Análise da Disponibilização de um Índice Invertido em P2P

Nuno Lopes, Carlos Baquero
{nuno.lopes,cbm}@di.uminho.pt

Departamento de Informática, Universidade do Minho

Resumo

A procura de informação com base em conteúdos é uma funcionalidade fundamental na construção de sistemas de partilha de ficheiros e recursos. Contudo, as soluções eficientes para a partilha *peer-to-peer* que têm por base DHTs - *distributed hash tables* - perdem a capacidade de efectuar buscas ao requerer a existência de identificadores unívocos. Tendo em conta que a manutenção de um índice invertido de conteúdos é essencial para a concretização eficiente de buscas, apresenta-se neste trabalho um modelo de construção deste índice, com base em árvores-B+, que permite continuar a tirar partido das propriedades de escala dos actuais mecanismos de DHT.

1 Introdução

Os sistemas P2P (*peer-to-peer*) têm vindo a permitir a um número muito significativo de utilizadores o acesso a mecanismos autónomos de partilha de informação. Mecanismos estes que tiram partido do elevado número de nodos computacionais que dispõem de conectividade Internet, mesmo que de carácter temporário.

A primeira geração destes sistemas partilhava ficheiros que eram encontrados através da procura de palavras. De entre os primeiros sistemas, a rede Gnutella [3] é um de entre estes onde o modelo de interligação é genuinamente P2P. Neste sistema o processo de procura assenta em técnicas de disseminação epidémica das palavras procuradas. Tal característica torna o sistema bastante ineficiente por requerer demasiados recursos na comunicação, em particular largura de banda, tendo inclusivé conduzido à sua erradicação de diversas redes universitárias.

A segunda geração de sistemas P2P é baseada no conceito de *tabela de hash distribuída* (DHT), sendo emblemáticos os modelos desenvolvidos nos sistemas Chord [7] e Pastry [5]. Estes sistemas são extremamente eficientes no uso de recursos para a comunicação entre participantes, tendo por base a construção de mecanismos de encaminhamento com excelentes propriedades de adaptação à escala. Contudo, não oferecem directamente a possibilidade de pesquisar a informação com base no conteúdo, uma vez que assumem a existência de identificadores já conhecidos para o acesso a cada conjunto de dados disponíveis.

Uma pesquisa num ambiente P2P pode ser vista como usando um modelo *local* ou *global*, quando se tem em conta o grau de disseminação da informação sobre os conteúdos. Num modelo local, efectuam-se pesquisas individuais em cada participante com vista à procura de conteúdos destes que possuam a informação em pesquisa. No caso do sistema Gnutella, o modelo local é levado ao extremo pois as pesquisas são disseminadas a todos os participantes numa zona topologicamente próxima, independentemente do teor da pesquisa. Uma procura baseada na recolha de informação sobre a topologia de conteúdos da rede permite em teoria, após um período de aprendizagem, orientar as pesquisas para os locais onde há maior probabilidade de se encontrar a informação requerida [4]. No entanto, os sistemas P2P poderão ser demasiado dinâmicos para permitir que um sistema baseado neste modelo consiga efectivamente convergir para um estado de conhecimento abrangente e preciso. Como, tipicamente, cada participante P2P está em média ligado ao sistema menos de uma hora [6], o sistema poderá não ter tempo suficiente para se adaptar à informação residente em participantes que exibam padrões de conectividades muito intermitentes. A viabilidade destes modelos dependerá pois da análise do comportamento das redes P2P reais.

Em alternativa, o modelo global assenta no anúncio, por parte dos participantes, dos seus conteúdos. Este anúncio concretiza-se no fornecimento, por cada nodo, de associações $palavra \mapsto local$ com base na ocorrência de palavras nos conteúdos por si exportados. O registo descentralizado desta informação dá origem a um índice invertido distribuído que terá de ser mantido pelo conjunto dos nodos activos no sistema. A procura de cada palavra nos conteúdos exportados concretiza-se na recolha, via índice invertido, do conjunto de locais onde a sua ocorrência foi previamente anunciada.

Importa contudo analisar alguns dados estatísticos sobre a ocorrência de palavras em documentos. Sabe-se que para documentos genéricos a ocorrência de palavras segue uma distribuição Zipf [2, 1]. Esta distribuição indica que certas palavras do índice invertido, as mais comuns nos documentos, tenham um número de ocorrências extremamente elevado. Isto, apesar da maior parte das palavras conter um número mais reduzido de ocorrências, e como tal derivarem um conjunto de locais mais reduzido. A variação do número de ocorrências de cada palavra leva a que o índice exiba uma grande variabilidade no que toca ao tamanho do conjunto de locais (identificando documentos nos nodos) que este associa a cada palavra.

Um índice invertido requer uma funcionalidade semelhante à oferecida por um sistema DHT. Estes sistemas DHT são já adaptáveis a escalas de milhões de participantes, conseguindo comunicar utilizando pouca memória de estado em cada participante, consumindo pouca largura de banda e efectuando um número reduzido de saltos entre nodos da rede. Estas características quase óptimas tornam estes sistemas bons candidatos para servir como base à gestão de um índice invertido distribuído. No entanto, os sistemas DHT, apesar de manterem associações $chave \mapsto valor$, apresentam uma limitação que impede a sua aplicação directa ao registo das associações $palavra \mapsto locais$ que que estão por base à manutenção de índices invertidos.

Num sistema DHT permite-se a associação de uma chave a uma dada imagem, podendo estas imagens ser substituídas ou obtidas através do fornecimento da respectiva chave. Na prática, a forma como os DHTs são concebidos leva-os a ter de limitar o tamanho da imagem, ou, no caso particular de se tratar de uma imagem estática, à sua

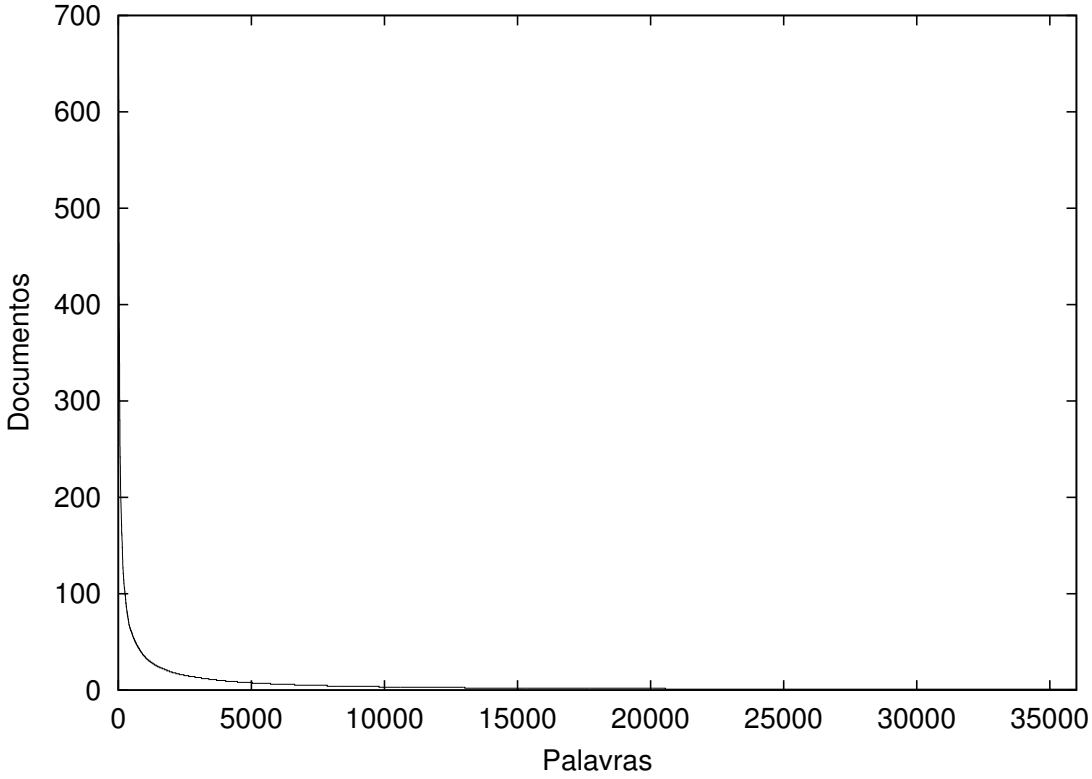

Figura 1: Distribuição do número de documentos associados a cada palavra. Observa-se uma típica distribuição Zipf.

divisão numa sequência de blocos com tamanho limitado. A não limitação de tamanho nas imagens/blocos associadas às chaves acarreta, nos actuais algoritmos DHT, grandes assimetrias na distribuição de carga pelos nodos, ou uma gestão ineficiente de imagens grandes que tenham um carácter não estático.

A Figura 1 mostra qual a carga a distribuir pelos nodos caso se tente armazenar directamente um índice invertido. Observa-se facilmente que os nodos a quem fosse alocado o armazenamento das primeiras dezenas de palavras mais frequentes, teriam de suportar uma carga muito superior à carga média dos outros nodos.

A gestão eficaz de um índice invertido requer, quer a capacidade de adicionar e remover independentemente associações entre chaves e locais de ocorrência, quer a gestão de grandes conjuntos de localizações. Este artigo introduz um modelo de composição que permite criar um índice invertido com as propriedades desejadas, tomando por base a gestão de blocos de tamanho limitado que é oferecida eficientemente pelos algoritmos DHTs já disponíveis.

2 Apresentação do Modelo

2.1 Interfaces de Operação

Um índice invertido cria uma associação entre palavras e referências. A cada palavra estão associadas referências para documentos onde essa mesma palavra ocorre. Como tal, as principais operações de um índice são definíveis da seguinte forma:

- INSERT(*word*,*reference*) – Insere uma referência no conjunto associado à palavra.
- REMOVE(*word*,*reference*) – Remove a referência do conjunto associado à palavra.
- HAS_REF(*word*,*reference*):*bool* – Verifica se a referência existe no conjunto associado à palavra.
- GET_REF(*word*):*reference* – Devolve a primeira referência do conjunto associado à palavra.
- NEXT_REF(*reference*):*reference* – Itera sobre um conjunto, de uma referência para outra seguinte.

Por seu turno, um sistema DHT genérico oferece uma estrutura distribuída capaz de fazer uma associação entre chaves e valores. Esta estrutura oferece de um modo transparente, escalabilidade na comunicação e tolerância a faltas de nodos. As chaves utilizadas internamente pelo DHT têm um tamanho fixo que resulta da aplicação de uma função de *hash* a uma chave original. Esta função de *hash* permite uniformizar o tamanho da chave e distribuir equilibradamente as chaves pelo universo de nodos. Os valores que correspondem à imagem da chave devem ser limitados no tamanho, por forma a respeitar a capacidade de armazenamento dos nodos e a oferecer uma granularidade apropriada para que se obtenha uma distribuição uniforme da carga. As funções básicas de um DHT genérico podem ser resumidas da seguinte forma:

- INSERT(key_h,$value$) – Insere o par (chave,valor) no sistema.
- GET(key_h):*value* – Retorna o valor associado à chave.
- UPDATE(key_h,$value$) – Actualiza o valor associado à chave.
- REMOVE(key_h) – Remove o par com esta chave.

2.2 Decomposição do Índice Invertido

Um índice invertido requer a utilização de duas estruturas abstractas básicas: associações e conjuntos. Enquanto que a funcionalidade requerida pelas associações pode ser obtida directamente através de um DHT, os conjuntos já requerem propriedades distintas.

Um conjunto, utilizado para guardar as referências de um índice invertido, tem de ser capaz de suportar um número elevado de elementos sem penalizar as operações normais de procura, inserção e remoção. Além disso, o conjunto deve suportar uma actividade constante de inserções e remoções sem degradar o comportamento geral do sistema.

Torna-se pois interessante encontrar uma estrutura que ofereça a funcionalidade de um conjunto e que seja adaptável a um sistema distribuído suportado por DHTs.

As árvores-B+ são estruturas de dados desenhadas para serem eficientes quando utilizadas em memórias de acesso secundário, como por exemplo discos-duros. Estas estruturas são tipicamente utilizadas para conter mapeamentos de pares, $chave \mapsto valor$, mas podem ser também aplicadas para implementar conjuntos de elementos com a mesma eficácia, uma vez que o *valor* pode ser mantido a *nil*.

A memória secundária é caracterizada por uma latência elevada no acesso a dados, pelo que o objectivo destas estruturas concentra-se na minimização do número de acessos aos blocos constituintes da árvore. É evidente o paralelo que se pode estabelecer com a comunicação em rede, nomeadamente quando se observa uma latência elevada na comunicação entre nodos e se pretende minimizar o número de mensagens trocadas e os nodos consultados.

O modelo protagonizado neste artigo propõe a utilização de uma estrutura do tipo árvore-B+ sobre um sistema DHT base por forma a disponibilizar um mecanismo escalável que permita armazenar conjuntos com um conteúdo altamente dinâmico.

3 Descrição do Sistema

O DHT, acedido pelo interface apresentado na Secção 2.1, constitui-se como o mecanismo base para o armazenamento persistente de blocos de informação que são acedidos através de uma chave. Os blocos de informação guardados no DHT têm, para este efeito, um tamanho constante e suficientemente pequeno de modo a que possam ser armazenados e transferidos com eficácia.

As palavras (chaves no índice invertido) vão funcionar como chaves no DHT para aceder ao bloco raiz que constitui o ponto inicial de acesso às suas referências. Para conseguir armazenar um número elevado de referências para uma só palavra, o sistema vai utilizar mais blocos de informação que são acedidos através de um segundo tipo de chaves no DHT.

A chave para o bloco raiz contém o *hash* da palavra do índice e um indicador de tipo que a identifica como sendo uma chave para um bloco raiz. As palavras usadas no sistema vão ter uma dispersão no universo de identificadores dada pela função de *hash*. Esta propriedade vai fazer com que o DHT disperse uniformemente pelos seus nodos, todos os blocos raiz das palavras indexadas.

O bloco raiz corresponde à raiz de uma árvore-B+ cujo conteúdo será o conjunto de referências (para os documentos) onde a palavra do índice ocorre. Cada bloco irá conter uma estrutura clássica de bloco de uma árvore-B+, que inclui algumas referências e disporá de ligações para outros blocos.

As ligações entre os blocos ao longo dos níveis de uma árvore são concretizadas por chaves que identificam univocamente os blocos dentro dessa mesma árvore. Esta propriedade é obtida pela utilização de um identificador único do bloco pai e de um identificador interno único para todos os filhos desse bloco pai.

As várias árvores-B+ irão coexistir no DHT, sendo as chaves dos seus blocos tornadas disjuntas pela inclusão da palavra, que identifica univocamente a árvore-B+ a que o bloco pertence, na composição da chave. O valor final da chave irá conter um tipo,

identificando-a como sendo uma chave de blocos não–raiz, e o resultado do *hash* da junção da palavra com os identificadores provenientes do bloco pai.

A utilização de um campo tipo de chave elimina colisões entre chaves dos dois tipos (raiz ou secundários). Uma vez que as chaves deste tipo utilizam valores únicos para cada bloco, o *hash* gerado terá poucas probabilidades de gerar colisões.

Apesar do sistema ter uma boa dispersão das chaves geradas para o DHT, poderá existir contenção nos nodos que guardam os blocos raiz de palavras muito populares. Uma adequada utilização de técnicas de *caching* irá aliviar a sobrecarga nesses mesmos nodos. Mais, se as palavras que são muito frequentes nas buscas também tiverem um elevado número de referências, a maior parte das operações irá centrar-se nos blocos dos níveis inferiores da árvore-B+, pelo que a raiz terá pouca probabilidade de ser invalidada por operações de modificação. Esta concentração nos níveis inferiores já ocorre nas inserções de palavras frequentes. À medida que se vai descendo de nível na árvore, o aumento do número de blocos vai diminuir a probabilidade de contenção num bloco em particular. Para evitar problemas de coerência em cópias desactualizadas, o *caching* não funciona para os blocos folha, isto é, os blocos no último nível da árvore-B+. A detecção de blocos inválidos em *cache* utiliza a informação de estado contida nos blocos. A modificação de um bloco (não-folha) leva a que a informação nele contida sobre os seus filhos seja diferente da informação contida na versão em *cache*. Neste caso, quando um bloco, agindo como cliente, usa a sua versão em *cache* de um outro bloco recém modificado, a informação de estado dos filhos que vai utilizar estará desactualizada. Ao fazer um acesso real a um bloco filho do bloco inválido, a informação de estado que o cliente lhe envia irá ser diferente da que ele próprio contém. Esta situação será detectada e o bloco cliente avisado de que está a utilizar uma versão desactualizada do bloco.

4 Análise das Operações

A implementação de um índice distribuído deve ter em conta a escalabilidade das operações à medida que no sistema vão aumentando o número de nodos. Esta análise começa por descrever algumas propriedades dos sistemas DHT para depois estudar o impacto, em termos de complexidade, das principais operações sobre o índice.

4.1 Sistema DHT

No seguimento desta exposição tomar-se-á por base o sistema Chord como concretização de um DHT, sendo este sistema bastante eficiente nas suas operações básicas. Para executar uma operação sobre uma chave, o sistema tem primeiro de encontrar o nodo responsável pela respectiva chave. Enquanto que cada operação não acarreta consumos significativos em termos da sua execução local, a comunicação entre os nodos e a memória de estado requerida podem desenvolver crescimentos não escaláveis. Uma operação distribuída pode ser considerada eficiente, num contexto P2P, quando o número de mensagens necessárias à sua execução e a memória de estado que exige em cada um dos nodos cresce com um factor logarítmico em relação ao número de nodos do sistema.

Para encontrar o nodo que guarda uma qualquer chave, o sistema Chord necessita em média de comunicar com $O(\log N)$ nodos a partir de qualquer nodo, considerando um

sistema com N nodos. Cada nodo necessita de manter uma tabela de encaminhamento com somente $O(\log N)$ entradas, sendo esta utilizada para indicar o nodo conhecido mais próximo da chave. Ou seja, qualquer operação sobre uma chave irá utilizar um número logarítmico de mensagens, sendo o estado mantido em cada nodo também logarítmico face ao número de nodos presentes no sistema.

4.2 Busca de uma Referência para uma Palavra

Para encontrar o bloco que contém uma referência para uma palavra, é necessário percorrer a árvore-B+ dessa palavra. A raiz da árvore-B+ é o bloco associado à chave do DHT com o *hash* da palavra. Os outros blocos da árvore são acedidos através das chaves contidas internamente nos blocos.

O acesso a um bloco implica utilizar $O(\log N)$ mensagens para o sistema DHT encontrar o nodo responsável por essa chave. Uma árvore-B+ requer em média $O(\log_t n)$ acessos a blocos para aceder a um elemento, que corresponde à sua altura média, sendo t o número de referências contidas num bloco e n o número total de elementos na árvore-B+. Pode-se assumir que em média, encontrar o bloco responsável por uma referência de uma palavra irá requerer $O(\log N * \log_t n)$ mensagens, para N nodos no sistema.

Note-se que a componente $O(\log_t n)$ irá depender do tamanho do conjunto de referências, n, sendo que, na situação média, este tamanho será pouco expressivo e facilmente amortizável desde que se escolha um valor suficientemente grande para t. O custo de suportar eficientemente conjuntos de grande cardinalidade acaba por ser pago localmente nesses mesmos conjuntos, sem afectar as palavras com menos referências.

4.3 Inserção e Remoção de Referências de uma Palavra

Qualquer destas operações requer encontrar o bloco responsável pela referência. Encontrado o bloco, este é lido, o seu conteúdo actualizado e depois novamente escrito no DHT.

A operação de inserção pode ocasionar situações em que é necessário dividir um bloco em dois. Neste caso, haverá a criação de um novo bloco e a modificação do bloco pai para conter uma nova chave para o bloco novo. Esta operação secundária não prejudica a escalabilidade da operação de inserção em termos de acesso a blocos, uma vez que a divisão têm na grande maioria dos casos um impacto localizado.

A operação de remoção pode também necessitar de reorganizar blocos devido ao redimensionamento do bloco onde a chave foi removida. No entanto, esta operação não vai, em média, necessitar de novos acessos a blocos, apesar da possibilidade de necessitar de efectuar mais que um acesso leitura/escrita ao mesmo bloco.

Enquanto que no caso simples tanto a operação de inserção como a de remoção se encontram circunscritas a um único bloco, foi já exposto que há situações em que estas operações requerem a modificação de outros blocos. Estas situações podem criar problemas de concorrência quando vários clientes executam estas operações de modificação em simultâneo sobre a mesma árvore-B+ e em particular sobre os mesmos blocos.

No caso das árvores-B+ o controlo de concorrência pode ser efectuado pela detecção de qual a sub-árvore que é afectada por uma dada operação, que transborde um único bloco, e a aplicação de um mecanismo de serialização dos respectivos pedidos no bloco raiz da sub-árvore. Em operação normal, as zonas de serialização serão bastante afastadas

da raiz da árvore, uma vez que as reorganizações que actuam perto da raiz são comparativamente raras, pelo que o controlo de concorrência não deverá constituir um factor de contenção significativo.

5 Simulação

O protótipo simula um sistema base DHT no qual são enviadas mensagens entre blocos, sendo cada bloco identificado através de uma chave única. A chave é utilizada pelos nodos para encontrar o nodo associado ao bloco pretendido. O processo de *routing* depende do DHT usado, podendo assumir-se que, em média, este requer $O(\log N)$ passos até encontrar o nodo pretendido de entre os N nodos no sistema.

No que respeita à modelação da comunicação entre nodos da rede P2P são assumidas diversas simplificações: As mensagens enviadas entre dois nodos são consideradas operações atómicas bem sucedidas; É apenas contabilizado o número de comutações entre blocos, não sendo feita explicitamente uma alocação dos blocos a nodos concretos.

As mensagens trocadas representam invocações das operações apresentadas no interface de gestão do índice e operações de manutenção para as árvores-B+. Ao nível do simulador estas operações traduzem-se na invocação de funções locais que implementam o interface.

Foram utilizados como documentos, pequenos artigos de notícias, com em média cerca de 350 palavras únicas por artigo. Ao todo foram utilizados 1000 documentos com um total de 36499 palavras únicas. Como já foi referido, a distribuição das palavras pelos documentos segue uma distribuição Zipf (ver Figura 1) em que um pequeno conjunto de palavras aparece em quase todos os documentos, enquanto que a maioria das palavras ocorre apenas pontualmente.

A Figura 2, com uma escala logarítmica ao nível das palavras, mostra a relação entre o número de documentos referenciados e os blocos DHT usados para armazenamento e as palavras indexadas pelo sistema, ordenadas por número decrescente de referências.

O número de blocos utilizado por cada palavra depende do seu número de referências e do tamanho do bloco. Sendo este último um valor constante, pode-se constatar e observar que as duas linhas estão totalmente correlacionadas.

Pode inferir-se da Figura 2 que a maior parte das palavras necessita apenas de um bloco devido à frequência média de referências ser muito baixa, próxima de 1. Mais concretamente, para blocos capazes de armazenar 10 referências, constata-se que 90% das palavras indexadas neste exemplo requerem o uso de um único bloco, e como tal podem ser localizadas com um único acesso ao substrato DHT.

O tamanho do bloco deverá ser dimensionado atendendo aos factores de utilização do espaço em disco, largura de banda e custo das operações. Se o tamanho do bloco for grande demais irá desperdiçar espaço para a maioria das palavras e gastar mais largura de banda ao ser transferido entre nodos. Por outro lado, um bloco pequeno vai aumentar a altura da árvore-B+ e exigir mais acessos a cada operação do índice em palavras com bastantes elementos. Um melhoramento potencial poderá passar pela utilização de blocos com tamanho variável, nomeadamente os blocos raiz.

A concretização, ou simulação, das operações de inserção e pesquisa de associações no índice invertido, conduz rapidamente à necessidade de optar entre dois cenários de

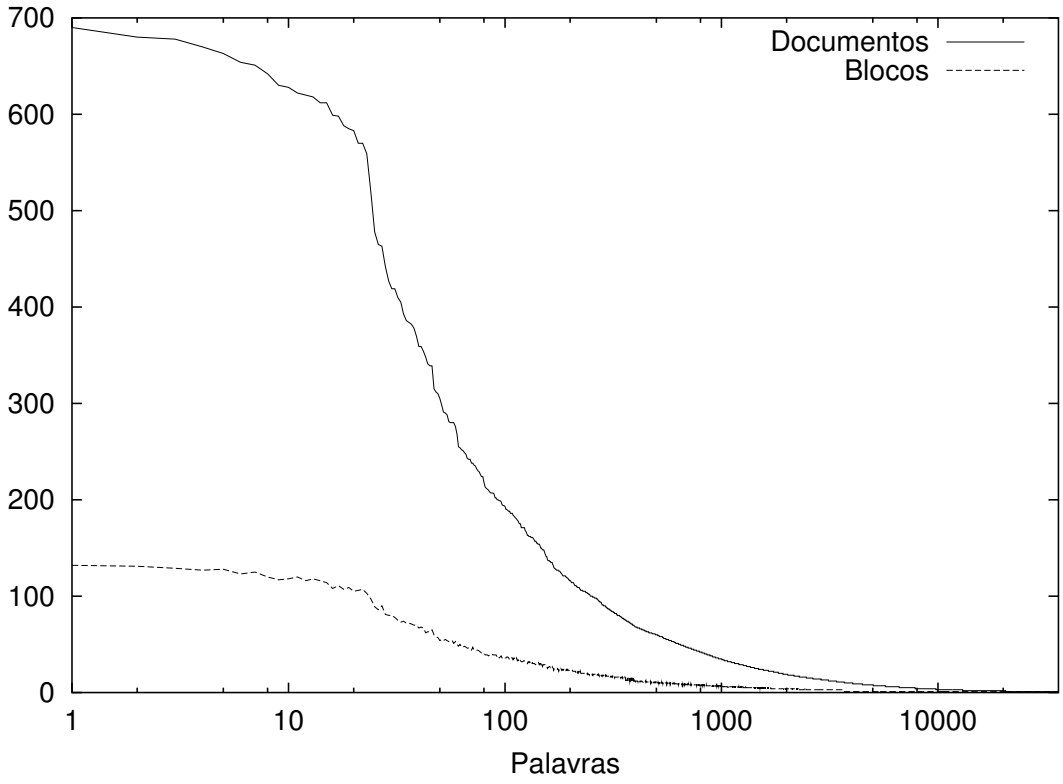

Figura 2: Distribuição em escala logarítmica das ocorrências de palavras e blocos necessários ao seu armazenamento.

realização: Execução com base no nodo cliente da operação; Delegação da execução a partir do nodo cliente da operação. No primeiro caso a carga requerida para o processamento da operação encontrar-se-á mais concentrada no nodo cliente, enquanto no segundo caso se procura uma maior uniformidade de distribuição de carga.

Sem pôr de parte a necessidade de avaliar as duas opções, a presente simulação tomou por base a primeira opção e considerou que cada nodo, cliente de uma dada operação, fica com o ónus de contactar em sequência todos os nodos envolvidos nessa execução. Contudo, caso a execução de uma dada operação acarrete reestruturações da estrutura da árvore, a iniciativa de desencadear esse processo e a sua coordenação já será da responsabilidade dos nodos que controlam os blocos que deram origem à reestruturação.

A simulação do mecanismo de *caching* dos blocos mais recentemente usados irá considerar esta *cache* como local a cada nodo, constituindo assim uma primeira abordagem à gestão do mecanismo de *caching*, que parte igualmente do pressuposto de que as operações são geridas a partir dos nodos que lhes dão origem. Não existe, nesta fase, uma composição em camadas do mecanismo de *caching*.

A execução da simulação consistiu em inserir todos os documentos, agrupando-os em conjuntos de 100 documentos. Estes grupos de 100 documentos representam um eventual conteúdo distinto a anunciar em cada nodo. O simulador executou a inserção dos conjuntos de documentos utilizando uma *cache* diferente para cada conjunto, o que corresponde à *cache* que cada nodo teria como resultado das suas operações e da sua visão sobre o sistema. A inserção é precedida pela construção local de um índice invertido com os 100 documentos e prossegue com a inserção palavra a palavra das associações $palavra \mapsto local$ no índice invertido distribuído, criando-se deste modo alguma localidade de referência no acesso ao índice.

Não foi simulada nenhuma operação de procura uma vez que o seu impacto no sistema é, grosso modo, equivalente ao da operação de inserção. Ambos os tipos de operação fazem uma travessia vertical da árvore-B+ e consequentemente vão utilizar o mesmo padrão de acesso a blocos.

A Figura 3 mostra o número total de acessos em cada bloco do sistema. Como se pode observar, existe uma assimetria no acesso aos blocos, que está relacionada com a alta frequência de determinadas palavras. Os blocos raiz das palavras mais frequentes são alvo de contenção pois qualquer operação requer sempre o acesso à raiz da árvore-B+ que utiliza. Nas palavras mais frequentes observam-se mais de 700 acessos aos respectivos blocos raiz.

Como se pode ver na Figura 4, a utilização de uma *cache* com capacidade para 10 blocos em cada nodo, reduz em uma ordem de grandeza o número de acessos aos blocos mais requisitados, passando o número máximo de acessos para menos de 60. Adicionalmente, pode-se constatar uma maior uniformidade no número de acessos que incide sobre a generalidade dos blocos, o que vai de encontro ao objectivo de eliminar pontos de contenção.

A presente simulação é um primeiro passo para a verificação de que a adaptação de DHTs para a disponibilização de índices invertidos é possível sem que tal acarrete a introdução de zonas de contenção não derrimíveis. Estão assim estabelecidas as bases para uma compreensão mais aprofundada dos impactos das diversas opções de alocação de nodos e de outras políticas de *caching*.

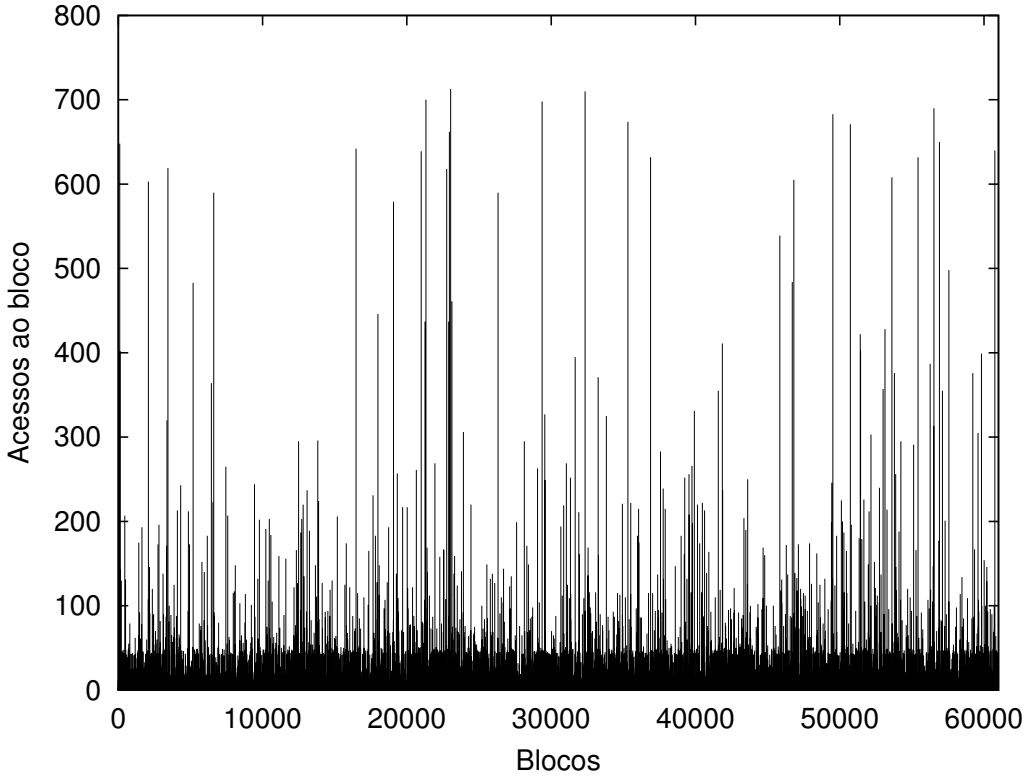

Figura 3: Carga no acesso a blocos sem a presença de *caching* nos nodos clientes.

6 Conclusões

É praticamente uma evidência que a disponibilização autónoma de conteúdos com base numa infra-estrutura P2P necessita de ser acompanhada de mecanismos de pesquisa. Mesmo na situação em que conteúdos multimédia são anunciados por pequenas frases descritivas, continuam vigentes as propriedades estatísticas que condicionam a construção de índices. Não é espectável que os actuais e ineficientes modelos de partilha P2P, que ainda recorrem em grande parte ao modelo Gnutella, venham a ser substituídos por abordagens eficientes com boa adaptação à escala, nomeadamente DHTs, se tal implicar a perda da capacidade de busca.

Neste artigo procurou-se perspectivar uma solução para o problema da construção de um índice invertido sobre P2P a partir da aplicação de um modelo de armazenamento com base em árvores-B+ e em que o mecanismo DHT serve de suporte à gestão dos blocos de armazenamento. Tal, é feito sem que novos pontos significativos de contenção sejam introduzidos na solução distribuída e sem grandes penalizações nas propriedades de escala dos DHT.

Em aberto, e como trabalho futuro, podem ser referidas duas linhas relevantes: Uma completa análise dos valores médios de complexidade das diversas operações, quando se tome em conta os efeitos da *cache* de nodos e do controlo de concorrência, face às distribuições usuais de conteúdos; A extensão do modelo de acesso ao índice invertido por forma a enquadrar com eficiência a aplicação de operadores conjuntivos e disjuntivos nas expressões de busca.

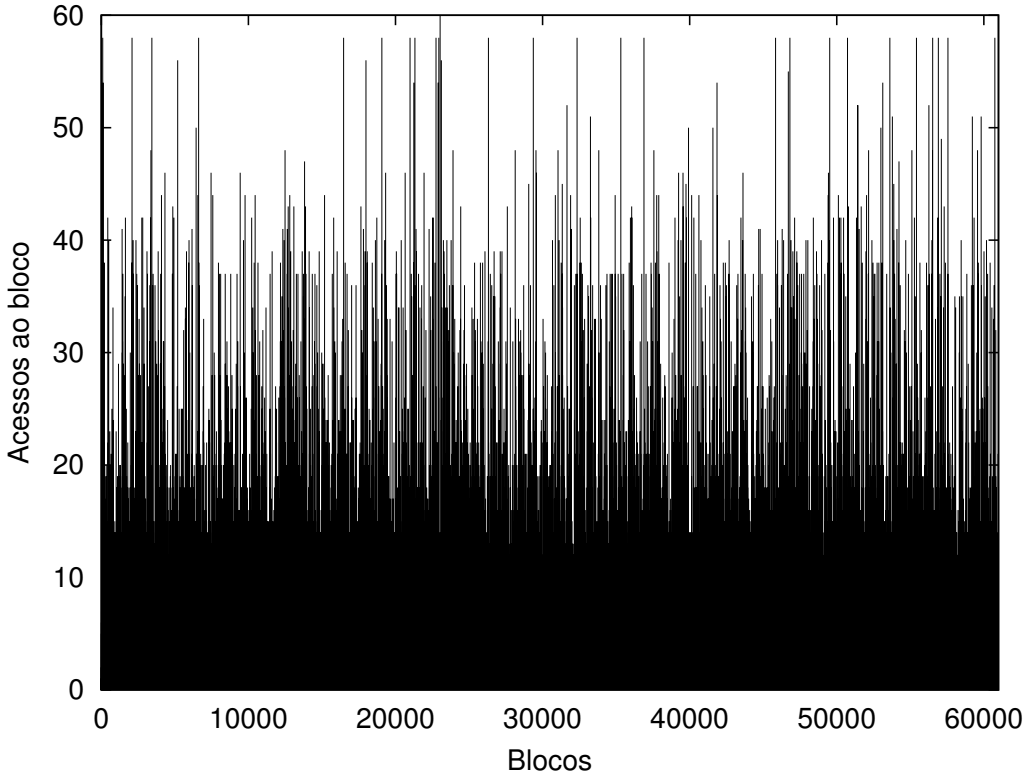

Figura 4: Carga no acesso a blocos com *caching*, nos nodos clientes, dos 10 blocos mais recentemente usados (LRU).

Referências

[1] R. Baeza-Yates and Ribeiro-Neto B. *Modern Information Retrieval*. ACM Press, NewYork, 1999.

[2] Zipf G. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, 1949.

[3] Gnutella website. http://gnutella.wego.com/.

[4] Fernando Pedone, Nelson Duarte, and Mario Goulart. Probabilistic queries in large-scale networks. In *Proceedings of the 4th European Dependable Computing Conference*, pages 209–226, 2002.

[5] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*, Germany, 2001.

[6] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN'02)*, San Jose, CA, USA, January 2002.

[7] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM'01 Conference*, pages 149–160, 2001.