Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática - CEEI
Departamento de Sistemas de Computação - DSC
Análise de desempenho de sistemas discretos
Professor Reinaldo Gomes

Equipe

- Acácio Fernandes Leal
- Leticia Farias Wanderley
- Martha Michelly Galvão Menezes

Projeto de Medição

1. **Planejamento**
    a. **Descrição**

Sistema de auxílio à interação entre clientes e restaurantes in loco que funciona como uma comanda digital, permitindo que os clientes realizem pedidos e paguem a conta através do aplicativo e que o restaurante gerencie os pedidos feitos pelos clientes. O sistema tem duas abordagens principais: a dos clientes, um aplicativo mobile que será ativado dentro do restaurante contendo o cardápio e as opções de realizar pedidos, dividir a conta entre os ocupantes da mesa e pagar a conta, e a dos restaurantes, um sistema web no qual a gerência poderá cadastrar pratos, receber pedidos dos clientes, atualizar os clientes com relação ao preparo dos pratos e receber o pagamento. O restaurante poderá atualizar os clientes sobre o status dos pratos pedidos no decorrer da sua preparação, as informações geradas por essas atualizações podem auxiliar o restaurante a otimizar o preparo de pratos e cardápio. O sistema tem como meta melhorar a experiência dos clientes de restaurantes desde o início até o fim da interação. Reduzindo a espera por atendimento, fornecendo informações e updates sobre pratos e eliminando o desconforto da divisão da conta.

Para que o aplicativo cumpra seu objetivo e melhore a interação entre os clientes e o restaurante é preciso que as operações nele realizadas sejam processadas o mais rápido possível. Afinal, se o sistema for lento não há vantagem em utilizá-lo em substituição da maneira tradicional de interação cliente/restaurante. O ponto no qual a velocidade de processamento e notificação do sistema é mais crítica é o da transmissão de informações sobre pedidos entre os clientes e o restaurante e vice-versa. Quando um cliente faz um pedido ele quer que o restaurante seja notificado do existência do mesmo em tempo real, quando um cliente cancela um pedido o restaurante precisa receber essa informação imediatamente para que não desperdice ingredientes em um prato que muito provavelmente não será consumido. O cliente também quer ter informações atuais sobre o estado de preparo do seu pedido, estas fornecidas pelo restaurante.

    b. **Especificação do ambiente**

O sistema servidor, que será medido, está hospedado em uma VPS (Virtual Private Server), que usa o sistema operacional Ubuntu 15.04, com as seguintes especificações:

vendor_id	GenuineIntel
cpu family	6
model	45
model name	Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz
cpu MHz	1211.684
cache size	10240 KB
cpu cores	4
address sizes	46 bits physical, 48 bits virtual
RAM	1GB

c. Métricas de avaliação de desempenho

- Tempo de resposta a requisição HTTP;
- Tempo gasto na requisições ao banco de dados.

d. Fatores e Níveis

- Tipo de requisição http
  - POST
  - PUT
- Quantidade de requisições por minuto
  - 50 requisições por minuto
  - 250 requisições por minuto
- Memória RAM disponível
  - 1 Gigabyte
  - 500 Megabytes
  - 100 Megabytes

2. Método

As medições foram realizadas através de procedimentos que arquivavam os timestamps referentes aos instantes em que as operações aconteciam, tanto as operações banco de dados quanto às operações feitas da máquina cliente, essas identificações foram todas padronizadas em timestamps no formato de segundos desde 1970 (padrão UNIX) e todas estavam com o timezone em UTC. Foi decidido utilizar este tipo de marcação por facilitar os cálculos e por ela ser mais específica.

No servidor o timestamp foi coletado no instante imediatamente anterior à chamada, ou conjunto de chamadas, ao banco e no instante imediatamente posterior. Para isso foi usada a função *microtime* da linguagem PHP, que retorna uma string representando o timestamp UNIX atual em segundos e a fração de segundo restante. No cliente os tempos de resposta foram coletados utilizando dois timestamps do sistema, um antes da requisição HTTP ser feita e outro depois. Para encontrar o timestamp foi utilizado o comando *gdate* no sistema Mac OS X. Este comando retorna o tempo UNIX atual em milisegundos. O comando *gdate* foi utilizado em detrimento do comando *date* porque este último, no sistema Mac OS X, retorna apenas o timestamp UNIX em segundos inteiros, o que, no caso desta análise, não é representativo, pois as requisições tendem a demorar frações de segundo. O comando *gdate*, por sua vez, tem em seu retorno as frações de segundo correspondentes ao timestamp atual.

Para mudar a quantidade memória disponível se utilizou, no servidor virtual, o comando linux *ulimit*, este comando permite limitar o valor de memória virtual disponível no sistema. Antes de executar os script de realização de requisições e coleta de dados a memória virtual disponível era restringida ou liberada, de acordo com o nível do fator "Memória RAM disponível" que estava sendo medido. Na figura ao lado de pode ver um exemplo do funcionamento do comando, no ambiente do servidor virtual, acessado por *ssh*.

```
rafaelclp@vps:~$ ulimit -Sv 100000
rafaelclp@vps:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 256887
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 256887
virtual memory          (kbytes, -v) 100000
file locks                      (-x) unlimited
```

Foram criados quatro scripts bash para realizar as requisições de forma automática, um fazendo 50 requisições PUT por minuto, outro fazendo 250 requisições PUT por minuto e outros dois com as mesma quantidades de requisições por minuto descritas mas fazendo requisições POST. Foram realizados ensaios para requisições de cada tipo, para cada uma das quantidades de memória RAM disponíveis e para cada uma das médias de requisições.

A quantidade de ensaios* para cada uma das situações é descrita na tabela abaixo:

	Requisição **PUT**	Requisição **POST**
50 requisições por minuto	20 ensaios	4 ensaios
250 requisições por minuto	4 ensaios	20 ensaios

*A quantidade de ensaios feita foi a mesma para os três valores de memória RAM descritos, por isso este fator não está descrito na tabela.

No total, para cada conjunto de níveis dos fatores foram feitas aproximadamente 1000 requisições ao sistema. Portanto, como existem três fatores, dois com dois níveis e um com três níveis, existem 12 combinações diferentes dos níveis e foram feitas 12000 requisições ao sistema durante todo o processo de coleta, espalhadas por 144 ensaios diferentes.

3. Instrumentação

Cada ensaio gera como saída 2 arquivos contendo logs de timestamp das operações, o primeiro com os dados de tempo de acesso ao servidor e o segundo com os dados de tempo de acesso ao banco de dados.

- Tempo de resposta a requisição HTTP

O arquivo com os dados de tempo de acesso ao servidor foi gerado no seguinte padrão:

<timestamp> <requisição HTTP>,<flag indicando coleta anterior à requisição>

<timestamp> <requisição HTTP>,<flag indicando coleta posterior à requisição>

Como se pode ver no exemplo abaixo, este exemplo foi retirado do arquivo que contém os dados das requisições POST, utilizando 1 gigabyte de memória e fazendo 50 requisições por minuto.

1498617558119031000 POST,B

1498617558398537000 POST,A

Este exemplo se trata de uma requisição post que iniciou no dia 28 de junho de 2017 às 2 horas, 39 minutos e 18.119 segundos e terminou no dia 28 de junho de 2017 às 2 horas, 39 minutos e 18.399 segundos. A flag *B* é a representação para “before”, antes da requisição e *A* para “after”, depois da requisição.

- Tempo gasto na requisições ao banco de dados

O arquivo com os dados de tempo de acesso ao banco foi gerado no seguinte padrão:

<requisição HTTP>,<flag indicando coleta anterior à requisição>,<fração de segundo> <timestamp em segundos>

<requisição HTTP>,<flag indicando coleta posterior à requisição>,<fração de segundo> <timestamp em segundos>

As linhas abaixo representam o tempo de acesso ao banco da mesma requisição tratada no exemplo acima:

POST,B,0.32125200 1498617558

POST,A,0.32245000 1498617558

No caso do tempo de acesso ao banco é comum que o timestamp inteiro seja o mesmo tanto antes quando depois do retorno do banco, isso acontece porque as operações realizadas tendem a ser muito rápidas. Nos dados acima só é necessário levar em conta as frações de segundo que se passaram entre as duas linhas. Subtraindo os dois valores decimais se tem o tempo de acesso ao banco de dados, 0.001198 segundo. Novamente, A flag *B* é a representação para “before”, antes da operação no banco e *A* para “after”, depois da operação.

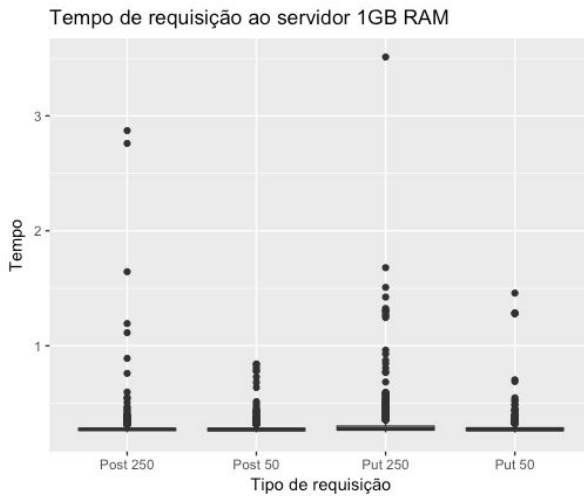
4. Resultados e Análise

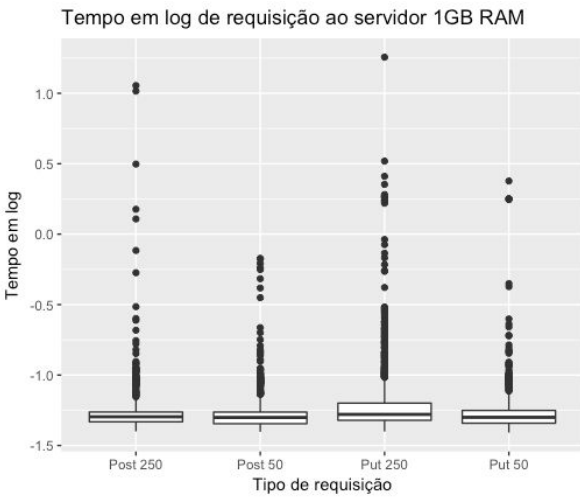
O que se procura analisar é a diferença de tempo gasto em operações que atualizam um objeto. De acordo com o protocolo HTTP atualizações devem ser resposta à uma requisição PUT, o corpo desta requisição só contém os campos que devem ser atualizados, e criações devem ser consequência de uma requisição POST, o corpo desta requisição contém um objeto completo. No entanto, é comum que desenvolvedores usem requisições POST para atualizar objetos, e é este hábito que o projeto procura analisar. Para replicar estes comportamentos no servidor a requisição PUT faz um update no banco de dados e a requisição POST remove a linha que está sendo atualizada e insere outra linha, com os dados atualizados.

Os dados coletados pelo script foram analisados com a linguagem R. A análise tem como objetivo comparar os diferentes conjuntos de níveis de acordo com a medição feita. Para isso foram gerados gráficos do tipo boxplot, estes gráficos são os mais indicados para a análise porque mostram em uma única visualização diversas métricas do conjunto de dados, como por exemplo a mediana, os quartis e os outliers. Para a construção de gráficos representativos e significativos foi necessário aplicar uma operação de normalização nos intervalos de tempo das requisições. Isso aconteceu porque, principalmente no caso do acesso ao banco, os intervalos de tempo são muito pequenos, têm valores muitos próximos a zero. Para deixar as pequenas diferenças mais visíveis utilizou-se uma escala logarítmica nos valores do eixo y, os valores de tempo.

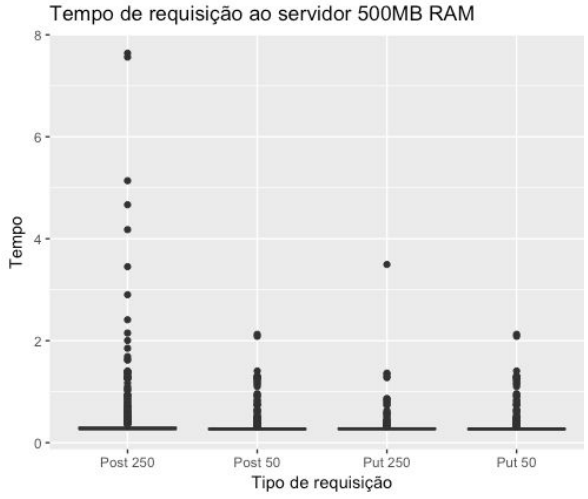
É importante também informar que durante as medições ocorreram alguns timeouts no servidor, ou seja, algumas requisições não foram respondidas pelo servidor por algum erro interno e isso causou que a requisição retornasse um timeout. Como este comportamento só ocorreu poucas vezes em poucos ensaios estes dados foram excluídos do conjunto analisado.

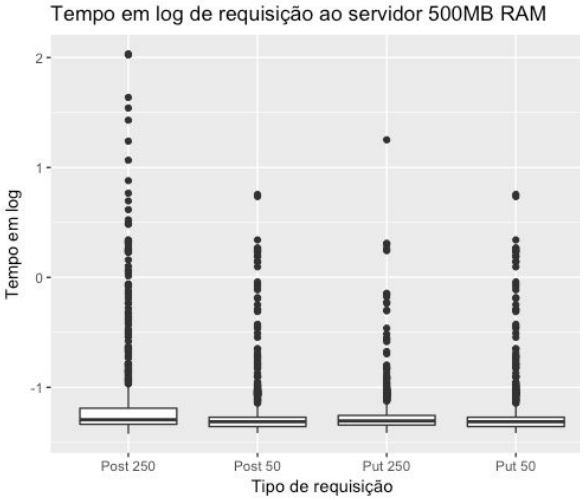
- Tempo de resposta a requisição HTTP





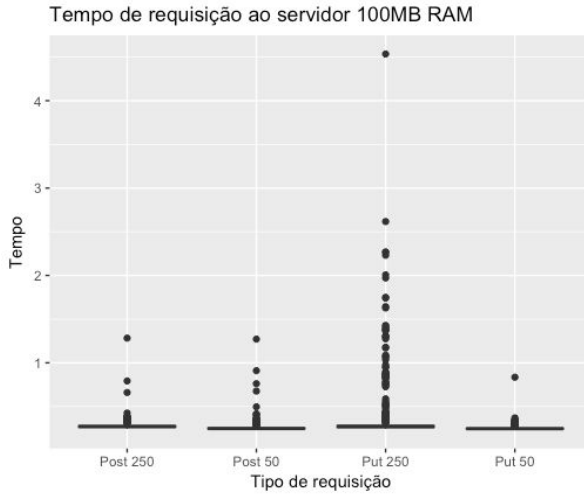
À primeira vista, o tempo de resposta a requisições HTTP não parece variar tanto quando se observa o resultado das requisições com 1 gigabyte de memória disponível. É possível ver que normalmente, as requisições demoram milésimos de segundo para terminar. Existem alguns outliers, valores que diferem muita da mediana do conjunto, nesse caso eles são sempre maiores que a mediana, chegando a atingir 3.5 segundos em uma das requisições PUT feitas na taxa de 250 por minuto. Também se pode ver que as requisições PUT feitas a uma média de 250 requisições por minuto foram as que mais variaram no tempo de resposta à requisição HTTP. Existem algumas diferenças entre os resultados da medição como a variação maior para requisições do tipo PUT, mas medianas para todos os casos são muitos semelhantes, levando em conta tanto o tipo de requisição quanto a quantidade de requisições por minuto.

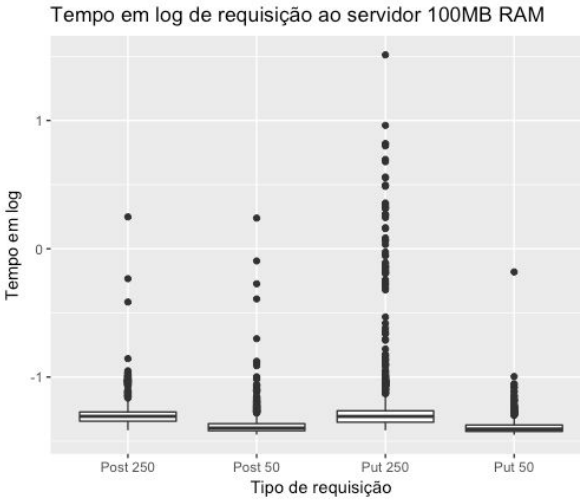




Os resultados quando existiam 500 megabytes de memória disponíveis também apresentam tempo de resposta a requisição HTTP, em torno de algumas frações de segundo. Existem alguns outliers, que chegam a atingir 7 segundos, principalmente nos ensaios que tratam de requisição POST a uma taxa de 250 requisições por segundo. Os tempos das requisições desses ensaios também foram os que mais variaram. Porém, novamente, as medianas de todos os subconjunto de níveis foi semelhante. Ao comparar os resultados para 1 gigabyte de memória e 500 megabytes se vê que o tempo de resposta

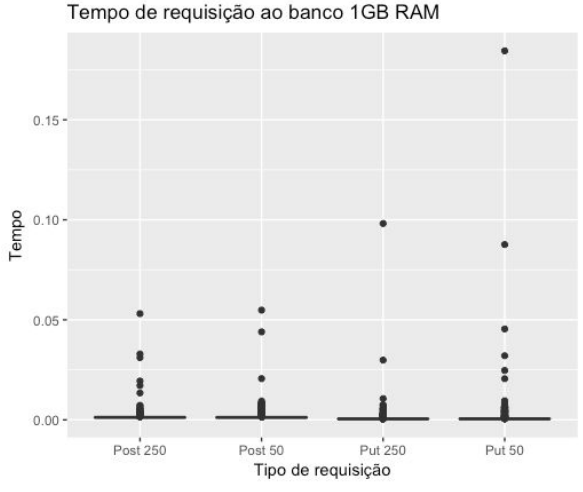
aumentou um pouco quando havia menos memória disponível. As medianas estavam em torno de 0.2 quando se tinha 1 GB e em torno de 0.35 quando se tinha 500MB.

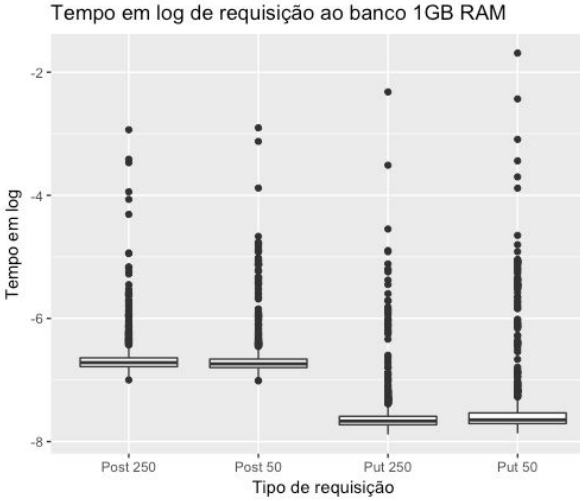




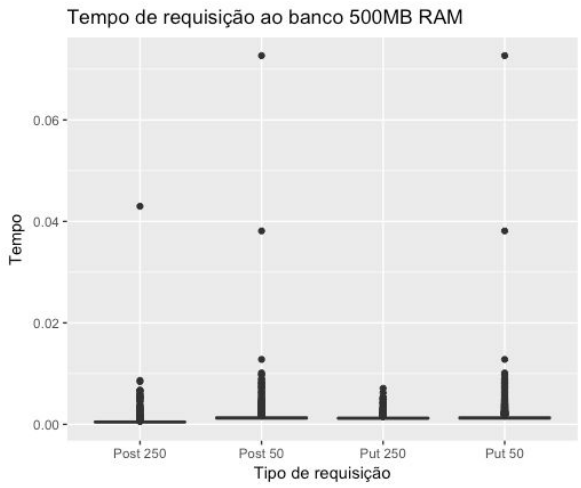
Quando havia 100 megabytes de memória disponível foi possível ver mais diferença entre as medianas dos tempos de resposta. A diferença mais visível está no fator de quantidade de requisições por minuto, nos casos, tanto de requisições POST quanto PUT em que eram feitas menos requisições por minuto o tempo de resposta mediano foi menor. Não houve muita diferenciação entre os tipos de requisição, as medianas da taxa de 250 requisições por minuto foram similares para POST e PUT, bem como para a taxa de 50 requisições por minuto as requisições de POST e PUT o tempo de resposta mediano teve valores semelhantes.

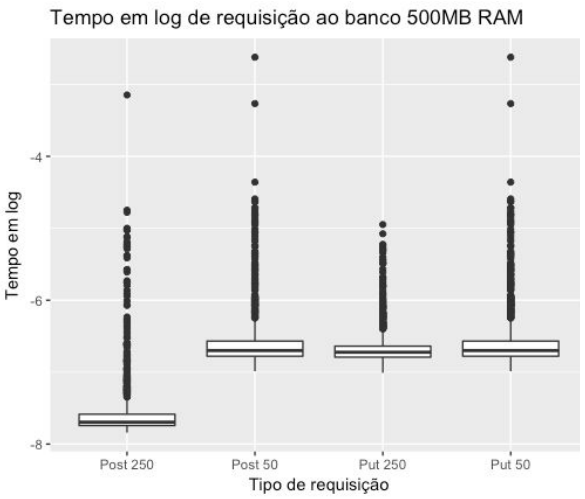
- Tempo gasto na requisições ao banco de dados



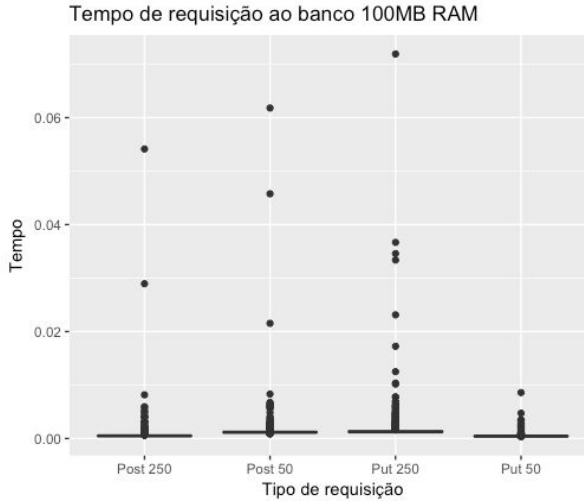


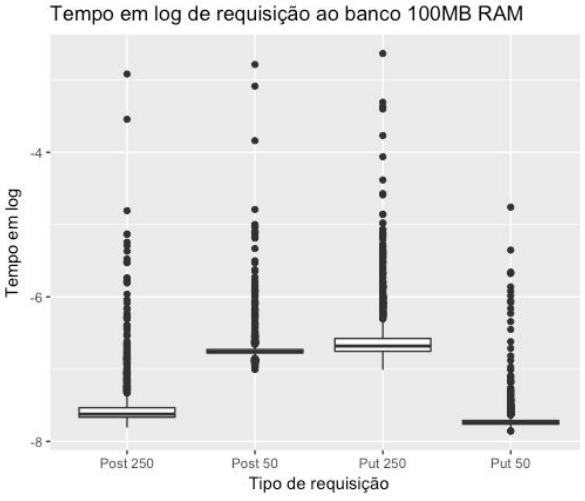
Os resultados da análise dos tempos de acesso ao banco de dados foram menos intuitivos. Quando a memória disponível era 1 gigabyte os tempos das requisições do tipo PUT, ou seja, um *update* no banco de dados, foi menor que os tempos das requisições POST, um *delete* e um *insert* no banco de dados, tanto para a taxa de 50 requisições por minuto, quanto para a taxa de 250 requisições por minuto. As medianas dos tempo gastos por requisições do mesmo tipo, sem levar em conta a taxa de requisições por minuto, foi semelhante e teve valores muito baixos, alguns poucos milissegundos.





Ao limitar o uso de memória para 500 megabytes aconteceram algumas situações estranhas. As medianas dos tempos de acesso ao banco de dados das requisições PUT feitas a taxa de 50 e 250 requisições por minuto e das requisições POST feitas a taxa de 50 requisições do minuto foram muito semelhantes e maiores que a mediana do tempo de acesso ao banco das requisições POST feitas a 250 requisições por minuto. Isso pode ter acontecido devido a algum mecanismo de cache próprio do banco de dados. Porém, é preciso investigar mais a fundo para ter uma explicação concreta. Como para quando se utilizou 1 gigabyte de memória, as requisições ao banco demoraram alguns milissegundos.





Limitando o uso de memória para 100 megabytes o tempo mediano de resposta ao acesso ao banco não variou muito do tempo mediano dos outros valores de memória. Nesse caso os conjuntos de fatores que gastaram mais tempo para acessar o banco de dados foram os de requisições POST a 50 requisições por minuto e requisições PUT a 250 requisições por minuto. Sendo as requisições PUT a 250 as que mais variaram no tempo de resposta. O conjunto de tipo de requisição e taxa de requisições que teve a menor mediana de tempo de resposta foi a requisição PUT a uma taxa de 50 requisições por minuto. As requisições POST a uma taxa de 250 requisições por minuto tiveram um resultado atípico similar ao visto quando se usou 500 MB de memória.
Os scripts e os resultados da medição podem ser encontrados neste repositório.