Tese de Mestrado em Engenharia Informática e de Computadores

Terracotta Scheduling para Execução de Gridlet Engine e Aplicações Cycle-Sharing

João Neto, Nº 49665

Resumo

Este relatório descreve a fase inicial de investigação e desenho de uma solução de computação grid para aumentar a escalabilidade e extensibilidade num overlay peer-to-peer simulado. É composto por uma introdução aos conceitos de computação grid e os seus usos e benefícios reais, ao conceito da máquina virtual Java distribuída Terracotta e ao conceito de rede e overlay peer-to-peer. De seguida são enunciados os vários projectos actualmente existentes nas áreas mencionadas, sendo estes classificados segundo critérios relativos à sua avaliação (ex. velocidade de procura num overlay peer-to-peer, tipo de sistema grid, escalabilidade de um simulador de overlay, etc.). Contém também uma abordagem preliminar à solução do problema e às métricas de avaliação a usar, e qual o estado actual da arte no que toca a sistemas grid, overlays peer-to-peer, e simulação de redes e sistemas peer-to-peer.

Palavras-chave: Terracotta peer-to-peer grid simulação paralelismo overlay computação distribuída

1. Introdução

> *"On 16 September 2007, Folding@home, a distributed computing network operating from Stanford University (USA) achieved a computing power of 1 petaflop -- or 1 quadrillion floating point operations per second. The project uses the power of peoples' home computers, as well as their PlayStation3s, to simulate the processes inside living cells that can lead to diseases, such as Alzheimer's Disease."*
>
> *"Dia 16 de Setembro de 2007, a Folding@Home, uma rede de computação distribuída a correr na Universidade de Stanford (EUA) atingiu a marca de 1 petaflop em poder de computação (o equivalente a um quadrilião (10^{15}) de operações de vírgula flutuante por segundo. O projecto usa as capacidades de processamento de computadores pessoais (ou suas PlayStations 3), para simular os processos que ocorrem dentro de células vivas que podem levar a doenças, como é exemplo a Doença de Alzheimer."*
>
> *Guiness World Book of Records*

Desde que se criaram os primeiros mecanismos para programação em várias threads de execução que se tornou claro que a computação paralela seria uma ferramenta altamente valiosa para acelerar processos que outra forma seriam impraticáveis. O primeiro grande salto foi dado quando começaram a ser desenvolvidos os primeiros sistemas com vários CPUs, passou a ser possível efectuar computação paralela real por meio de hardware. Hoje em dia com as possibilidades dadas pela *World Wide Web* e as velocidades das ligações cada vez mais rápida, um sistema terá hipoteticamente acesso a um super computador virtual em constante crescimento, com um poder computacional semelhante a um recente super computador da IBM. É neste contexto que se introduz o conceito de *Grid Computing*.

1.1. Grid Computing

A primeira definição para *Grid Computing* foi dada em meados dos anos 90 por Ian Foster como uma analogia para um sistema onde se pudessem aceder a recursos partilhados tão facilmente como podemos aceder a uma rede de electricidade (*Power Grid*) [1]. Define-se por um sistema *Grid* como uma infra-estrutura que permita e facilite a rápida partilha de recursos computacionais em larga escala para que estes estejam disponíveis para aplicações de computação distribuída. Esta infra-estrutura deve conter ferramentas para a gestão de dados partilhados entre os vários intervenientes, planeamento, agendamento e gestão de pedidos de execução de operações sobre os dados [2].

Desde o início que foi óbvia a utilidade de um sistema destes para suprir as largas necessidades computacionais de projectos científicos. Os projectos *Folding@Home* e *Seti@Home* são os proeminentes exemplos do potencial de um sistema grid,

funcionando em prol de causas científicas. Seguindo de novo a definição de Ian Foster, um sistema *Grid* deve permitir e facilitar a partilha de recursos em computadores ligados à Internet.

Normalmente um sistema grid não faz uso de um cluster ou de máquinas puramente dedicadas para a execução de processos da grid. Por cluster entende-se um conjunto de máquinas sob administração comum, que poderão estar disponíveis para processamento paralelo local, para projectos instalados directamente nestas máquinas. O sucesso dos projectos *Folding@Home* e *Seti@Home* prende-se justamente com o facto de estes poderem fazer uso de computadores pessoais por todo o mundo, aproveitando os tempos mortos (em que o cpu não está em uso intensivo) para efectuarem a recepção de dados e sua computação. O grosso dos cálculos acaba por ser feito e enviado para o servidor sem o utilizador do computador se aperceber disso, ou sem notar carga extra na máquina. Este tipo de aplicações que usam máquinas não dedicadas (mas sim voluntariadas) para efectuar as operações denominam-se de *Cycle Sharing,* na medida em que um indivíduo partilha os ciclos de CPU do seu computador para outras actividades. Nem todos os projectos grid no entanto são apenas baseados em desktops pessoais, havendo vários que fazem uso dos recursos disponibilizados num cluster.

Entende-se portanto por computação grid, um sistema de computação distribuída em que várias máquinas partilham os seus recursos e efectuam trabalho coordenado por uma entidade orquestradora com vista efectuar trabalho de elevada carga computacional, que à partida seria incomportável numa máquina apenas. A entidade central tem como tarefa principal a divisão do trabalho total em partes usáveis por um dos intervenientes (também conhecido como *Task* ou *Job).*

Posto isto estamos em condições para definir concretamente os conceitos de computação paralela/distribuída, cluster e computação Grid:

- **Computação paralela/distribuída:** paradigma de computação em que um programa ou processo faz uso dos recursos (cpu, memória, etc.) de mais do que uma máquina de modo a acelerar a sua performance. Para atingir isto, o programa/processo é dividido em várias partes capazes de ser processadas singularmente, e são enviados para múltiplas máquinas capazes de comunicar em rede, de modo a poderem ser processadas em paralelo.

- **Cluster**: conjunto de computadores sob total controlo administrativo directo disponíveis para qualquer tipo de sistema de computação paralela/distribuída que os seus administradores decidam utilizar.

- **Computação Grid**: modelo de aplicações e infra-estrutura computacional baseado na agremiação de recursos computacionais altamente heterogéneos distribuídos para contribuição e resolução de um problema/tarefa definido.

Para se construir uma rede grid é necessária uma infra-estrutura capaz de organizar os vários intervenientes (máquinas em partilha de recursos) e possibilite o fácil endereçamento, encaminhamento, entrada e saída de nós, e parcialmente

descoberta de recursos. Uma vez que a Internet não fornece suporte directo para este tipo de operação torna-se necessário introduzir uma camada de abstracção por cima do protocolo TCP/IP, denominada de rede *Overlay (Overlay Network)*. Uma rede Overlay designa-se em poucas palavras por uma rede super imposta numa rede TCP/IP. É uma simples camada de abstracção por cima de uma rede que permite novas funcionalidades que seriam difíceis de conseguir de outra forma.

As infra-estruturas comuns para computação grid são infelizmente, pouco resistentes devido a todo o trabalho ser coordenado por uma entidade central. Com o objectivo de criar uma rede sem controlo centralizado que permita a execução de tarefas e a partilha de recursos/dados pelos seus intervenientes surge o conceito de overlay *Peer-to-peer*.

1.2. *Peer-to-peer*

Uma rede P2P define-se por uma infra-estrutura de rede que permite facilidade na distribuição de dados ou recursos por várias máquinas (nós ou pares/peers) associados a um contexto, sem que haja obrigatoriedade de organização central por parte de um servidor. As redes p2p estão hoje em dia muito associadas à partilha de ficheiros online, provando deste modo a viabilidade de um sistema distribuído com ênfase na eficiente pesquisa e transmissão de dados, mas existem no entanto outras aplicações desta tecnologia, sendo exemplos disso software de conversação online ou software de armazenamento distribuído.

Um rede p2p permite portanto a construção de um sistema organizado para transmissão de dados entre vários computadores heterogéneos (nós) sem necessidade de controlo central, em que todos os nós podem participar activamente, sendo estes capaz de resistir a falhas em seus participantes e manter conectividade sem necessidade de intervenção de um agente coordenador.

1.3. Objectivo do Trabalho

É no contexto destas tuas tecnologias que se insere o trabalho a efectuar nesta tese de mestrado. O objectivo do trabalho consistem em implementar um motor que permita que uma aplicação desenhada para ser executada com base num overlay peer-to-peer possa ser executado num ambiente simulado mas com a possibilidade da mesma simulação correr num ambiente de execução paralela num cluster com mais do que uma máquina (ao invés de correr apenas num computador). É portanto criada uma cama de abstracção entre um overlay peer-to-peer completamente simulado (em que não existe uma ligação directa entre um nó do overlay e uma máquina física) e um sistema grid que parte do overlay peer-to-peer e distribui a sua carga computacional por várias máquinas.

A modularidade e portabilidade do Java permite-nos atingir este objectivo com mais segurança pois esta linguagem está implementada nos mais variados sistemas operativos e suporta um conjunto bastante heterogéneo de hardware, e permite-nos

acima de tudo fácil integração com o simulador de overlays peer-to-peer "PeerSim", implementado em Java.

Como ferramenta essencial na construção na camada inferior da aplicação surge o Terracotta.

1.4. Terracotta

Com a emergência destas tecnologias *Grid* e *Peer-to-Peer* e a proeminência do Java como a principal linguagem de programação completamente grátis e aberta (e com implementação para vários tipos de sistema operativo e hardware), começaram a surgir ferramentas para poder executar aplicações Java de uma forma distribuída por uma rede de várias máquinas. O Terracotta é constituído por uma máquina virtual java baseada numa arquitectura cliente/servidor em que é possível executar programas feitos originalmente uma só máquina num cluster, sem haver necessidade de alterar a sua implementação, ou adicionar código. O programa é executado numa máquina sendo esta o coordenador da execução do programa e as seguintes máquinas que executarem o mesmo programa, ligam-se ao coordenador e continuam a execução no mesmo estado em que todas as outras se encontram. A implementação do programa é portanto complemente agnóstica da execução partilhada.

1.5. Peersim

Um simulador *peer-to-peer* consiste numa aplicação que simule uma rede *peer-to-peer* com um número de nós arbitrário numa só máquina, e que permite a aplicação de protocolos e operações sobre esses mesmos nós. Este software permite simular o comportamento de um algoritmo ou estrutura de código numa rede de computadores, sem ter que efectivamente ser feita a instalação de várias máquinas e de uma rede física.

O Peersim é um simulador de overlays peer-to-peer desenhado em Java e que tem à partida à sua disposição a implementação de vários protocolos, além de ter maior escalabilidade do que todos os outros simuladores disponíveis em Java, conseguindo suportar com razoável eficiência até 1 milhão (10^6) de nós simulados [3][4]. É facilmente configurável na medida em que é possível escrever novos protocolos de peer-to-peer de modo a acomodar qualquer tipo de problema.

1.6. Documento

Este documento apresentará os objectivos do projecto relativo a esta tese de mestrado seguido de um levantamento e classificação dos principais projectos relativos às tecnologias apresentadas nesta introdução (overlays *peer-to-peer*, computação em grid e simulação). Por fim o documento termina com uma apresentação da arquitectura da solução e uma metodologia de avaliação capaz de medir o rendimento e performance do sistema.

2. Objectivos do Trabalho

O objectivo desta tese de mestrado é desenvolver uma possível extensão ao Peersim de modo a que este possa correr as suas simulações num conjunto de máquinas em rede em vez de correr apenas num CPU. O propósito deste trabalho será construir uma camada de abstracção que permita construir um sistema grid que permita ao peersim, dividir a sua carga por várias máquinas num regime de cycle-sharing configurável. Cada máquina teria uma configuração que permite saber qual a política de utilização do(s) CPU(s) disponíveis, de modo a que esta apenas partilhe parte dos seus recursos (ou apenas participe na execução quando está em repouso ou *idle)* deixando a máquina disponível para outros usos. Para atingir este objectivo far-se-á uso da máquina virtual Java distribuída Terracotta. Sendo o Peersim um projecto open source escrito em Java, é possível alterar a sua source e desenvolver uma possível extensão ou plug-in que faça uso das funcionalidades do Terracotta, devidamente adaptadas, para atingir o objectivo pretendido de estender um simulador de topologias *overlay peer-to-peer* em clusters.

3. Trabalho Relacionado

3.1. Redes *Peer-to-peer*

As várias implementações diferentes de redes *peer-to-peer* podem ser categorizadas segundo o seu grau de centralização e se a rede é ou não estruturada.

Uma rede *peer-to-peer* estruturada mantém em execução uma estrutura semelhante a uma *Hash Table* distribuída onde mantém informação sobre todos os intervenientes da rede de forma a manter a sua localização constantemente disponível e mantém também catalogados todos as chaves de modo a que seja rápida a sua localização no overlay. Uma chave representa algo pelo qual um nó é responsável (numa aplicação de partilha de dados, uma chave poderá representar um ficheiro ou um bloco de dados único, num servidor de DNS uma chave representará um par nome/endereço único, numa biblioteca poderá representar um artigo, etc.). A maneira como esta estrutura é mantida depende de implementação para implementação, pois a maneira como os nós contactam uns com os outros varia, sendo que normalmente cada nó só conhece um conjunto limitado de outros nós na rede. A cada nó, é atribuído um identificador único na rede, e a cada chave é também é atribuída um identificador único. Isto permite a construção de um grafo de toda a rede em que rapidamente se identifique qual o nó responsável por determinada chave.

A manutenção de uma rede estruturada é bastante complexa, pelo que é complicado e ineficiente manter coerência quando existe um grande volume de nós a entrar e a sair da rede com o seu conjunto de chaves, pelo que quase todas as populares são redes de partilhas de dados não são estruturadas.

Uma vez que todos os overlays *peer-to-peer* estruturados são completamente descentralizados só se torna relevante analisar o grau de centralização em redes não estruturadas. Este é definido pela necessidade de existir uma semi-coordenação

central na rede. Semi-coordenação, porque havendo total coordenação por parte de um agente servidor (na medida em que todas as operações são controladas por um servidor central), a semântica por trás de uma rede *peer-to-peer* deixa de estar associada ao sistema em questão. Um overlay *peer-to-peer* pode ser portanto:

- **Completamente descentralizado.** Todos os nós têm o mesmo estatuto na rede na rede. Esta alternativa tem a desvantagem de não haver um mínimo de controlo sobre o estado total da rede, e limita as opções para uma máquina se ligar ao overlay, pois este tem que conhecer um participante arbitrário na rede para se poder juntar à rede.
- **Parcialmente descentralizado ou híbrido**. Existem alguns nós denominados de *super-peers* que coordenam as entradas e saídas da rede, mas que por si só, apenas um nó pode não ter conhecimento da rede inteira.
- **Completamente centralizado.** Existe um servidor central que controla todas as entradas e saídas da rede e que pode ou não ter conhecimento dos objectos ou blocos de dados contidos nos nós. A desvantagem neste desenho é óbvia. O servidor central é um ponto de falha para toda a rede. [5]

Portanto, à semelhança de [6] a seguinte lista de trabalho efectuado na área peer-to-peer será classificado segundo a sua centralização e estruturação sendo também comparados os seguintes aspectos:

- **Aplicação**, no que toca a qual a utilização final dos programas que implementa o algoritmo (uso geral, partilha de ficheiros, etc.)
- **Tempo de Procura**, dado por uma função do número de nós da rede e outras variáveis da rede (apenas nas redes estruturadas)
- **Tipo de Procura,** descrição simples do algoritmo de lookup (ex.: procura no servidor, procura no *super-peer*, procura por flood, etc.) (apenas nas redes não estruturadas)

Os projectos seguintes serão apresentados com a seguinte classificação: **Nome do Projecto [Estruturação, Centralização, Aplicação, Tempo de Procura /Tipo de Procura].**

3.1.1. Projectos correntes

Chord [Estruturado, Completamente descentralizado, Uso geral, O(logN)]. O Chord é um sistema de lookup de chaves em que os nós estão organizados num espaço de identificadores circular. Cada um dos nós é representado por uma identificador, derivada de uma função de hashing sobre o seu endereço. Os nós são ordenados pelo seu identificador num anel virtual (chord ring) em que um nó x é designado de sucessor de y se o seu identificador for igual ou imediatamente a seguir ao identificador de x. Chegando ao fim do espaço de endereçamento de nós, o sucessor é o nó cujo identificador é igual ou imediatamente a seguir a 0. Nenhum nó

conhece a totalidade da rede, mantendo informação apenas um número máximo de identificadores em memória (mais especificamente log(N) [6] para N igual ao número de peers na rede). Este desenho tem a clara vantagem de permitir entradas e saídas de nós sem haver necessidade de propagar a alteração por toda a rede (apenas os nós vizinhos precisam de alterar o seu estado quando ocorre uma saída ou uma entrada).

A cada chave é atribuído também um identificador sendo uma função de hash sobre a própria chave. Cada nó é responsável por aproximadamente o mesmo número de chaves sendo que uma chave k é armazenada no nó cujo identificador seja igual a k ou imediatamente seguinte chamando-se o nó sucessor da chave k. Quando um nó `n` entra na rede, o seu nó sucessor transfere-lhe algumas chaves que estavam sob seu controlo mas que agora pertencem a `n`. Quando um nó sai da rede, todas as suas chaves são transmitidas para o seu sucessor.

Quando um peer precisa de localizar uma chave envia uma mensagem ao seu vizinho, que será propagada até chegar a um nó com identificador igual ou superior ao identificador da chave. O endereço do nó é depois devolvido ao remetente. Se a chave estiver armazenada no predecessor do peer, seria necessário percorrer todos os nós da rede para chegar à chave pretendida. Para isso, o chord faz uso de uma tabela denominada de *finger table.* Esta tabela contém `m` entradas, sendo m o número de bits que existem no espaço de identificadores da rede. Sendo k o identificador do peer, as entradas da *finger* table correspondem aos nós onde estão armazenadas as chaves de $k+2^0$ até $k+2^m$. Usando esta tabela, o peer pode procurar directamente o nó imediatamente abaixo da chave pretendida e começar o lookup a partir daí, optimizando o processo. [7][8]

Tapestry [Estruturado, Completamente descentralizado, Uso geral, $O(\log_B N)$]. O Tapestry é um sistema de lookup semelhante ao Chord na medida em que os nós estão organizados num espaço distribuído de identificadores mas com um mecanismo de lookup completamente diferente. Este sistema usa uma variante da técnica de procura Plaxton para localizar o nó correspondente à chave pretendida. Cada nó mantém uma tabela de endereçamento dos seus vizinhos, com vários níveis, correspondentes ao número de dígitos do maior identificador possível. Cada nível armazena x entradas (Sendo x a base dos endereços de endereçamento. x é igual a 10 caso os endereços sejam decimais, 2 caso sejam binários, 16 caso sejam hexadecimais, etc).

	Level 5	Level 4	Level 3	Level 2	Level 1
Entry 0	07493	x0493	xx093	xxx03	xxxx0
Entry 1	17493	x1493	xx193	xxx13	xxxx1
Entry 2	27493	x2493	xx293	xxx23	xxxx2
Entry 3	37493	x3493	xx393	xxx33	xxxx3
Entry 4	47493	x4493	xx493	xxx43	xxxx4
Entry 5	57493	x5493	xx593	xxx53	xxxx5
Entry 6	**67493**	x6493	xx693	xxx63	xxxx6
Entry 7	77493	x7493	xx793	xxx73	xxxx7
Entry 8	87493	x8493	xx893	xxx83	xxxx8
Entry 9	97493	x9493	xx993	xxx93	xxxx9

Figura 1 Exemplo de tabela de vizinhos de um nó com identificador 67493 [5]

Como exemplo do funcionamento do algoritmo Plaxton [9], temos a tabela na figura 1 que mostra a tabela correspondente a um nó com identificador 67493. Caso este nó queira enviar uma mensagem para o nó 12345, ele irá procurar a entrada número 5 do nível 1, que lhe indica o sufixo xxxx5. Cada uma destas entradas traduz-se para o endereço do nó mais perto de 67493 cujo identificador termina em 5. Vamos assumir a título de exemplo que é o nó 45235. Este novo nó irá reencaminhar o pedido usando o mesmo procedimento mas partido do nível 2 até chegar ao nó destinatário.

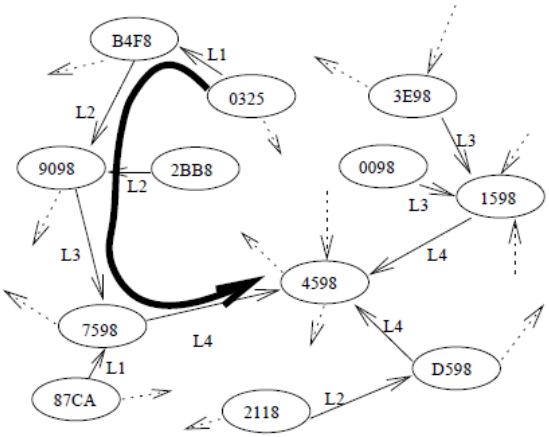

Figura 2 Exemplo do reencaminhamento de uma mensagem de 0325 para 4598 [8]

Esta técnica de Plaxton garante que no máximo, a mensagem irá ser reencaminhada por B-1 máquinas, sendo B o número de níveis na tabela. [6][7][10]

Pastry [Estruturado, Completamente descentralizado, Uso geral, $O(\log_B N)$]. O pastry é um overlay com um funcionamento muito semelhante ao Tapestry na medida em que funciona também com uma procura baseada na técnica de Plaxton, mas que no entanto usa uma tabela de vizinhança bastante mais complexa (continuando a ser baseada no entanto na separação dos vários dígitos do identificador de um nó. A construção da tabela de vizinhança deixa de ser baseada na base dos endereços mas sim no número de bits da base dos endereços. A procura é efectuada primeiro pelo prefixo do endereço e depois pela chave funcionando de modo semelhante ao sistema Chord, na medida em que os identificadores têm que estar também organizados num espaço de endereçamente circular. Este sistema é mais eficiente na medida em que na maior parte dos casos consegue fazer o lookup de um nó na rede em menos do que $O(\log_B N)$ "saltos" [6] sendo B a base dos identificadores[7][11]

Kademlia [Estruturado, Completamente descentralizado, Uso geral, $O(\log_B N)+c$]. O sistema Kademlia difere-se dos anteriores na medida em que é baseado numa métrica de distância entre os seus nós. Esta distância é medida usando uma operação lógica XOR (eXclusive OR) entre os bits que representam os identificadores. Esta distância é usada para efectuar a procura de uma determinada chave. Cada nó mantém uma tabela com 160 níveis (correspondentes aos 160 bits da chave), contendo cada nível uma lista de nós cujo valor da função de distância está entre 2^i e 2^{i+1}. Esta lista é ordenada pela última visita feita aos nós em questão estando em primeiro lugar os nós visitados à menos tempo. Quando é efectuada a procura de uma chave o nó escolhe X nós da lista, calcula a sua distância e envia pedidos de procura da chave em paralelo e de forma assíncrona para eles repetindo o processo. O nó de destino é o que apresentar a distância mais curta. [6][12]

CAN [Estruturado, Completamente descentralizado, Uso geral, $O(d.N^{1/d})$]. O CAN é um overlay *peer-to-peer* cujos nós se encontram organizados segundo um espaço cartesiano de n dimensões (comparativamente ao anel circular Chord e a malha Plaxton do Tapestry). A cada nó é atribuído um ponto no espaço de endereçamento cartesiano. O espaço total de endereçamento é particionado de modo a que cada nó ocupe uma "zona" cujo centro são as coordenadas do mesmo ponto. Assim sendo, dependendo do número de dimensões cartesianas na configuração do sistema, cada nó vai ter um conjunto de zonas adjacentes ocupadas que serão os vizinhos deste nó.

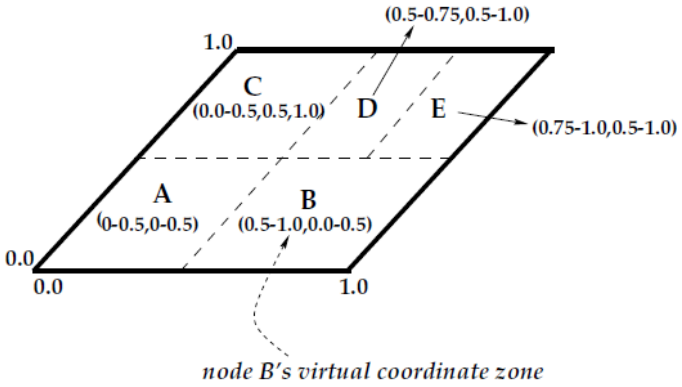

Figura 3 Espaço cartesiano virtual onde os nós são endereçados

Cada pedido de procura de uma chave vai ser propagado pelos vários vizinhos até ser encontrado. Embora cada nó só mantenha uma tabela com os endereços dos seus vizinhos, a performance deste sistema pode ser vantajosa devido à possibilidade de configurar o formato do espaço cartesiano e no fundo em quantas "direcções" são propagados os pedidos de procura de uma chave. A performance da procura de uma chave é então $O(d.N^{1/d})$ [6] sendo d o número de dimensões do espaço de endereçamento. [13]

Viceroy [Estruturado, Completamente descentralizado, Uso geral, O(logN)]. A rede Viceroy foi concebida de modo a aproveitar vários conceitos de desenhos anteriores de modo a encontrar uma solução óptima combinando com as vantagens de uma rede *Butterfly*. Os nós são mapeados num anel virtual (semelhante ao usado no Chord, em que uma chave é armazenada no nó cujo identificador é igual ou imediatamente seguinte ao identificador da chave). Além do seu identificador (que é um decimal atribuído aleatoriamente entre 0 e 1), um nó também é associado a um nível que é escolhido aleatoriamente entre 1 e log N sendo N uma estimativa do total de nós na rede. Além do anel global de endereçamento, o sistema também possui um anel por cada nível onde se relacionam os nós pertencentes ao mesmo nível. Os nós ligam-se entre níveis também, por meio de uma rede *Butterfly* em que cada nó de um nível está ligado a dois nós do nível L +1 e a um nó do nível L -1 sendo L o nível do nó.

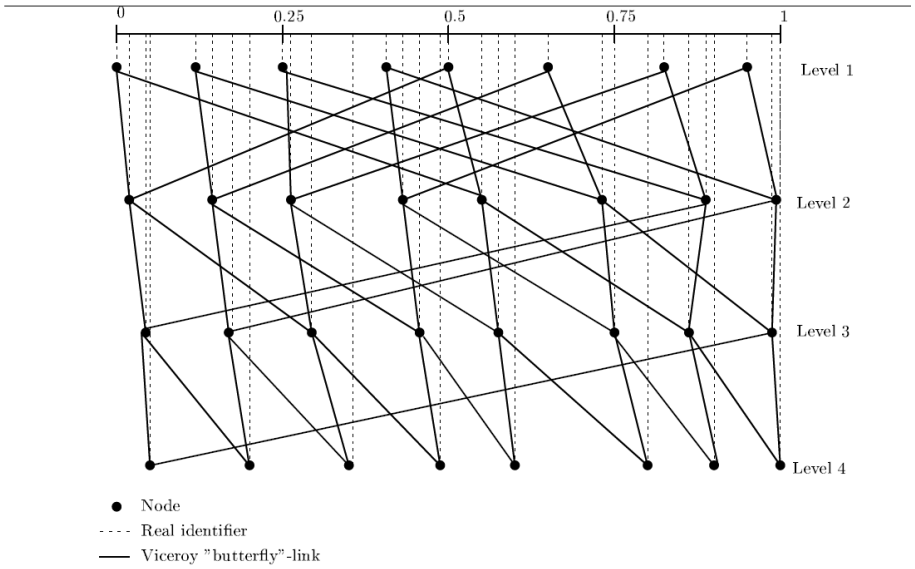

Figura 4 Rede Viceroy com 4 níveis sendo a primeira linha o anel onde estão endereçados todos os nós

A procura de uma chave é efectuada começando por seleccionar o nó nível acima (l -1) e propagando o pedido para os níveis de baixo usando as ligações disponíveis. O algoritmo calcula a distância entre o nó actual e a chave pedida e, sabendo que no nível i a distância ao nó nunca é mais do que $1/(2^{i-1})$, se esta for igual superior a $1/(2^{i})$ decide seguir a ligação da direita, caso contrário segue a da esquerda. Quando é atingido um nó no nível mais abaixo ou cujo identificador seja superior à chave (e não a tenha), o algoritmo percorre o anel global no sentido correcto até encontrar a chave.

A performance deste algoritmo é de O(log N) [6] sendo N o número de nós na rede. [7][14]

Koorde [Estruturado, Completamente descentralizado, Uso geral, O(log n/ log log n)]. O Koorde é um overlay baseado no Chord, que consegue manter a performance de procura (O(log N)), mantendo um número constante de vizinhos por nó. A performance de procura é variável consoante o número de vizinhos que cada nó mantém, O(log n/ log log n) sendo n o número de nós e O(log n) o número de vizinhos mantidos por nó. Para este efeito, os vizinhos do Koorde não são mantidos por uma finger table (Chord) mas sim por um grafo de De Bruijn. Através de operações de shift nos bits dos identificadores, os pedidos de routing são propagados pelo grafo e terminam no nó pedido. [15]

Cyclone [Estruturado, Completamente descentralizado, Uso geral, O(log n)]. O Cyclone é um overlay que permite formar vários clusters de nós e agregá-los numa rede apenas, mantendo uma hierarquia. Os identificadores são atribuídos aleatoriamente mas são divididos num prefixo e num sufixo, sendo o sufixo o

identificador do cluster. Usando apenas o prefixo os vários clusters podem-se organizar segundo anéis tipo Chord autónomos. Usando o identificador completo, os vários clusters podem ser combinados num único cluster. As procuras são efectuadas à semelhança do Chord sendo os pedidos reencaminhados para outro cluster quando é atingido o maior nó pertencente ao cluster actual e que é menor do que o nó de destino. Esta aplicação do Cyclone em redes do tipo Chord denomina-se de Whirl. [16]

	Centralização	Aplicação	Tempo de procura	Variáveis do Sistema
Chord	Completamente descentralizado.	Infra-estruturas capazes de suportar vários tipos de aplicação.	O(logN)	N = número de nós da rede
Tapestry			$O(\log_B N)$	N = número de nós da rede B = número de bits de endereçamento dos nós
Pastry			$O(\log_B N)$	N = número de nós da rede B = número de bits de endereçamento dos nós
Kademlia			$O(\log_B N+C)$	N = número de nós da rede B = número de bits de endereçamento dos nós C = constante pequena
CAN			$O(d.N^{1/d})$	N = número de nós da rede D = número de dimensões do espaço cartesiano virtual
Viceroy			O(logN)	N = número de nós da rede
Koorde			O(logN/ log logN) sendo logN o número de vizinhos por nó	N = número de nós da rede
Cyclone			O(logN)	N = número de nós da rede

Tabela 1 Comparação de overlays peer-to-peer estruturados

Gnutella [Não Estruturado, Completamente descentralizado, Partilha de ficheiros, Procura por flood pelos vizinhos]. O Gnutella é um serviço de partilha de ficheiros completamente descentralizado em que os seus participantes assumem tanto a tarefa de clientes (fazendo pesquisas de ficheiros pela rede e recebendo os resultados) como a tarefa de manter o funcionamento da rede reencaminhando os pedidos de procura de ficheiros para outros nós. Quando é feita uma procura, o nó envia o pedido todos os vizinhos que este tem conhecimento. À medida que os nós contactados encontrarem o ficheiro em questão, respondem para o remetente. Uma vez que a rede é completamente descentralizada, os nós podem entrar e sair à vontade sem grande impacto no funcionamento global da rede. Esta é também altamente resistente a falhas pois não existem nenhuns pontos de falha tais como servidores ou índices centrais. Para um nó se juntar à rede, este tem que conhecer pelo menos um nó que esteja activo no overlay. [7]

FreeHaven [Não Estruturado, Completamente descentralizado, Publicação anónima de dados]. Além do simples serviço de ficheiros, os overlays *peer-to-peer*

vieram suprir uma necessidade pendente de um serviço de armazenamento de documentos/ficheiros com total anonimato, em que os participantes da rede não podem responsabilizados pelos conteúdos armazenados e que é resistente a tentativas de eliminar os mesmos documentos. O FreeHaven foi a resposta a esta necessidade. É um overlay totalmente descentralizado e a própria falta de estrutura permite total anonimato dos autores, publicadores, leitores, servidores e documentos (na medida em que não existe nenhuma ligação entre o documento e os vários intervenientes).

A sua arquitectura é baseada num conjunto de nós sem controlo central funcionando com um sistema de confiança. Para se publicar um documento é preciso primeiro encontrar um servidor disposto a armazená-lo. Este ficheiro é repartido em várias partes e é alojado e replicado pelos vários nós da rede e vai sendo movido de nó em nó consoante o seu grau de confiança (cada servidor vai acumulando os endereços sobre todos os nós na rede à medida que vão sendo transferidos blocos de dados). O grau de confiança é estabelecido pelo historial de armazenamento. Quando um servidor aceita armazenar um documento, compromete-se a fazê-lo por um período definido de tempo específico. Se o nó libertar o documento ou sair da rede antes desse tempo o seu grau de confiança diminui. Cada nó mantém uma base de dados com os nós que mantêm ficheiros que este lhe enviou, e também um registo com o grau de confiança dos nós com que já trabalhou. Tal como o Gnutella, um nó/servidor Free Haven tem que ser inicializado com um endereço de um nó conhecido na rede. [7][17]

Freenet [Levemente Estruturado, Completamente descentralizado, Publicação anónima de dados, Pesquisa por chave e texto descritivo nó a nó]. Semelhante ao Free Haven, a rede freenet tem como objectivo a publicação de textos e ficheiros sob anonimato no que se denomina numa rede *censorship-free.* A rede freenet pode-se categorizar por ser uma rede completamente descentralizada mas apenas semi-estruturada, na medida em que cada nó mantém uma tabela de encaminhamento com alguns nós com correspondência entre chaves (representando documentos, ficheiros, blocos de dados, etc.) e nós na rede representando sugestões e os nós mais visitados. [36] As pesquisas são efectuadas através da escolha de parte dos nós conhecidos e são reencaminhados ao longo da rede toda, até se atingir um resultado. Uma vez que a tabela de encaminhamento é construída dinamicamente e pode não abranger os resultados requeridos, a procura pode não devolver resultados levando a que o overlay não possa ser considerado estruturado. [7][18]

Kazaa [Não Estruturado, Parcialmente centralizado, Partilha de ficheiros, Pesquisa pelos ficheiros indexados no super-peer]. O Kazaa é uma rede de partilha de ficheiros muito popular muito semelhante à rede Gnutella mas com a existência de uma hierarquia de nós em que passam a existir super nós (Supernodes) responsáveis por algum controlo na operação da rede. Cada nó normal na rede está associado a um super nó que mantém conhecimento do catálogo de ficheiros que o nó normal está a partilhar. O sistema torna-se bastante mais rápido nas suas procuras que uma rede Gnutella, na medida em que as pesquisas são realizadas apenas nos super nós mas é obviamente menos resistente a falhas, uma vez que a falha de um super nó implica que os seus nós associados se tenham que ligar à rede de novo (e a um novo super nó). [19]

eDonkey [Não Estruturado, Parcialmente centralizado, Partilha de ficheiros]. A rede eDonkey é constituída à semelhança da rede Kazaa por um conjunto de nós servidores (Supernodes) que contém também os metadados relativos aos ficheiros partilhados pelos nós clientes associados a ele. A rede eDonkey, ao particionar os ficheiros em várias porções transmitíveis, permite aos clientes fazer a transferência do mesmo ficheiro de vários nós ao mesmo tempo fazendo um download multi-parte à semelhança de um download por um browser de um servidor http. Cada ficheiro partilhado é identificado por uma função de hashing que permite que o mesmo ficheiro com nomes diferentes seja identificado pelo servidor/super-node como sendo o mesmo ficheiro. Nós com o ficheiro na lista de transferências mas que ainda não possuem o ficheiro inteiro também podem enviar as suas partes a outros clientes. À semelhança da rede Kazaa e Gnutella, as transferências são feitas directamente entre os nós clientes. Quando um nó cliente procura um ficheiro o seu nó servidor devolve-lhe a lista de ficheiros que correspondam ao nome pretendido e que nele estejam indexados. [6][20]

BitTorrent [Não Estruturado, Parcialmente centralizado, Partilha de ficheiros]. A arquitectura da rede bit torrent é também centralizada, mas contrariamente às redes Kazaa e eDonkey, os servidores são dedicados e não são nós participantes na rede. Estes servidores (chamados trackers) são participantes que se dedicam exclusivamente a manter uma lista de todos os nós que possuem o conjunto de ficheiros em questão. Os torrents são ficheiros que descrevem um conjunto de ficheiros (possivelmente organizado em directorias com vários ficheiros), seus tamanhos, nomes e checksums e o endereço do tracker responsável pelo conjunto de ficheiros. As transferências são depois efectuadas directamente entre os nós clientes exactamente à semelhança da rede eDonkey. Quando um tracker não está em funcionamento os ficheiros pelos quais este é responsável deixam de estar disponíveis. A rede Bit Torrent manté também um sistema de incentivo que favorece nós que têm elevada taxa de partilha e upload dando-lhes mais prioridade nas listas de espera. Os ficheiros torrent não estão disponíveis na própria rede e têm que ser servidos externamente à rede, de modo a que quando um nó deseja procurar um ficheiro ou conjunto de ficheiros, este tem já que ter o ficheiro torrent correspondente. Sites como TorrentReactor.net ou o Mininova.org, alojam ficheiros torrent e possuem motores de busca internos para sua pesquisa. [6][22]

Napster [Não Estruturado, Híbrido descentralizado, Partilha de música]. A rede Napster foi provavelmente o primeiro sistema *peer-to-peer* de partilha de música a tornar-se conhecido e a ser usado em larga escala (tendo sido também o primeiro a ser alvo de um processo judicial bastante mediático que levou ao fim do seu funcionamento original). A sua arquitectura é composta por um único servidor central e os nós clientes que armazenam os ficheiros em si. O servidor central serve apenas de directório central com todos os ficheiros disponíveis na rede. Quando um cliente se liga à rede, transmite a sua lista ao servidor, e quando efectua uma procura o mesmo servidor devolve-lhe a list de nós cujo nome do ficheiro coincide com o padrão de procura. Tal como todas as outras redes analisadas, as transferências são efectuadas entre os pares. Uma vez que o servidor central não coordena todas as operações na

rede, a rede é classificada como Híbrida Descentralizada (sendo um sistema híbrido entre uma rede puramente centralizada e uma rede semelhante à Kazaa). [7]

	Centralização	Aplicação	Tipo de procura	Tipo de decentralização
Gnutella	Completamente descentralizado.	Partilha de ficheiros	Procura por flood pelos vizinhos	Todos os nós são equivalentes, não existindo hierarquia.
FreeHaven		Publicação anónima de dados		Todos os nós são equivalentes, sendo classificados por um sistema de confiança.
Freenet		Publicação anónima de dados	Publicação anónima de dados, Pesquisa por chave e texto descritivo nó a nó.	Aproximação ao modelo estruturado. Os nós estão endereçados mas não existe um algoritmo de procura dependente da estrutura de endereçamento de nós.
Kazaa	Parcialmente centralizado	Partilha de ficheiros	Pesquisa pelos ficheiros indexados no *super-peer*.	Os nós são associados aos seus super-peers e os super-peers comunicam uns com os outros.
eDonkey		Partilha de ficheiros	Pesquisa pelos ficheiros indexados nos nós servidores.	O nós são associados a um dos vários servidores disponíveis.
BitTorrent		Partilha de ficheiros	A procura é efectuada pelo tracker com que o software foi inicializado (através do respectivo ficheiro .torrent).	Não existe um servidor central nem super-peers mas um tracker que gere os nós que contêm os ficheiros pelo qual é responsável.
Napster	Híbrido descentralizado	Partilha de música	Pesquisa do ficheiro no servidor central.	Existe apenas um servidor central com todos os ficheiros disponíveis indexados e o nó a qual o ficheiro pertence.

Tabela 2 Comparação de overlays peer-to-peer não estruturados

3.2. Sistemas Grid e Middleware para processamento paralelo/distribuído

O conceito de sistema Grid engloba um conjunto muito vasto de projectos diferentes com métodos e objectivos radicalmente diferentes. Desde a aplicação final do projecto; qual o regime de partilha de recursos, tais como:

- **Desktop grid**: A aplicação corre como uma normal aplicação de desktop em que o utilizador voluntaria-se a oferecer parte ou totalidade dos recursos do seu computador pessoal para o projecto. Qualquer tipo de computador pessoa pode ser utilizado e pode apenas contribuir para grid quando este não está a ser utilizado.
- **Aplicação dedicada**: A aplicação transforma o computador numa unidade dedicada a 100% para computação na grid.
- **Marketplace:** O vários participantes na grid registam os seus recursos e aceitam cedê-los a outros participantes da rede para que estes possam executar as suas tarefas.

e qual o tipo de grid, segundo a seguinte classificação:

- **Grid Institucional:** Grid dedicada ao serviço interno de uma instituição ou empresa.
- **Grid Cooperativa/Voluntária:** Grid aberta a quem estiver disposto a oferecer os seus recursos para um objectivo comum num modelo aproximado a uma rede *peer-to-peer*.
- **Toolkit:** Software desenhado para construir aplicações de computação distribuída/paralela e sistemas grid.

Os projectos seguintes serão apresentados com a seguinte classificação: **Nome do Projecto [Tipo de Grid, Regime de Partilha, Aplicação].**

3.2.1. Projectos correntes

SETI@Home [Grid Cooperativa/Voluntária, Desktop grid, Processamento de sinal áudio]. O projecto SETI @ Home foi concebido para suprir a falta de poder computacional para processar os dados obtidos pelo projecto SETI (Search for Extra Terrestrial Intelligence). O projecto SETI faz uso do maior radio telescópio do mundo, localizado em Arecibo, Porto Rico para pesquisar emissões no espectro das ondas de rádio em busca de sinais artificiais, possivelmente oriundos de civlizações extra terrestres. O radio telescópio armazena todos as suas observações em fitas de dados (fitas de 35 GB, gravadas num rácio de 5 MB por segundo, contendo cada fita 35GB). Estas fitas depois de cheias são enviadas por correio comum para a universidade de Berkeley nos Estados Unidos (pois não existe ligação de banda larga disponível em Arecibo) onde são introduzidas no servidor do sistema. Os dados são extraídos das fitas e particionados segundo as várias bandas de frequências e depois em blocos de 107 segundos. Este blocos são colocados no servidor de dados e resultados e são posteriormento transmitidos para máquinas com o software cliente a correr, processados e enviados de volta.

Para uma máquina ser participante da rede, tem apenas que possuir uma ligação à internet e ter o software cliente instalado. Este corre como uma aplicação cycle-sharing na medida em que pode ser configurado para correr apenas quando a máquina está em *idle* ou para correr constantemente com a prioridade de processo no mínimo, mantendo a disponibilidade da máquina para outros trabalhos. A aplicação mantém também estado em disco para que os resultados da computação não se percam caso o servidor esteja em baixo. [24]

Folding@Home [Grid Cooperativa/Voluntária, Desktop grid, Processamento de simulações em bioquímica]. À semelhança do projecto SETI@Home, o projecto Folding@Home surgiu a partir de um problema com um custo computacional demasiado elevado para as máquinas disponíveis na altura. Neste caso trata-se de um problema relativo a medicina e biologia denominado de *Protein Folding*. A pesquisa deste problema permite ganhar conhecimento acerca do desenvolvimento de terapias/vacinas para várias doenças. Para dar resposta a isto foi criado um modelo de simulação que faz uso de uma rede semelhante à SETI@Home

para enviar os dados para várias máquinas e receber os seus resultados. Sendo um projecto com uma importância imediata maior do que o SETI @ Home, rapidamente foram desenvolvidos outros tipos de clientes para que a rede possa ser alargada além dos desktops em cycle-sharing. Exemplos são o cliente para Playstation 3 e principalmente um cliente para tirar partido de GPUs em placas gráficas. Este último permite correr a aplicação cliente em cluster de placas gráficas (normalmente usados para fazer tratamento de vídeo e CGI em filmes) e usar os seus processadores de alto desempenho mantendo-se parte da grid como mais um nó participante. [25][26]

BOINC [Toolkit, Desktop grid, Várias aplicações]. Tanto o projecto SETI@Home como o projecto Folding@Home são construídos sobre uma framework de construção de sistemas Grid denominada de BOINC. A infra-estrutura BOINC foi desenhada para que um cientista de investigação consiga instalar um sistema Grid com pouco trabalho inicial e que permita ajudá-lo nas suas necessidades de computação usando uma rede de computadores voluntários. Esta infra-estrutura começou como parte do projecto SETI@Home e acabou por ser separada num infra-estrutura de middleware dado o potencial da tecnologia em outros projectos.

A framework permite correr uma aplicação escrita em C ou C++ com muito poucas alterações ao código, e permite manter a rede em funcionamento com bastante pouca manutenção semanal. [27]

Grid4All [Grid Institucional, Marketplace, Partilha de recursos entre os vários intervenientes]. O projecto Grid4All faz parte de uma iniciativa da União Europeia para criar uma rede completamente democrática e aberta que permita a utilização de recursos computacionais disponíveis por qualquer pessoa ou empresa. O projecto pretende facilitar a gestão e os custos da criação e manutenção de recursos informáticos para suporte de infra-estruturas TI criando um sistema que seja completamente autónomo e gerido por si próprio, baseado em tecnologia *peer-to-peer*. O formato abstracto da rede assemelha-se a um mercado onde existem ofertas e pedidos de recursos de computação ou armazenamento sendo estes recursos alocados a um indivíduo ou uma organização por um determinado preço por unidade de tempo. [28][29]

OurGrid [Toolkit, Marketplace/Aplicação dedicada, Partilha de recursos entre os vários intervenientes]. Num objectivo semelhante ao projecto Grid4All surge o projecto OurGrid que consiste numa rede aberta e grátis onde os participantes servem simultaneamente de fornecedores e requerentes de recursos. A OurGrid oferece um toolkit que permite construir um pequeno cluster que pode ser rapidamente usado para correr aplicações distribuídas e que pode também ser ligado à comunidade OurGrid onde irá partilhar os seus recursos com o resto da rede.

As aplicações a correr na rede são limitadas apenas a aplicações que podem ser divididas em várias tarefas (*tasks*) divisíveis e não dependentes umas das outras. Estas aplicações são denominadas de aplicações BoT (Bag of Tasks). Estas tarefas são executadas em máquinas virtuais criadas na altura de correr a tarefa e destruídas no final, mantendo total segurança e anonimato na tarefa a correr. [30]

Xgrid [Toolkit, Aplicação dedicada, Computação Distribuída]. O Xgrid é um software desenvolvido pela Apple para o seu sistema operativo Mac OS para efectuar computação distribuída numa rede local de computadores Mac OS. Usa o serviço Rendezvous para fazer a descoberta de nós para a rede, sem necessidade de configuração e o overlay *peer-to-peer* BEEP[27] para efectuar a comunicação necessária. Fazendo uso de uma configuração user friendly baseada em interface gráfica, o objectivo deste software é apelar a que um investigador ou cientista consiga montar uma grid local sem necessidade de efectuar configurações nem de perder tempo a descobrir novas tecnologias. [31]

XtremWeb [Toolkit, Aplicação dedicada/Desktop Grid, Computação Distribuída]. Um projecto muito semelhante ao OurGrid é o XtremWeb baseado em tecnologia Java. Oferece as mesmas funcionalidades, permitindo várias configurações possíveis, desde uma grid privada sem ligação com o exterior até a uma comunidade semelhante à OurGrid. É possível configurar como participante da grid desde um pc desktop voluntário a correr um cliente em screensaver, até uma server farm por detrás de um router. À semelhança da infra-estrutura BOINC também é possível configurar um servidor de agregação de resultados para onde as máquinas em regime voluntário fazem upload de resultados directamente. [32]

GiGi [Grid Cooperativa/Voluntária, Desktop Grid, Computação Distribuída]. O GiGi é um projecto português em Grid Computing que consiste num motor de execução de gridlets altamente configurável baseado num overlay *peer-to-peer*. Cada gridlet consiste numa tarefa ou unidade de execução, os dados a processar e informação sobre o custo de execução da unidade, em função de memória ocupada e ciclos de cpu necessários. Os gridlets são enviados de nó em nó até a sua execução terminar sendo convertidos em gridlet-results e reenviados para os nós que os submeteram para a rede. Esta função de custo permite um balanceamento da carga da grid mais eficiente que não é possível noutros sistemas. [33]

	Tipo de Grid	Regime de Partilha	Aplicação
SETI@Home	Grid Cooperativa/Voluntária	Desktop grid	Processamento de sinal áudio captado pelo radiotelescópio do programa SETI.
Folding@Home			Processamento de simulações em proteínas para investigação em doenças incuráveis.
BOINC	Toolkit		Toolkit para construção de desktop grids à semelhança dos projectos @Home.
Grid4All	Grid Institucional	Marketplace	Mercado para oferta e procura de recursos computacionais fornecidos pelos intervenientes à semelhança de um mercado de acções.
OurGrid	Toolkit		Toolkit para construção de grids para computação distribuída ou para participação em sistema marketplace semelhante ao Grid4All.

Xgrid		Aplicação dedicada	Toolkit da Apple para a fácil construção de pequenas grids para computação distribuída orientado a projectos pessoais de cientistas.
XtremWeb		Aplicação dedicada/Desktop Grid	Tookit semelhante ao projecto BOINC que permite construir uma grid para computação distribuída.
GiGi	Grid Cooperativa/Voluntária	Desktop Grid	Infraestrutura para um sistema grid baseado em gridlets, com função de custo de execução associada

Tabela 3 Comparação de sistemas Grid

3.3. Simulação de redes/overlays

Como já foi falado na introdução, um simulador de overlays *peer-to-peer* é desenhado de modo a criar um conjunto de nós virtuais e simular a sua interacção como se de uma rede física se tratasse. À semelhança de [31] e [35] os seguintes simuladores serão classificados segundo a sua arquitectura:

- **Eventos Discretos:** O simulador usa um sistema de eventos em que os vários nós enviam mensagens uns aos outros que são consequentemente sincronizadas e potencialmente atrasadas pelo simulador de modo a manter coerência da simulação.
- **Query-Cycle:** O total da simulação é composta por um conjunto de ciclos de queries, que consistem num conjunto de operações ordenadas de comunicação entre os pares que correm todas sequencialmente até ao final. Um ciclo só termina (e posteriormente só é iniciado o seguinte ciclo) quanto todos os nós tem a resposta a sua query.

extensibilidade a protocolos feitos por medida e escalabilidade (número máximo de nós).

Os projectos seguintes serão apresentados com a seguinte classificação: **Nome do Projecto [Arquitectura, Extensibilidade, Número máximo de nós].**

3.3.1. Projectos correntes

GPS [Eventos Discretos, Extensível a novos protocolos, N/A]. O GPS foi um simulador de overlays *peer-to-peer* baseado em Java e capaz de simular vários protocolos cujo desenvolvimento se encontra inactivo. A framework de simulação é baseada em eventos discretos. Os eventos podem ser acções por parte do utilizador, mensagens trocadas entre os nós ou eventos internos ao simulador. O GPS permite também usar vários tipos de protocolo peer-to-peer na medida em que o simulador é configurável por meio de um componente extensível em que se podem re-implementar funções como pesquisa de nós/chaves, entrada e saída de nós. [34]

P2Psim [Eventos Discretos, Extensível a novos protocolos, 3000 nós]. O P2PSim é um simulador feito em C++ capaz de simular à partida os overlays Chord, Koorde, Kademlia, Accordion, Kelips e Tapestry. Este simulador suporta até 3000 nós usando o motor de simulação baseado em eventos discretos, com garantia de correr a simulação em tempo útil e fornece algumas estatísticas de base sem necessidade de escrever novo código. É também possível implementar novos protocolos, à semelhança do GPS implementado estendendo o software existente e implementando as funções de pesquisa e junção de nó à rede. Infelizmente o projecto encontra-se com pouca actividade actualmente e tem muito pouca documentação. [3][35]

OverSim [Eventos Discretos (inclui simulação de camada TCP/IP), Extensível a novos protocolos, 100.000 nós]. O OverSim é uma framework de simulação de overlays baseada também em eventos discretos, que oferece grande escalabilidade, uma fácil configuração de protocolos (para permitir a implementação de overlays estruturados e não estruturados), um modulo de estatísticas bastante completo, e um visualizador na forma de uma interface gráfica que permite facilmente efectuar a depuração de protocolos do overlay como visualizar a topologia da rede (tanto a nível do overlay como a nível da simulação da camada inferior de simulação). O Oversim permite também modelar a rede ao nível da camada inferior (TCP/IP, UDP, etc) de modo a permitir a simulação de uma rede realista com limites de largura de banda, congestionamento e perda de pacotes ou a utilização de uma rede sem limites de eficiência para o estudo puro do comportamento da rede com elevado número de nós. À semelhança dos outros simuladores referidos, a implementação de novos protocolos faz-se definindo não só as funções de entrada, saída, e procura mas também o tratamento de mensagens, e os mecanismos para se poder efectuar a visualização gráfica da rede. O oversim corre com tempos aceitáveis numa rede simulada até 100.000 nós. [36]

Peersim [Eventos Discretos e Query-Cycle à escolha, Extensível a novos protocolos, 1 milhão de nós]. O Peersim que já foi descrito na introdução, será o simulador a usar no projecto. Este projecto é baseado em Java e permite a utilização tanto de um motor baseado em eventos como de uma simulação baseada no paradigma Query-Cycle. À semelhança dos anteriores projectos, este simulador também é extensível a protocolos feitos por medida mas apresenta uma maior escalabilidade registada do que os mesmos. [3]

	Arquitectura	Extensibilidade	Número máximo de nós
GPS	Eventos Discretos	Extensível a novos protocolos	N/A
P2Psim			3000 nós
OverSim			100.000 nós
Peersim	Eventos Discretos e Query-Cycle à escolha		10^6 (1 milhão) de nós usando o motor Query-Cycle

Tabela 4 Comparação de simuladores *peer-to-peer*

4. Arquitectura da solução

A solução deste projecto de tese de mestrado passará, como já referido, por estender a funcionalidade do simulador P2P Peersim de modo a que passe a efectuar o seu processamento em paralelo sobre um cluster, numa topologia grid, usando para isso as funcionalidades do projecto Terracotta.

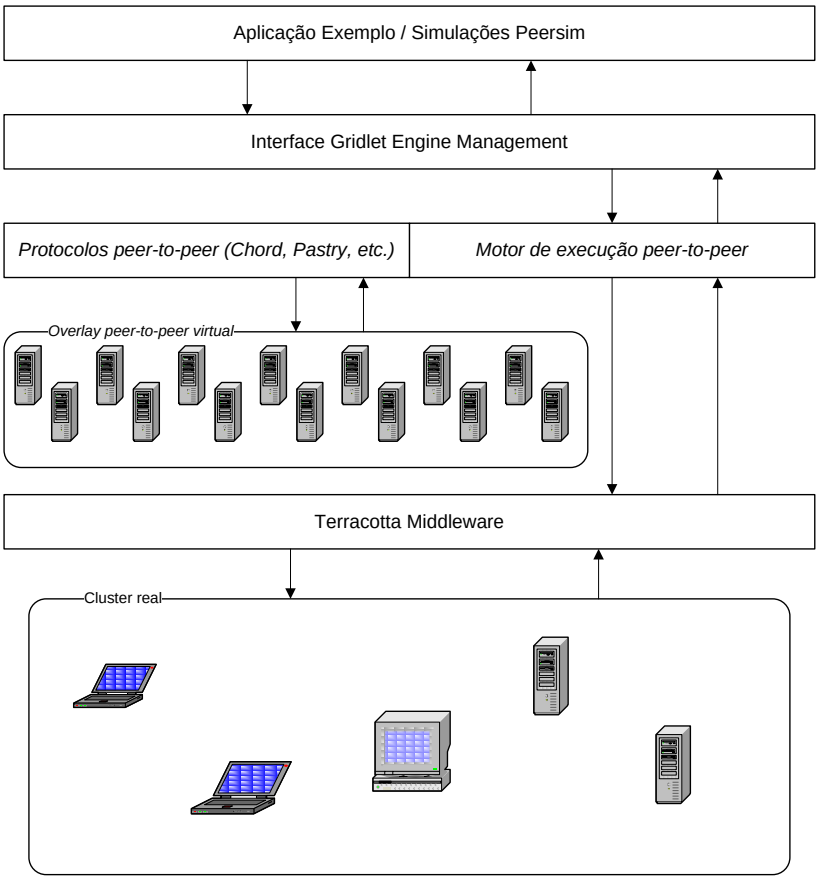

Figure 5 Arquitectura simplificada do simulador *peer-to-peer* modificado

A aplicação exemplo passa a estar em contacto com uma interface do Gridlet Engine em vez de estar em contacto directamente com o Peersim. Esta interface trata de providenciar a mesma interface que o Peersim, mas em vez de enviar os pedidos para o motor de simulação, envia os mesmos para o novo motor de execução *peer-to-peer*. Isto permitir-nos-á usar as próprias simulações providenciadas pelo Peersim para fazer a avaliação do projecto, pois poderá ser feita uma comparação directa com o Peersim original.

A interface do Gridlet Engine irá encaminha os pedidos da simulação para o novo motor de execução peer-to-peer. Esta camada da aplicação irá ser baseada no motor de simulação do Peersim, mas irá integrar as funcionalidades do Terracotta para efectuar a distribuição do trabalho pelos vários intervenientes da grid. Será da sua responsabilidade sincronizar os resultados de modo a manter a simulação coerente (como se corresse numa máquina apenas).

Este motor continuará a suportar os vários protocolos *peer-to-peer* que já eram suportados pelo Peersim. Estes terão que ser adaptados de modo a acomodar a nova implementação do simulador. Na camada inferior da aplicação irá estar o Middleware Terracotta que tratará de fornecer ao motor de execução *peer-to-peer* as suas funcionalidades de memória partilhada entre várias máquinas virtuais Java.

5. Metodologia de avaliação do trabalho

A avaliação do trabalho será feita principalmente com base em medições de tempos de execução de simulações do peersim em várias configurações do sistema. Existem várias conclusões a tirar sobre a eficiência do sistema entre uma grid de várias máquinas ligadas em rede, ou uma grid com menos máquinas mais mais cpus/cpu cores por máquina:

- Medir qual o número de nós máximo/óptimo para uma determinada topologia de rede (Gigabit Lan, Internet, etc) nas várias simulações disponíveis.
- Medir quantos saltos são efectuados para entregar uma simulação
- Medir a memória ocupada durante as simulações e comparar com a execução do simulador original apenas numa máquina.
- Medir a eficiência do sistema no que toca tarefas processadas vs. tarefas perdidas durante a execução paralela da simulação.
- Efectuar testes de stress para determinar qual o máximo de nós suportados no overlay virtual em função do poder computacional da grid (em termos de número de CPU, respectivas velocidades de relógio e total de memória partilhada disponível na grid).

6. Conclusões

Como foi estabelecido na introdução, as tecnologias de computação em grid e *peer-to-peer* estão actualmente a entrar num período de intensa actividade, em que se estão a tornar nas bases de grande parte do software actual. Depois de introduzido este background foi fácil perceber a utilidade do modelo do projecto apresentado nesta tese de mestrado.

Posto isto foi feita uma investigação exaustiva nas áreas de *peer-to-peer,* simulação e tecnologias de computação distribuída e computação em grid. Os

projectos enunciados foram classificados segundo vários parâmetros para que pudesse ser feita uma comparação eficaz entre eles.

Os projectos correntes peer-to-peer englobaram os principais algoritmos de overlay estruturado e alguns exemplos de redes não estruturadas em uso actualmente. Parte dos overlays estruturados virão a ser usados no nosso gridlet engine, portanto a sua classificação segundo tempos de procura e variáveis de execução será importante na avaliação da performance do sistema.

Os projectos correntes em tecnologias grid levantaram as várias escalas de actuação dos projectos e os vários métodos de partilha de recursos entre os intervenientes da grid. Estes exemplos serão úteis de modo a integrar no nosso projecto uma configuração de regime de partilha.

De seguida foi feita uma proposta de arquitectura do sistema simplificada com todos os principais componentes e suas responsabilidades identificadas. O seu desenho feito em três camadas principais permite esconder por completo a natureza distribuída do projecto das camadas de cima, podendo a camada superior (com as qual interage o programa exemplo da simulação) manter uma interface igual à do simulador original.

Por fim, foi planificada uma metodologia de avaliação capaz simultaneamente validar a utilidade e funcionalidade do desenho proposto e efectuar uma comparação directa entre os resultados obtidos e os projectos correntes classificados neste relatório.

7. Anexo

7.1. Planeamento e Calendário do Projecto

- Testar o Peersim usando uma versão compilada a partido seu código fonte para que possa ser feito um estudo do seu código fonte. Efectuar o mesmo procedimento para o Terracotta. **10 de Janeiro a 25 de Janeiro.**
- Identificar código responsável pela geração de mensagens entre nós e sua sincronização no motor de execução *peer-to-peer*. Identificar os problemas levantados na divisão das mensagens por várias máquinas e dificuldades de sincronização. **25 de Janeiro a 15 de Fevereiro.**
- Estudar a possibilidade de exteriorizar o Terracotta de modo a que este funcione como uma biblioteca/plugin externo de modo a servir a comunicação e sincronização de mensagens do motor de execução *peer-to-peer*. **15 de Fevereiro a 30 de Fevereiro.**
- Implementação do primeiro protótipo do Gridlet Engine, consistindo numa versão funcional do projecto capaz de efectuar a divisão dos vários nós da simulação pelas máquinas usadas para teste, e devolver um resultado correcto e consistente com uma simulação feita no Peersim original**. 1 de Março a 15 de Abril**

- Efectuar medições de teste com várias máquinas e correcções necessárias à implementação do primeiro protótipo. **16 de Abril a 7 de Maio.**
- Implementação do segundo protótipo do Gridlet Engine de modo a incorporar configuração de partilha de recursos/prioridade do processo e respectivo balanceamento da carga baseado na configuração de hardware da máquina em questão. **8 de Maio a 8 de Junho.**
- Efectuar novas medições de teste de modo a avaliar a melhoria de performance relativamente ao primeiro protótipo e efectuar as respectivas correcções. **9 de Junho a 20 de Junho**
- Estudar optimização do desenho testando de novo o sistema com várias configurações de hardware. Efectuar resolução de eventuais problemas e correcções necessárias. **21 de Junho a 31 de Julho**
- Produção da dissertação da tese em paralelo com a fase final de implementação. **10 de Junho a 8 de Agosto**
- Entrega final da dissertação. **15 de Setembro**

8. Referências

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 2004.

[2] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," *PHYSICS TODAY*, vol. 55, 2002, pp. 42-47.

[3] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers, "The state of peer-to-peer simulators and simulations," *ACM SIGCOMM Computer Communication Review*, vol. 37, 2007, pp. 95-98.

[4] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai, "A Survey of Peer-to-Peer Network Simulators," *Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK*, 2006.

[5] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing," *HP Laboratories Palo Alto, March*, 2002.

[6] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys & Tutorials, IEEE*, 2005, pp. 72-93.

[7] S. ANDROUTSELLIS-THEOTOKIS and D. SPINELLIS, "A Survey of Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys*, vol. 36, 2004, pp. 335-371.

[8] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM New York, NY, USA, 2001, pp. 149-160.

[9] C.G. Plaxton, "Accessing Nearby Copies of Replicated Objects in a Distributed Environment," *Theory of Computing Systems*, vol. 32, 1999, pp. 241-280.

[10] B.Y. Zhao, J. Kubiatowicz, and A.D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," *Computer*, vol. 74, 2001.

[11] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," *Lecture Notes In Computer Science*, vol. 2218, 2001, pp. 329-350.

[12] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Springer, 2002, p. 263.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," *Proceedings of the 2001 SIGCOMM conference*, ACM New York, NY, USA, 2001, pp. 161-172.

[14] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: a scalable and dynamic emulation of the butterfly," *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, ACM New York, NY, USA, 2002, pp. 183-192.

[15] M.F. Kaashoek and D.R. Karger, "Koorde: A Simple Degree-Optimal Distributed Hash Table," *LECTURE NOTES IN COMPUTER SCIENCE*, 2003, pp. 98-107.

[16] M.S. Artigas, P.G. Lopez, J.P. Ahullo, and A.F.G. Skarmeta, "Cyclone: a novel design schema for hierarchical DHTs," *Peer-to-Peer Computing, 2005. P2P 2005. Fifth IEEE International Conference on*, pp. 49-56.

[17] R. Dingledine, M.J. Freedman, and D. Molnar, "The Free Haven Project: Distributed Anonymous Storage Service," *LECTURE NOTES IN COMPUTER SCIENCE*, 2001, pp. 67-95.

[18] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," *LECTURE NOTES IN COMPUTER SCIENCE*, 2001, pp. 46-66.

[19] J. Liang, R. Kumar, and K.W. Ross, "Understanding KaZaA," *Manuscript, Polytechnic Univ*, 2004.

[20] O. Heckmann and A. Bock, *The eDonkey 2000 Protocol*, Tech. Rep. KOM-TR-08-2002, Multimedia Communications Lab, Darmstadt University of Technology, Dec. 2002, .

[21] A. Legrand, L. Marchal, and H. Casanova, "Scheduling Distributed Applications: The SimGrid Simulation Framework," *Proceedings of the Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03), May*, 2003.

[22] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The Bittorrent P2P File-Sharing System: Measurements and Analysis," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 3640, 2005, p. 205.

[23] X. Li and J. Wu, "Searching techniques in peer-to-peer networks," *Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks*, 2006, pp. 613–642.

[24] D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@ home: an experiment in public-resource computing," *Communications of the ACM*, vol. 45, 2002, pp. 56-61.

[25] S.M. Larson, C.D. Snow, M. Shirts, and V.S. Pande, "Folding@ Home and Genome@ Home: Using distributed computing to tackle previously intractable problems in computational biology," *Computational Genomics*, 2002.
[26] "Folding@home Homepage," *http://folding.stanford.edu/*.
[27] D.P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," *5th IEEE/ACM International Workshop on Grid Computing*, 2004, pp. 365-372.
[28] "Grid 4 All - Grid 4 All," *http://www.grid4all.eu/*.
[29] K. Kotis, G.A. Vouros, A. Valarakos, A. Papasalouros, X. Vilajosana, R. Krishnaswamy, and N. Amara-Hachmi, "The Grid4All ontology for the retrieval of traded resources in a market-oriented Grid."
[30] N. Andrade, L. Costa, G. Germoglio, and W. Cirne, "Peer-to-peer grid computing with the ourgrid community," *Proceedings of the SBRC*, 2005.
[31] D. Kramer and M. MacInnis, "Utilization of a Local Grid of Mac OS X-Based Computers using Xgrid," *High Performance Distributed Computing: Proceedings of the 13 th IEEE International Symposium on High Performance Distributed Computing*, 2004, pp. 264-265.
[32] C. Germain, V. Neri, G. Fedak, and F. Cappello, "XtremWeb: Building an Experimental Platform for Global Computing," *LECTURE NOTES IN COMPUTER SCIENCE*, 2000, pp. 91-101.
[33] L. Veiga, R. Rodrigues, and P. Ferreira, "GiGi: An Ocean of Gridlets on a "Grid-for-the-Masses"," *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, IEEE Computer Society Washington, DC, USA, 2007, pp. 783-788.
[34] W. Yang and N. Abu-Ghazaleh, "GPS: A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent," *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*, 2005, pp. 425-434.
[35] "p2psim: a simulator for peer-to-peer (p2p) protocols," *http://pdos.csail.mit.edu/p2psim/*.
[36] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," *IEEE Global Internet Symposium, 2007*, pp. 79-84.