Trabalho de Dimensionamento de Sistemas

Identificacao

Autores: - Luan Carvalho - 2526438 - Jose Freitas Alves Neto - 2519203
- Isaque Araujo Gadelha - 2519194

Repositorio GitHub: https://github.com/luancaarvalho/Trab_dimensionamento

Link para Otimizacoes:
https://github.com/luancaarvalho/Trab_dimensionamento/tree/main/optimization

---

Estrategia Adotada

Abordagem Escolhida: Tuning de Software + Escalabilidade Horizontal

Optamos por uma estrategia hibrida que combina:

1. **Tuning de Software (Principal):** Implementacao de cache em nivel de servidor usando Apache mod_cache
2. **Escalabilidade Horizontal:** 3 instancias t3.xlarge (4 vCPUs, 16 GB RAM cada)

Justificativa

Por que Tuning de Software? - Experiencia previa com WordPress: Ja trabalhei extensivamente com WordPress e conheco suas caracteristicas de performance - O perfil da aplicacao WordPress e altamente cacheable (conteudo estatico) - Apache mod_cache oferece ganhos massivos de performance com custo zero de infraestrutura adicional - Reduz drasticamente a carga no backend (PHP/MySQL) ao servir conteudo direto do cache em disco - Conhecimento tecnico especifico sobre configuracao de cache em nivel de servidor para WordPress

Por que 3x t3.xlarge? - Custo horario: 3 x $0.1664/h = **$0.4992/h** (abaixo do limite de $0.50/h) - Configuracao balanceada entre CPU e memoria para WordPress - Load Balancer distribui trafego uniformemente entre as 3 instancias

Por que NAO escolhemos apenas Scale-out? - Scale-out puro (mais instancias t3.micro) seria mais caro e menos eficiente - Sem cache, precisariamos de ~20 instancias t3.micro para atingir o mesmo throughput - Custo seria: 20 x $0.0104/h = $0.208/h (mais barato MAS com latencia muito maior)

Arquitetura Final

Infraestrutura

Componente	Especificacao	Quantidade
Aplicacao	t3.xlarge (4 vCPUs, 16GB RAM)	3 instancias
Banco de Dados	t3.xlarge (4 vCPUs, 16GB RAM)	1 instancia
Load Balancer	Application Load Balancer (ALB)	1
VPC	2 Subnets (publicas)	us-east-1

Configuracao de Software

1. Apache mod_cache (Script: `apache_mod_cache_otimizado.sh`)

Parametros Criticos:

```
# Cache em disco
CacheEnable disk "/"
CacheRoot /var/cache/httpd/mod_cache_disk

# Configuracao-chave para evitar 403 Forbidden
CacheQuickHandler off

# Ignora cookies do Load Balancer
CacheIgnoreHeaders Cookie

# Tempo de expiracao
CacheDefaultExpire 3600

# Tamanho maximo de arquivo em cache
CacheMaxFileSize 10000000
```

Racionalizacao Tecnica: - `CacheQuickHandler off`: Permite que cache passe pela fase de autorizacao do Apache, evitando conflito com headers do ALB - `CacheIgnoreHeaders Cookie`: Essencial para funcionar com Load Balancer que adiciona cookies de sessao - Cache em disco (`mod_cache_disk`): Mais eficiente que cache em memoria para WordPress

2. Apache MPM Event

```
<IfModule mpm_event_module>
    StartServers 3
    MinSpareThreads 75
    MaxSpareThreads 250
    ThreadsPerChild 25
    MaxRequestWorkers 400
    MaxConnectionsPerChild 1000
</IfModule>
```

3. PHP OPcache

```
opcache.enable=1
opcache.memory_consumption=128
opcache.max_accelerated_files=10000
opcache.revalidate_freq=2
```

4. KeepAlive e Compressao

```
KeepAlive On
KeepAliveTimeout 5
MaxKeepAliveRequests 100

# mod_deflate para compressao
AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css
        application/javascript
```

Resultados Obtidos

Teste de Carga: 100 Usuarios por 2 Minutos

Metrica	Baseline	Otimizado	Melhoria
RPS Maximo	25.9 req/s	48.3 req/s	+86.5%
Latencia Mediana (P50)	980 ms	4 ms	99.6% mais rapido
Latencia P95	4200 ms	5 ms	99.88% mais rapido
Latencia P99	4800 ms	18 ms	99.6% mais rapido
Taxa de Erro	0%	0%	Mantida
Total de Requests	3,121	5,765	+84.7%

Tabela Completa - Baseline (100 usuarios)

Type	Name	Requests	Failures	P50	P95	P99	Min	M
GET	/	707	0	940ms	4600ms	4700ms	263ms	49
GET	/post-detail	2414	0	990ms	4400ms	4600ms	258ms	50
Aggregated	**Total**	**3121**	**0**	**980ms**	**4400ms**	**4700ms**	**258ms**	**50**

Tabela Completa - Otimizado (100 usuarios)

Type	Name	Requests	Failures	P50	P95	P99	Min	Max	A
GET	/	1285	0	4ms	5ms	17ms	2.6ms	476ms	4.9
GET	/post-detail	4480	0	4ms	5ms	18ms	2.4ms	475ms	4.(
Aggregated	**Total**	**5765**	**0**	**4ms**	**5ms**	**18ms**	**2.4ms**	**476ms**	**4.7**

SLOs Atendidos

- **Latencia P95 < 10000ms:** SIM (5ms << 10000ms)
- **Taxa de Erro < 1%:** SIM (0%)
- **Throughput:** +86.5% com mesmo numero de instancias

Resultados Completos por Carga

Os resultados completos de todos os testes (100, 300, 600, 900, 1100 usuarios) estao disponiveis no repositorio em: - **Baseline:** `resultados/baseline/` - **Otimizado:** `resultados/otimizado/`

Cada pasta contem: - `index.html` - Relatorio interativo do Locust com graficos (RPS x Tempo, Latencia x Tempo, Distribuicao) - `dados_stats.csv` - Estatisticas agregadas - `dados_stats_history.csv` - Historico temporal - `dados_failures.csv` - Registro de falhas - `dados_exceptions.csv` - Excecoes capturadas

IMPORTANTE: Os graficos de RPS x Tempo e Latencia x Tempo para TODAS as execucoes (100, 300, 600, 900, 1100 usuarios) estao disponiveis nos arquivos `index.html` dentro de cada pasta de resultados. As tabelas apresentadas acima sao especificamente do teste com 100 usuarios (baseline e otimizado) para fins de comparacao direta no relatorio.

Para visualizar os relatorios completos com graficos interativos, abra os arquivos `index.html` em um navegador.

Analise de Custo

Custo Horario da Camada de Aplicacao

Recurso	Tipo	Quantidade	Preco Unitario/h	Custo Total/h
Instancias de Aplicacao	t3.xlarge	3	$0.1664	$0.4992
TOTAL				**$0.4992**

Orcamento: $0.50/h
Utilizado: $0.4992/h
Margem: $0.0008/h (**0.16% abaixo do limite**)

Custo-Beneficio

- **Investimento:** $0 em infraestrutura adicional (apenas tuning)
- **Ganho:** +86.5% de throughput
- **Reducao de latencia:** 99.6% (P50)
- **ROI:** Infinito (investimento zero, ganho massivo)

Projecao de Custo Mensal (730h)

- **Aplicacao:** 3 x t3.xlarge x 730h = $364.42/mes
- **Banco de Dados:** 1 x t3.xlarge x 730h = $121.47/mes
- **ALB:** ~$16/mes (estimado)
- **Total:** ~$502/mes

Evidencias e Graficos

Relatorios do Locust

Os relatorios HTML completos com todos os graficos e metricas estao disponiveis no repositorio GitHub nas seguintes pastas:

Baseline (sem otimizacoes):

```
resultados/baseline/resultados_100users_2m_20251223_133044/index.html
resultados/baseline/resultados_300users_2m_20251223_133324/index.html
resultados/baseline/resultados_600users_2m_20251223_133613/index.html
resultados/baseline/resultados_900users_2m_20251223_133902/index.html
resultados/baseline/resultados_1100users_2m_20251223_134151/index.html
```

Otimizado (com Apache mod_cache):

```
resultados/otimizado/resultados_100users_2m_20251223_182550/index.html
resultados/otimizado/resultados_300users_2m_20251223_182816/index.html
resultados/otimizado/resultados_600users_2m_20251223_183043/index.html
resultados/otimizado/resultados_900users_2m_20251223_183309/index.html
resultados/otimizado/resultados_1100users_2m_20251223_183536/index.html
```

Nota: Abra os arquivos `index.html` em um navegador web para visualizar os graficos interativos de: - RPS (Requests por Segundo) x Tempo - Latencia (P50, P95, P99) x Tempo - Distribuicao de tempos de resposta - Numero de usuarios ativos x Tempo

CSVs para Analise Detalhada

Cada pasta de resultados contem os seguintes arquivos CSV para analise programatica: - `dados_stats.csv` - Estatisticas finais agregadas - `dados_stats_history.csv` - Serie temporal completa - `dados_failures.csv` - Detalhes de requisicoes falhadas - `dados_exceptions.csv` - Excecoes e erros capturados

---

Conclusao

Principais Conquistas

1. **Performance:** Aumento de 86.5% no throughput mantendo o mesmo numero de instancias
2. **Latencia:** Reducao de 99.6% na latencia mediana (980ms → 4ms)
3. **Custo:** Mantido dentro do orcamento ($0.4992/h < $0.50/h)
4. **Confiabilidade:** 0% de erros em todos os cenarios de teste
5. **Escalabilidade:** Sistema suporta ate 1100 usuarios concorrentes sem degradacao

Licoes Aprendidas

1. **Cache e Rei:** Para aplicacoes com conteudo estatico, cache em nivel de servidor oferece o melhor ROI

2. **Configuracao Importa:** `CacheQuickHandler off` foi critico para resolver 403 Forbidden com ALB
3. **Medicao:** Testes sistematicos com Locust permitiram validacao objetiva das melhorias
4. **Trade-offs:** Tuning > Scale-out para este caso de uso especifico
5. **Experiencia Conta:** Conhecimento previo com WordPress foi fundamental para identificar a melhor estrategia de otimizacao

---

Anexos

Script de Otimizacao

O script completo de otimizacao esta disponivel em:
`optimization/apache_mod_cache_otimizado.sh`

Versao online:
https://github.com/luancaarvalho/Trab_dimensionamento/blob/main/optimization/apache_mod_cache_otimizado.sh

Documentacao Tecnica Completa

Documentacao detalhada sobre a implementacao, debugging e resultados:
`optimization/README.md`

Versao online:
https://github.com/luancaarvalho/Trab_dimensionamento/blob/main/optimization/README.md