Proposta de uma metodologia de avaliação de sistemas peer-to-peer baseados em tabelas hash distribuídas

Paulo Ricardo Zanoni[1], **Luis Carlos Erpen de Bona**[1], **Eduardo Cunha de Almeida**[1]

[1]Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19.081 – CEP 81.531-980 – Curitiba – PR – Brasil

{paulo,eduardo,bona}@c3sl.ufpr.br

Abstract. *Among the most used types of peer-to-peer (P2P) networks are the distributed hash tables (DHTs), which are data structures that allow the insertion of data indexed by keys. There is a big amount of DHTs, which may have multiple implementations and be configured through many parameters. However, there is no consensus about the best way to evaluate a DHT performance, which complicates the choice of the ideal implementation for each case. This paper proposes a methodology for evaluating the performance of DHTs that uses a standard set of performance tests based on the most used evaluations found in the literature. An implementation for this methodology and its initial results are also presented.*

Resumo. *Dentre os tipos de redes par-a-par (P2P,* peer-to-peer*) mais utilizados estão as tabelas hash distribuídas (DHTs,* distributed hash tables*), que são estruturas de dados que permitem a inserção de dados indexados por chaves. Existe uma grande quantidade de DHTs, que podem possuir várias implementações e serem configuradas através de diversos parâmetros. Entretanto, não há um consenso sobre a melhor maneira de avaliar o desempenho de uma DHT, o que dificulta a escolha da implementação ideal para cada caso. Este trabalho propõe uma metodologia de avaliação de desempenho de DHTs que utiliza um conjunto padrão de testes de desempenho baseado nas avaliações mais utilizadas encontradas na literatura. É apresentada também uma implementação para essa metodologia e seus resultados iniciais.*

1. Introdução

Ao longo da última década as redes par-a-par (P2P, *peer-to-peer*) tornaram-se muito populares, surgindo uma grande variedade de casos de uso e implementações que possuem até milhões de participantes, como no caso da rede Gnutella [Gnutella Protocol Development]. Uma rede P2P é composta por um conjunto de participantes (nodos) dinâmicos sem o controle de uma autoridade central. Em geral as redes P2P são projetadas para gerenciarem uma grande quantidade de nodos entrando e saindo do sistema a todo momento.

As redes P2P podem ser classificadas de acordo com suas estruturas em três grupos: as redes P2P *estruturadas*, as *não-estruturadas* e as *fracamente estruturadas* [Androutsellis-Theotokis and Spinellis 2004]. Nas redes estruturadas os participantes mantém uma topologia pré-definida e seguem regras que definem como as informações são transmitidas e armazenadas. Nas redes P2P não-estruturadas não há regras para a

manutenção da topologia e local de armazenamento das informações, diminuindo o custo de manutenção da rede mas impossibilitando o estabelecimento de garantias quanto ao desempenho de certos aspectos da mesma. As redes P2P fracamente estruturadas são caracterizadas por não definirem estritamente a localização do conteúdo armazenado, mas afetarem a sua localização através de seus algoritmos de roteamento.

Os exemplos mais comuns de redes P2P estruturadas são as tabelas hash distribuídas (DHTs, *distributed hash tables*). As DHTs são estruturas de dados distribuídas que permitem a inserção de dados indexados por chaves. Cada chave é um identificador único, que deve ser guardado e utilizado para encontrar os dados previamente inseridos. Dentre as DHTs mais utilizadas estão Chord [Stoica et al. 2001], Pastry [Rowstron and Druschel 2001], Tapestry [Zhao et al. 2004] e CAN [Ratnasamy et al. 2001].

Existe uma grande quantidade de DHTs, cada uma com suas próprias características, parâmetros e topologia, o que dificulta o processo de escolha da DHT ideal para um determinado sistema. Além disso, cada DHT pode possuir diversas implementações, aumentando ainda mais o número de opções. Para facilitar o processo de escolha da DHT ideal para um determinado sistema, metodologias de avaliação de desempenho podem ser utilizadas. Nos últimos anos foram publicados uma série de trabalhos que propõem metodologias de avaliação de desempenho. Contudo, não existe um consenso de um conjunto mínimo de testes de desempenho suficientes para realizar esta avaliação de maneira imparcial, o que dificulta a avaliação e comparação dos resultados apresentados.

Neste artigo é proposta uma metodologia de avaliação de desempenho de DHTs. Essa metodologia é composta por diversos testes de desempenho, que por sua vez são definidos através dois elementos: cargas de trabalho e métricas de avaliação. O objetivo dos testes é cobrir de maneira imparcial os principais casos de uso das DHTs. As cargas de trabalho e métricas de avaliação são baseadas nos testes de desempenho mais utilizados pelos trabalhos que propõem a avaliação de desempenho de DHTs. Uma implementação para esta metodologia também é apresentada, bem como seus resultados iniciais.

A Seção 2 apresenta e compara alguns dos diversos trabalhos que avaliam desempenho de diferentes DHTs. Com base na análise realizada, a Seção 3 apresenta uma proposta de uma metodologia de avaliação de sistemas P2P baseados em DHTs, que tem sua implementação apresentada na Seção 4. A Seção 5 apresenta os resultados obtidos através dos experimentos iniciais realizados com a ferramenta. Por fim, a Seção 6 apresenta as considerações finais.

2. Avaliação de sistemas baseados em DHTs

Um teste de desempenho de DHT consiste na aplicação de uma carga de trabalho (*workload*) à rede e na análise de diversas métricas enquanto a mesma está sendo aplicada. A definição da carga de trabalho deve ser a mais completa possível, envolvendo informações como a DHT, seus parâmetros, o número de nodos, as ações executadas por cada um dos nodos (comportamento, o que inclui desde as operações `put` e `get` realizadas até os momentos em que o nodo entra e sai da rede) e até, se possível, a localização geográfica e organização topológica dos nodos, tanto nas camadas de aplicação quanto nas outras camadas de rede. Alguns trabalhos utilizam cargas de trabalho obtidas através da monitoração de redes P2P reais e outros criam suas próprias cargas de trabalho.

Apesar de ainda não haver um conjunto de testes de desempenho amplamente aceito como padrão para a avaliação de desempenho de DHTs, já foi apresentada uma grande quantidade de trabalhos que avaliam o desempenho de DHTs. Alguns desses trabalhos propõem também metodologias de avaliação de desempenho de DHTs, como [Oppenheimer et al. 2004], [Li et al. 2004] e [Kato and Kamiya 2007], porém nenhum deles foi amplamente utilizado em trabalhos subsequentes ou procurou estabelecer suas metodologias com base em trabalhos relacionados. Esta seção descreve alguns destes trabalhos (dos encontrados, os que possuem maior foco em avaliação de desempenho), analisando as avaliações de desempenho propostas e apresentando um resumo com as seguintes informações: as DHTs – ou redes – utilizadas, o ambiente sobre o qual as avaliações foram realizadas, o tamanho das redes em questão, o comportamento dos nodos e as métricas utilizadas. Esses dados são apresentados na Tabela 1.

Cada uma das redes utilizadas pelos trabalhos analisados foi associada a uma das seguintes categorias: (i) implementações de DHTs (i.e., prontas para serem utilizadas por aplicações reais); (ii) DHTs providas por simuladores; (iii) infraestruturas que possibilitam a criação de DHTs, como no caso de Plaxton, utilizado pelas DHTs Pastry e Tapestry; ou (iv) modelos teóricos de DHTs ou de infraestruturas que possibilitam a criação de DHTs, como no caso de [Kong et al. 2006]. As redes utilizadas nos trabalhos analisados são: Chord [Stoica et al. 2001], Pastry [Rowstron and Druschel 2001], Tapestry [Zhao et al. 2004], Kademlia [Maymounkov and Mazières 2002], CAN [Ratnasamy et al. 2001], Kelips [Gupta et al. 2003], Symphony [Manku et al. 2003], Plaxton [Plaxton et al. 1999], Accordion [Li et al. 2005] e Bamboo [Rhea et al. 2004].

Os ambientes utilizados nos diversos trabalhos analisados foram classificados como: (i) *reais*, para os casos mais próximos de uma utilização real de DHTs, como o PlanetLab [Chun et al. 2003]; (ii) *simulação*, para os casos nos quais simuladores de redes P2P foram utilizados; (iii) *emulação*, para os casos nos quais foram utilizadas implementações reais em ambientes restritos (e.g., centenas de nodos virtuais em um único nodo físico ou uma pequena rede local) ou (iv) *analíticos*, para os casos nos quais os estudos realizados foram puramente teóricos e os resultados apresentados são, por exemplo, provas matemáticas. Os simuladores utilizados nos trabalhos analisados são: p2psim [p2psim], ACME framework [Oppenheimer et al. 2003] e Microsoft Research Pastry Simulator v3.0A [Bjurefors et al. 2004]. Alguns trabalhos não mencionam os simuladores utilizados.

A coluna *tamanho da rede* descreve a quantidade de nodos utilizados nas avaliações realizadas pelos trabalhos apresentados. Atenção especial deve ser dada para o trabalho [Kong et al. 2006], onde a análise teórica é realizada para o caso no qual o tamanho da rede tende ao infinito.

A coluna *comportamento dos nodos* descreve o comportamento adotado pelos nodos na rede, o que inclui as operações `put` e `get`, além das entradas e saídas de cada nodo da rede (juntas, essas entradas e saídas definem o dinamismo dos nodos na rede, também conhecido como *churn*). Para os casos onde houve vários testes de desempenho com comportamentos diferentes, os diversos tipos de comportamento são listados. Na maioria dos casos analisados o comportamento dos nodos era simplesmente realizar um certo número de requisições em um determinado intervalo de tempo. Em alguns casos

os comportamentos utilizaram cargas de trabalho (*workloads*) observadas em aplicações reais, como em [Castro et al. 2005]. Em outros casos os comportamentos foram apenas inspirados nos possíveis comportamentos reais, ou então a escolha do comportamento adotado simplesmente não foi justificada. Houve também avaliações onde foram analisadas apenas as mensagens de controle geradas pelas DHTs, portanto os nodos não realizaram requisições, como em [Bjurefors et al. 2004].

Cada trabalho realizado possui seu próprio método para analisar desempenho e medir os resultados. Entretanto certas métricas como a latência foram analisadas de maneiras diferentes em diversos trabalhos (e.g., em [Li et al. 2004] a latência é simplesmente o intervalo de tempo entre uma requisição `get` e sua resposta, mas em [Rhea et al. 2003] a latência é expressa como a relação entre o intervalo de tempo obtido e o tempo de *ping* entre o nodo que busca a chave e o nodo que a possui). Apesar dessas pequenas diferenças, nós organizamos as diversas medições realizadas pelos trabalhos entre os seguintes grupos:

latência: mede o intervalo de tempo entre uma ou mais requisições e suas respostas;
tráfego de rede: mede a quantidade de mensagens – em valor absoluto ou em bytes – gerada pelos nodos durante suas operações;
taxa de sucesso: mede a quantidade de requisições completadas com sucesso pelos nodos;
proximidade de réplicas: compara a distância da réplica obtida através de uma requisição `get` com a distância da réplica mais próxima do nodo;
consistência: compara a consistência dos resultados de um conjunto de operações iguais – como buscas pela mesma chave – realizadas por nodos diferentes quase ao mesmo tempo;
caminhos falhos: mede a quantidade de nodos que pode ser alcançada por cada nodo da rede.

Observamos que algumas DHTs como Chord e Pastry foram utilizadas em quase todos os trabalhos realizados, portanto podem ser consideradas como as DHTs "mais populares". Além disso, quase todos os trabalhos analisados utilizaram ambientes simulados ou emulados, sendo apenas um deles baseado em um ambiente real. Outro fato observado foi que apesar das discussões sobre escalabilidade e aplicações que podem possuir até milhões de nodos, boa parte dos trabalhos não analisou mais do que apenas algumas centenas ou milhares de nodos. Ainda, apesar do comportamento dos nodos nas avaliações realizadas envolver quase somente a realização de buscas periódicas, a caracterização das redes envolvidas variou bastante, principalmente com relação aos algoritmos e técnicas utilizados internamente pelas DHTs. Por fim, observamos também que as métricas mais utilizadas nas avaliações de desempenho foram latência, tráfego de rede e taxa de sucesso.

3. Definição da metodologia

Em uma avaliação de desempenho de DHT há uma série de fatores que devem ser analisados, portanto cada avaliação deve ser composta por diversos testes de desempenho. Uma das maiores dificuldades na avaliação é que cada uma das inúmeras possíveis aplicações que utilizam DHTs pode apresentar uma carga de trabalho diferente. Portanto o conjunto

Trabalho	Rede(s)	Ambiente	Tamanho da rede	Comportamento dos nodos	Métricas
[Rhea et al. 2003]	Chord e Tapestry	Real: PlanetLab	79 – 83 nodos	(i) buscas sem churn	(i) latência, (ii) proximidade de réplicas
[Bjurefors et al. 2004]	Pastry	Simulação: Microsoft Research Pastry Simulator v3.0A	30-3000 nodos	(i) inicialização, (ii) *i* com buscas, (iii) *ii* com tabelas particionadas	(i) tráfego de rede
[Li et al. 2004]	Chord, Tapestry, Kelips e Kademlia	Simulação: p2psim	1024 nodos	(i) buscas com churn	(i) tráfego de rede, (ii) latência
[Oppenheimer et al. 2004]	Chord, Pastry e Tapestry	Emulação: ACME	150 nodos	(i) buscas com churn	(i) latência, (ii) taxa de sucesso, (iii) tráfego de rede, (iv) consistência
[Castro et al. 2005]	Pastry, Heterо-Pastry e Super-Pastry	Simulação	10000 – 37000 nodos	(i) workload real sem queries, (ii) workload real, (iii) churn	(i) tráfego de rede, (ii) taxa de sucesso, (iii) latência
[Kong et al. 2006]	CAN, Chord, Kademlia, Symphony e Plaxton	Analítico	Infinito	(i) rede com nodos falhos	(i) caminhos falhos
[Kato and Kamiya 2007]	Accordion, Bamboo, Chord e Pastry	Emulação: rede local	91 – 991 nodos	(i) buscas sem churn, (ii) buscas com churn, (iii) outros modelos mais complexos	(i) taxa de sucesso, (ii) latência, (iii) tráfego de rede
[Harvesf and Blough 2007]	Pastry	Simulação	1024 nodos	(i) buscas com nodos falhos e replicação	(i) taxa de sucesso, (ii) latência

Tabela 1. Trabalhos que medem o desempenho de DHTs

de testes de desempenho relevantes para uma determinada aplicação pode ser completamente diferente do conjunto de testes de outra. Com isso, a determinação de um conjunto a ser adotado como padrão para a maioria dos casos não é trivial. A proposta desta metodologia é, a partir da análise realizada, definir um conjunto de testes de desempenho mínimo que seja capaz de refletir as medições mais comuns entre os trabalhos que medem o desempenho de DHTs.

A metodologia proposta define cada teste de desempenho como uma combinação de dois elementos: a *carga de trabalho* e as *métricas* a serem analisadas. A carga de trabalho é a definição das ações a serem executadas pelos nodos da DHT. Como o objetivo é que cada teste possa ser utilizado por diversas DHTs e em diversos ambientes, algumas informações como por exemplo a disposição dos nodos na DHT ou os detalhes das camadas de rede mais inferiores devem ser omitidas da carga de trabalho. As métricas definem os dados que devem ser coletados pelos nodos, como latência das operações, taxa de sucesso e outras. O conjunto de dados a serem coletados deve ser independente da DHT a ser utilizada.

Identificamos dois grupos de cargas de trabalho: as cargas de trabalho baseadas em aplicações reais e as cargas de trabalho que visam exercitar alguma operação ou funcionalidade específica da DHT. Um exemplo muito comum do primeiro grupo são as cargas de trabalho de aplicações de compartilhamento de arquivos. Já sobre segundo grupo pode-se citar as cargas de trabalho com alto grau de *churn*, que visam testar a estabilidade das DHTs, e as cargas de trabalho onde os nodos efetuam grandes quantidades de buscas, visando obter medidas como latência, corretude e outras.

As métricas também podem ser divididas em dois grupos: as métricas gerais, que podem ser medidas em todos os tipos de testes de desempenho e as métricas específicas,

que só devem ser medidas em testes que possuem cargas de trabalho específicas. Do primeiro grupo pode-se citar as três métricas identificadas como mais analisadas: latência das mensagens, tráfego de rede gerado e taxa de sucesso das operações. Do segundo grupo pode-se citar, por exemplo, as métricas que avaliam a distância e disponibilidade das réplicas armazenadas em DHTs com suporte a réplicas.

O conjunto básico de testes de desempenho a ser escolhido deve apresentar ambos os tipos de cargas de trabalho e ambos os tipos de métricas a serem analisadas. Esses testes devem contemplar as funcionalidades essenciais das DHTs: escalabilidade, tolerância a *churn* (readaptação da topologia, propagação de chaves, roteamento), inserção de chaves e busca de chaves.

Com base no exposto e na análise feita na Seção 2, um conjunto básico de testes para compor uma avaliação de desempenho deve conter as seguintes cargas de trabalho: (i) sem buscas, com e sem *churn*, viabilizando métricas que envolvam a análise dos custos de manutenção da DHT, como o tráfego de rede, o tempo de estabilização e a transferência de chaves entre nodos; (ii) com buscas e diversos níveis de *churn*, viabilizando métricas que envolvam as operações `get` e `put`, como latência e taxa de sucesso de operações; e (iii) reais, obtidas através da análise de aplicações reais que utilizam DHTs.

4. A ferramenta proposta

A ferramenta proposta tem como objetivo permitir a aplicação de testes de desempenho de DHTs. Sua elaboração foi realizada com os seguintes objetivos: (i) ser facilmente adaptável aos diversos tipos de DHTs existentes; (ii) permitir facilmente a adição de diversos testes de desempenho; (iii) possuir uma boa escalabilidade; (iv) funcionar sem a necessidade de obter informações específicas das DHTs (e.g, lista de nodos vizinhos) e (v) funcionar sem a necessidade de alteração nas DHTs a serem avaliadas.

Com base nos requisitos estabelecidos acima um modelo para a implementação da ferramenta foi elaborado. Este modelo é formado por três entidades: o *mestre*, os *controladores* e os *nodos*, conforme ilustrado na Figura 1. As linhas contínuas representam a comunicação entre as entidades realizada através da ferramenta e as linhas pontilhadas representam a comunicação entre as entidades *nodo* realizada através das DHTs.

O mestre é a entidade hierarquicamente superior: ele possui a definição da carga de trabalho a ser aplicada na rede e deve aplicá-la comunicando-se com os controladores. Essa troca de informações deve funcionar de maneira externa à DHT a ser avaliada e ser rápida e confiável o suficiente para não afetar negativamente o resultado dos testes de desempenho realizados. Como o mestre é apenas um e deve gerenciar muitos controladores, cuidado especial deve ser tomado para que a sua implementação não comprometa a escalabilidade da ferramenta: quanto menos informações ele precisar enviar aos controladores durante a execução dos testes, maior será o seu grau de escalabilidade.

Para cada nodo existente na DHT deve existir uma entidade chamada controlador. O controlador é responsável por operar o funcionamento de um nodo participante da DHT a ser avaliada. Ele recebe do mestre as informações sobre como será seu comportamento durante a execução dos testes e quais dados deverá coletar para obter seus resultados. A partir dessas informações ele controla seu nodo associado para executar as ações necessárias. Cada implementação de DHT a ser utilizada exige a implementação de um controlador diferente.

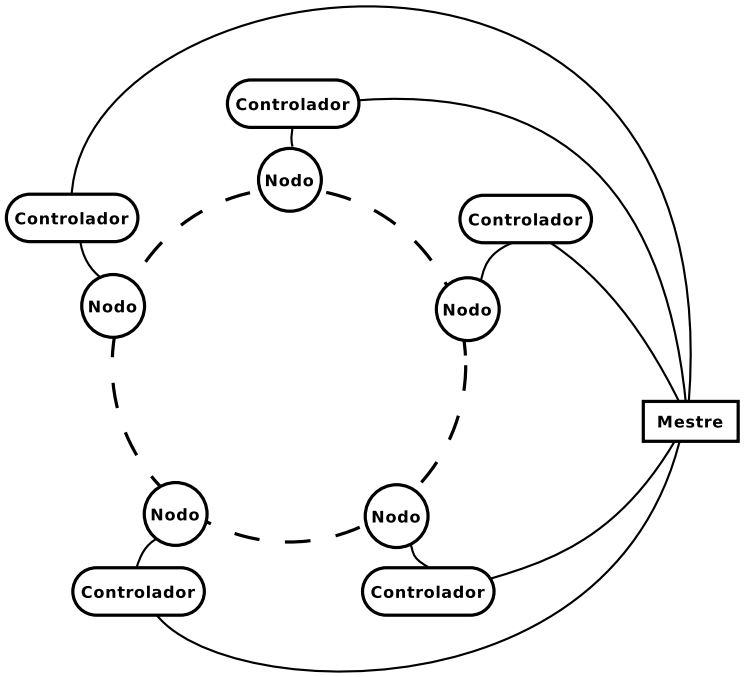

Figura 1. Representação esquemática da ferramenta proposta

Os nodos são as entidades que fazem parte da DHT, portanto podem possuir as mais variadas implementações. Apesar das possíveis diferenças, as implementações dos nodos precisam expor para os controladores as funcionalidades básicas das DHTs: iniciar uma nova DHT, entrar na DHT a partir de algum nodo que já faça parte da mesma, inserir chaves (`put`), buscar chaves (`get`) e sair da DHT. Todas as operações realizadas pelos controladores durante a avaliação devem depender apenas desse conjunto básico de funcionalidades.

Os resultados obtidos por cada controlador devem ser armazenados e enviados ao mestre após a realização da avaliação, para que então este possa processá-los e obter os resultados finais.

4.1. Multidhtshell

O Multidhtshell é a implementação realizada da ferramenta proposta e permite a utilização das DHTs providas pelo conjunto de ferramentas (*toolkit*) Overlay Weaver [Overlay Weaver]: Chord, Kademlia, Koorde, LinearWalker, Pastry e Tapestry. As entidades *mestre* e *controlador* são programas escritos na linguagem Ruby. A entidade *nodo* é representada pelo *owdhtshell*, uma ferramenta do Overlay Weaver que permite o controle de um nodo em uma DHT através de um pequeno *shell*. O Overlay Weaver foi escolhido por ser uma ferramenta que está em constante processo de desenvolvimento e possuir diversas DHTs que podem ser manipuladas através de uma única interface (o *owdhtshell*), o que torna necessária apenas uma implementação da entidade controlador. Através do Overlay Weaver pode-se utilizar tanto DHTs reais como simuladas, porém o Multidhtshell utiliza apenas DHTs reais.

Uma limitação da versão inicial dessa ferramenta é a comunicação entre as enti-

dades *mestre* e *controlador*. Essa comunicação não se dá através da rede e sim através de *threads*, portanto todos os nodos devem ser executados em um mesmo sistema. Com isso, pode-se afirmar que as avaliações realizadas são do tipo *emulação*. Uma vantagem desse método é que ele elimina parâmetros como conexão, latência e transmissão entre máquinas, aumentando a importância do desempenho da entidade *nodo* e aumentando também a reprodutibilidade das avaliações realizadas. É importante ressaltar que a comunicação entre os nodos das DHTs (pertencente ao código do Overlay Weaver) ainda assim é feita através dos protocolos de rede convencionais, portanto mesmo sendo realizada entre nodos de uma mesma máquina está sujeita a *bufferização*, espera de confirmações (ACK), etc.

A versão inicial do Multidhtshell possui dois testes de desempenho implementados, chamados de *estabilização* e *latência*. O teste *estabilização* serve para medir a taxa de propagação de chaves para nodos que entram na DHT. A aplicação de sua carga de trabalho consiste em quatro etapas: (i) um nodo é iniciado, criando a DHT, (ii) esse único nodo insere uma certa quantidade de chaves na DHT, (iii) outros nodos são adicionados na rede e (iv) paralelamente, cada um dos nodos tenta fazer `get` em cada uma das chaves inicialmente adicionadas até que consiga pelo menos um `get` com sucesso para cada chave. As métricas analisadas são o número de tentativas realizadas por cada nodo e o tempo decorrido. O teste *latência* tem como objetivo medir a latência das operações `get` das DHTs, dada pelo intervalo de tempo que ocorre do momento em que o nodo inicia a busca por uma chave até o momento em que ele a encontra, o que pode incluir comunicação com muitos nodos. Sua carga de trabalho é dividida da seguinte maneira: (i) um nodo é iniciado, criando a DHT, (ii) outros nodos são iniciados, (iii) espera-se até que todos os nodos entrem na DHT e (iv) paralelamente, cada nodo tenta fazer uma certa quantidade de requisições `get` em chaves aleatórias. A métrica analisada é o tempo de latência médio das requisições de cada nodo.

Versões futuras do Multidhtshell eliminarão a limitação acima citada, possuirão implementações de controladores capazes de controlar outros tipos de DHTs além daquelas providas pelo Overlay Weaver e possuirão mais testes de desempenho, de maneira a cumprir os objetivos propostos no início desta seção.

5. Resultados experimentais

Esta seção descreve os resultados obtidos nos experimentos iniciais realizados com o Multidhtshell. Como a versão inicial do Multidhtshell possui a limitação de que todos os nodos devem estar presentes no mesmo sistema, todos os nodos foram emulados em um computador com processador Intel(R) Core(TM) i5 3.20GHz, 4GB de memória RAM e sistema operacional Mandriva Linux 2010.1 Alpha 3 (Linux Kernel 2.6.33.1). A versão do Overlay Weaver utilizada foi a 0.9.9 com o *patch* oficial para melhorar a tolerância a *churn* (distribuído junto com o Overlay Weaver). A máquina virtual do Java utilizada pelo Overlay Weaver foi a Java HotSpot(TM) Server VM (build 16.0-b13, mixed mode). O objetivo dos experimentos realizados é apenas demonstrar que a ferramenta apresentada é funcional, portanto os valores escolhidos nos testes (como número de nodos e operações `get`) podem ser considerados pequenos.

O objetivo do teste *estabilização* é verificar o tempo de propagação das chaves para nodos que entram na DHT. O número de chaves presentes da DHT foi 100 e o número

de nodos inseridos – além do nodo inicial – foi 10. A Figura 2 mostra o tempo decorrido em cada DHT (i.e, tempo necessário para que todos os nodos da DHT realizassem pelo menos um `get` com sucesso em cada chave). A Figura 3 mostra o número máximo de buscas por uma única chave realizadas por um único nodo durante a avaliação (não necessariamente o nodo que mais demorou para obter todas as chaves).

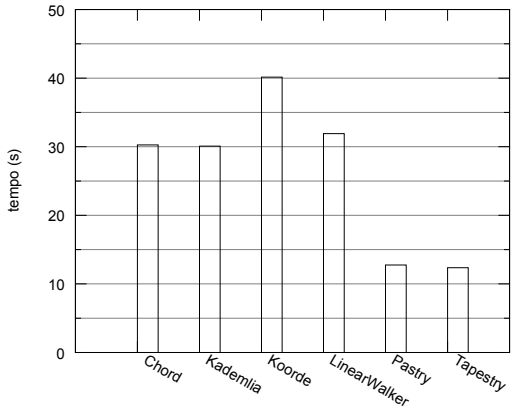

Figura 2. Estabilização: tempo de estabilização de cada DHT analisada. *Default reput interval*: 30s.

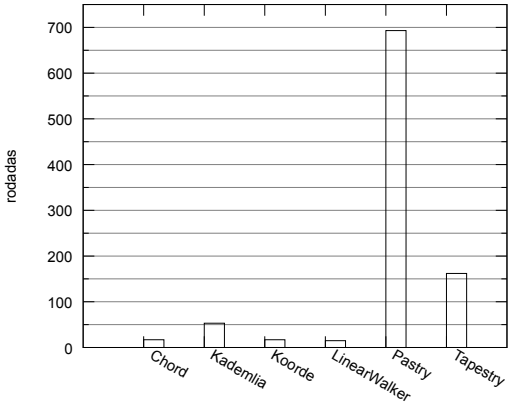

Figura 3. Estabilização: máximo de buscas por uma única chave por nodo. *Default reput interval*: 30s.

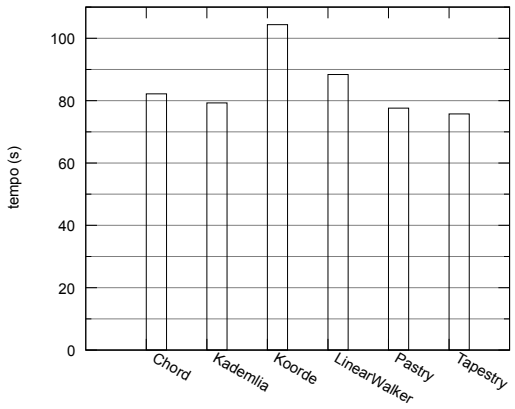

Figura 4. Estabilização: tempo de estabilização de cada DHT analisada. *Default reput interval*: 100s.

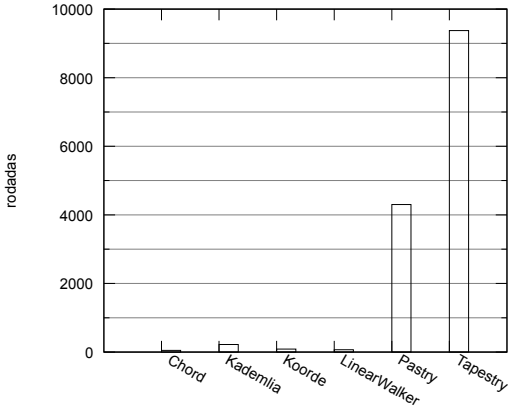

Figura 5. Estabilização: máximo de buscas por uma única chave por nodo. *Default reput interval*: 100s.

A implementação do Overlay Weaver utilizada possui um parâmetro chamado `DEFAULT_REPUT_INTERVAL`, que define a frequência com a qual as chaves são reinseridas na DHT (visando aumentar a tolerância a *churn*). As figuras 4 e 5 mostram os resultados obtidos quando o valor do parâmetro é modificado de 30 segundos para 100 segundos. Como esperado, o tempo de estabilização das DHTs analisadas aumentou. Observa-se que esse parâmetro exerce grande influência na capacidade de estabilização da rede, porém sem a presença de mais testes de desempenho não é possível medir seus outros efeitos. Além disso, as figuras 3 e 5 mostram uma grande diferença das

implementações das DHTs Pastry e Tapestry para as outras, pois estas são capazes de realizar uma quantidade de buscas muito maior em um intervalo de tempo menor. Maiores investigações sobre a razão deste comportamento devem ser realizadas através de análises no código-fonte do Overlay Weaver.

O teste *latência* tem como objetivo calcular a latência média das mensagens `get` em cada DHT. Em uma primeira avaliação foi utilizada uma DHT com 10 nodos, onde cada nodo realizou 100 buscas. A Figura 6 mostra, para cada DHT, a latência média dos nodos que apresentaram a menor e a maior latência, além da latência média entre todos os nodos. Outra avaliação foi realizada, porém com 50 nodos e 500 buscas para cada nodo. Seus resultados, apresentados na Figura 7, mostram que mesmo com um número muito maior de nodos e buscas o desempenho relativo entre as DHTs analisadas permanece o mesmo, com exceção das DHTs Kademlia, que apresentou um melhor desempenho relativo se comparado com a avaliação anterior e Pastry, que não pode ser avaliada devido a um *bug* na implementação do Overlay Weaver.

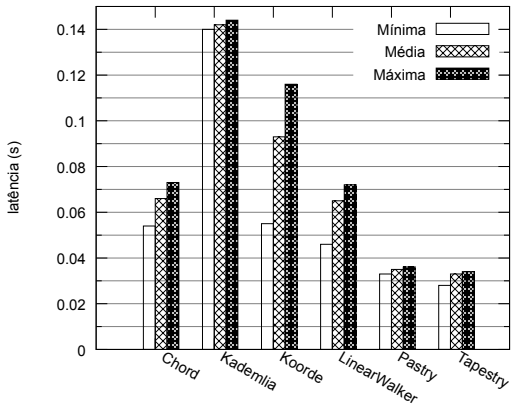

Figura 6. Latência: 100 buscas para cada um dos 10 nodos

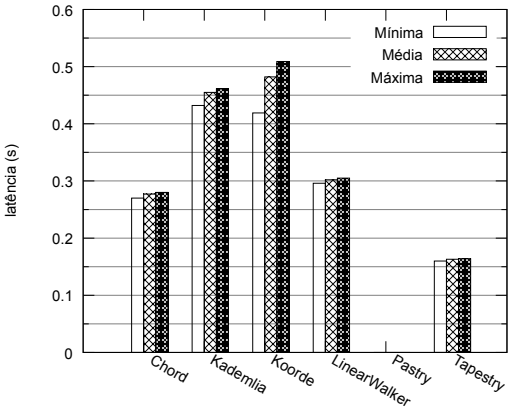

Figura 7. Latência: 500 buscas para cada um dos 50 nodos

6. Conclusão

Ao longo da última década as redes P2P tornaram-se bastante populares, sendo as DHTs os exemplos mais comuns de redes P2P estruturadas. Cada DHT pode possuir diversas implementações, que por sua vez podem ser ajustadas através de diversos parâmetros. Essa grande diversidade torna difícil a escolha da DHT ideal para cada aplicação. Uma possível solução para esse problema é a utilização de avaliações de desempenho. Nos últimos anos foram apresentados diversos trabalhos que propõem métodos para a avaliação de desempenho de DHTs, porém não há um consenso de um conjunto mínimo de testes de desempenho suficientes para realizar, de maneira imparcial, uma avaliação de desempenho de DHTs.

Este artigo analisou os trabalhos que efetuaram avaliações de desempenho de DHTs, visando identificar os seus pontos em comum. Com base nessa análise foi proposta uma metodologia de avaliação de desempenho de DHTs que utiliza um conjunto de testes de desempenho baseado nos testes mais utilizados pelos trabalhos analisados. Cada teste de desempenho foi definido como uma combinação de uma carga de trabalho e um conjunto de métricas a serem analisadas.

Uma implementação inicial para a metodologia proposta foi apresentada. Essa implementação é capaz de avaliar as DHTs implementadas pelo Overlay Weaver e possui dois testes de desempenho. Os resultados iniciais da utilização da ferramenta foram os esperados.

Referências

Androutsellis-Theotokis, S. and Spinellis, D. (2004). A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4):335–371.

Bjurefors, F., Larzon, L. A., and Gold, R. (2004). Performance of pastry in a heterogeneous system. In *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on*, pages 278–279.

Castro, M., Costa, M., and Rowstron, A. (2005). Debunking some myths about structured and unstructured overlays. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation*, pages 85 – 98, Berkeley, CA, USA. USENIX Association.

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. (2003). PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12.

Gnutella Protocol Development. `http://rfc-gnutella.sourceforge.net/`. Acessado em 9 de dezembro de 2009.

Gupta, I., Birman, K., Linga, P., Demers, A., and Van Renesse, R. (2003). Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead. *Lecture Notes in Computer Science*, pages 160–169.

Harvesf, C. and Blough, D. M. (2007). The Design and Evaluation of Techniques for Route Diversity in Distributed Hash Tables. In *Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing*, pages 237 – 238. Citeseer.

Kato, D. and Kamiya, T. (2007). Evaluating DHT Implementations in Complex Environments by Network Emulator. In *Proceedings of the IPTPS 2007*.

Kong, J. S., Bridgewater, J. S. A., and Roychowdhury, V. P. (2006). A General Framework for Scalability and Performance Analysis of DHT Routing Systems. *Arxiv preprint cs/0603112*.

Li, J., Stribling, J., Gil, T. M., Morris, R., and Kaashoek, M. F. (2004). Comparing the performance of distributed hash tables under churn. In *Proceedings of the 2nd Bertinoto Workshop on Future Directions in Distributed Computing (FuDiCo II): Survivability: Obstacles and Solutions, Bertinoro, Italy*, pages 87–99. Citeseer.

Li, J., Stribling, J., Morris, R., and Kaashoek, M. F. (2005). Bandwidth-efficient management of DHT routing tables. In *Proc. 2nd Symposium on Networked Systems Design and Implementation*.

Manku, G. S., Bawa, M., and Raghavan, P. (2003). Symphony: Distributed Hashing in a Small World. In *USENIX Symposium on Internet Technologies and Systems (USITS)*. Stanford InfoLab.

Maymounkov, P. and Mazières, D. (2002). Kademlia: A Peer-to-peer Information System Based on the XOR Metric. *Proceedings of the IPTPS02*, 1:53 – 65.

Oppenheimer, D., Vatkovskiy, V., and Patterson, D. A. (2004). Towards a framework for automated robustness evaluation of distributed services. In *Proceedings of the 2nd Bertinoto Workshop on Future Directions in Distributed Computing (FuDiCo II): Survivability: Obstacles and Solutions, Bertinoro, Italy*. Citeseer.

Oppenheimer, D., Vatkovskiy, V., Weatherspoon, H., Lee, J., Patterson, D. A., and Kubiatowicz, J. (2003). Monitoring, Analyzing, and Controlling Internet-scale Systems with ACME. Technical report, UC Berkeley Technical Report UCB-CSD-03-1276.

Overlay Weaver. `http://overlayweaver.sourceforge.net`. Acessado em 27 de março de 2010.

p2psim. `http://pdos.csail.mit.edu/p2psim/`. Acessado em 27 de março de 2010.

Plaxton, C. G., Rajaraman, R., and Richa, A. W. (1999). Accessing Nearby Copies of Replicated Objects in a Distributed Environment. *Theory of Computing Systems*, 32(3):241–280.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. (2001). A Scalable Content-Addressable Network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161 – 172. ACM.

Rhea, S., Geels, D., Roscoe, T., and Kubiatowicz, J. (2004). Handling Churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference*, pages 127–140.

Rhea, S. C., Roscoe, T., and Kubiatowicz, J. (2003). Structured peer-to-peer overlays need application-driven benchmarks. *Lecture notes in computer science*, pages 56–67.

Rowstron, A. and Druschel, P. (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, volume 11, pages 329 – 350. Citeseer.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149 – 160. ACM.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. D. (2004). Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on selected areas in communications*, 22(1):41 – 53.