / balancer://mephistos/
  ProxyPassReverse / balancer://mephistos/
  ErrorLog /tmp/mephisto_proxy_errors_log
  CustomLog /tmp/mephisto_proxy_access_log
</VirtualHost>
```

<4> Many Mongrels Behind Apache

```
> mongrel_rails start -d -e production \
     -p 3000 --pid log/mephisto0.pid
> mongrel_rails start -d -e production \
     -p 3001 --pid log/mephisto1.pid
> mongrel_rails start -d -e production \
     -p 3002 --pid log/mephisto2.pid
> rcapache2 start

# Browse to http://localhost/admin
```

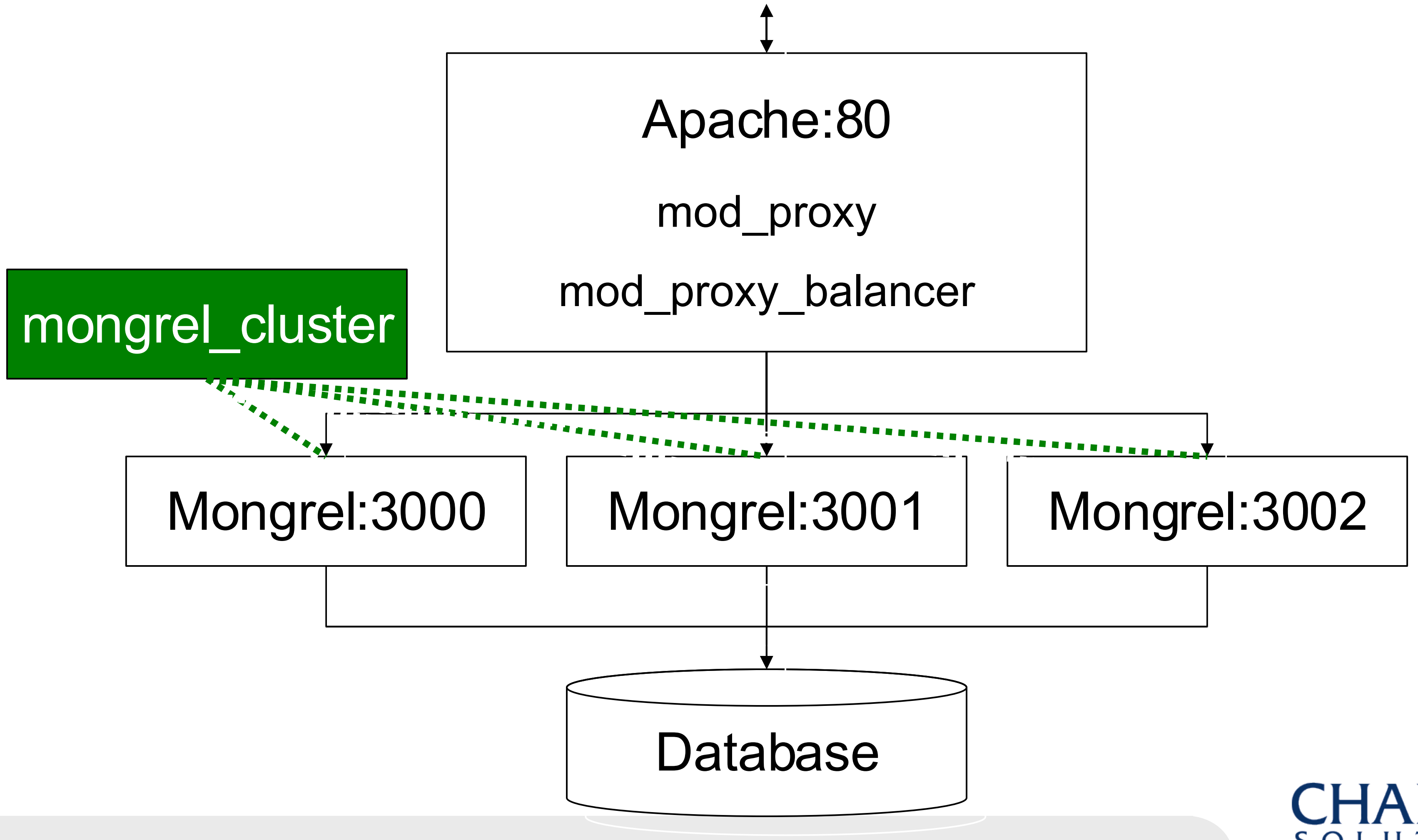
CHARIOT
SOLUTIONS

<4> Many Mongrels Behind Apache

- Works great, but:
  - That's a lot of typing to start up the mongrels
  - What if one goes down?
  - How do I upgrade my app?

Simplifying Server Management

<5> Mongrel_Cluster + Apache



‹5› Mongrel_Cluster + Apache

```
> gem install mongrel_cluster
# This writes out config file to config/
> sudo mongrel_rails cluster::configure \
  -e production -p 3000 -N 3          \
  -c /home/mephisto-0.7.3 -a 127.0.0.1  \
  --user mongrel --group mongrel
# This writes out pids to files in log/
> sudo mongrel_rails cluster::start
> sudo mongrel_rails cluster::stop
```

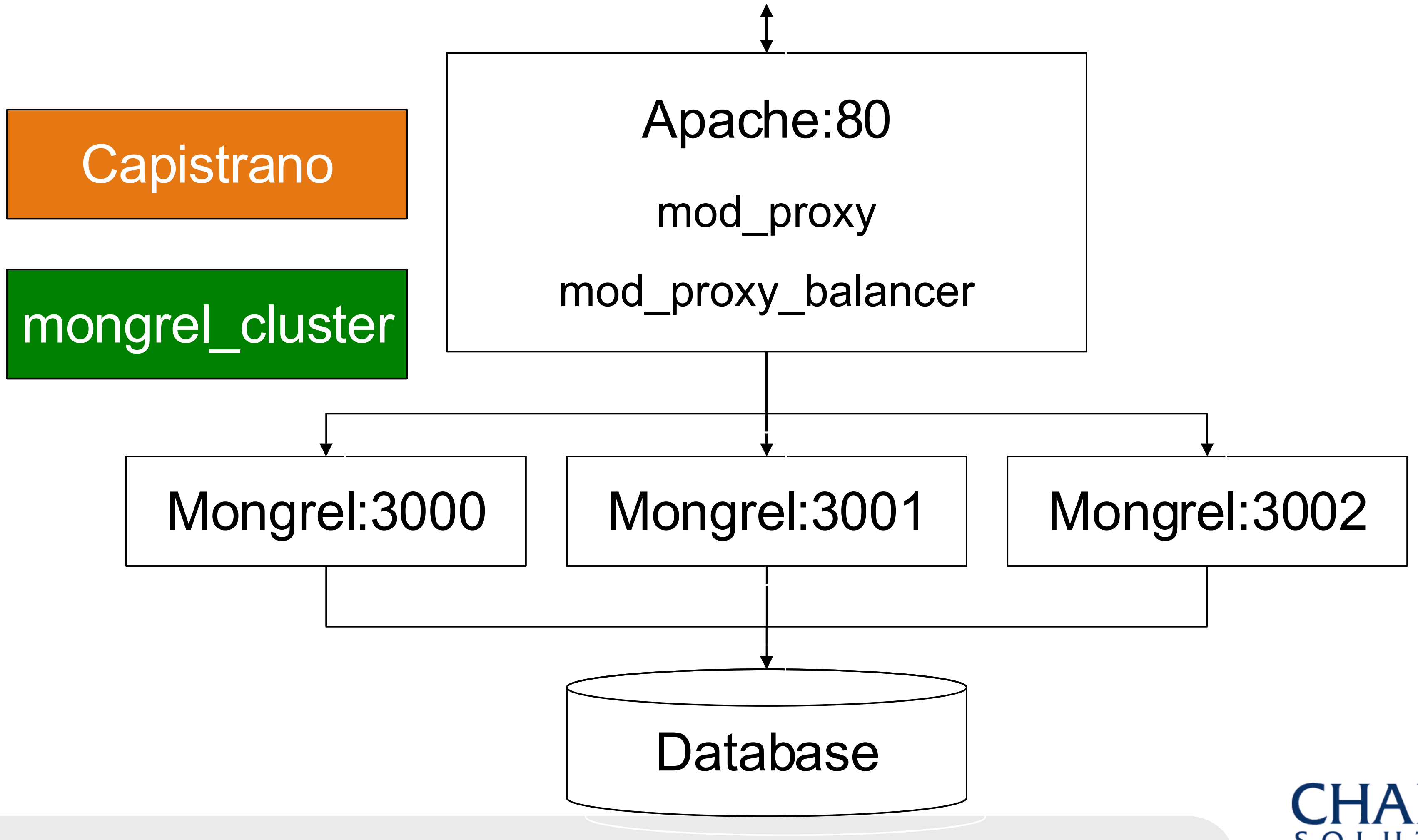
<5> Mongrel_Cluster + Apache

```
> cat config/mongrel_cluster.yml
---
user: mongrel
cwd: /home/mephisto-0.7.3
port: "3000"
environment: production
group: mongrel
address: 127.0.0.1
pid_file: log/mongrel.pid
servers: 3
```

<5> Mongrel_Cluster + Apache

- Simplified management of a cluster of Mongrel instances
- Still under active development
- Latest pre-release versions give you more fine-grained control and work better with monit

<6> Mongrels + Apache + Capistrano



CHARIOT SOLUTIONS

‹6› Mongrels + Apache + Capistrano

```
> gem install -s \
    http://gems.rubyonrails.com capistrano
# After checking mephisto into local svn
> svn co file:///svn/mephisto
> cd mephisto
> capify
> vi config/deploy.rb
> svn commit
> cap deploy:cold
> cap deploy:stop
```

<6> Mongrels + Apache + Capistrano

```
require 'mongrel_cluster/recipes'

set :application, 'mephisto'
set :repository,  'file:///files/svn/mephisto'
set :deploy_to,   "/tmp/deploy/#{application}"
set :domain,      'localhost'

set :mongrel_conf, "/home/mephisto-0.7.3/config/mongrel_cluster.yml"
set :mongrel_clean, true

role :app,  domain
role :web,  domain
role :db,   domain, :primary => true
...
```

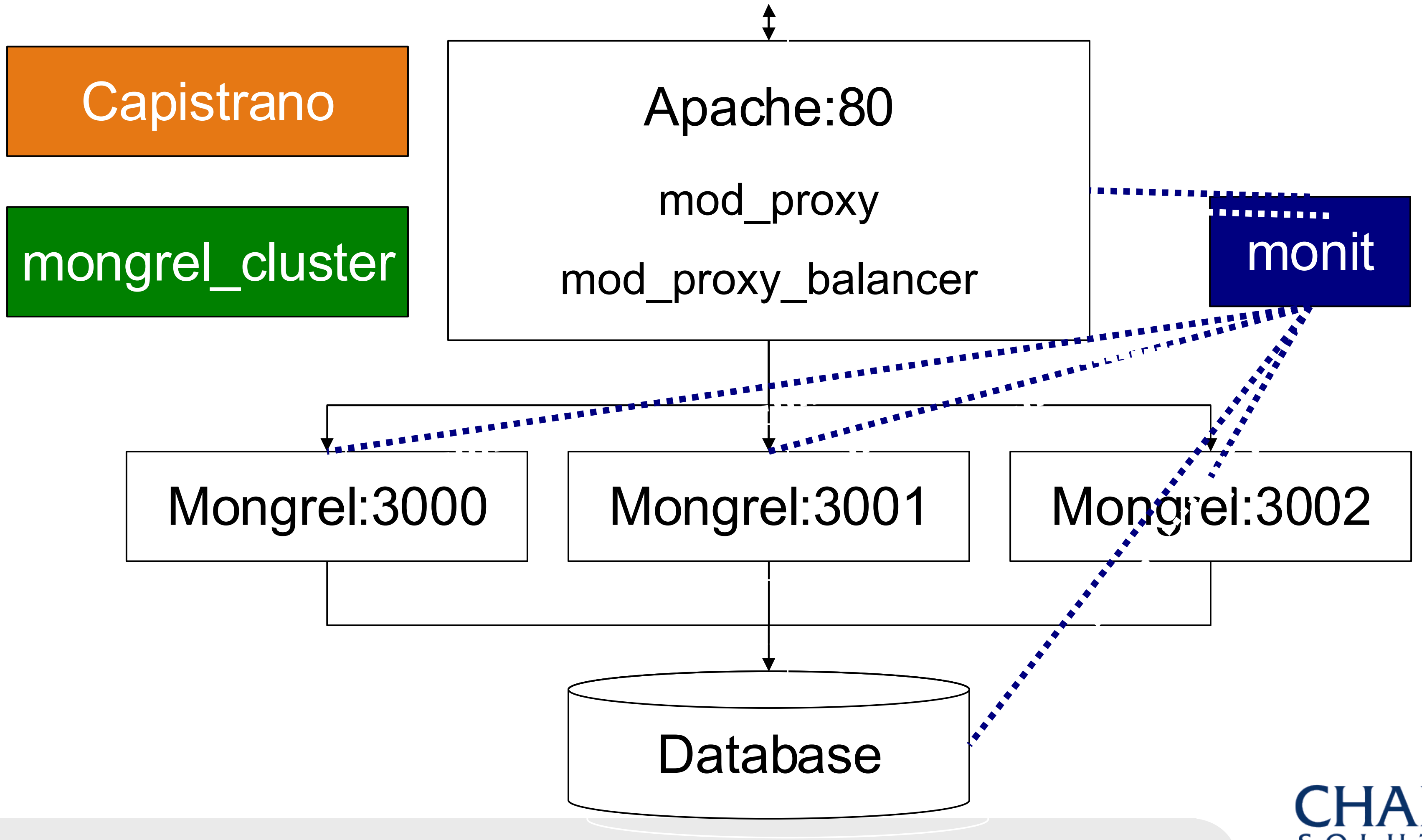
CHARIOT SOLUTIONS

```
namespace :deploy do

  task :cold do
    update
    migrate
    setup_mongrel_cluster
    start
  end

  # until mongrel_cluster updates to cap2...
  task :start,    :roles => :app do start_mongrel_cluster end
  task :stop,     :roles => :app do stop_mongrel_cluster end
  task :restart,  :roles => :app do restart_mongrel_cluster end

  task :setup_mongrel_cluster do
    sudo "cp #{release_path}/config/mongrel_cluster.yml #{mongrel_conf}"
    sudo "chown meara:users #{mongrel_conf}"
    sudo "chmod g+w #{mongrel_conf}"
  end
end
```

<6> Mongrels + Apache + Capistrano

- Capistrano is great
- Too much to cover in this talk
- Let's schedule another!

<7> Mongrels + Apache + Cap + Monit



<7> Mongrels + Apache + Cap + Monit

```
# /etc/monitrc

set daemon 60
set mailserver localhost
set mail-format { from: monit@phillyonrails.org }
set alert admin@phillyonrails.org

set httpd port 2812 and
    use address localhost  # only accept connection from localhost
    allow localhost        # allow localhost to connect to the server

# Continued on next slide...
```

```
check process mongrel_<%= @username %>_3000
   with pidfile /data/<%= @username %>/shared/log/mongrel.3000.pid
   start program = "/usr/bin/mongrel_rails cluster::start -C /data/<%
= @username %>/current/config/mongrel_cluster.yml --clean --only 3000"
   stop program = "/usr/bin/mongrel_rails cluster::stop -C /data/<%=
@username %>/current/config/mongrel_cluster.yml --clean --only 3000"
   if totalmem is greater than 110.0 MB for 4 cycles then
restart       # eating up memory?
   if cpu is greater than 50% for 2 cycles then
alert                 # send an email to admin
   if cpu is greater than 80% for 3 cycles then
restart               # hung process?
   if loadavg(5min) greater than 10 for 8 cycles then
restart          # bad, bad, bad
   if 20 restarts within 20 cycles then
timeout                        # something is wrong, call the sys-admin
   group mongrel

# REPEAT FOR OTHER MONGREL INSTANCES (e.g. 3001, 3002, etc.)
```

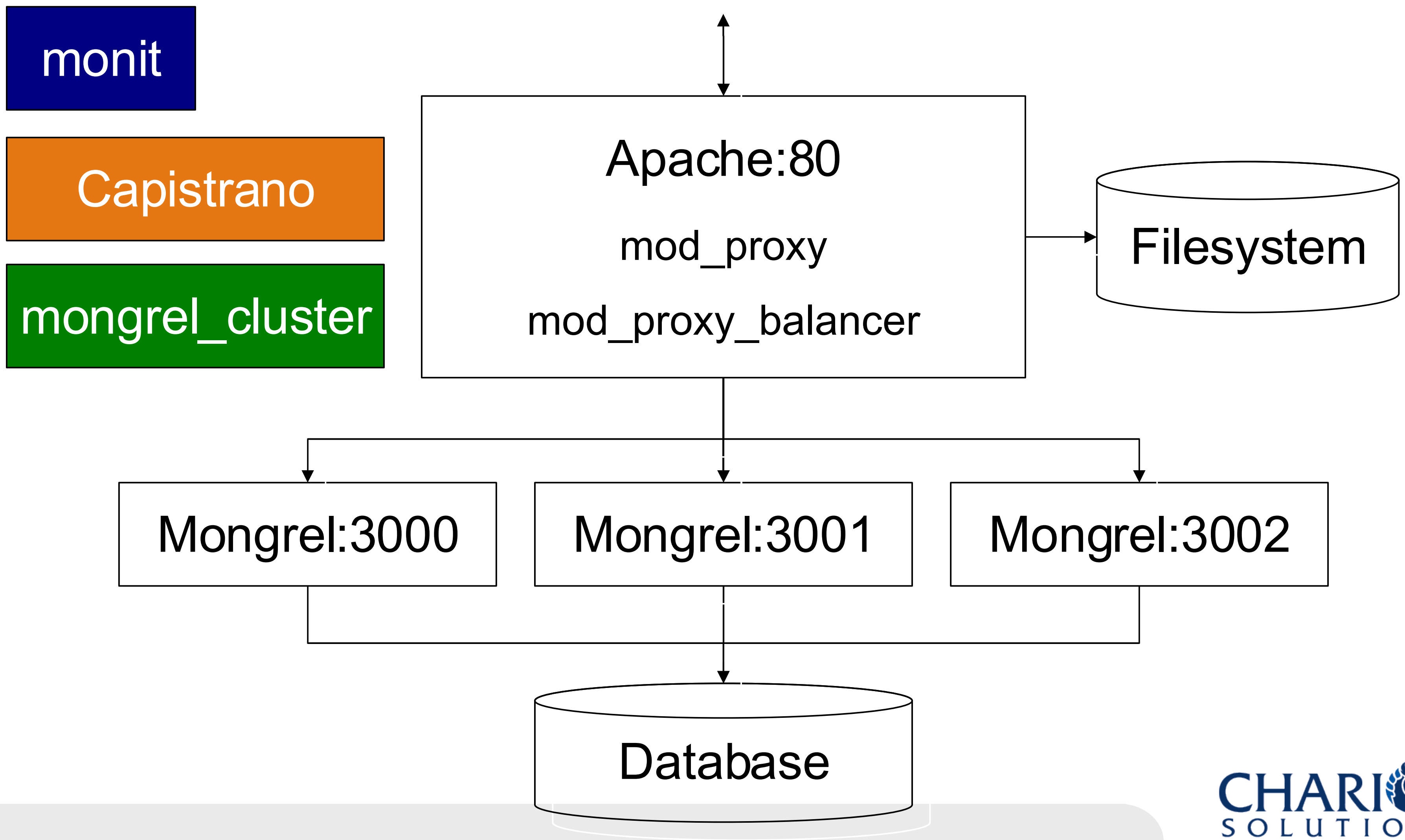
<7> Mongrels + Apache + Cap + Monit

- Sometimes mongrel instances die
- Monit keeps track and restarts them
- Can monitor/restart web server and database server too

Optimizing the Single-Server Deployment

CHARIOT SOLUTIONS

<8> Serve Static Files through Apache



CHARIOT SOLUTIONS

<8> Serve Static Files through Apache

```
DocumentRoot /tmp/mephisto
<Directory "/tmp/mephisto">
    Options FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
RewriteEngine On
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -f
RewriteRule (.*) $1 [L]
```

Don't forget to turn on mod_rewrite first, like you did for mod_proxy and mod_proxy_balancer

<8> Serve Static Files through Apache

```
# Another more specific option, replaces ProxyPass

RewriteRule \
"^/(images|stylesheets|javascripts)/?(.*)" "$0" [L]

RewriteRule ^([^.]+)$ $1.html [QSA]

RewriteCond %{REQUEST_FILENAME} !-f

RewriteRule "^/(.*)" "http://localhost:3000/$1" \
 [P,QSA,L]
```

<8> Serve Static Files through Apache

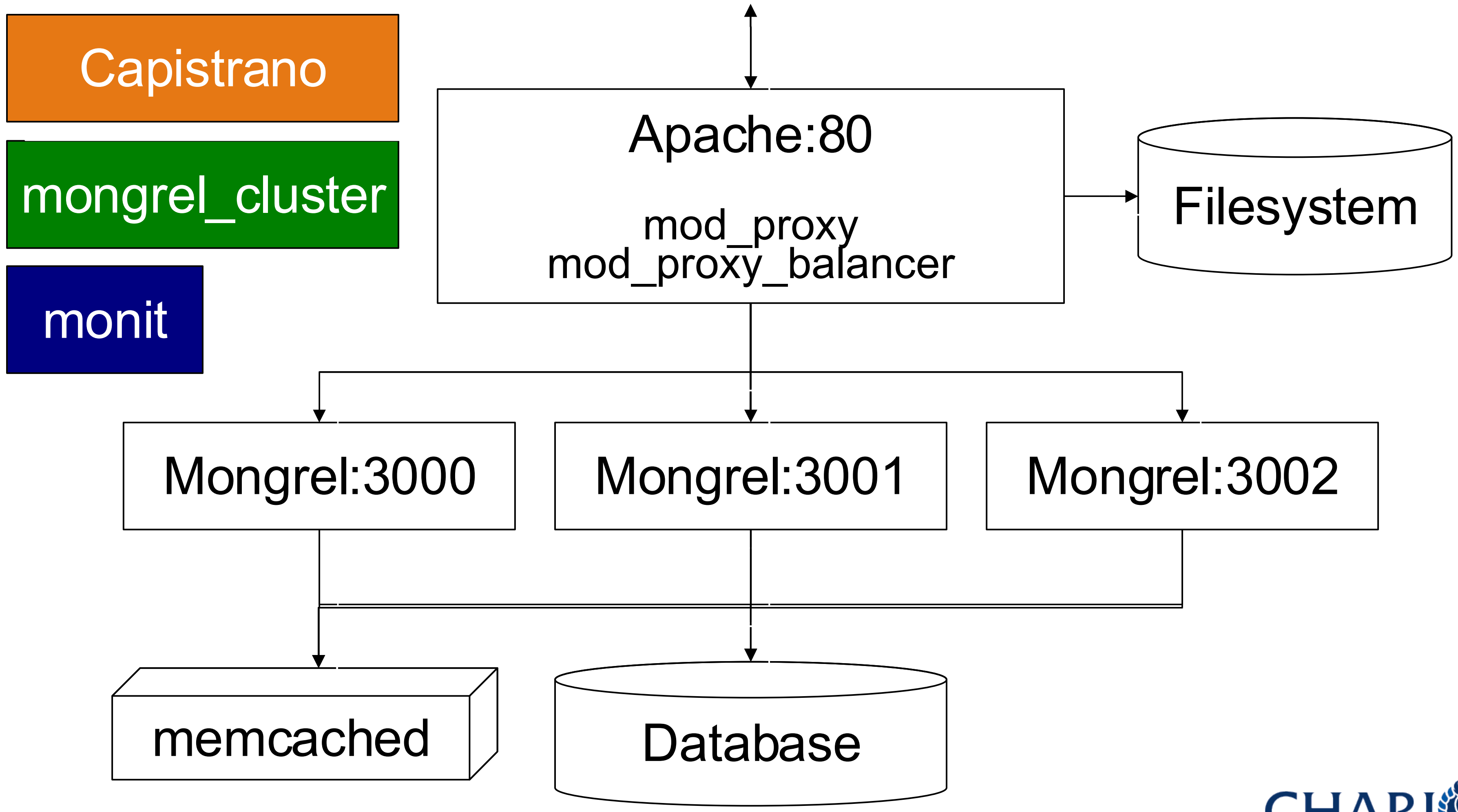
```
# Yet more possibilities

ProxyPass /images !
ProxyPass /stylesheets !

Alias /images /myapp/public/images
Alias /stylesheets /myapp/public/stylesheets
```

CHARIOT
SOLUTIONS

<8> Serve Static Files through Apache

- Lots of options here
- Bottom line:
  - Don't serve static content through Mongrel
  - Use rewrite rules and/or ProxyPass exceptions to set this up
- Think about security and maintenance too:
  - Set up static maintenance file rule
  - Disallow requests to .svn directories

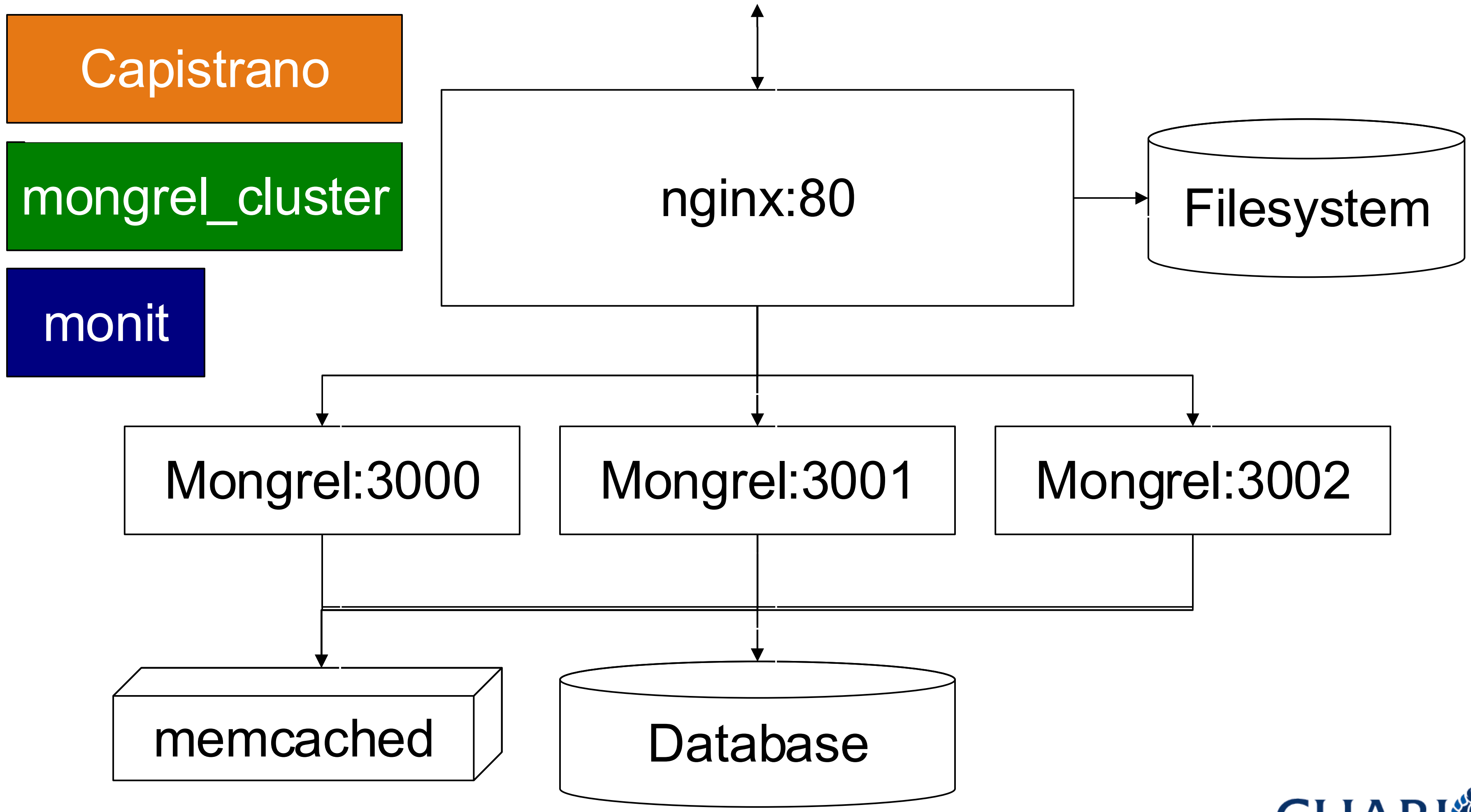
‹9› Reduce DB Hits with memcached



<9> Reduce DB Hits with memcached

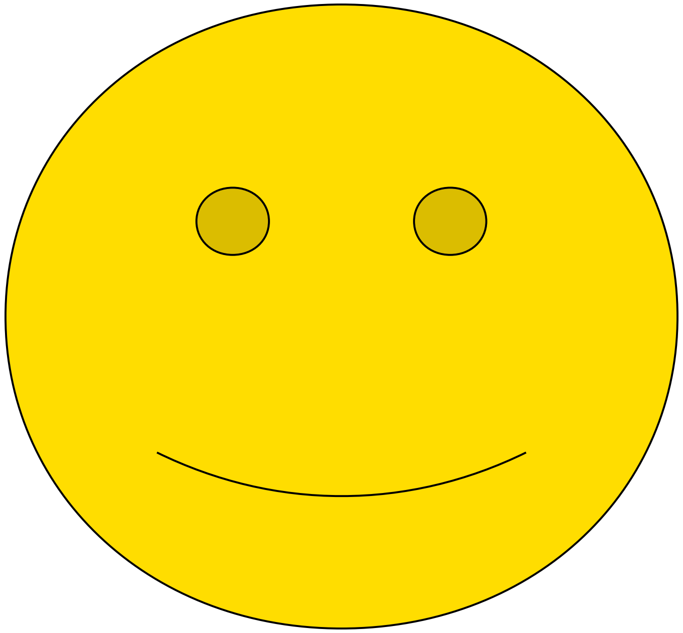
- Database is often the bottleneck
- Lots of repeated queries
- Try rails-level caching
- Eventually, add memcached
  - It's a big hash
  - Use cache_fu to integrate with ActiveRecord
- Multiple mongrel instances can share one
- Huge deployments might have many

<10> Replace Apache with nginx



<10> Replace Apache with nginx

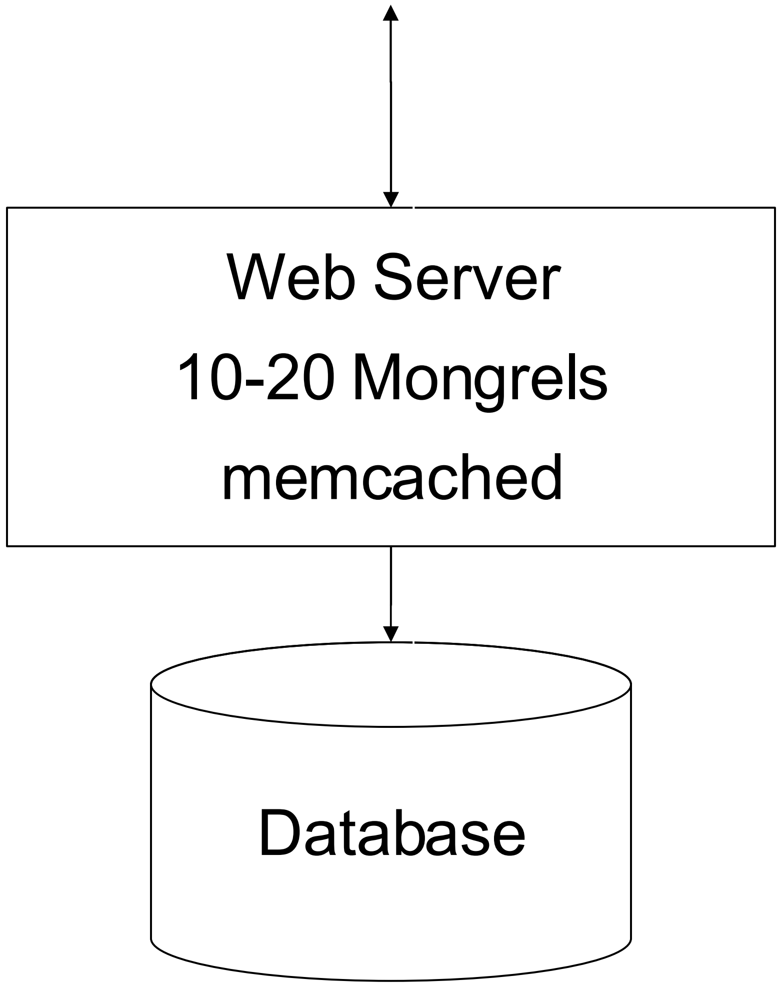
- Implementation left as an exercise for the reader



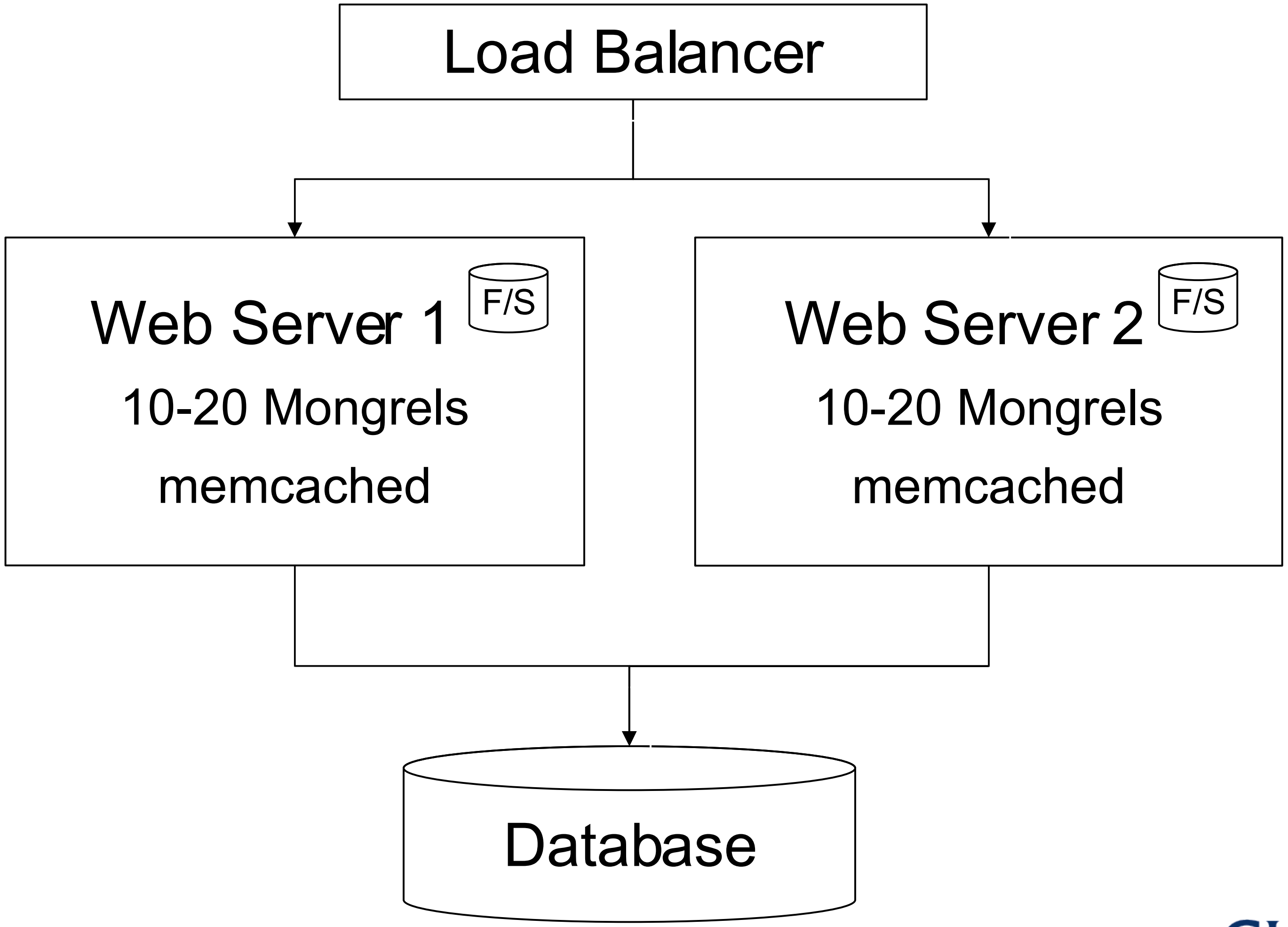
Beyond One Server

CHARIOT SOLUTIONS

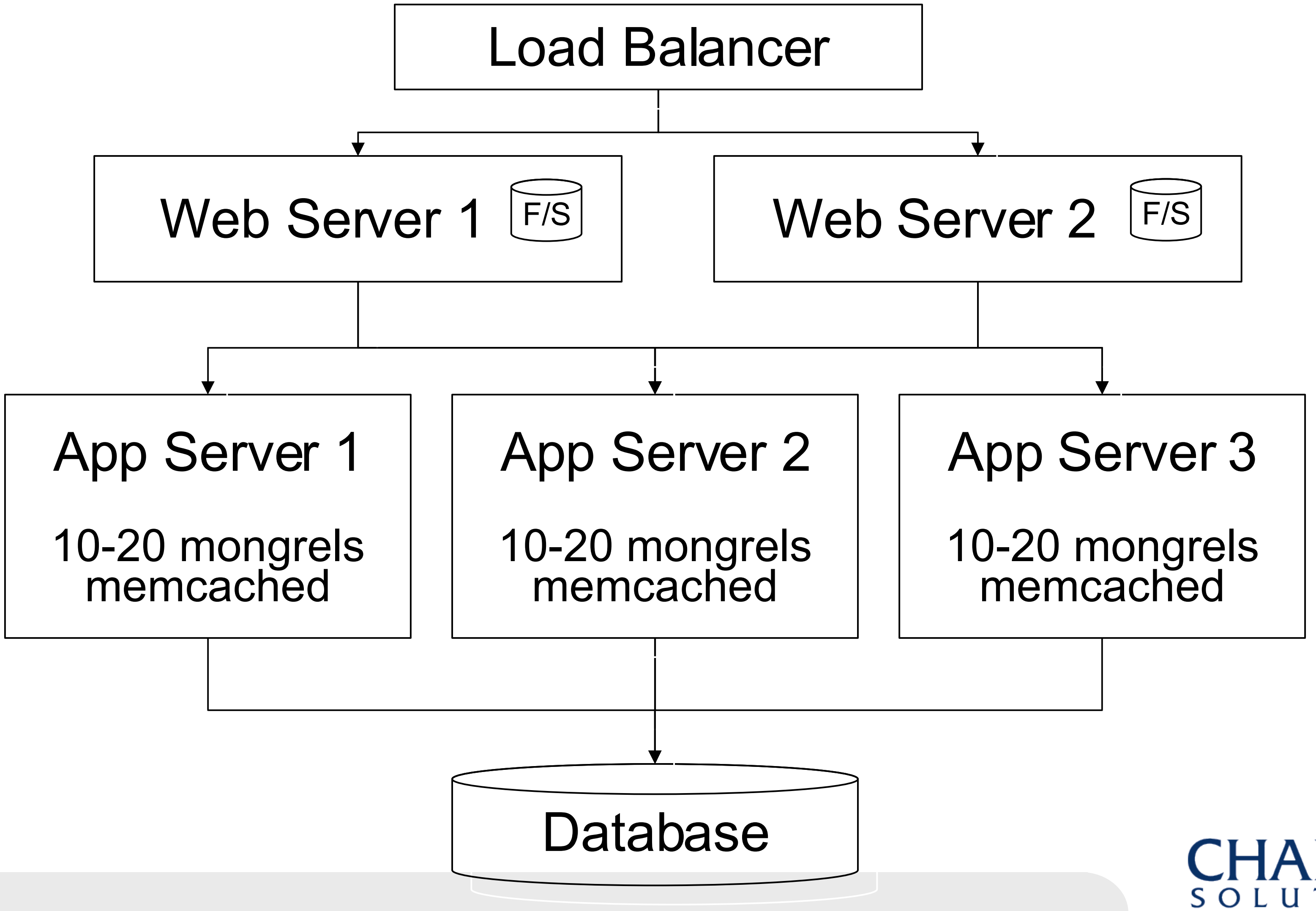
<11> Split Out the Database



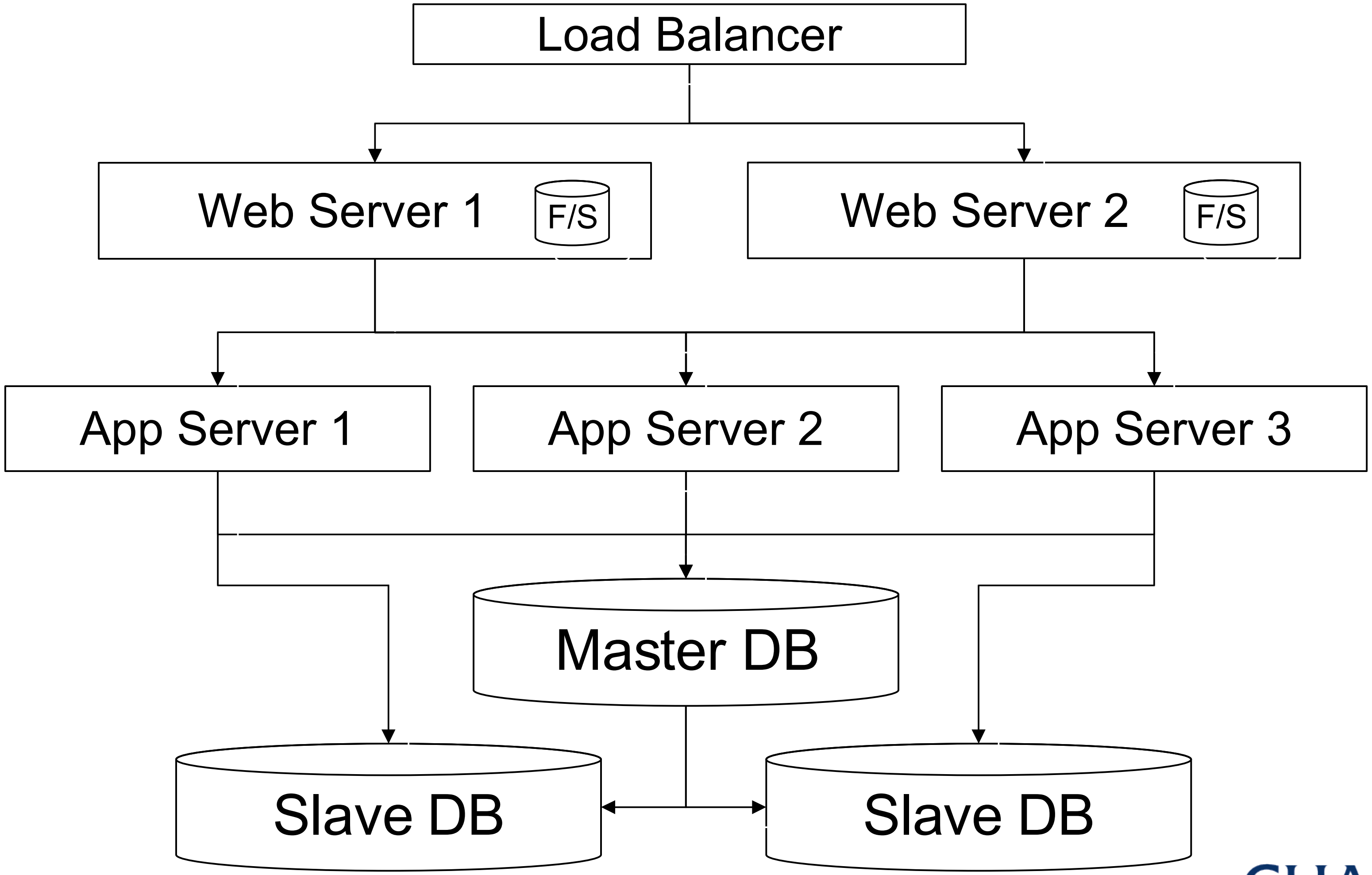
<12> Add More Web Servers



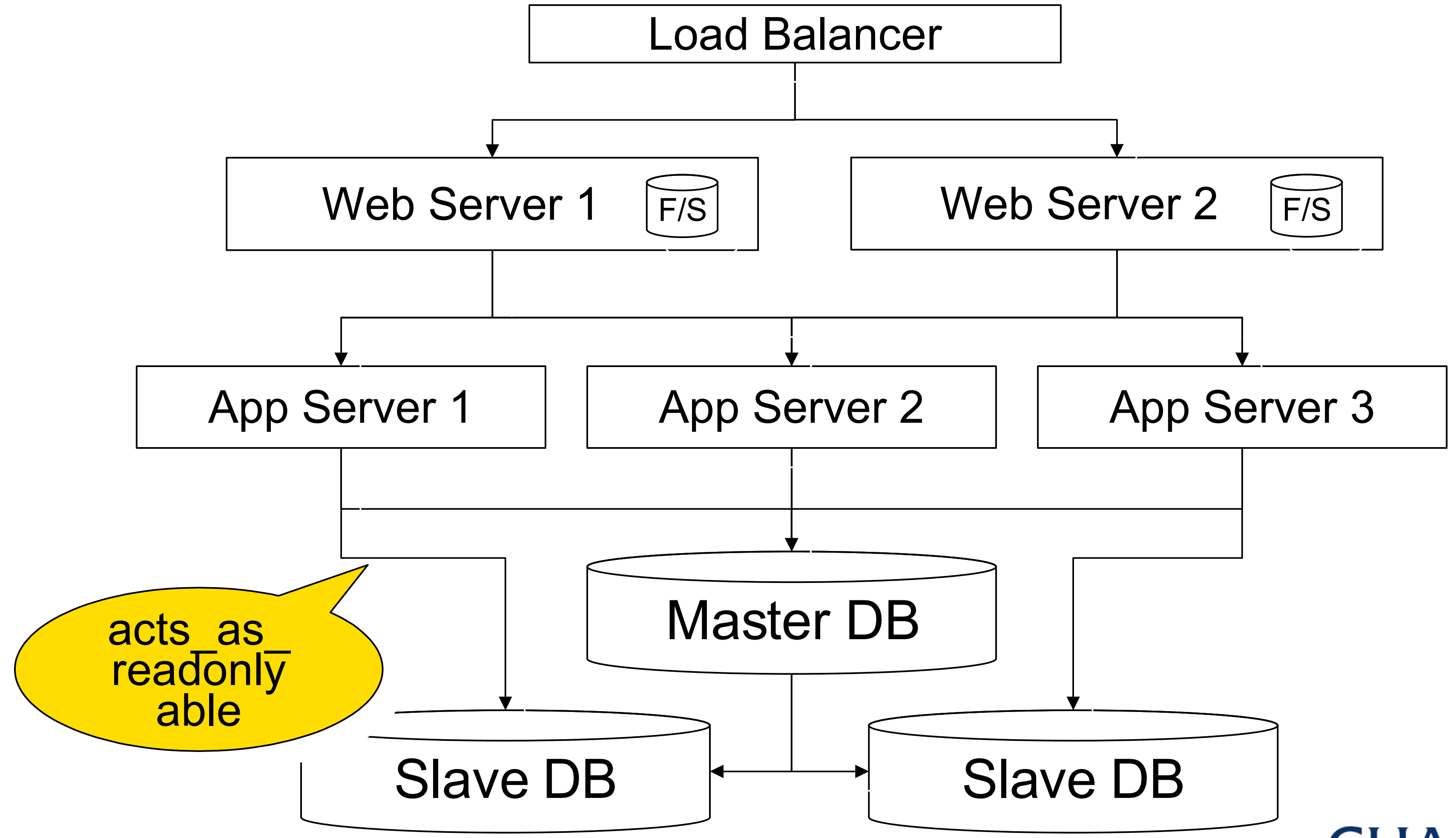
‹13› Split Web and App Servers



‹14› Add Read-Only Database Slaves



‹14› Add Read-Only Database Slaves



‹15› Database Clustering

- Advantages
  - Ideal for applications with lots of writes
  - All nodes are synchronized
- Disadvantages
  - Complex to configure
  - Need very fast network (e.g. gigabit or better)
  - Usually need extra nodes for management services
  - May need expensive SAN

‹16› Virtualization

- Each node is a VM, not a physical machine
- Can be run on same or different hardware
- Advantages
  – Very easy to scale out on demand
  – Can arrange VMs to maximize utilization
- Disadvantages
  – Can be difficult to tune performance
  – May not work well for database clusters

Hosting

- Hosts available for most of these scenarios
- Site5 and others support single-node shared hosting
- Rails Machine, Engine Yard and others support virtualization
- Can roll your own hosting with virtual dedicated servers, dedicated servers and/or grids like Amazon's EC2
- Need another presentation to cover all this

Ideas for Follow-Up Presentations

- Hosting options
- Performance Tuning
- Capistrano
- Virtualization
- Extreme Caching
- MySQL vs. PostgreSQL
- Hands-on Database Clustering
- Deployment with Apache 1.x, FastCGI, Lighttpd, etc.