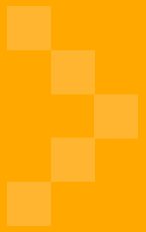
Desenvolvendo aplicações Web escaláveis

Elton Luís Minetto

CONAPHP
CONGRESSO NACIONAL DE PHP

Quem?



- ✓ Graduado e pós-graduado em Ciência da Computação. Cursando MBA em Gerenciamento de Projetos
- ✓ Trabalha com PHP desde 2000
- ✓ Autor do livro Frameworks para Desenvolvimento em PHP - Editora Novatec
- ✓ Membro do PHPSC
- ✓ Arquiteto de Software da empresa Network Vox

Terminologia

- ✓ Performance: a habilidade que uma aplicação tem de atingir um objetivo, como por exemplo responder no menor tempo possível
- ✓ Capacidade: a carga total que uma aplicação pode suportar
- ✓ Escalabilidade: a habilidade de uma aplicação manter a performance quando a carga de trabalho aumenta. É a junção da capacidade e da performance

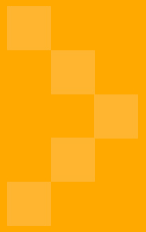
Ou…

Ou...

- ✓ Performance: a velocidade máxima do carro
- ✓ Capacidade: o limite de velocidade e o número de pistas da estrada
- ✓ Escalabilidade: quantos carros e pistas eu posso adicionar sem diminuir a velocidade do tráfego
  - ✓ “Performance is a problem. Scaling your performance is a bigger problem”

Camadas

Web Servers

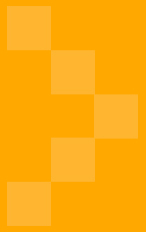


- ✓ Apache: Prefork X Worker
- ✓ Prefork
  - ✓ Usa fork(). Cada processo filho trata uma conexão
  - ✓ Grande uso de memória
  - ✓ Rápido
  - ✓ Bom para até duas CPUs
- ✓ Worker
  - ✓ Apache 2.0 e superior
  - ✓ Múltiplas threads dentro de cada filho (poucos filhos)
  - ✓ Diminui o uso de memória
  - ✓ Mais escalável
  - ✓ Melhor com múltiplos processadores

Web Servers

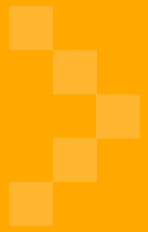
- ✓ “Usar servidores dedicados para conteúdo estático“ (Flickr)
- ✓ Apache pode ser substituído por outro mais leve: lighttpd, tux, thttpd
- ✓ Usar domínios diferentes para conteúdo estático e dinâmico. Exemplo:
  - ✓ http://www.eltonminetto.net (Servidor Apache)
  - ✓ http://static.eltonminetto.net (Servidor thttpd)
  - ✓ Usa a possibilidade dos navegadores acessarem múltiplos domínios ao mesmo tempo. Não sobrecarrega o servidor de conteúdo dinâmico

MySQL



- ✓ “Nenhum dos problemas de escalabilidade que enfrentaram era relacionado com PHP. Os maiores problemas encontrados eram relacionados com base de dados” (Digg)
- ✓ “Sharding é usado para quebrar a base de dados em várias porções menores” (Digg)
- ✓ “Fazer o tunning do MySQL durante a escolha da engine de armazenamento das tabelas. Usar InnoDB quando precisa de transações e MyISAM quando não precisa” (Digg)

MySQL

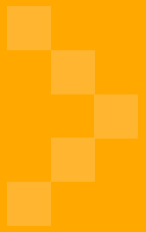


- ✓ “Desnormalização ou cacheamento são as únicas formas de gerar uma tag cloud em milissegundos para milhões de tags” (Flickr)
- ✓ “Não usar conexões persistentes ao banco de dados “ (Friendster)

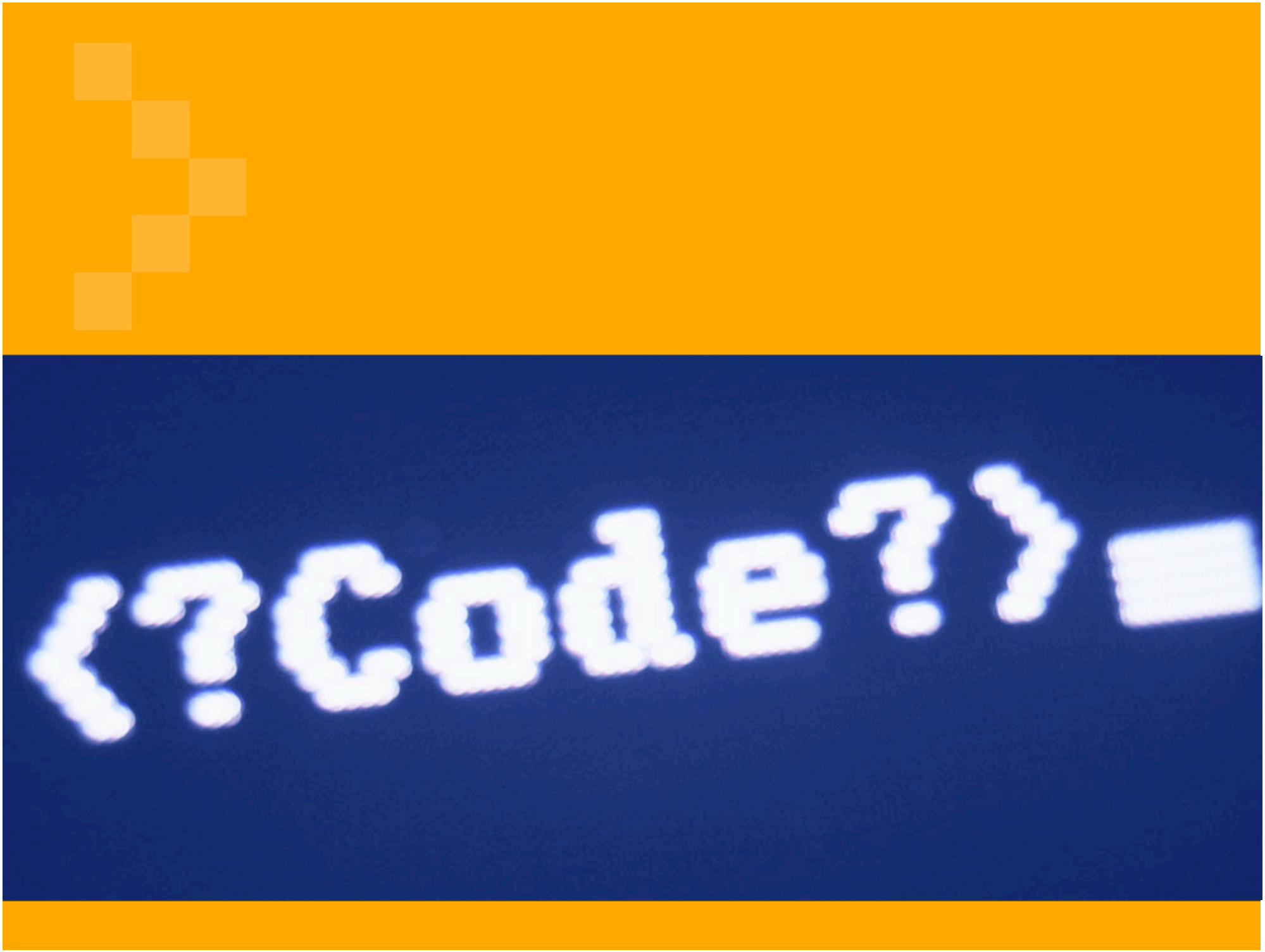
MySQL

- ✓ Usar replicação Master-Slave.
- ✓ Dividir a carga entre servidores. As requisições de modificação (INSERT, UPDATE,DELETE) podem ser enviadas para o Master. Os dados são replicados para os Slaves. As requisições de leitura (SELECT) são enviadas direto para os Slave
- ✓ Usar o tipo correto de dados na tabela:
  - ✓ INT x SMALLINT x TINYINT
  - ✓ CHAR x VARCHAR

MySQL



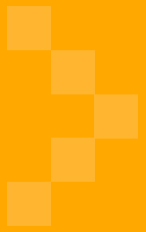
- ✓ **Master-Slave tem o problema do tempo de sincronização. Resposta: sharding.**
  - ✓ **“Uma base de dados pode ser sharded por tabelas, dados ou faixas (ranges). É similar ao particionamento, mas possui algumas diferenças. Sharding envolve separar os dados em máquinas fisicamente distintas, enquanto que particionamento geralmente ocorre em mesmo hardware. MySQL não suporta nativamente sharding, mas sim tabelas particionadas, tabelas federadas (federated) e clusters.”**


<?Code?>

PHP

- ✓ Usar “boas práticas de programação”
  - ✓ echo é mais rápido que print
  - ✓ require_once() é lento
  - ✓ etc,etc
  - ✓ http://reinholdweber.com/?p=3
  - ✓ http://www.devolio.com/blog/archives/314-Practical-and-impractical-PHP-Optimizations.html

PHP



- ✓ PHP tem um compilador JIT que gera um código intermedário chamado opcode que é então interpretado. Por default essa compilação ocorre em todas as execuções do script. Para otimização e caching desse opcode, existem algumas soluções:
  - ✓ Pacote APC do PECL
  - ✓ XCache
  - ✓ Zend Platform ($$$)

Cache

- ✓ **Memcached: é um sistema de cache de objetos em memória distribuída de alta performance.Ele foi desenvolvido de maneira que se consiga armazenar qualquer tipo de informação mas é largamente usado em aplicações web para armazenar conteúdos como resultados de queries SQL, sessões de usuários, arquivos CSS, etc.**
- ✓ **É possível criar “memcached farms”, aumentando a escalabilidade**

Cache

```
<?php //exemplo usando cliente PECL
$memcache = new Memcache;
$memcache->connect('localhost', 11211) or die ("Could not connect");
$version = $memcache->getVersion();
echo "Server's version: ".$version."<br/>\n";
$tmp_object = new stdClass;
$tmp_object->str_attr = 'test';
$tmp_object->int_attr = 123;
$memcache->set('key', $tmp_object, false, 10) ;
echo "Store data in the cache (data will expire in 10 seconds)<br/>\n";
$get_result = $memcache->get('key');
echo "Data from the cache:<br/>\n";
var_dump($get_result);
?>
```

80-90% of the end-user response time is spent on the frontend.

Javascript

- ✓ “Algumas pessoas reclamam que o Digg é lento. Mas isso é mais devido ao uso de grandes bibliotecas javascript e não da arquitetura de backend (Digg)
- ✓ JavaScript pode ser comprimido
  - ✓ http://developer.yahoo.com/yui/compressor/
  - ✓ Usar compresão no Apache.

Average Time to Load Toolkit (non cached, gzipped, minified)

Toolkit	Time Avg	Samples
jquery-1.2.1	709.2752	5130
dojo-1.0.1	885.1917	5118
prototype-1.6.0	894.8619	5121
yahoo-utilities-2.4.0	908.6602	5118
protoculous-1.0.2	1115.0848	5106

Average Time to Load Toolkit (cached, gzipped, minified)

Toolkit	Time Avg	Samples
yahoo-utilities-2.4.0	115.2847	4956
jquery-1.2.1	129.9630	5142
prototype-1.6.0	131.4928	4955
dojo-1.0.1	165.4857	5139
protoculous-1.0.2	264.2389	5135

Imagens

- ✓ Editores de imagens possuem grandes ferramentas para otimizar o tamanho do arquivo sem perder qualidade visual. Mesmo assim elas adicionam informações extra no documento, como nome da ferramenta, data de criação, etc.
- ✓ O site Smushit.com permite fazer upload de uma imagem ou indicar uma URL com a imagem a ser otimizada. Lançado em 10/2008 por pesquisadores do Yahoo! Exceptional Performance Team.

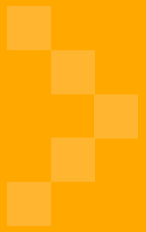
Métricas

Métricas

Métricas


O que você ainda tem
www32.flickr.mud.yahoo.com CPU last day
Percent
100
50
0
Tue 00:00
Tue 12:00
área
"segura"
@85% CPU
User CPU
Nice CPU
System CPU
WAIT CPU
Idle CPU

Ferramentas



- ✓ Apache ab (aplicação)
- ✓ Ganglia (S.O. e serviços)
- ✓ Firebug e YSlow (javascript, aplicação)
- ✓ Xdebug (profiling do PHP)
- ✓ Outras...

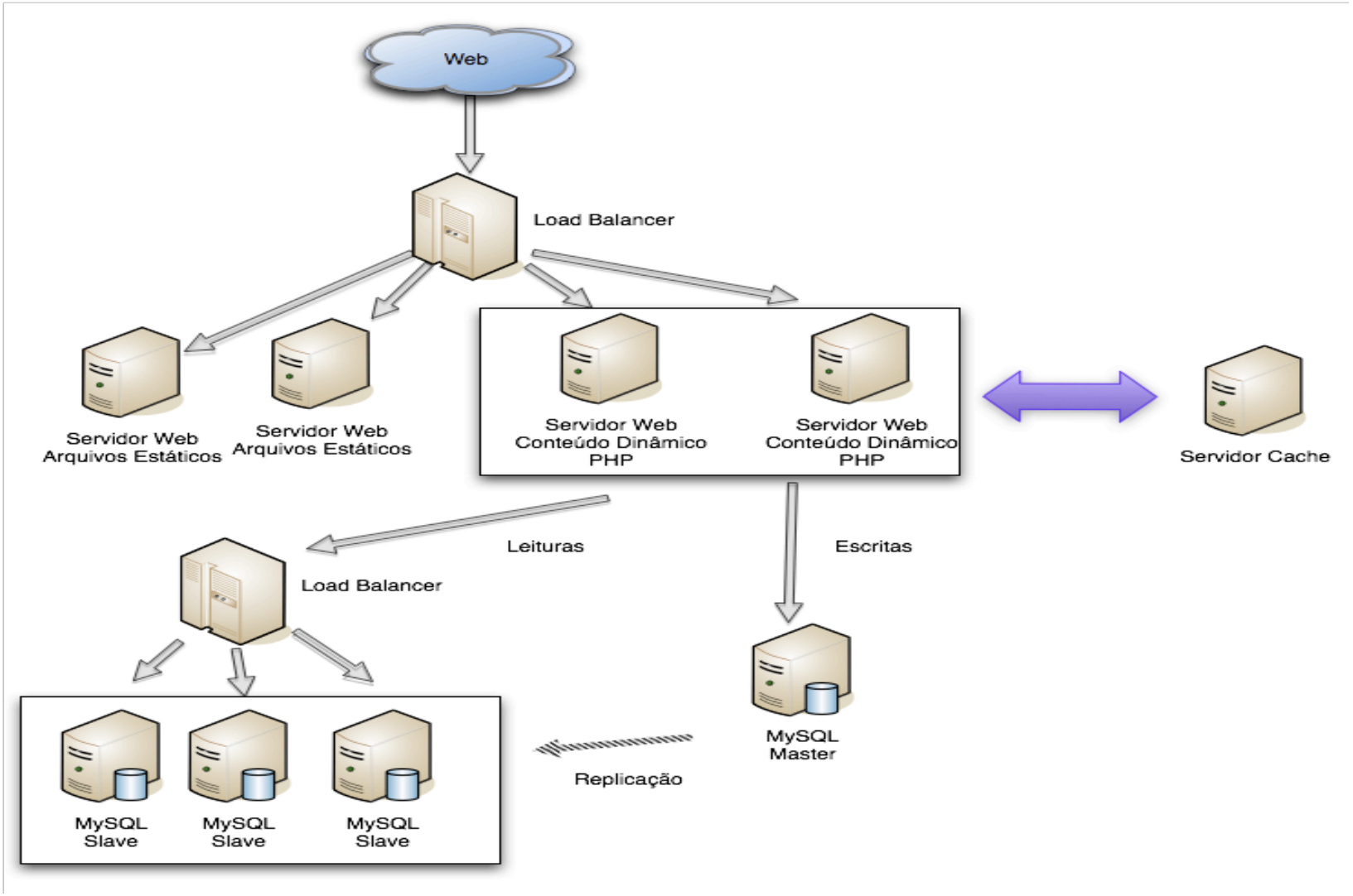
Arquitetura

- ✓ Load Balancers
- ✓ Cache servers
- ✓ Bancos de dados Master/Slave, Sharding
- ✓ Scale-Out Wins Over Scale-Up (escalar horizontalmente adicionando mais máquinas é melhor do que verticalmente adicionando mais memória/CPU)

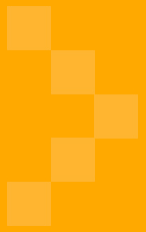
Arquitetura

- ✓ Sobre o Youtube:
  - ✓ “Eles seguiram uma evolução comum: servidor único, único master e múltiplos slaves para leitura, depois particionaram a base de dados e finalmente usaram uma abordagem baseada em sharding.”

Arquitetura



Arquitetura – Observações



- ✓ Como agora um cliente pode ser atendido por diversos servidores Web durante o uso, as sessões dos usuários devem ser salvas no banco de dados ou nos servidores de cache (Memcached)
- ✓ O MySQL Master é um SPOF(Single Point of Failure) – adicionar mais Masters em um esquema de replicação Master/Master. Ou usar sharding
- ✓ Diversos servidores de cache podem ser adicionados

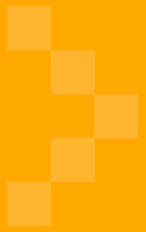
Lições Aprendidas

- ✓ Dividir a carga
- ✓ Usar servidores dedicados para conteúdo estático
- ✓ Ao invés de comprar máquinas grandes e centralizadas, é melhor comprar um monte de pequenas e baratas.
- ✓ Base de dados é um gargalo. Atacar e corrigir consultas lentas
- ✓ Coletar várias estatísticas de performance
- ✓ Usar InnoDB quando precisa de transações e MyISAM quando não precisa

Lições Aprendidas

- ✓ Fazer medições realísticas. Capacity planning deve ser feito baseado em dados reais e não abstratos.
- ✓ Começar lentamente. Não comprar muito equipamento apenas porque está apavorado/feliz com o fato de que o seu site vai explodir.
- ✓ Descobrir o máximo de cada servidor para poder saber se está perto do limite

Receita para tratar crescimento rápido



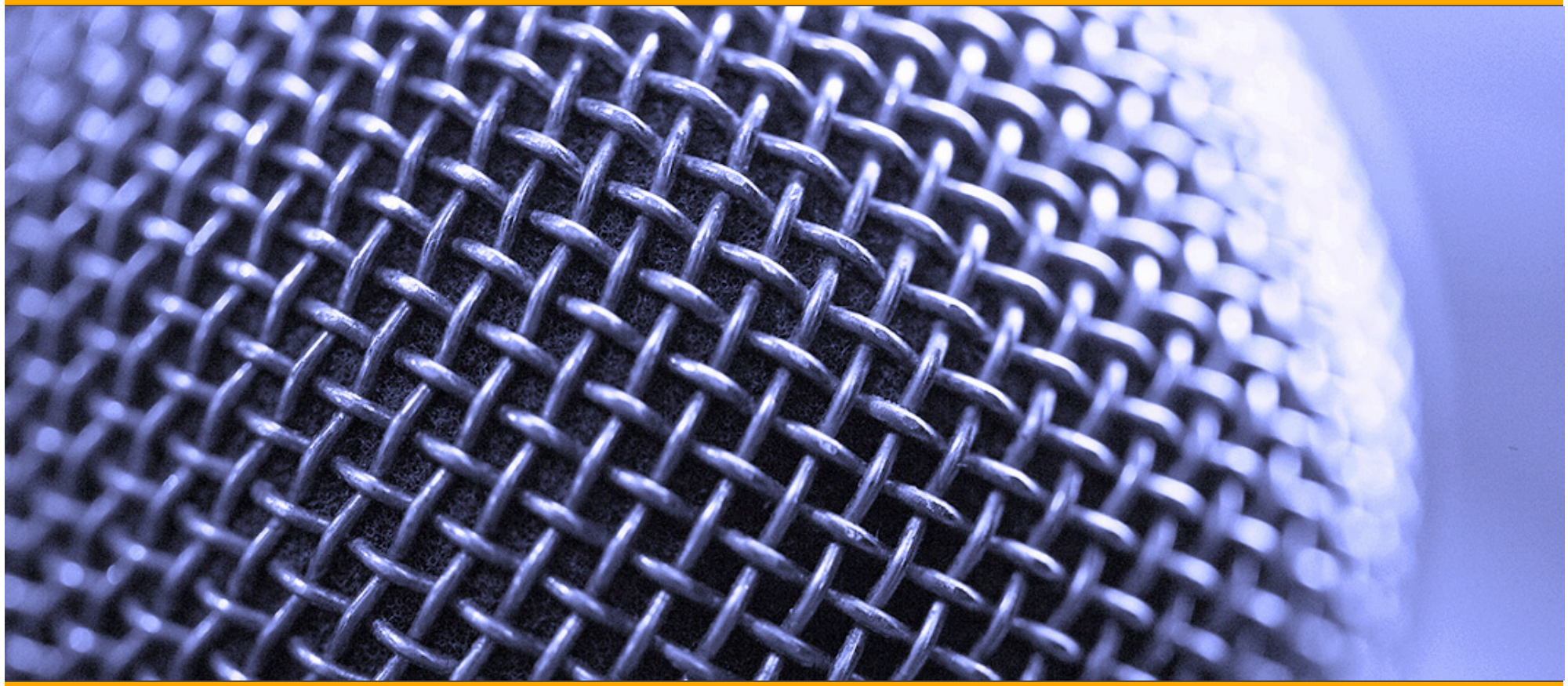
```
while (true)
{
 identify_and_fix_bottlenecks();
 drink();
 sleep();
 notice_new_bottleneck();
}
```

Fonte: YouTube Team

Referências

- ✓ http://developer.yahoo.com/performance/
- ✓ http://highscalability.com/unorthodox-approach-database-design-coming-shard
- ✓ http://blog.feliperibeiro.com/2008/04/slides-da-palestra-php-para-desenvolvimento-de-aplicacoes-de-grande-porte.html
- ✓ http://www.xdebug.org
- ✓ http://devzone.zend.com/content/zendcon_07_slides/White_Eli_zendcon-2007-high-perf.pdf
- ✓ http://jst.pbwiki.com/
- ✓ http://www.slideshare.net/jallspaw/velocity2008-capacity-management1-484676

Perguntas?



Contato

```
<?php
$card = array(
  ‘nome’ =>  ‘Elton Luís Minetto’,
  ‘site’ => ‘http://www.eltonminetto.net’,
  ‘e-mail’ => ‘eminetto@gmail.com’,
  ‘fone’ => ‘(47) 9189 6359’
);
var_dump($card);
?>
```

Estamos contratando!
Envie currículos para
elton.minetto@networkvox.com.br