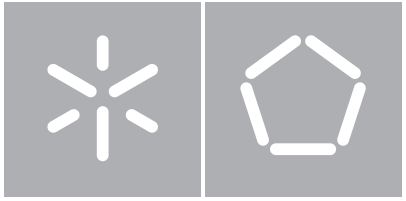
Universidade do Minho
Escola de Engenharia

João Miguel Barbosa Gomes

Modelo de simulação de discos

Junho de 2013



Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Miguel Barbosa Gomes

Modelo de simulação de discos

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de
Professor José Orlando Roque Nascimento Pereira

Junho de 2013

Agradecimentos

Quero agradecer a todos os meus amigos e familiares que contribuíram indiretamente para a realização da dissertação. Em especial quero agradecer:

Ao professor e orientador José Orlando Pereira por toda a disponibilidade e ajuda que demonstrou durante toda a elaboração da dissertação, e pelos conselhos muito úteis que me foi dando durante o desenvolvimento da mesma.

Aos meus colegas de casa André Pimenta e João Brandão por todo o apoio e companheirismo que mostraram durante este último ano.

A todas as pessoas que me acompanharam em todo o percurso académico na universidade do Minho, especialmente aos que iniciaram esse mesmo percurso comigo.

Por fim, quero agradecer aos meus pais Maria José Barbosa e João Gomes por toda a dedicação, apoio e incentivo que me deram durante todo o meu percurso académico e de vida, e também à minha irmã Joana Gomes pelo constante apoio demonstrado.

Resumo

A simulação de componentes é uma importante ferramenta para o auxílio no desenvolvimento de sistemas, realização de testes e uma melhor compreensão acerca desses mesmos componentes por parte de investigadores e desenvolvedores. Esta pode ser realizada utilizando diferentes abordagens, mas tem de permitir uma reprodução fiável do ambiente.

A presente dissertação assenta sobre uma plataforma já existente, o Minha. Esta plataforma permite simular sistemas distribuídos e é capaz de simular todas as interações entre várias máquinas ao nível da rede. Embora a plataforma seja capaz de realizar a simulação ao nível da rede, esta não era capaz de realizar qualquer simulação ao nível dos discos das máquinas simuladas, até à realização do dissertação. É este o problema que a presente dissertação se propõe resolver, criando um módulo que realize a intercepção das operações sobre o disco e que trate as mesmas de forma a simular a existência de um disco independente para cada uma das máquinas simuladas.

Esta dissertação tem como objetivo dotar a plataforma de um novo módulo que permita que a mesma consiga simular sistemas que necessitem de recursos do disco, como bases de dados. Até à realização da dissertação a plataforma não fazia qualquer controlo sobre os recurso requeridos do disco, o que provocava resultados de simulação inconsistentes devido à partilha não controlada do disco da máquina onde a simulação era realizada.

O modelo de simulação apresentado é validado experimentalmente com um micro-benchmark e com TPC-B sobre a base de dados HyperSQL.

De realçar que o resultado da dissertação em questão já se encontra integrado na plataforma e disponível no repositório oficial da plataforma Minha que se encontra alojado em *http://code.google.com/p/minha/*.

Palavras-chave: Simulação, Discos, Sistemas Distribuídos

Abstract

The simulation components is an important tool to support the development of systems, testing and a better understanding of those components by researchers and developers. This can be accomplished using different approaches, but must allow faithful reproduction of the environment.

This dissertation is based on an existing platform, Minha. This platform allows simulate distributed systems and is capable of simulating all the interactions between multiple machines at the network level. Although the platform is able to perform the simulation at the network level, this was not able to perform any simulation to the level of the disks of machines simulated until completion of the dissertation. Is this the problem that this dissertation proposes to solve by creating a module that performs the interception operations on the disk and that treat the same in order to simulate the existence of an independent disk for each of the simulated machines.

This dissertation aims to provide a platform for a new module that allows simulate systems that require disk resources, such as databases. Until completion of the dissertation platform made no control over the resources required from disk, which caused inconsistent simulation results due to uncontrolled sharing disk of the machine where the simulation was performed.

The simulation model is validated experimentally presented with a micro-benchmarks and TPC-B on the HyperSQL database.

Note that the result of the dissertation is already integrated into the platform and available in the official repository of Minha platform that is available in http://code.google.com/p/minha/.

Key Words: Simulation, Disks, Distributed Systems

Índice

Agradecimentos i

Resumo iii

Abstract v

Índice viii

Glossário x

Lista de Figuras xi

1 Introdução 1

1.1 Objetivos e Contribuição 2

1.2 Estrutura 3

2 Background 4

2.1 Simulação em Sistemas Distribuídos 4

2.2 Plataforma Minha [2] 4

3 Estado da Arte 8

3.1 DiskSim 8

3.2 Vesper 9

3.3 GemDisk 9

3.4 HP 97560 Disk Drive 10

4 Problema 13

4.1 Assunções 14

4.2 Abordagem 14

4.3 Validação 15

5 Modelo de simulação 17

5.1 Assunções 17

5.2 Sistema de ficheiros 18

5.3 Dispositivos de armazenamento 20

6 Calibração e avaliação 27

6.1 Caracterização do desempenho 27

6.2 Resultados 31

6.3 Caso de estudo (HSQL) 33

7 Conclusão 37

8 Referências 39

Glossário

SSD	Solid-State Drive
JVM	Java Virtual Machine
HSQLDB	Hyper Structured Query Language DataBase
IP	Internet Protocol
API	Application Programming Interface
I/O	Input/Output
MB	Megabyte

Lista de Figuras

Figura 1 - Ficheiro de configuração 7

Figura 2 - Estrutura do GemDisk[5] 10

Figura 3 - Classe File Fake 19

Figura 4 - Exemplo de configuração 21

Figura 5 - Adicionar pedido 24

Figura 6 - Tratamento de pedidos 26

Figura 7 – Execução sobre o simulador 32

Figura 8 – Execução sobre o sistema 32

1 Introdução

O desenvolvimento de sistemas cada vez mais elaborados e complexos é uma constante nos dias de hoje. Quando um sistema exige velocidade e alto desempenho surge a necessidade de o otimizar o máximo possível, e uma preocupação com pormenores que vão para além da parte lógica dos sistemas. Neste contexto surge a necessidade de se adaptar e desenhar o sistema tendo em atenção o hardware onde o mesmo será implementado de forma a tirar um maior partido das suas características. Cada vez mais a variedade de hardware existente e a velocidade com que é atualizado torna muito custoso, do ponto de vista económico e de tempo, executar testes sobre vários discos e arquiteturas diferentes com o objetivo de perceber qual a mais adequada às necessidades do sistema. Esta foi a motivação para o aparecimento dos primeiros simuladores de hardware, permitindo ter numa única máquina um ou mais componentes simulados, com características de um componente real. Desta forma tornou-se possível efetuar testes sobre diferentes componentes de forma a perceber qual a configuração que melhor se adapta ao sistema. O aparecimento de simuladores de hardware tornou possível o teste de várias otimizações, tanto ao nível de algoritmos como de configurações de hardware, de forma a maximizar o desempenho dos sistemas. Estas ferramentas vieram dotar os desenvolvedores de inúmeras possibilidades de configuração de hardware num único ambiente, de uma forma simples e rápida, que facilita o desenvolvimento de sistemas.

A simulação de componentes tem sido estudada e aperfeiçoada ao longo de vários anos. Os simuladores de hardware têm aumentado quer em termos de quantidade quer em variedade, sendo que a qualidade que estes oferecem em termos de realismo tem vindo a ser cada vez maior. O aumento da utilização de dispositivos e serviços web levou ao crescimento dos sistemas, tornando-os cada vez mais complexos e elaborados. Este desenvolvimento levou a que os sistemas deixassem de se restringir a uma só máquina passando a distribuir a sua carga por várias máquinas de forma a obter um maior poder

de processamento e uma maior redundância (sistemas distribuídos). Com esta nova mudança, os simuladores até antes usados, deixaram de conseguir proporcionar uma facilidade tão grande ao desenvolvedor pelo facto de não serem capazes de simular um ambiente distribuído que era agora exigido. Esta nova metodologia de desenvolvimento de sistemas levou ao problema inicial, que se prende com o facto de fazer testes sobre os sistemas e os problemas em termos de tempo e custo que estes testes implicam. Foi neste contexto que surgiram simuladores mais orientados a testes e suporte na resolução de problemas em sistemas distribuídos, passando a permitir simular um ambiente distribuído numa única máquina. A plataforma Minha é um exemplo disso, permitindo fazer a simulação da rede e das interações entre várias máquinas simuladas.

1.1 Objetivos e Contribuição

A plataforma Minha tem como objectivo a simulação de uma rede de computadores. Tal como uma rede real, cada máquina tem os seus próprios componentes de rede, processador, discos, etc. Atualmente a plataforma Minha já é capaz de realizar a simulação de uma rede de computadores, do ponto de vista do reencaminhamento dos pacotes para os diferentes nós, da velocidade de transmissão, e permite também simular perdas e atrasos dos pacotes que nela circulam, basicamente tudo que está relacionado com a rede. O mesmo não acontece no que toca às operações sobre o disco, visto não ser capaz de realizar qualquer simulação sobre este contexto, deixando as máquinas simuladas sem um dispositivo de armazenamento próprio que permita independência desta em relação às restantes, coisa que seria esperada.

Atualmente utiliza o disco da máquina onde está a ser executada para servir todas as máquinas que estão a ser simuladas sendo as escritas e leituras de disco feitas todas para o mesmo componente de armazenamento. Sem simulação ao nível do disco, todas as máquinas são simuladas tendo os componentes de rede e processador próprios mas realizam todas as operações sobre um mesmo disco partilhado, influenciando o tempo de execução de cada uma. Este é problema que esta dissertação aborda.

A presente dissertação tem como objectivo desenhar e implementar um modelo de discos para ser embebido na plataforma Minha de forma a que esta se torne mais completa

e que consiga apresentar resultados o mais realistas possível, através da virtualização do dispositivo de armazenamento em cada máquina. Não sendo o objetivo principal da dissertação, esta completa também a plataforma de mais classes e métodos que ainda não existam e sejam necessários para o trabalho.

1.2 Estrutura

Ao longo dos próximos capítulos é dada uma visão mais detalhada sobre simulação, uma visão resumida da plataforma Minha e o seu problema da simulação no que toca às operações sobre discos, que é o alvo da presente dissertação. São ainda analisados alguns dos projetos existentes atualmente capazes de simular o armazenamento em disco, sendo feita a análise crítica desses mesmos projetos de forma a aproveitar o melhor de cada um com vista à resolução do problema. São também expostas algumas assunções necessárias à construção do modelo e a forma como é feita a sua implementação, tanto ao nível do código que é introduzido na plataforma, como do software auxiliar para a extração de tempos bem como os testes realizados. Em seguida é explicada a forma como a implementação é validada de forma a garantir que a solução proposta para o problema é realmente capaz de o resolver. Visto isto, é explicada a abordagem ao problema e as decisões tomadas durante o desenvolvimento.

É explicado o funcionamento do sistema de ficheiros que suporta o armazenamento dos vários nós virtuais e a forma como a informação é guardada e organizada. É também analisada e clarificada a forma como os componentes do disco estão a ser simulados em termos de software. Com o intuito de facilitar a extração de informação de forma válida de outros discos a serem simulados foi criado um programa que terá como objetivo a extração da informação de discos, de forma a ser simulado na plataforma. É então analisada a forma como a informação é extraída dos discos através do programa de extração, programa esse que é usado também para validar o modelo. Por fim, é utilizado um caso de estudo de forma a mostrar a aplicabilidade e fiabilidade do simulador em problemas reais.

2 Background

2.1 Simulação em Sistemas Distribuídos

Durante a última década os sistemas distribuídos tiverem um enorme crescimento. No início, toda a computação era realizada apenas num único nó, onde os programas executavam recorrendo apenas a recursos existentes na mesma máquina onde estavam alocados. Com o passar dos anos houve uma maior necessidade de aumento dos recursos computacionais. Esta necessidade deveu-se em grande parte à massificação da utilização da internet visto que as máquinas que disponibilizavam recursos na rede estarem cada vez mais sobrecarregadas devido ao enorme aumento de utilizadores da rede. Foi neste contexto que apareceram os primeiros sistemas distribuídos. Estes sistemas executam em várias máquinas e comunicam entre si de forma a cooperarem com o objetivo de aumentar o seu poder computacional.

Com o aparecimento dos sistemas distribuídos surgiram também novos problemas, tais como a concorrência de acessos e a sincronização de dados. O desenho dos sistemas, inicialmente centralizados, também sofreu alterações de forma a tirar partido desta nova metodologia. Por vezes, a implementação destes sistemas continha alguns problemas que apenas se tornavam evidentes quando executavam em grande escala, problemas estes que não estavam previstos nos modelos. A deteção de alguns destes problemas tornava-se cada vez mais complicada devido à necessidade de executar o sistema em várias máquinas de forma a serem testados em ambientes reais, o que trazia elevados custos, tanto a nível de tempo como a nível financeiro. Foi com vista ao auxílio no desenvolvimento destes sistemas que apareceram os primeiros simuladores capazes de recriar o ambiente de um sistema distribuído. Estes permitiam recriar as características de uma rede e das suas máquinas numa única máquina para que se pudesse simular o sistema de uma forma distribuída numa única máquina, permitindo uma melhor observação do seu comportamento. Uma das plataformas que realiza este tipo de simulação é o Minha[1], projeto que está na base desta dissertação.

2.2 Plataforma Minha [2]

O Minha é uma plataforma que permite simular a rede entre várias máquinas de forma a tornar possível a análise do comportamento e interações entre elas. Antes do aparecimento da plataforma Minha, realizar um teste numa única máquina passava pela execução do sistema em várias JVM/s estabelecendo a comunicação entre elas localmente. Esta poderia ser uma solução mas a quantidade de memória gasta, a partilha dos componentes, de uma forma não controlada, por todas as JVM/s e a inexistência de falhas e erros, como perdas de pacotes, que só aparecem quando se está a testar num ambiente realmente distribuído torna este tipo de abordagem pouco apropriada.

A plataforma Minha surge com o intuito de resolver estes problemas. Esta agrupa as várias JVM/s numa única, diminuindo drasticamente a quantidade de memória utilizada para a execução do sistema [1]. Para realizar a simulação, a plataforma cria uma instância de uma máquina virtual por cada máquina real, sendo que posteriormente interseta toda a informação que inicialmente iria para a rede e reencaminha para a máquina simulada correta. Esta forma de simular a rede internamente na plataforma permite um controle total de todas as interações e informação que nela circula permitindo inserção de atrasos e perdas de pacotes, entre outros problemas existentes na comunicação real, tornando a simulação mais realista. Isto torna o Minha numa ferramenta com enorme potencial no que toca à depuração e teste de sistemas distribuídos visto estes problemas existirem e apenas poderem ser observados quando o sistema é executado num ambiente real. Para além da simulação, a plataforma Minha permite também a geração de relatórios de execução e de tráfego de informação na rede de forma a permitir uma maior facilidade na análise dos potenciais problemas ou características do sistema. Esta plataforma utiliza uma técnica denominada de simulação de eventos discretos que lhe permite introduzir avanços controlados no tempo de forma a tornar mais rápidas as simulações que dependem de esperas [1].

Para a realização da simulação o Minha contém entre outras classes, duas que se podem considerar as mais importantes e com as quais a dissertação em questão lida mais diretamente. As duas principais classes são a *Timeline*, que é responsável pelo controle de tudo que está relacionado com o tempo e ordem da simulação dentro da plataforma, e a

classe *ClassLoader*, que é responsável por carregar as classes necessárias para a simulação.

Timeline: esta é a classe responsável pela ordenação dos eventos que o Minha irá executar. A *Timeline* pode ser vista como a classe responsável por orquestrar a execução de todos os eventos e por gerir o tempo dentro da plataforma [1]. Para o trabalho em questão as principais características da *Timeline* são o facto de executar os eventos por ordem e o de permitir parar e avançar no tempo interno da plataforma.

A *Timeline* trata eventos que derivem da classe *Event*, para isso qualquer classe que necessite de utilizar este mecanismo tem de ser estendida da classe abstracta *Event*. No caso desta dissertação a única classe que necessita de ter estas características é a classe *Storage*. Esta necessita de executar em certos intervalos de tempo, e para isso tem de poder adicionar à *Timeline* sempre que seja necessário, sendo este processo é explicado em pormenor no Capítulo 5.3. A classe abstrata *Event* possui um método também ele abstracto que a classe *Storage* terá de herdar, sendo esse método que a *Timeline* invocará quando for executar o evento.

Visto a *Timeline* ser responsável por controlar o tempo interno da plataforma é através dela que é feita a inserção de avanços no tempo. O tempo da simulação pode ser parado e reiniciado para qualquer tempo à frente. Esta propriedade é indispensável e será utilizada para a plataforma avançar o tempo inserindo o custo da simulação dos pedidos a disco. A forma como o processo de avanço é feito é abordado no Capítulo 5.3.

ClassLoader: A plataforma Minha é capaz de executar vários nós dentro de uma mesma JVM e para que isto seja possível, é utilizado o *classLoader* responsável por carregar as classes para posteriormente as executar de forma controlada, interceptando os acessos aos recursos, como a rede ou relógio [1]. Para que as intercepções sejam realizadas, a plataforma tem um conjunto de classes próprias que contém classes com o mesmo nome e métodos com a mesma assinatura que os originais, mas que estão alterados de forma a possibilitar a simulação e captura de dados, sendo que o próprio retorno por vezes também é influenciado.

Visto que nem todas as classes existentes precisam de ser substituídas, a plataforma dispõe de um ficheiro de configuração, apresentado na Figura 1, que contém quais as clas-

ses que esta irá substituir pelas suas classes *fake*. Durante o carregamento das classes a plataforma verifica o nome da classe que vai carregar, após uma consulta no seu ficheiro de configuração, verifica se a classe em questão se encontra marcada como *fake*. Caso a classe esteja marcada como *fake* esta é então substituída pela classe falsa correspondente da plataforma.

```
# Remap some core classes to simulation models
java.lang.Thread=fake
java.lang.Runtime=fake
java.lang.System=fake
java.util.concurrent.locks.ReentrantLock=fake
java.io.RandomAccessFile=fake
java.io.File=fake
java.io.FileDescriptor=fake
java.io.FileInputStream=fake
java.io.FileOutputStream=fake
java.io.ObjectOutputStream=fake
```

Figura 1 - Ficheiro de configuração

3 Estado da Arte

A simulação de discos é um mecanismo que já existe há bastantes anos, existem inúmeros projetos neste sentido que devem ser tomados em consideração e analisados para a criação de um modelo de discos mais fiável para a plataforma Minha. Embora a simulação de discos seja algo objetivo, há muitas formas de abordar o problema e os vários softwares existentes atualmente para a realização desta tarefa fazem-no de forma diferente uns dos outros. Alguns com uma maior preocupação com a especificidade da simulação, simulando apenas um modelo de disco específico, outros que permitem adaptar-se a configurações diferentes, obtendo informações de discos para depois reproduzir o seu comportamento, outros mais focados na alta configurabilidade que disponibilizam ao utilizador permitindo que este configure inúmeros parâmetros como velocidades dos barramentos e o número de discos ligados a cada barramento.

De forma a ter uma visão geral de algumas das abordagem ao problema existentes é apresentada uma pesquisa sobre simuladores de disco com abordagens e desenho bastante diferentes uns dos outros. A lista seguinte tem como objetivo permitir uma análise do que de melhor existe em cada um dos simuladores e os problemas existentes, de forma a desenvolver um modelo o mais bem conseguido possível para integrar a plataforma Minha.

3.1 DiskSim

O DiskSim [3] é o mais completo simulador de discos existente atualmente. O estudo do desempenho dos discos, a optimização de algoritmos é a motivação para o seu aparecimento, sendo este simulador desenvolvido com o intuito de compreender o desenho dos discos e avaliar novas arquiteturas de escalonamento de forma a aumentar o desempenho e desenvolver novas metodologias de ligação entre os componentes do disco.

Graças ao seu desenho e arquitetura, o DiskSim é um simulador eficiente, preciso e altamente configurável. O DiskSim está também dividido em subcomponentes que aquando da sua criação podem ser ligados e utilizados da forma que for mais adequada a simulação em questão, sendo este a principal mais valia do DiskSim.

O DiskSim ainda tem a seu favor o facto de estar sempre atualizado em relação às novas tecnologias de armazenamento que vão aparecendo, e com o tempo vai ganhando mais e mais extensões. A última atualização que possuí, aquando da escrita do documento em questão, é a extensão para a simulação de Solid State Disks (SSD) baseados em memória FLASH. [3]

O DiskSim é usado como referência para o desenvolvimento de modelos de simulação de discos, sendo em alguns casos usado como complemento para outras ferramentas de simulação, exemplo disso é o GemDisk (ver Secção 3.3).

3.2 Vesper

Vesper [4] é uma ferramenta que nasceu com uma motivação mais académica, com o intuito de fornecer uma maior abstração do hardware.

Embora baseado no DiskSim [3], o Vesper possui um nível muito mais baixo de detalhe em relação ao anterior, existindo mesmo assim um elevado grau de performance e realismo. Esta ferramenta de simulação não simula os componentes de armazenamento em separado, nem faz qualquer distinção sobre estes. Para o Vesper todos os componentes são como uma caixa negra em que são feitos pedidos e são obtidos resultados de uma forma totalmente abstrata. O seu grau de abstração sobre os componentes e ao mesmo tempo de realismo deve-se em grande parte a sua metodologia de simulação que se baseia na obtenção de dados de tempos de operações sobre outros discos que posteriormente são usadas para o cálculo dos valores de leituras/escritas que irá simular. Esta metodologia é o profile-driven, que consiste na realização de vários testes sobre ao disco que pretende simular para posteriormente poder simular o seu comportamento.[4]

3.3 GemDisk

O GemDisk [5] nasceu como ferramenta de suporte ao projeto geom sched [6]. O GemDisk faz a ponte entre o geom_gate [7] e o DiskSim e disco rígido para fazer a simulação do disco. O geom gate intercepta as operações de input e output no kernel, enquanto que o DiskSim permite fazer a simulação de disco. O GemDisk é uma camada que se encontra entre o DiskSim e o geom gate, utilizando o interface do geom gate para trazer as operações do kernel de volta para o espaço do utilizador onde as gere. Com as operações no espaço do utilizador este utiliza o DiskSim para a obtenção de tempos precisos, enquanto faz executa as operações, devolvendo os resultados do disco rígido, bem como os tempos obtidos no DiskSim de volta para o kernel.

Podemos ver o GemDisk como uma simples camada que permite a utilização do DiskSim em cooperação com o geom gate. Este é uma abordagem diferente onde não existe uma pura simulação por parte de nenhum dos componentes, mas é interessante a forma como faz a ponte entre várias ferramentas de forma a tornar a sua implementação fácil e eficaz. A Figura 2 mostra o modelo usado o GemDisk. Pode-se observar a forma como o GemDisk utiliza as funcionalidades do geom gate para interceptar os pedidos da aplicação no kernel e trazê-los de volta para o espaço do utilizador. Já no espaço do utilizador requerer os tempos das operações ao DiskSim que será responsável por simular essas mesmas operações como de um disco real se trata-se.[5]

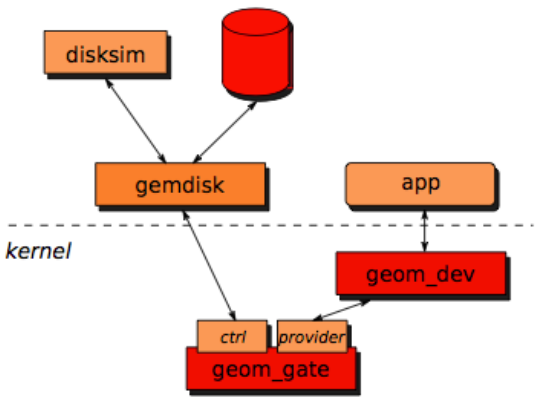


Figura 2 - Estrutura do GemDisk[5]

3.4 HP 97560 Disk Drive

O HP 97560 [8] é um modelo de discos muito mais específico, este está focado apenas na simulação de um disco em particular. Embora não seja de todo este o alvo do projeto é interessante ver a forma como as estruturas são criadas e os passos que são seguidos para uma correta simulação de um disco em particular, para depois se poder ter uma visão do que é realmente fixo, seja qual for a simulação que se pretende, e o que pode ser configurável e totalmente flexível.

No HP97560, a atenção das funcionalidades de configuração disponibilizadas focam-se na sua maioria na parte de ligações entre os componentes. Exemplo disso são as configurações ao nível das ligações a barramentos que os discos permitem, podendo ligar-se dois discos ao mesmo barramento, em barramentos separados, embora não haja qualquer possibilidade de configuração do barramento em si. Quando o disco é criado para a simulação possui booleanos que indicam a persistência ou não do conteúdo que neles é escrito.

Embora se trate da modelação de um disco rígido que possui inúmeras características, existem assunções que são feitas para permitir uma correta modelação do mesmo visto nem tudo ser possível simular ou não funcionar exatamente da mesma forma quando simulado. O interface que disponibiliza ao programador é muito simples e apenas permite inicializar o disco e escolher a que barramentos ligar, embora a sua simulação seja muito detalhada em termos do componente em si, tendo em consideração até mesmo as caches.

4 Problema

A plataforma Minha é um simulador ainda em desenvolvimento. Uma das componentes ainda não disponibilizada pela mesma é a de simulação de discos. O problema baseia-se no facto de quando se pretende simular um sistema distribuído pretende-se que cada máquina do sistema seja simulada de forma independente tendo os seus próprios componentes, como acontece na realidade. O que acontecia até a realização da dissertação era que todos os pedidos a disco feitos por qualquer uma das máquinas simuladas estavam a ser realizados diretamente sobre o disco da máquina onde a plataforma estava a correr sem que esses pedidos sofressem qualquer alteração. Isto resulta em dois problemas, o facto de se estar a executar o sistemas mediante velocidades do disco da máquina onde está a correr a plataforma, e o facto de todas as máquinas simuladas estarem a realizar operações sobre o mesmo disco. O problema das velocidades tem relevância na medida em que se queremos simular um sistema distribuído convém que este seja simulado com tempos de acesso ao disco e com as velocidades que aconteceriam num disco onde se pretende alojar o sistema. O outro problema é a partilha das velocidades e buffers do mesmo disco, deixando as máquinas simuladas de ser independentes ao nível do disco umas das outras.

Neste contexto, é necessária a introdução na plataforma de um mecanismo que permita simular a existência de um disco físico por cada máquina simulada existente, permitindo que cada máquina possa criar uma árvore de diretorias totalmente independente das outras máquinas simuladas, sendo que as diretorias usadas por uma máquina simulada não são visíveis para as outras máquinas simuladas. É também importante que cada máquina tenha um disco com o seu próprio buffer impedindo que as outras máquinas possam influenciar o espaço deste.

4.1 Assunções

Devido ao trabalho já desenvolvido na plataforma Minha a linguagem a ser utilizada para o projeto será Java, visto ser a mesma utilizada para o desenvolvimento de toda plataforma.

O modelo de simulação terá de ser implementado diretamente na plataforma e não poderá ser usado um software externo para o auxílio da simulação, ou seja, não poderá ser tirado partido direto de nenhum software já existente como o DiskSim ou utilizar uma metodologia parecida com o GemDisk. As soluções anteriormente abordadas seriam demasiado pesadas em termos de tempos de simulação, o que tornaria a simulação no Minha uma tarefa bastante mais demorada. Pretende-se encontrar um meio termo entre uma abordagem que seja rápida e ao mesmo tempo precisa o suficiente para produzir simulações realistas.

4.2 Abordagem

Tendo em conta as assunções acima citadas são desenvolvidos e implementados modelos de forma a darem resposta a todos os requisitos exigidos na dissertação.

É implementado um modelo para a criação da árvore de discos com o objetivo de dotar a plataforma de um mecanismo que permita o reconhecimento das diretorias que irão representar os discos de cada máquina simulada, e configurar a mesma para que reencaminhe todos os pedidos das diferentes máquinas para as diretorias correspondentes. Este modelo tem em consideração alguns aspectos que foram tidos em atenção, como o facto da existência da árvore de discos ter de ser totalmente transparente para a plataforma e todos os pedidos terem de ser restringidos à diretoria associada ao disco da máquina simulada. A criação da árvore de discos pode ser feita de duas formas. Pelo utilizador que utiliza a plataforma para correr testes sobre um sistema, permitindo a inicialização da estrutura com alguns ficheiros que possam ser úteis para a simulação a ser efectuada, sendo que esta árvore é reconhecida automaticamente pela plataforma desde que criada tendo em conta que o nome da diretoria que simula uma máquina virtual contenha o mesmo nome que o ip da máquina simulada. Caso o utilizador não queira criar a

árvore manualmente esta é criada durante a execução da simulação, em tempo de execução, sendo apenas criadas as diretorias que são usadas durante o desenrolar da mesma, ou seja, máquinas que não efetuem qualquer interação com os discos não terão diretoria associada a si, pois esta não é criada.

É também modelado e implementado um mecanismo de extração de informação de discos, este é um requisito que surgiu após a análise da melhor forma de facilitar a reprodução de inúmeros ambientes de uma forma simples. Este requisito tem como objectivo a realização de um programa que extraia informação do disco onde for executado de forma a gerar um ficheiro de configuração que é usado pela plataforma de forma a simular as características desse mesmo dispositivo. O programa em questão realiza vários testes de escrita e leitura sobre o disco a simular de forma a calcular tempos de acesso do mesmo.

A modelação e implementação do modelo de discos propriamente dito é o principal foco da dissertação. A abordagem passa por detectar todos os pontos onde existe interação com o disco e criar um bloco de código que faça a paragem de tempo na *Timeline*, faça uma chamada ao disco virtual e em seguida volte correr o código real avançando no tempo da simulação. O avanço no tempo é calculado pelo modelo e reflete o tempo que aquele pedido demoraria quando executado sobre o disco simulado. Este método é bastante parecido com o que já é feito na componente de rede da plataforma Minha, onde é igualmente feita uma paragem do tempo na *Timeline* quando é enviado algum pacote para a rede.

4.3 Validação

Em relação à validação das novas funcionalidades da plataforma Minha, esta passa pela criação de um caso de estudo que teste o modelo e implementação realizada sobre a plataforma, sendo que este caso será executado em ambiente real, e posteriormente sobre a

plataforma Minha. É expectável que os resultados obtidos sejam o mais parecidos quanto possível entre as duas execuções.

Durante a elaboração do modelo de simulação e do modelo de extração de dados foram elaborados exemplos para testes longo de toda a implementação com o objetivo de ir validando o projeto bem como os valores obtidos. Foram feitos testes que validam desde a construção da árvore de discos, tempos de execução, retornos da API falsa criada, até testes de validação dos resultados obtidos, passando também pela validação de todos os mecanismos intermédios.

A validação da criação da árvore de disco tem como objetivo testar a plataforma e verificar se são criadas todas as diretorias e com os nomes corretos. Testa também se a plataforma é capaz de reconhecer uma árvore criada manualmente e se é capaz de utilizar os ficheiros previamente alocados em cada disco virtual. Foi também necessária a validação dos tempos de execução do código sem qualquer chamada ao modelo de armazenamento de forma a validar que os valores obtidos não são dispares, de forma a não interferirem com os resultados obtidos aquando da simulação utilizando ao modelo de armazenamento.

Uma parte importantíssima da validação é a validação do tipo de retorno que se obtém com a invocação de alguns métodos do sistema de ficheiros, como o *getCanonicalPath*. Visto ter de se interceptar as invocações à classe *File*, entre outras, é importante garantir que o novo retorno iria de encontro ao que seria expectável numa situação normal, e que a transparência estaria assegurada. Um exemplo disso é a invocação do método *getAbsolutePath* que originalmente retornaria todo o caminho no disco real, mas o que é pretendido é que ele apenas retorne o caminho dentro do disco virtual, e um erro neste retorno pode causar bastantes erros em todo o fluxo de um programa simulado na plataforma.

A validação final passa por um teste que abrange todas as funcionalidades de uma só vez. Este teste passa inevitavelmente pela execução do extrator de informação sobre a plataforma e a execução de um caso de estudo.

5 Modelo de simulação

5.1 Assunções

Para o desenho e implementação do modelo de simulação a integrar na plataforma Minha surgiram algumas questões sobre a melhor forma de implementar o simulador de discos. Em relação a implementação e simulação das escritas para disco a abordagem seguida passa pela abstração dos componentes integrantes do disco focando apenas no que é indispensável para a simulação de forma a ser o mais rápida e realista possível. Assumindo que não há preocupações ao nível da divisão do disco em sectores, que os diferentes componentes do disco, como barramentos, não são sub-módulos, o disco é visto como uma caixa negra onde são feitos pedidos é obtido o tempo que esses pedidos levariam a executar. É também assumido que o tempo de procura não é tido em consideração, ou seja, em relação às leituras assume-se que toda a informação se encontra disponível na memória primária.

Sendo o simulador de disco um módulo da plataforma Minha este tem de ser desenvolvido e implementado mediante algumas regras a plataforma assim exige. Se se trata-se de uma simulação de discos em um outro qualquer programa em Java, a abordagem passaria por criar uma thread que trataria os pedidos em intervalos de tempo pré definidos. Mas visto que o simulador de discos é um componente integrante da plataforma esta abordagem não é válida neste contexto. Para assumir um comportamento similar ao explicado anteriormente, mas num contexto válido dentro da plataforma, é criada uma classe *Storage* que se auto adiciona à *Timeline* mediante um intervalo de tempo, e cada vez que for executada realiza o tratamento de pedidos pendente. Como visto no Capitulo 2.2 para que a classe *Storage* possa ser adicionada e tratada pela *Timeline* esta é estendida da classe abstracta Event que possui, tal como a classe *Runnable*, um método abstracto *run* que é o método invocado pela *Timeline* para iniciar a execução dos eventos. A classe *Storage* é adicionada à *Timeline* do Minha apenas quando tiver pedidos para tratar, não estando em constante escalonamento de forma a poupar recursos. Caso existam vários pedidos que não possam ser tratados numa única execução, a classe *Storage*

volta a ser adicionada à *Timeline* de forma a ser executada novamente num tempo futuro. A forma como os eventos são adicionados e tratados pela *Timeline* foi abordada no Capitulo 2.2.

Sendo a plataforma Minha uma ferramenta de simulação orientada a sistemas distribuídos, o desenho do modelo de simulação e a integração que é feita com a plataforma teve esse facto em consideração. O desenho é pensado de forma a poder ser facilmente adaptado de forma a receber várias configurações sendo apenas necessárias alterações mínimas na criação das máquinas simuladas na plataforma, permitindo uma maior abrangência a maleabilidade. O modelo permite que o disco seja utilizado como individual, tendo cada máquina o seu próprio disco virtual independente, para isso basta manter o *clone* na criação da instância da máquina simulada. Pode assumir as características de um único disco para várias máquinas, aproximando-se da metodologia da cloud, apenas retirando o *clone* durante a criação da máquina. Sendo também possível a criação de máquinas com discos com diferentes especificações, passando em argumento ficheiros de configuração diferentes. Embora esta última alteração seja um pouco mais exigente do ponto de vista das alterações necessárias à plataforma, visto que a forma como a plataforma está neste momento a fazer a separação dos argumentos tenha de ser um pouco alterada, torna-se mesmo assim bastante intuitivo pela maneira como a mesma está estruturada e modelada.

5.2 Sistema de ficheiros

De forma a tornar mais intuitivo para o utilizador da plataforma a adição de novos ficheiros aos discos e a visualização do conteúdo dos mesmos após a execução do sistema, é desenvolvido e implementado um módulo que tem como objetivo a criação e gestão da árvore de diretorias. Este módulo tem como objetivo a separação de cada disco virtual em diretorias próprias, sendo que cada diretoria passa a ser um disco físico para cada máquina. Assim sendo todos os pedidos feitos ao disco das máquinas virtuais são tratados e reencaminhados para diretorias associadas a cada máquina respectivamente. O nome destas diretorias, de forma a tornar mais intuitiva a percepção de qual diretoria que pertence a cada máquina virtual, tem como nome o ip da máquina em questão.

Todas as interações são também filtradas e é ocultado para o programa que corre sobre a plataforma que esta virtualização é feita e que os pedidos são reencaminhados para diferentes diretorias. Esta abordagem permite transparência no tratamento dos pedidos ao disco para o programa que corre sobre a plataforma.

Toda esta simulação e reencaminhamento de pedidos para o sistema é feita a partir da criação de uma nova classe *File* que substitui a classe original do Java. Esta nova classe contém, entre outras variáveis necessárias à simulação do sistema de ficheiros, uma variável de instância que é uma instância da classe *File* original que é usada para interagir com o disco real do sistema (Figura 3).

```
public class File {
    private final String path;
    private final java.io.File impl;
    private final String hostName;
```

Figura 3 - Classe File Fake

No momento da instanciação da classe falsa é também instanciada a verdadeira mas utilizando um novo caminho previamente modificado. A criação do novo caminho passa pelo acréscimo de um sufixo ao caminho original no momento da instanciação da classe falsa, sufixo esse que é o principal responsável pelo reencaminhamento.

A classe falsa é como que uma cópia precisa da classe original, do ponto de vista do programa que corre sobre a plataforma. A classe *File* falsa implementa todos os métodos e com a mesma assinatura que da classe *File* original.

De realçar que nesta classe, como em todas que são simuladas pela plataforma, quando é necessário algum método especial para resolver algum problema interno, esse método, dependendo do nível onde precisa de ser invocado, utiliza os modificadores *protected* e

package-private. Desta forma é garantido que apenas ficam visíveis internamente da plataforma, ficando apenas visíveis publicamente os métodos que a classe original também fornece.

Quando é feita uma invocação a uma instância da classe *File* falsa esta é reencaminhada para a instância da classe *File* real, que se encontra guardada dentro da instância *File* falsa, procedendo-se ao tratamento do parâmetro a ser passado caso seja necessário. No caso do método necessitar de retornar algum valor este também é previamente tratado caso necessário, ou seja, tudo é filtrado da forma mais conveniente em cada caso.

Durante a execução são criadas as diretorias referentes a cada máquina de forma a que seja possível simular os respectivos discos associados às diretorias, sendo que o disco virtual só é criado em tempo de execução se existir alguma chamada ao sistema de ficheiros por parte da máquina simulada a ele associada. O módulo de armazenamento criado na dissertação encontra-se também preparado para reconhecer árvores de discos virtuais criados previamente para que possam ser introduzidos ficheiros necessários a um correta execução do sistema. A criação do sistema de ficheiros tem de ser criado segundo regras pré estabelecidas. Sendo que a árvore de discos tem que ser criada na raiz onde a plataforma Minha está a ser executada e o nome da diretoria que corresponde ao disco tem de ter o nome igual ao ip da máquina simulada correspondente.

5.3 Dispositivos de armazenamento

O método de armazenamento de disco baseia-se na simulação das operações sobre o disco, simulando o custo dos processos de escrita e leitura sobre o disco e fazendo a gestão do débito do disco.

Quando é requerida uma operação de leitura sobre um disco, existe inicialmente uma deslocação da cabeça do disco para a zona onde essa mesma informação se encontra armazenada e após isto a informação é copiada para o sistema. No caso da simulação de discos implementada na plataforma o movimento das cabeças não é considerado, como referido no Capitulo 5.1. No caso das escritas, existe também um movimento das cabeças para a área onde se iniciará a escrita. Inicialmente a informação é copiada para o

buffer do disco e só depois escrita definitivamente em disco. Caso o buffer esteja cheio o pedido espera até haver espaço para ser copiado.

No que diz respeito à implementação, o núcleo da simulação de dispositivos propriamente dita está armazenada numa única classe, *Storage*, que é responsável pela gestão de buffers associados ao disco, simulação de débito do disco e cálculo de custos das operações de I/O.

Quando a plataforma Minha é iniciada, entre informação de outros módulos, é carregada toda a informação necessária à simulação de discos através de um *Properties File* que contém o conjunto de valores que são usados para a simulação das operações sobre o disco. O ficheiro deverá conter uma forma similar ao da Figura 4, tendo obrigatoriamente de conter todas as propriedades na figura.

```
#Sun Jun 30 13:08:31 WEST 2013
writeStable=7176.47288715193
writeDecl=3.5271128480703466
bucketSize=8388608
diskDebit=58462
readStable=4516.394723986321
readDecl=0.511333277357108
```

Figura 4 - Exemplo de configuração

O ficheiro de configuração contém seis campos que são carregados e tratados pela classe *Storage*.

Os campos que têm como sufixo *Decl* e *Stable*, tanto no *write* e *read*, correspondem às variáveis da fórmula da recta, que é usada para calcular o custo de escrita e leitura de uma quantidade de bytes, representando as constantes m e q, respectivamente.

$$y = mx + q$$

Através do uso destes valores é possível calcular o custo das escritas e leituras de uma quantidade de bytes. Em relação ao campo *diskDebit*, este indica a quantidade de bytes

que são escritos efetivamente em disco a cada um milissegundo, e não o custo de copiar para o buffer. Por último, o campo *bucketSize* é o tamanho do buffer do disco em mb.

Quando uma nova simulação é iniciada, a plataforma Minha cria uma nova instância da classe *Storage*, passando uma instância *Properties* que contêm as propriedades do ficheiro de configuração já carregadas. Recebendo a informação do ficheiro de configuração é da responsabilidade da classe *Storage* a leitura dos parâmetros e posterior inicialização das suas variáveis de instância com os parâmetros obtidos através da instância da classe *Properties*. Após uma primeira instanciação da classe *Storage* a plataforma invoca o método *clone* sobre a mesma instância de forma a obter novas instâncias iguais à primeira. É então passada uma nova instância para cada máquina virtual iniciada, sendo que cada máquina da plataforma terá uma instância independente e responsável pela simulação do seu armazenamento.

A classe *Storage* possui, para além das variáveis necessárias para o cálculo do custo das operações, estruturas responsáveis por suportar todo o processo de gestão interno do disco, como buffers, uma coleção das streams ativas e outra de pedidos que estão bloqueadas à espera de poderem ser processados para posteriormente retornarem.

Quando um sistema que está a executar sobre a plataforma Minha requere alguma interação sobre o sistema de ficheiros é criada uma nova instância da classe *Request*. O construtor da classe *Request* recebe o tipo de pedido, este pode ser de escrita ou leitura, o tamanho do pedido, o identificador único da stream à qual está associado, a instância da classe *Storage* associada à máquina simulada em questão, e um *Event* de *wakeup* que servirá para o caso da máquina ter de esperar pelo retorno da *Storage* esta a poder iniciar novamente. Após a inicialização do *Request* é invocado o método *execute* sobre a instância que é responsável por adicionar o pedido à classe Storage para ser tratado. Após se adicionar o pedido é invocado o método sobre o Request que retorna o custo associado à operação, em nano segundos. Caso o pedido ainda não tenha sido processado a máquina simulada fica em espera e quando a classe Storage executar o pedido a máquina em espera é notificada, através do evento de *wakeup*, acordando a que volta a requerer novamente o custo usando o para avançar no tempo da simulação.

Com o intuito de ser possível simular o método *sync*, método responsável por garantir que a informação se torna realmente persistente em disco, quando um pedido é adicionado à *Storage* o seu identificador é incrementado numa coleção que é responsável por

guardar todas as streams ativas. O identificador de stream só é decrementado quando o mesmo deixar de ser utilizado na instância de *Storage*, ou seja, quando ele deixar de estar no buffer de disco e passa a estar efetivamente escrito em disco. Quando a chamada de *sync* é interceptada é invocado um método sobre a *Storage* que apenas retorna quando o identificador desse stream estiver a zero na coleção, ou caso este não exista lá.

Quando o *Request* é adicionado à classe de *Storage* este é filtrado para aferir qual o seu tipo de forma a poder ser reencaminhado dentro da *Storage*.

As leituras e escritas são tratadas de forma diferente pelo modelo. A diferença de tratamento pode ser observada na Figura 5 onde é apresentado, na forma de diagrama de atividades em UML, a maneira como cada um é tratado quando adicionado à *Storage*.

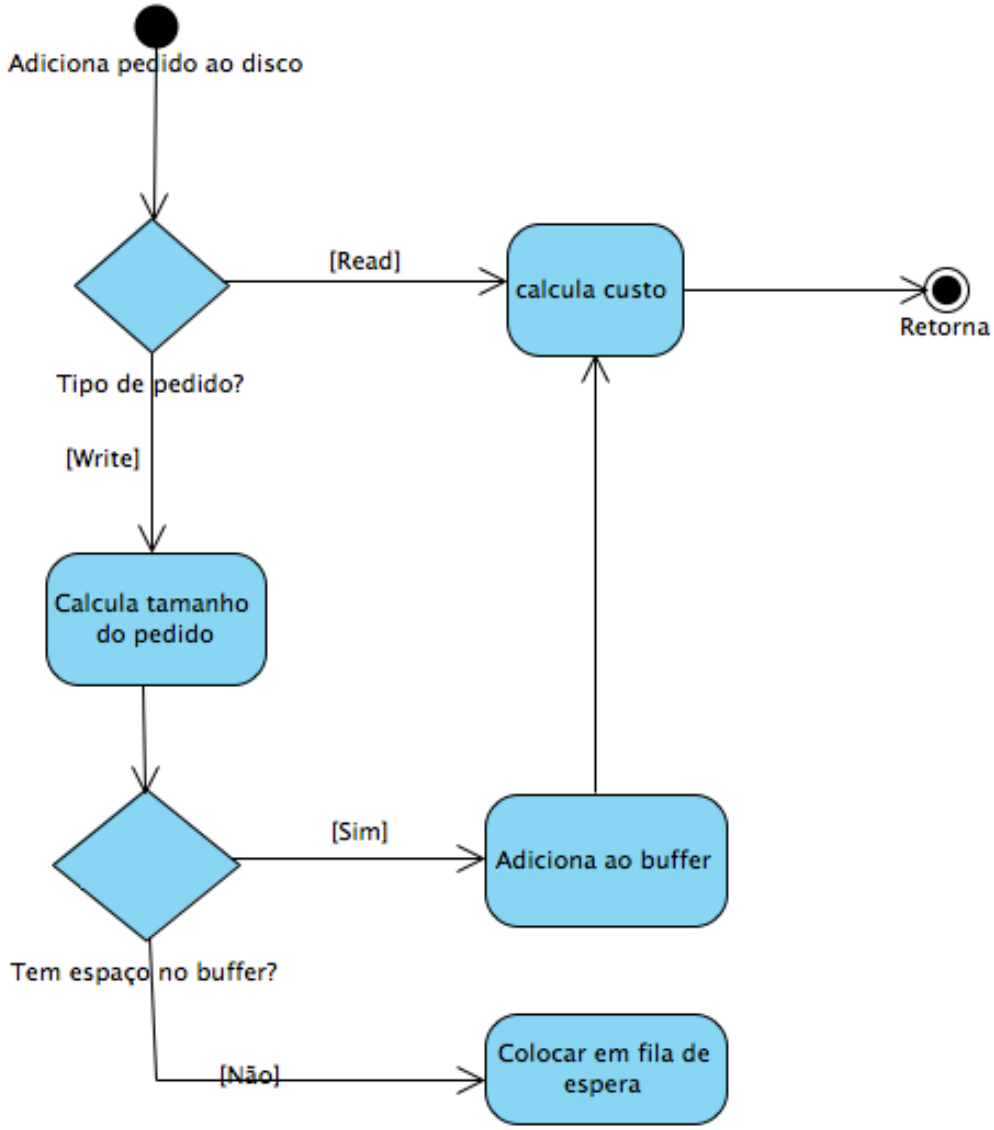


Figura 5 - Adicionar pedido

No caso de se tratar de um pedido de leitura, apenas é calculado o custo de efetuar a operação e é adicionado o custo à instância do pedido para poder ser lida.

No caso de se tratar de um pedido de escrita, o processo torna-se um pouco mais complexo. Primeiramente é analisado se existe espaço no buffer do disco para copiar o pedido para lá. Caso exista, o pedido é adicionado ao buffer, é calculado o custo e associado ao pedido para poder ser lido. Caso o buffer não tenha espaço disponível para a cópia do pedido, este é colocado na fila de espera impedindo-o de retornar de imediato

ficando a aguardar que o buffer esvazie e que este possa ser copiado para lá. Apenas após a cópia para o buffer é que o custo é calculado e adicionado ao pedido.

Até agora apenas foi abordada a forma como os diferentes pedidos são tratados quando adicionados ao disco, a forma como o custo da operação é calculada e a maneira como os pedidos são guardados temporariamente em disco à espera de serem efetivamente escritos.

No que se refere às escritas persistentes em disco estas são processadas a cada um milissegundo, e na quantidade indicada pela variável de débito do disco. A forma como o processo se desenrola pode ser observada, na forma de diagrama de atividades em UML, na Figura 6.

Quando um pedido de escrita é adicionado à classe *Storage* este é inicialmente adicionado ao buffer para ser tratado, caso não exista espaço no buffer é adicionado à fila de espera. Após adicionado um pedido de leitura a classe de *Storage* inicia o processo de tratamento de pedidos. Existindo pedidos no buffer, a cada milissegundo é processada uma quantidade de bytes do primeiro pedido, que se encontra no buffer. No caso do débito ser maior que a quantidade de bytes por processar no pedido, este é retirado da fila, e é processado o próximo pedido, sendo que a quantidade de bytes a ser processados desta vez é o débito total menos a quantidade processada do pedido anterior. Este ciclo é realizado até que a quantidade total de bytes tratada seja igual ao débito total do disco ou que já não existam mais pedidos no buffer para serem processados. A cada pedido que é processado, é libertado espaço no buffer e é testado se existe algum pedido em fila de espera para ser movido para o buffer, caso exista espaço em buffer esse pedido é movido para lá. Cada vez que um pedido é removido do buffer é também decrementada a stream a ele associada, indicando que este pedido está permanente em disco.

Para evitar que se esteja sempre a tentar processar pedidos mesmo que eles não existam, a instância *Storage* possui uma variável booleana que indica se o disco está ativo ou não. Caso não esteja, quando chega um novo pedido de escrita esse booleano passa a verdadeiro, e é nessa altura que é iniciado o tratamento de pedidos a cada milissegundo até que já não existam mais pedidos a tratar e então o booleano volta a passar a falso até

voltarem a existir pedidos. Este ciclo de tratamento de pedido é orquestrado pela *Time-line* que executa o método *run* do *Event* que corresponde ao tratamento de pedidos na classe *Storage*.

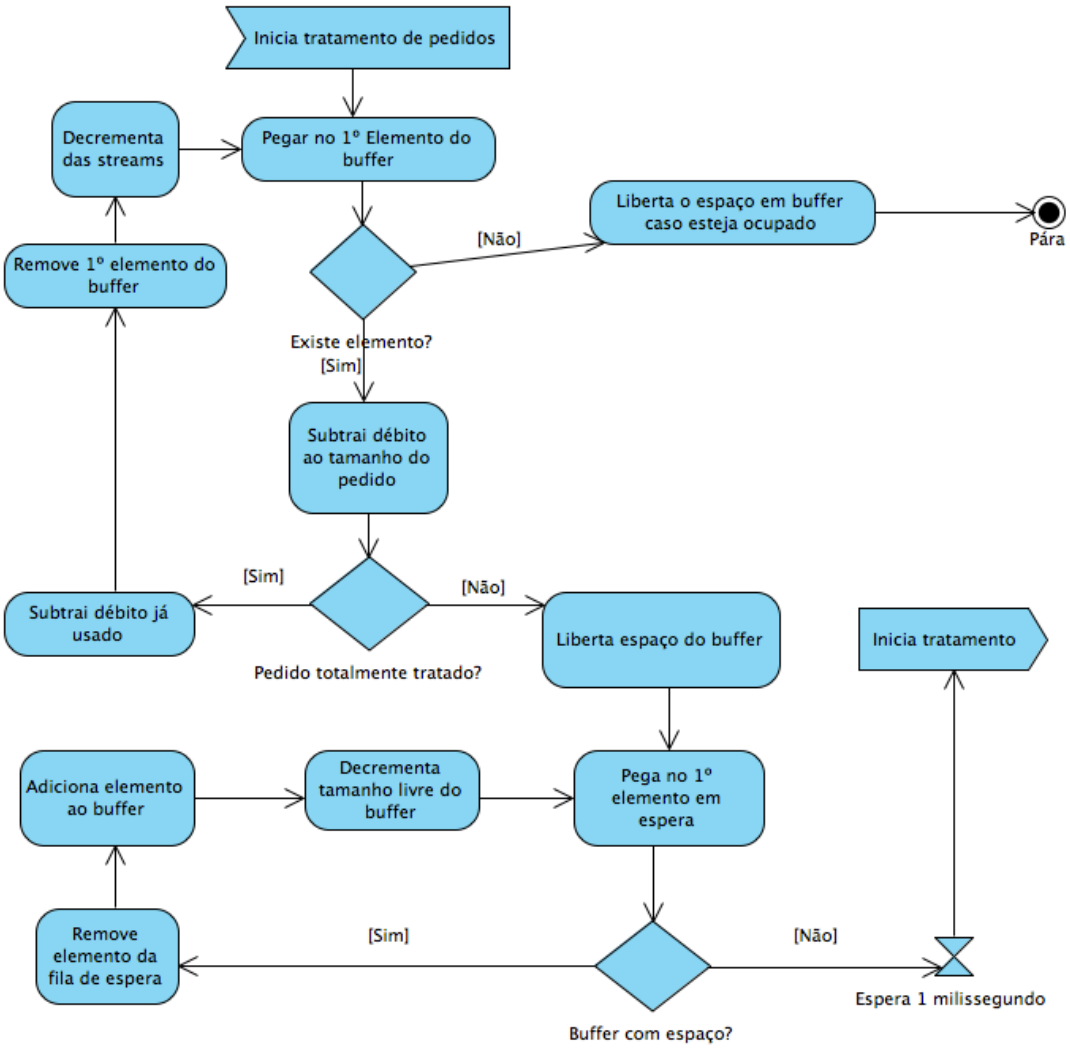


Figura 6 - Tratamento de pedidos

6 Calibração e avaliação

A forma como os valores usados para a simulação do disco são inicializados tem influência na forma como toda a simulação irá decorrer. De forma a ter um maior realismo na simulação esses valores tem de ser obtidos através de mecanismos que garantam que estes são os valores que realmente a plataforma está preparada para receber. Foi com este intuito que foi criado o extrator de informações de disco que se compromete a realizar a extração de forma adequada para uma simulação realista. Para se provar o realismo da simulação é também necessária uma avaliação cuidada do desempenho do modelo de simulação de discos.

Nesta secção é abordada a forma como o extrator de informações de disco funciona, apresentando todos os parâmetros que este se compromete a medir e a forma como esta medição é realizada. Será também abordada a forma como estas medições são testadas de forma a comprovar que tudo está a ser realizado de forma correta. Por último é mostrado o seu desempenho recorrendo a um caso de estudo.

6.1 Caracterização do desempenho

Como complemento ao modelo de simulação de disco explicado no Capitulo 5, foi desenvolvida um aplicação com o objectivo de recolher informação de um disco que se pretenda simular na plataforma. Esta ferramenta tem como principal objetivo agilizar o processo de extração de valores de um ambiente onde será implementado o software e transferir essas características para a plataforma Minha de forma rápida, simples e eficaz. Para isso foi criada uma aplicação que quando executada sobre um determinado ambiente realiza testes sobre o disco e o sistema criando um ficheiro de configuração do mesmo tipo que a plataforma Minha utiliza para a configuração dos seus discos.

O valores necessários para a simulação são as constantes necessárias para a reprodução de uma recta de onde serão extraídos os valores do custo da execução de pedidos ao disco, o débito do disco, bem como o tamanho do buffer de disco. Cada uma das medidas foi extraída separadamente testando o disco da forma mais adequada para cada constante.

Pontos da recta de escrita: Esta recta caracteriza o tempo que uma certa quantidade de bytes demora a ser copiada para o buffer do disco, e não o tempo que demora ser escrita efetivamente em disco. Isto porque, quando é requerida ao sistema a transferência de uma quantidade de bytes para o disco, este primeiro efetua uma cópia para os buffers do disco, sendo este tempo que pretendemos medir, visto que posteriormente a quantidade de bytes que o disco consegue escrever num intervalo de tempo estará relacionada com o seu débito. De sublinhar que os tempos de procura em disco não são tidos em consideração neste modelo de simulação. Visto isto, a forma de obter as constantes desta recta é através da obtenção de dois pontos, e posterior cálculo do declive e da constante, valores referido no Capitulo 5.3. Os pontos utilizados para a obtenção desta recta têm algumas restrições para garantir que o que está a ser medido é realmente o tempo que demora a copiar uma quantidade de bytes para o buffer e que o débito do disco não afecta estes valores, ou seja, que não se está a ultrapassar o tamanho buffer e a ter de esperar que ele esvazie para copiar o resto da informação. Para que isto seja possível foram escolhidas duas quantidades de informação que nos garantem estas restrições, um byte e dois kilobytes. Sendo que estes valores foram escolhidos por serem suficientemente pequenos para que o tamanho do buffer não interfira mas ao mesmo tempo separados o suficiente para que seja possível o cálculo da recta de uma forma aceitável. Escolhida a quantidade de bytes a ser testada é calculado o tempo que cada uma demora a ser copiada para o buffer do disco. Para isso é inicializado um *FileOutputStream* e inicializado um array de bytes com o tamanho da quantidade de bytes a copiar. São então efetuadas várias escritas é obtido o tempo médio em nanosegundos que demorou essa operação a ser realizada. Para evitar que o tempo perdido com a obtenção do tempo através do método *System.nanoTime()* influencie os resultados, são efetuadas várias escritas da mesma quantidade e medido o tempo que todas elas demoram. Visto que apenas interessa saber o custo uma operação é então dividido o tempo pelo número de operações para obter o custo de uma única operação. Tendo os custos de escrever as duas quantidades

de informação é então calculado o declive e a constante da função da recta e guardados os valores para serem escritos no ficheiro de configuração.

Pontos da recta de leitura: A forma como os pontos da recta de leitura são calculados é muito similar ao que acontece com os de escrita. É calculado o tempo que demora uma certa quantidade de bytes a ser copiada, neste caso o tamanho do buffer a ser copiado é de um kilobyte e um megabyte. No caso das leituras é assumido, como referido no Capitulo 4.1, que a informação a ser lida já se encontra na memória principal, ou seja, o que se pretende medir é o custo de copiar essa informação da memória principal para o programa. Para garantir que isto acontece, que a informação já se encontra na memória principal, é inicialmente preparado o buffer, fazendo um certo número de cópias para o programa, mas sempre do mesmo bloco de informação. Assim sendo, é criado uma instância da classe *RandomAccessFile*, e por cada iteração que copia uma certa quantidade de informação e move o apontador para o inicio do ficheiro para garantir que na próxima iteração é copiado o mesmo bloco novamente. Tendo sido preparado o buffer, e estando garantido que este bloco já se encontra na memória principal, é então obtido o tempo que demora uma certa quantidade a ser copiada da memória principal para o programa. Mais uma vez são feitas várias cópias e posteriormente dividido o tempo obtido pelo número de cópias efectuadas. Tendo os custos de ler as duas quantidades de informação é calculado o declive e a constante da função da recta e guardados os valores para serem escritos no ficheiro de configuração.

Débito do disco: O débito do disco pode ser definido como a quantidade de informação que é escrita efetivamente em disco num determinado espaço temporal. Este é dos parâmetros mais importantes visto que influencia todo o comportamento do sistema, sendo o responsável por definir a velocidade a que o buffer de disco vai esvaziando, tornando-o apto para receber mais informação. Para medir o débito do disco, e aproveitando esta medição para futuramente testar o funcionamento da fila de espera onde são colocados os pedidos quando o buffer está cheio, tem de se ter em consideração dois aspectos. O primeiro é que a quantidade de informação a ser copiada tem de ser maior que o buffer do disco para garantir que o buffer fica realmente cheio. Visto que o buffer vai esvaziando enquanto o estamos a encher, é usado como valor de teste o dobro do tamanho do

buffer, sendo que esta medida nos dá alguma segurança que o buffer fica realmente cheio e que alguns pedidos têm de ficar pendentes, ou seja, a fila de espera é usada. O tamanho do disco é uma variável que consta no ficheiro de parâmetros do programa de extração de informação, esta deve ser alterada conforme o sistema a testar, sendo que se encontra iniciada a 4084 mb. O segundo ponto é o facto de se queremos medir é o débito do disco, ou seja, teremos de esperar que a informação se torne realmente persistente em disco.

Para garantir que realmente se está a medir o débito do disco e não a velocidade com que a informação é copiada para os buffers de disco, é obtido inicialmente o tempo em milissegundos, em seguida é escrita uma grande quantidade de informação para o disco e para garantir que a informação se torna realmente persistente em disco. Após o pedido de escrita da informação é invocado o método *sync* sobre o descritor de ficheiro associada à stream, que apenas retorna quando a informação se torna realmente persistente em disco, sendo o medido o tempo em milissegundos no final destas operações. Sabendo o quantidade de informação escrita e o tempo que essa informação demorou para se tornar persistente em disco é feita a divisão da quantidade de informação escrita pelo tempo que esta operação demorou de forma a obter-se o débito do disco.

A unidade de tempo usada no débito do disco é o milissegundo visto ser o valor mais sensato obtido a partir dos testes efetuados. Quando testado de nano em nano segundo observou-se que era um intervalo muito pequeno e que exigia demasiadas operações, e a influência no resultado era quase nula, sendo que no teste feito segundo a segundo mostrou impactos significativos quando se estava a escrever quantidades relativamente grandes e muitas vezes, provocando o enchimento do buffer, não pela quantidade de bytes mas pelo elevado espaço entre os débitos.

Tamanho do buffer de disco: O tamanho do buffer do disco é um parâmetro que pode não ser muito relevante quando estamos a falar de escritas de poucos dados, mas quando esse tamanho aumenta de uma forma considerável este parâmetro torna-se essencial no que toca à velocidade das escritas. Embora importante este é um parâmetro muito difícil de medir, sendo que aptou-se por deixar esse valor ao critério de cada utilizador, sendo que durante a criação do ficheiro de configuração o software de extração de tempos utilizará 8 mb como valor padrão, visto ser o valor mais comum para a maioria dos discos.

Em relação ao desenho do programa, todos os valores de configuração, desde a quantidade de testes, a quantidade de informação usada para o aquecimento dos buffers do disco, o espaçamento dos pontos e até o diretório onde serão escritos e lidos os ficheiros é totalmente configurável. Estando estas variáveis contidas todas numa única classe sendo variáveis estáticos facilmente alteráveis e que tem impacto em todo o programa. De realçar que quando o programa é iniciado este cria o diretório onde os testes serão realizados, e apenas efetuará operações dentro dele, após o término do programa esse diretório tal como todos os ficheiros utilizados serão automaticamente apagados, deixando o ambiente completamente limpo. O ficheiro de configuração será também criado na mesma diretoria onde o extrator de informação estiver a ser executado.

6.2 Resultados

Tendo sido testada toda a parte de interação entre o programa e o sistema, e garantido que o tempo de execução do código simples não afectaria os resultados obtidos, como mostrado no Capitulo 4.3, passou-se para a validação dos resultados nos testes executados diretamente sobre o sistema e sobre a plataforma Minha.

Para garantir que os valores da simulação são fidedignos, executou-se o mesmo código para extração de informação do disco mas desta vez sobre a plataforma de forma a verificar os valores obtidos da extração de informação sobre o disco mas desta vez simulado. É expectável que os valores do ficheiro de propriedades gerado desta forma fosse parecido com o gerado quando o programa correu diretamente sobre o sistema. Os valores obtidos foram bastantes similares o que demonstra que o modelo de simulação é válido e fiável.

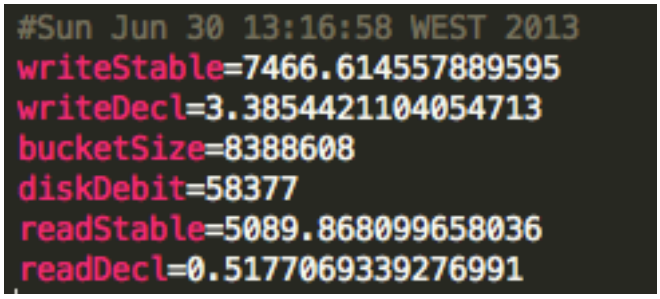


Figura 7 – Execução sobre o simulador

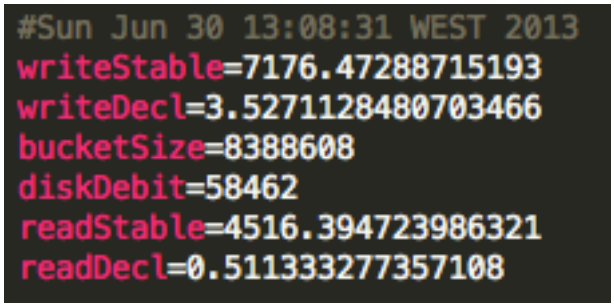


Figura 8 – Execução sobre o sistema

Na Figura 8 é apresentado o ficheiro de configuração gerado pelo programa após a extração da informação sobre o sistema. Os parâmetros apresentados na Figura 8 foram os mesmos que serviram de configuração para o modelo de armazenamento da plataforma Minha que originou o ficheiro apresentado na Figura 7 após a execução do programa responsável pela extração de dados desta vez sobre a plataforma.

O facto de executar o programa de extração sobre a plataforma Minha não comprova apenas a fiabilidade e validade dos tempos de simulação mas testa também todos fluxos internos do modelo. Esta execução esta o cálculo dos tempos de escrita e leitura, através de pedidos de leitura e escrita pequenos, mas testa também os buffers e o débito do disco quando se aumenta a carga de pedidos sobre o modelo. E por último visto utilizar o método sync, testa também o mecanismo que permite a simulação desta chamada ao sistema. Testa também partes mais pequenas e básicas do código como a transferência de pedidos da fila de espera para o buffer, para além de testar o sistema de ficheiros, e as intercepções nas streams. Resumindo, o facto de executar este programa sobre a plataforma pode ser considerado o real teste ao modelo de armazenamento. Sendo que o modelo passou em todos os testes ao qual foi sujeito.

Visto a API de Java ser muito extensa e não fazer de todo sentido implementar todas as classes e métodos existentes que interajam com o disco, e visto que o modelo de *Stora-*

ge ser um dos componentes da plataforma Minha, este segue a sua ideologia, ou seja, todos os métodos que forem fazendo falta com o passar do tempo irão sendo implementados.

6.3 Caso de estudo (HSQL)

A principal motivação da plataforma Minha é a simulação de sistemas distribuídos de forma a agilizar o processo de desenvolvimento e depuração dos mesmos. Visto isto e tendo em conta que a maior parte dos sistemas distribuídos, e não distribuídos, tem como base de armazenamento da sua informação bases de dados, surgiu a necessidade de testar como se comportaria o modelo de simulação de discos sobre estas condições. Uma vez que a plataforma é desenvolvida para a realização de simulações sobre a linguagem Java, teve de se escolher uma base de dados desenvolvida sobre esta linguagem. A escolha recaiu sobre o HSQLDB, visto ser totalmente desenvolvido em Java, e pelo facto de ser simples (tendo em conta o tamanho do código) e testar todas as funcionalidades da API de I/O.

Assim sendo optou-se por fazer um teste onde se punha à prova a capacidade da base de dados, mas também a capacidade do modelo de *Storage* conseguir simular o armazenamento de uma forma correta quando colocado sob um teste de stress. A melhor forma para realizar este teste é executar um benchmark da base de dados sobre a plataforma.

O benchmark utilizado para este teste foi o benchmark oficial do HSQL, o *Performance Test Case TPC-B* [5], que se encontra disponível no site oficial do HSQL. Este benchmark é um teste de stress à base de dados, que tem por objetivo medir o seu desempenho e comportamento sobre muita carga.

Visto as operações sobre a base de dados serem operações sobre o disco, este mesmo teste de stress à base de dados irá fazer o teste de stress ao modelo de armazenamento. Embora não testando todo o modelo, este teste tem como principal objectivo provar que o modelo funciona bem em ambientes de simulação reais.

A simulação do problema que o benckmark pretende recriar é o de uma base de dados de um banco. Cada banco tem várias agências em que cada uma delas contém 10 caixas e 100.000 contas. Uma transação atualiza uma linha em cada uma das agências, caixas e tabelas de conta, selecionando uma linha atualizada da tabela de conta, e inserindo uma linha na tabela de histórico. A cada transação são executadas 5 operações [5],

Com o objectivo de recriar as possíveis transações, neste modelo é usado um factor de escala de 40, ou seja, é simulada a existência de 40 filiais, 400 caixas e 4000 contas na base de dados. São realizados todos os testes com 4 conexões, cada uma realizando 8000 operações (32000 transações ou 160000 operações sob a base de dados por *round*) [5].

Este teste foi realizado na seguinte máquina:

- Processador Intel Core i5 dual core de 2,4GHz com 3MB de cache L3 compartilhado
- 4GB de memória DDR3 de 1333MHz (dois SO-DIMMs de 2GB)
- Disco rígido Serial ATA de 500GB, 5400 rpm

Após a realização do teste diretamente no sistema foram obtidos os seguintes resultados:

** Benchmark Report **

Time to execute 32000 transactions: 1.545 seconds.

Max/Min memory usage: 988034544 / 913646248 kb

0 / 32000 failed to complete.

Transaction rate: 20711.974110032363 txn/sec.

Quando o mesmo teste foi realizado sobre o simulador de disco carregado com as características do disco do teste real, os resultados foram os seguintes:

** Benchmark Report **

Time to execute 32000 transactions: 1.445 seconds.

Max/Min memory usage: 0 / 0 kb

0 / 32000 failed to complete.

Transaction rate: 20911.974110032363 txn/sec.

Como se pode constatar os valores são muito similares quando executados dentro ou fora de ambiente da plataforma.

A realização de um teste sobre um caso muito parecido com a realidade vem realçar o facto do simulador conseguir garantir medidas muito realistas, vindo demonstrar a sua perfeita aplicabilidade em situações de simulação de ambientes reais.

7 Conclusão

Para que a simulação possa ser utilizada como uma mais valia para facilitar o estudo e desenvolvimento de novos sistemas esta deve ser feita de forma coerente com o objetivo de gerar um ambiente de simulação que seja o mais real e confiável possível. A simulação nunca pode substituir a realidade, mas serve como ferramenta para agilizar o processo de desenvolvimento.

A dissertação acrescenta à plataforma Minha mecanismos que permitem que esta faça uma simulação ao nível dos discos. Esta encontra-se agora dotada de um sistema de ficheiros, responsável pela simulação do espaço de discos, de um módulo responsável pelo cálculo do custo dos pedidos e da gestão dos mesmos. Para além dos mecanismos ligados diretamente à parte da simulação propriamente dita, a plataforma possui um novo mecanismo que não se encontra ligado diretamente com a plataforma mas que serve de suporte a uma melhor simulação. Trata-se do programa de extração de informação de outros discos para serem reproduzidos pela plataforma de forma a proporcionar maior facilidade da reprodução de novos ambientes.

Durante todo o trabalho desenvolvido ao longo da dissertação a plataforma foi sendo completada e corrigida no seu todo, com a introdução de novas classes e métodos que foram fazendo falta durante todo o processo de análise e testes.

Tendo em conta os resultados obtidos através dos testes realizados ao sistema pode-se concluir que o objetivo foi alcançado e que a plataforma está agora capaz de simular os discos das máquinas simuladas de forma independente.

Todo o trabalho realizado na plataforma encontra-se já integrado com a plataforma Minha no ramo *diskio* no repositório oficial, repositório que é público[6].

8 Referências

[1] N. a. Carvalho, J. Bordalo, F. Campos, and J. Pereira, "Experimental evaluation of distributed middleware with a virtualized Java environment," *Proceedings of the 6th Workshop on Middleware for Service Oriented Computing - MW4SOC '11*, pp. 1–7, 2011.

[2] Minha: Middleware testing platform. Em http://www.minha.pt, consultado em Maio de 2013

[3] G. Ganger, B. Worthington, and Y. Patt, "The DiskSim simulation environment version 2.0 reference manual," 1999.

[4] P. Derosa and J. Pearson, "Realism and Simplicity : Disk Simulation for Instructional OS Performance Evaluation," 2006.

[5] F. Checconi and L. Rizzo, "GemDisk: a GEOM Class to Emulate Disk Drives," 2009.

[6] Geom-based Disk Schedulers. Em http://info.iet.unipi.it/~luigi/geom_sched/, consultado em Janeiro de 2013.

[7] GEOM Gate Network Devices. Em http://www.freebsd.org/doc/handbook/geom-ggate.html, consultado em Janeiro de 2013.

[8] David Kotz, Song Bac Toh and Sriram Radhakrishman, "A Detailed Simulation Model of the HP 97560 Disk Drive" 1994.

[9] HyperSQL - Performance tests. Em http://www.hsqldb.org/web/hsqlPerformanceTests.htm, consultado em Maio de 2013.

[10] Minha repository. Em http://code.google.com/p/minha/, consultado em Junho de 2013