Página 29

1. Projeto de benchmarks: Desenvolva quatro benchmarks para testar E/S, computação e duas combinações. Uma visão geral do método pode ser descrita pelas seguintes etapas: entre computação e entrada/saída.

2. Criar servidores: Crie três servidores usando threads de plataforma, threads virtuais e fluxos reativos. Todos os servidores implementarão os quatro benchmarks.

3. Experimentos de Aumento Gradual de Carga: Nos experimentos de aumento gradual de carga, uma máquina cliente enviará solicitações simultâneas a um servidor que hospeda uma das três tecnologias, para um dos quatro benchmarks. O nível de concorrência é determinado pelas solicitações por segundo e aumentará linearmente até que ocorra a saturação de recursos no servidor. Durante esses testes, as métricas do cliente e do servidor serão coletadas (como uso de CPU e latência).

4. Experimentos com Carga Constante: Semelhantes aos experimentos de aumento gradual de carga, exceto que agora a taxa de requisições é mantida constante em um nível correspondente a cerca de 70% de utilização de recursos (seja qual for o fator limitante para o benchmark em questão). Além disso, utilizamos uma única série de testes em vez de cinco, pois a variância foi muito menor do que para o aumento gradual de carga em nosso ambiente de teste.

5. Validação e Experimentos Adicionais: Realize experimentos de validação e adicionais para explicar os resultados e validar o trabalho. Isso inclui a criação de perfis de pilha, a análise da conexão de rede e a avaliação do impacto do software de criação de perfis.

Página 30

Teste de E/S - atraso fixo de 100 ms.

Página 31

Antes de cada iteração, a máquina atacante enviará uma solicitação HTTP que força o servidor a realizar uma coleta de lixo.

Página 32

O aquecimento é realizado executando um teste de carga prolongado com o mesmo método de teste antes de cada iteração.

O aquecimento é realizado executando um teste de carga prolongado com o mesmo método de teste antes de cada iteração. O aquecimento também visa forçar a compilação JIT. A máquina que executa o servidor possui um contador de limite para compilação JIT igual a 10.000 (acessado pelo comando `java -XX:+UnlockDiagnosticVMOptions -XX:+PrintFlagsFinal -version`), o que significa que um método pode ser compilado após ser chamado esse número de vezes.

Assim, as especificidades do aquecimento (número de requisições) serão projetadas para chamar os métodos mais de 10.000 vezes com margem de segurança. Outra opção potencial seria

desabilitar o compilador JIT. No entanto, o compilador JIT é parte integrante do Java e não é recomendável desabilitá-lo. Além disso, o compilador JIT provavelmente estará sempre em execução durante condições semelhantes às deste teste experimental (servidores), tornando o estudo mais realista com o compilador JIT habilitado.

Minha máquina: 10000

Máquina virtual: ??? Executar `java -XX:+UnlockDiagnosticVMOptions -XX:+PrintFlagsFinal -version | grep CompileThreshold`

Média de cinco testes

Monitoramento da rede entre as máquinas, a manutenção de um ambiente de teste consistente e estável e a padronização dos testes.

Página 33

uas máquinas idênticas (MacBook Pro 2019) com as seguintes especificações: processador Intel Core i7 de 6 núcleos a 2,6 GHz, memória de 16 GB DDR4 de 2667 MHz e macOS 14.2.1

Uma máquina hospedava um servidor e coletava dados de perfil, enquanto a outra realizava testes de carga por meio do Vegeta. As máquinas estavam conectadas por um cabo Ethernet de um gigabit.

A máquina cliente coleta as seguintes métricas: latência (mínima, máxima, média, percentis 50, 90, 95 e 99), taxa de transferência, taxa de amostragem, respostas do servidor (para taxa de sucesso), bytes de entrada e bytes de saída. Isso é feito através do Vegeta.

O servidor coleta métricas referentes à utilização da CPU, número de threads (ativas e iniciadas) e heap (tamanho total e tamanho utilizado)

Tanto a máquina atacante quanto o servidor foram monitorados em busca de erros (impressos no terminal ou nos arquivos de saída), que ocorreram apenas sob cargas elevadas ou em momentos em que os recursos (CPU, heap ou threads) estavam muito sobrecarregados.

Página 34

O comprimento e as taxas de requisição usados no aquecimento foram calculados para chamar o servidor mais de dez mil vezes com alguma margem de erro, já que até esse número a JVM poderia executar compilações JIT (como explicado no capítulo de contexto).

- Realizamos cada série de testes cinco vezes e calculamos as médias. À medida que cada série de testes era repetida, obtínhamos uma média de cinco repetições.
- Um teste foi interrompido após três requisições consecutivas com falha. Após esse evento, as latências foram de vários minutos, o que está fora do escopo deste estudo, pois estamos interessados em casos de uso práticos.
- Levou várias horas, fazer mais não era viável.
- Atrasos de 60 segundos antes da coleta de lixo e 20 segundos depois. Essa configuração foi definida empiricamente por meio do estudo de séries temporais do VisualVM, escolhendo

valores que permitissem que a utilização da CPU e o uso de memória retornassem aos valores nominais entre os testes.

Página 35

Aquecimento:

1. 60 segundos com 300 solicitações por segundo
2. Coleta manual de lixo
3. Operação em modo de espera por 20 segundos
4. Repita três vezes com taxas ajustadas para o método específico

As três taxas de aquecimento posteriores são ajustadas para 70% da taxa máxima do método.

Ataques:

1. Envie solicitações HTTP com a taxa 'RATE' durante 10 segundos. Salve os resultados
2. Operação em modo de repouso por 60 segundos
3. Coleta manual de lixo
4. Durma por 20 segundos
5. Aumente a 'TAXA' em 25/50 e repita

Página 36

Perfilamento de Pilha:

Para investigar as pilhas de chamadas e obter uma visão geral da sobrecarga nessas três tecnologias, foram utilizados FlameGraphs.

O procedimento de teste seguiu os mesmos protocolos apresentados na seção anterior, incluindo extensos aquecimentos. O teste em si, no entanto, foi mais curto (com 240 segundos).

As taxas foram ajustadas manualmente com o objetivo de não sobrecarregar os servidores durante os testes.

Página 37

Conexão de Rede:

Quando os testes de carga foram realizados neste estudo, eles foram executados até que o servidor falhasse ou a conexão de rede fosse interrompida.

Impacto do Software de Criação de Perfil:

É uma lei fundamental da natureza que, ao observar algo, você o afeta (Heisenberg, 1927).

1 Atenção

Scripts no appendix A, página 77

Este script foi usado no cliente para enviar solicitações HTTP com Vegeta e salvar os resultados de forma organizada em um arquivo.