Simulador De Scale Set

Nome: Gabriela Chavez Estevez nºUSP: 10295440
Nome: Lui Franco nºUSP: 10295558
Nome: Óliver Savastano Becker nºUSP: 10284890

Resumo

A partir do tema sorteado em aula, a equipe Cabeça nas Nuvens decidiu adotar como problema a ociosidade de máquinas virtuais de servidores em nuvem. Para resolvê-lo, adotou-se como solução o conceito de Scale Set, que propõe o gerenciamento das *VMs* ativas, para evitar a ociosidade e gastos desnecessários. Deste modo, o projeto apresenta um simulador de Scale Set, que demonstrará que o uso deste contribui com a redução de custos e ociosidade das *VMs*.

Introdução

O trabalho que será desenvolvido ao longo deste documento tem como objetivo apresentar a elaboração de um Simulador de Scale Set. Através deste, será demonstrado como o código funciona, como executá-lo e como a comparação de seus resultados evidencia a redução de custos, ao aplicar o conceito de Scale Set.

Em servidores na nuvem, máquinas virtuais são utilizadas para atender a demanda de processamento do servidor em questão, podendo este ser usado de diversos modos, como servidor web, de arquivos, de mídia, entre outros. Para servidores com o número de requisições variáveis ao longo do dia, quando a demanda de requisições é baixa, torna-se dispensável que haja uma alta quantidade de *VMs (Virtual Machines)* ativas. Levando tal cenário em consideração, como se paga uma certa quantia por máquina virtual a cada hora, o serviço pode sair caro de modo desnecessário, pois haverá máquinas virtuais ligadas que não estão sendo utilizadas. Para resolver o problema, reduzindo o custo, propomos que um Scale Set seja aplicado ao servidor.

O projeto foi pensado a partir do tema inicial, “Virtualização e a Nuvem”, com o intuito de resolver o problema da ociosidade de máquinas virtuIasde servidores em nuvem que possuem elevada variabilidade de requisições. O Simulador controlará a quantidade de máquinas virtuais ativas de acordo com o fluxo de requisições de um servidor de arquivos, ativando ou desligando máquinas virtuais de acordo com a demanda de processamento.

Um servidor de arquivos tem como fim armazenar diversos tipos de arquivos, como planilhas, apresentações, documentos, entre outros tipos de arquivos. O código que será apresentado, desenvolvido em C, simulará como um servidor de arquivos em nuvem funcionaria com o Scale Set, gerenciando o uso das *VMs* responsáveis pelo processamento. O servidor será encarregado de requisições como upload e download de fotos, sendo estas os dois tipos de requisição que este poderá receber.

Problema

Diante do crescente uso de servidores em nuvem para processamento de dados e requisições, somado ao elevado custo de manter as máquinas virtuais ativas, é possível notar que a ociosidade dessas máquinas, durante períodos de baixas requisições, gera grandes gastos desnecessários. Logo, percebe-se que a ociosidade das *VMs* é muito desvantajoso e que é preciso um algoritmo para controlar seu uso.

Abordagem

Para solucionar o problema, foi utilizada a abordagem Scale Set. Esta aproximação gerencia o número de *VMs* ativas de acordo com a demanda de processamento, isto é, ajusta a quantidade de *VMs* proporcionalmente à quantidade de requisições recebidas. Desta forma, durante os períodos de baixas requisições, as máquinas que estariam ociosas são desativadas, evitando o gasto desnecessário de mantê-las ativas.

Desenvolvimento

1. Etapas para a realização do projeto

Para desenvolver o projeto, várias etapas foram seguidas. Primeiro, os integrantes do grupo leram o capítulo proposto, do livro Sistemas Operacionais Modernos, de Tanenbaum. Em seguida, concluiu-se que um problema interessante de se resolver seria a ociosidade de *VMs* em servidores em nuvem, pois isso aumenta o custo do servidor. Para solucionar o problema, analisamos que se um Scale Set fosse aplicado ao servidor em nuvem, a questão seria resolvida de forma eficiente. Foi, então, iniciada a implementação de um simulador de Scale Set.

2. Descrição dos métodos, bibliotecas e linguagem utilizadas

A linguagem utilizada para a implementação do projeto foi a linguagem C, sem o uso de bibliotecas externas. A entrada do código deverá conter as seguintes informações: servidor com Scale Set (representado pelo valor 0) ou sem Scale Set (representado pelo valor 1), número inicial de *VMs* ativas, poder de processamento de cada *VM* e custo por *VM* em reais. Caso o primeiro dado seja 0, indicando que o servidor em nuvem terá o Scale Set aplicado, a entrada precisará conter as seguintes informações a mais: limite inferior de processamento, limite superior de processamento, número máximo de *VMs* removidas (se o limite inferior for atingido), número máximo de *VMs* adicionadas (se o limite superior for alcançado). Além disso, o programa também receberá valores aleatórios, gerados de modo controlado, que simularão as requisições.

O simulador executa em um loop, fazendo a leitura de um arquivo de entrada, o qual simula requisições ao servidor. Tais requisições são representadas por números inteiros, que correspondem à quantidade de download ou upload de arquivos no servidor, em um dado instante de tempo. Ainda no loop, o dinheiro gasto é contabilizado, de acordo com o custo por *VM* definido no *input*. O Scale Set pode ou não estar ativo no servidor em questão. Se não estiver ativo, o servidor rodará com um número fixo de máquinas virtuais.

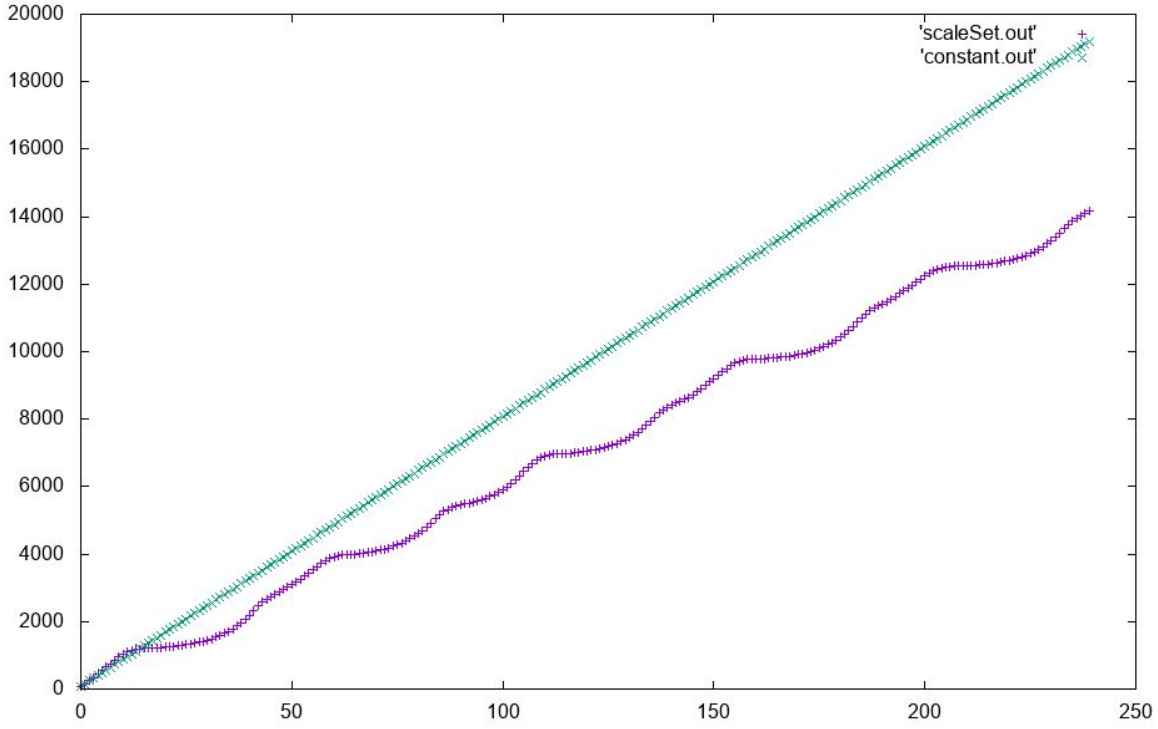
No caso do Scale Set ativo, checa-se a necessidade de adicionar ou remover *VMs* baseado no processamento demandado pelas requisições que chegam. A cada ciclo, verifica-se o processamento médio (razão entre a demanda de processamento das requisições e a soma do poder de processamento de cada VM ativa) do servidor no momento, analisando se tal número está entre os limites inferior e superior (sendo estes os limites passados como parâmetros do programa). Caso o processamento médio esteja abaixo do inferior, mostra que a demanda está pequena para a quantidade de máquinas ativas, indicando a possibilidade de desligar algumas destas, economizando dinheiro, sem comprometer a qualidade do serviço. Portanto, máquinas serão desativadas até que a razão esteja acima do limite inferior. Já no caso de processamento médio estar acima do limite superior, ocorrerá um processo análogo, porém ligando as *VMs* ao invés de desligá-las, de modo que o servidor não se sobrecarregue e consiga atender à demanda.

3. Resultados obtidos

Para realizar a comparação, usando o simulador, seguimos algumas diretrizes, e estas foram: mesma entrada (mesmo *input* de requisições), mesmo estado inicial (número inicial de máquinas virtuais, custo por hora das máquinas e poder de processamento das *VMs*). Além disso, houve a necessidade de discretizar o tempo, pois não haveria como simular, nestas condições, com um tempo contínuo. Para tanto, o processamento é separado em ciclos (os quais são denominamos como uma hora, mas poderiam representar qualquer instante de tempo).

Como resultado da simulação, obtivemos os gastos com a manutenção das máquinas virtuais ativas durante cada ciclo da iteração, para os dois casos de simulação (utilizando o Scale Set e quantidade constante de *VMs*). Tais custos podem ser vistos no Gráfico 1 abaixo.

Gráfico 1 - Resultados da simulação com Scale Set e *VMs* constante



Fonte: autoria própria.

No gráfico acima, o eixo vertical refere-se ao dinheiro gasto, em unidade monetária qualquer, para manter as máquinas ligadas. Já o eixo horizontal representa o tempo, medido em horas.

A partir da análise do gráfico, é possível notar que a que o servidor com Scale Set (pontos roxos) apresenta um custo menor de manutenção, quando comparado com o servidor que possui uma quantidade fixa de *VMs* (pontos azuis). Isto era o esperado, pois quando as requisições de uploads e downloads estão em baixa, o Scale Set reduz o número de máquinas virtuais ligadas, reduzindo os gastos com estas durante o momento estudado.

Conclusão

Durante o desenvolvimento do software, percebemos que a discretização do tempo acabou impactando bastante na precisão dos resultados, visto que acabou excluindo muito da dinamicidade encontrada nos servidores reais. Além disso, a operação de ligar e desligar máquinas teve de ser tratada como algo simples e imediato, algo que não retrata muito bem a realidade.

As maiores dificuldades foram, certamente, tentar reduzir ao máximo o impacto da discretização do tempo e a criação do arquivo de entrada, o qual contém a quantidade de requisições enviadas ao servidor. Isto porque, nos servidores reais, as requisições, por acontecerem de modo constante, costumam chegar com uma variação não muito grande, dando tempo para o Scale Set perceber a queda (ou aumento) no processamento de dados e ativando (ou desativando) as máquinas virtuais de uma maneira mais suave.

Para tentar criar um arquivo de entrada coerente e com variações durante as horas do dia (com maior quantidade de requisições durante a parte da tarde, e menos na madrugada), foi desenvolvido um outro programa para gerar tais números. A maneira utilizada para gerar a variação de horário foi criar, manualmente, um fator que irá multiplicar a quantidade máxima de processamento esperada para cada hora, além de multiplicar novamente por um pequeno número aleatório, a fim de alterar um pouco este resultado obtido, permitindo que a quantidade de requisições não permaneça sempre a mesma em horários iguais.

O software final está todo no arquivo “main.c”. Após compilado, o programa resultante poderá simular um servidor de imagens que utiliza Scale Set ou uma quantidade constante de máquinas virtuais, de modo a permitir uma comparação de custos entre estes dois cenários. A seleção e configuração dos modos é feita durante a execução do programa, pela *stdin*. Já a quantidade de requisições enviadas ao servidor em um instante de tempo está guardada no arquivo “input.in”, o qual é lido durante a execução do programa.

Referências

TANENBAUM, A.S. Sistemas Operacionais Modernos, tradução Ronaldo A. L. Gonçalves, Luís A. Consularo, Luciana do Amaral Teixeira, revisão técnica Raphael Y. de Camargo, 3ª edição, 2010. Pearson.

Anexo 1:

No arquivo compactado entregue há, além deste relatório, uma pasta chamada “simulacao”, e a funcionalidade dos arquivos lá contidos está descrita a seguir:

- “main.c”: código principal, no qual encontra-se implementado o simulador de servidor de *VMs*.
- “scaleSet.config” e “constant.config”: possui os parâmetros de configuração para execução do simulador. Estes serão lidos assim que o programa iniciar.
- “Graph.png”: imagem com o gráfico, já mostrado neste documento, proveniente de simulações já realizadas.
- “input.in”: arquivo de entrada do programa, contém o número de requisições enviadas ao servidor (quantidade de uploads e downloads de imagens do servidor) em um instante de tempo.
- “inputGenerator.c”: implementação do gerador do arquivo de input. Possui uma macro chamada ‘RAND_SEED’ que pode ser alterada para gerar números diferentes no arquivo “input.in”.
- “makefile”: arquivo para auxiliar na compilação e execução do simulador. Suas principais diretivas são: all - compila os códigos e gera os executáveis; plot-graph - utiliza o programa ‘gnuplot’ para criar o gráfico com os resultados das simulações e o salva no arquivo “Graph.png”; clear - remove os arquivos temporários.

Já os seguintes arquivos podem ser gerados a partir do makefile:

- “main”: programa que simula o servidor de mídia, podendo ser configurado para executar com uma quantidade fixa de *VMs* ou com um Scale Set, para controlar o uso das máquinas.
- “inputGenerator”: programa que gera o arquivo “input.in”. Para executar, basta digitar “./inputGenerator” e o input será atualizado.

- “scaleSet.out” e “constant.out”: arquivos de saída da simulação. Contém a quantidade acumulada de dinheiro gasto em cada momento. Estes serão os arquivos utilizados para plotar o gráfico.

- “.log”: arquivo temporário que contém os status da simulação em cada iteração.

Para realizar a simulação, é necessário executar o comando “**make**”, seguido de “**make plot-graph**” (certificando-se de estar em um sistema Linux com o programa gnuplot devidamente instalado). Em seguida, o arquivo “Graph.png” será criado e pode ser aberto para visualizar a quantidade de dinheiro gasto (eixo Y) de acordo com o tempo decorrido (eixo X), para os dois casos de simulação.