Uma Arquitectura para Múltiplas Tabelas de Hash Distribuídas em Ambiente Cluster

Relatório Técnico
versão 0.8.2 – Dezembro 2004

José Rufino
Instituto Politécnico de Bragança
rufino@ipb.pt

1 Introdução

Quando as aplicações necessitam de lidar com grandes volumes de dados, para lá da capacidade de armazenamento do sistema computacional hospedeiro, deixa de ser possível a utilização de estruturas de dados convencionais (centralizadas), sendo necessário o recurso a *Estruturas de Dados Distribuídas* (DDSs), realizadas através da cooperação de vários nós computacionais.

Idealmente, a realização de uma DDS deveria ser feita no quadro de uma política global de rentabilização dos recursos dos nós computacionais de suporte, que leva em linha de conta a sua eventual (ou melhor, inevitável) *heterogeneidade*.

Essa heterogeneidade decorre, em primeira instância, de perfis de *hardware* eventualmente diferentes – *heterogeneidade física*. Dependendo do ambiente distribuído, essa heterogeneidade pode ser em maior ou menor grau (*e.g.*, ambientes *Peer-to-Peer* (P2P) versus ambientes *cluster*), mas é relativamente *estática*, pois as configurações de hardware dos nós, que definem as suas *capacidades*, mudam pouco e lentamente.

Outra fonte de heterogeneidade é a execução concorrente, em cada nó, de múltiplas aplicações com requisitos específicos e dinâmicos, originando flutuações na *disponibilidade* dos seus recursos. Esta heterogeneidade, que classificamos de *heterogeneidade lógica* (porque originada ao nível do *software*) é inerentemente *dinâmica*, levantando por isso questões relativamente complexas.

Para além do tratamento da heterogeneidade do ambiente de execução, é também desejável suportar a co-existência de múltiplas instâncias de DDSs, eventualmente partilhadas por diversas aplicações. Assim, são necessários serviços capazes de gerir, de forma integrada, a participação simultânea de um nó computacional em diversas DDSs ao mesmo tempo que, em colaboração uns com os outros, asseguram uma utilização balanceada do conjunto de recursos que as sustentam.

Certas classes de aplicações estão particularmente interessadas em *dicionários distribuídos*, repositórios distribuídos em que o acesso a cada registo se faz por uma chave única. Neste contexto, Árvores e Tabelas de Hash distribuídas são, provavelmente, os paradigmas mais populares.

Neste documento descrevem-se os aspectos gerais da proposta de uma arquitectura que, no âmbito de um ambiente de computação paralela e distribuída do tipo cluster, pretende suportar a realização, exploração e gestão de múltiplas *Tabelas de Hash Distribuídas* (DHTs), levando em conta a heterogeneidade dos nós computacionais que as sustentam.

1.1 Relação com Trabalho Anterior

A arquitectura que se propõe surge na sequência de trabalho anterior, que originou modelos para:

- a fragmentação do contradomínio (ou *espaço de endereçamento*) de uma função de hash, e sua distribuição balanceada através de um conjunto de nós heterogéneos [1, 2];
- a localização daqueles fragmentos, com base em algoritmos de localização distribuída, desenvolvidos para a) levar em conta a maior proximidade entre tempos de comunicação e computação, no ambiente cluster, e b) realizar *encaminhamento agregado*, acelerando significativamente o processo de localização.

Assim, partindo dos modelos desenvolvidos, que fornecem a base a partir da qual é possível assegurar o balanceamento de uma DHT no *eixo do endereçamento*, atacamos agora o problema do balanceamento no *eixo do armazenamento*. É portanto necessário definir os mecanismos adequados ao balanceamento do *espaço de armazenamento* disponível no cluster (no âmbito da sua utilização por uma ou mais DHTs), e investigar de que forma se interligam com (e influenciam) os mecanismos de balanceamento do espaço de endereçamento.

2 Visão Geral da Arquitectura

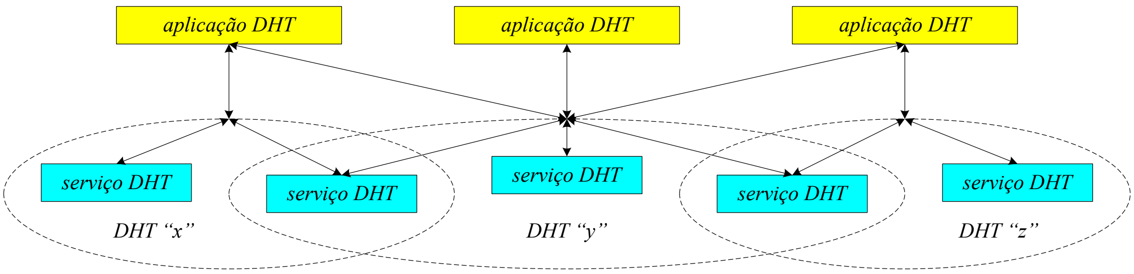

Figura 1: Arquitectura CoM_{dht} - entidades básicas e suas relações lógicas.

A figura 1 introduz as entidades básicas (e suas relações lógicas) da arquitectura que se propõe, que designamos por CoM_{dht}[1]. Assim, e numa perspectiva *top-down*:

- múltiplas aplicações (que designamos por *aplicações DHT*) necessitam de acesso a uma ou mais DHTs, podendo uma DHT ser partilhada entre várias aplicações;
- uma DHT é suportada por um conjunto de *serviços DHT*, podendo cada serviço participar na realização de várias DHTs.

A figura 2 ilustra as diversas possibilidades de disposição física das entidades básicas e evidencia a necessidade de certos *serviços mínimos* para a execução de aplicações e serviços DHT.

Discutem-se, de seguida, cada uma das entidades da arquitectura, numa perspectiva *bottom-up*.

[1]Abreviatura de *Cluster oriented Multiple DHTs*.

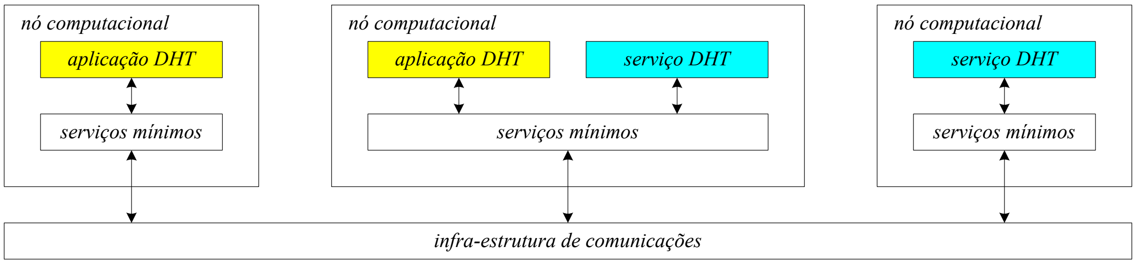

Figura 2: Arquitectura CoM_{dht} - possibilidades de disposição física das entidades básicas.

3 Serviços Mínimos

Serviços e aplicações DHT assumem a disponibilidade de uma infra-estrutura de serviços complementares, que designamos de *serviços mínimos*, e que devem contemplar pelo menos:

1. *Comunicação Nó-a-Nó entre Tarefas*;
2. *Directório Distribuído*;
3. *Execução Remota*.

A *Comunicação Nó-a-Nó entre Tarefas* é essencial para que aplicações e serviços DHT, que executam no ambiente distribuído do cluster, troquem mensagens entre si. Idealmente, este serviço deveria possibilitar a troca de mensagens entre fios de execução.

A troca de mensagens necessita, incontornavelmente, de um sistema de nomeação dos interlocutores, funcionalidade que poderá ser providenciada, por exemplo, por um serviço de *Directório Distribuído*. Este serviço poderá ainda ser usado como repositório de objectos ou estado (de pequena dimensão), partilhados entre aplicações e/ou serviços DHT[2].

Um serviço de *Execução Remota* permitirá não só efectuar o arranque de serviços DHT num conjunto inicial de nós, como ainda expandir, se necessário, esse conjunto. Esta funcionalidade beneficia também do Directório Distribuído para a nomeação e gestão das entidades envolvidas.

A plataforma $RoCL$ [3] oferece, de forma integrada, estes (e outros) serviços, através de uma implementação especialmente vocacionada para o ambiente cluster, constituindo um dos substratos possíveis para a realização da arquitectura CoM_{dht}.

4 Serviços DHT

Um *serviço DHT* é uma instância de software que participa na realização de uma ou mais DHTs, sendo assumido que, por cada nó computacional, existe no máximo um único serviço DHT em execução. Basicamente, é a cooperação entre os diversos serviços DHT em execução no cluster que permite fornecer a abstracção de uma ou mais tabelas de hash de dimensões virtualmente enormes, isolando as aplicações clientes dessas tabelas dos pormenores relativos à sua distribuição.

[2] No pior dos casos, o Directório Distribuído poderá registar a localização do objecto, em vez de conservar o próprio objecto, funcionando na prática como mecanismo de indirecção.

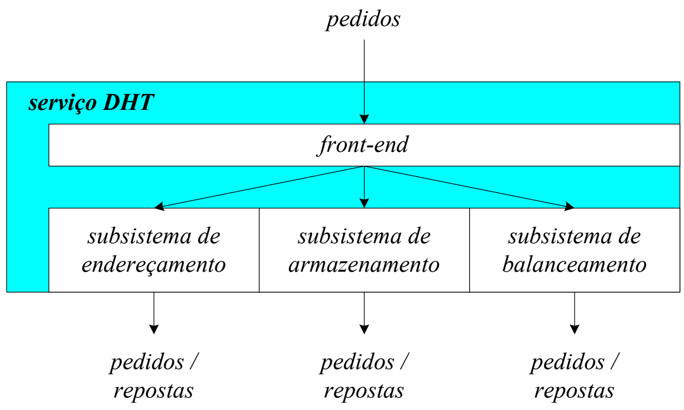

Figura 3: Serviço DHT - componentes lógicos.

Cada serviço DHT desempenha funções de *endereçamento*, *armazenamento* e *balanceamento*, em combinações diversas, através de subsistemas apropriados. Um *front-end* comum aos diversos subsistemas efectua a recepção e multiplexagem (balanceada) das mensagens dirigidas ao serviço DHT. Esta visão lógica da organização interna de um serviço DHT é veiculada pela figura 3.

4.1 Separação Endereçamento – Armazenamento

Seja h uma função de hash e H o seu contradomínio ou *espaço de endereçamento*. Então, a realização de uma DHT baseada em h implica a fragmentação total de H em *partições* e a *associação* de cada uma delas a uma entidade do sistema distribuído. Essa associação determina qual a entidade que passa a assumir a gestão do espaço de endereçamento correspondente à partição. Abreviadamente diremos que a entidade efectua a *gestão de endereçamento* da partição.

Em muitas das abordagens à realização de DHTs, se uma entidade efectua a gestão de endereçamento de uma partição, então implicitamente essa mesma entidade efectua a correspondente *gestão de armazenamento*, ou seja, é responsável pela salvaguarda de todos os dados que se mapeiam na partição, por via da aplicação da função de hash. Verifica-se portanto uma coexistência ou *união funcional*, na mesma entidade, entre as funções de endereçamento e armazenamento.

A abordagem preconizada pela arquitectura CoM_{dht} permite a *separação funcional* entre endereçamento e armazenamento, através de serviços DHT (entidades) diferentes, ou seja, se um determinado serviço assume a gestão de endereçamento de um conjunto de partições, poderá ser esse ou outro serviço a desempenhar a correspondente gestão de armazenamento. Neste contexto, designamos por *serviço de endereçamento* de uma partição o serviço DHT que efectua a gestão de endereçamento da partição. Concomitantemente, designamos por *serviço de armazenamento* de uma partição o serviço DHT responsável pela sua gestão de armazenamento.

A separação funcional entre endereçamento e armazenamento permitirá que um mecanismo global de balanceamento dinâmico atribua essas funcionalidades aos diversos serviços DHT, em gradações compatíveis com as *capacidades* e *disponibilidades* dos nós hospedeiros. Por exemplo, num nó as condições podem ser propícias à execução de funções de armazenamento (tipicamente *I/O bound* e dependentes de disponibilidade de memória secundária), ao passo que podem ser, noutro nó, adequadas ao desempenho de funções de endereçamento (execução de algoritmos de encaminhamento, tipicamente *CPU bound* e sensíveis à disponibilidade de memória primária).

A separação funcional entre endereçamento e armazenamento implica, todavia, cuidados especiais, de forma a que não introduza ineficiências desnecessárias. Para além de um acesso o mais *transparente* possível às DHTs, as aplicações DHT estão sobretudo interessadas num acesso *expedito*. A descoberta do serviço de armazenamento de qualquer partição deve ser, portanto, um processo relativamente eficiente.

4.2 Subsistema de Endereçamento

A descoberta do serviço de armazenamento de uma partição é realizada através da colaboração dos diversos serviços de endereçamento associados à DHT respectiva. Concretamente, nessa descoberta participam os *Subsistemas de Endereçamento* (SE) desses serviços, com base num esquema de localização escalável, quer no número de entidades envolvidas, quer na informação necessária.

4.2.1 Informação de Endereçamento

Um SE efectua a gestão de endereçamento de um conjunto de partições, possivelmente de DHTs diferentes. Tal implica a manutenção de *informação de endereçamento* para cada partição. Essa informação compreende, para além da identificação da partição, a) a sua *tabela de encaminhamento* e b) uma *referência de armazenamento* para o serviço de armazenamento da partição.

A tabela de encaminhamento de uma partição regista referências para o serviço de endereçamento de outras partições da mesma DHT, em número limitado e definidas deterministicamente.

A referência para o serviço de armazenamento representa uma indirecção no acesso aos dados da partição, à custa da qual se concretiza a separação funcional entre endereçamento e armazenamento. Essa referência pode representar uma definição *local* ou *remota* do serviço de armazenamento; no primeiro caso, o serviço de endereçamento e o serviço de armazenamento da partição são o mesmo serviço DHT; no segundo caso, são serviços DHT diferentes.

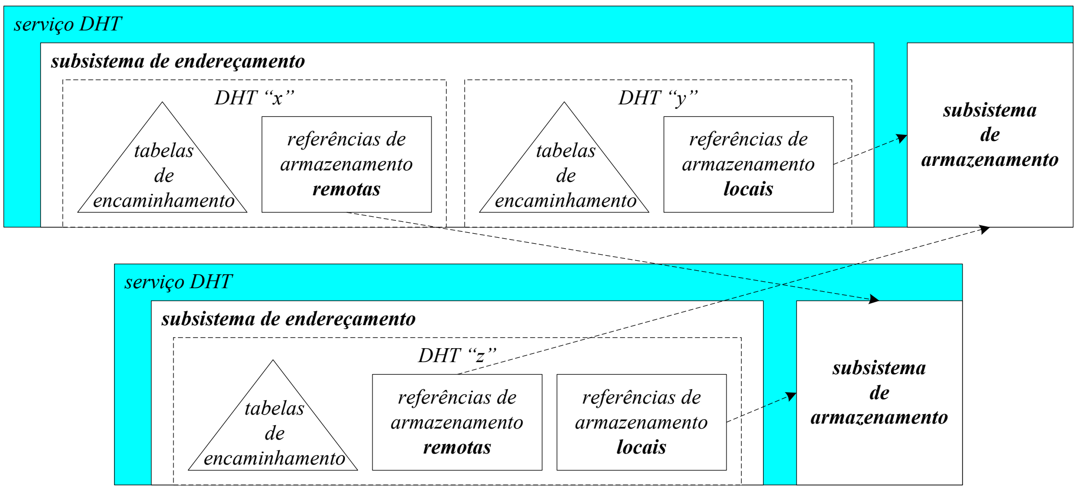

Figura 4: Subsistema de Endereçamento - organização lógica interna.

Ao nível de um SE, a informação de endereçamento é mantida em contextos separados por DHT. A figura exemplifica essa separação para as DHTs “x”, “y” e “z”. Adicionalmente, em cada SE representam-se, de forma separada, o conjunto das tabelas de encaminhamento e o conjunto das referências de armazenamento, para as partições de cada DHT. Na prática, é de esperar que ambos os conjuntos sejam reunidos numa única estrutura de dados, indexada por partição.

4.2.2 Localização Distribuída

A definição determinística da tabela de encaminhamento de todas as partições de uma DHT deriva de uma *topologia* de nível aplicacional, que interliga todas as partições da DHT e viabiliza um esquema de *localização distribuída* de qualquer partição. Neste contexto, “localizar uma partição” significa, em primeira instância, “localizar o seu serviço de endereçamento” para, em última instância, “localizar o seu serviço de armazenamento”.

A necessidade de um esquema de localização distribuída das partições radica na possibilidade de que o número destas seja muito elevado[3], ao ponto de se tornar incomportável a manutenção, em qualquer nó do cluster, do registo *total* das associações <*partição, serviço de armazenamento*>. Recorrendo a um esquema de localização distribuída, é apenas necessário manter um registo *parcial* dessas associações, em cada nó computacional.

Adicionalmente, um esquema de localização distribuída envolve um número *limitado* de entidades no processo de localização. No nosso contexto, a descoberta de uma partição exige a consulta de um número limitado de tabelas de encaminhamento de outras partições da mesma DHT. Essa consulta faz-se “navegando” na topologia que interliga as partições, à custa de *decisões de encaminhamento* realizadas por cada partição “visitada” durante uma *cadeia de encaminhamento*.

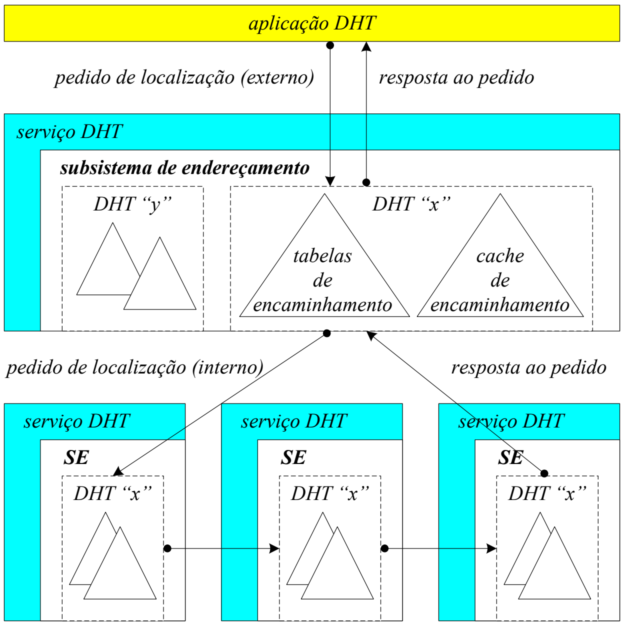

Figura 5: Subsistema de Endereçamento - exemplo de cadeia de encaminhamento.

[3] As métricas que ditam esse número caiem fora do âmbito deste documento.

A figura 5 ilustra o tipo de entidades envolvidas numa cadeia de encaminhamento, neste caso *externa*, pois é originada numa aplicação DHT. Se originada num serviço DHT dir-se-ía *interna*.

Cache de Encaminhamento É possível construir uma *cache de encaminhamento*, em cada SE, para cada DHT. Assim, sempre que um SE é ponto de partida de uma cadeia de encaminhamento interna, para uma certa DHT, a cache respectiva será alimentada com o resultado da resolução do pedido de localização, oriundo do SE final da cadeia, c.f. ilustrado na figura 5.

4.2.3 Topologias de Interligação de Partições

A arquitectura CoM_{dht} suporta duas topologias de interligação de partições: a) uma, adaptada da abordagem *Chord* para DHTs em ambientes P2P [4]; b) outra, baseada num grafo (directo) *de Bruijn*, para um alfabeto binário [5]. Cada uma destas topologias representa um compromisso diferente entre factores como: a) dimensão da tabela de encaminhamento, por partição; b) número (médio) de partições "visitadas" (ou de *saltos de encaminhamento*), por cadeia de encaminhamento; c) tempo (médio) consumido, por decisão de encaminhamento.

O facto da arquitectura CoM_{dht} ser dirigida ao cluster determina, para cada um destes factores, uma importância diferente da que teria se as decisões de encaminhamento fossem tomadas noutros ambientes, como é o caso dos ambientes P2P. Por exemplo, a disponibilidade habitual de tecnologias de comunicação de elevado débito (da ordem dos Gbps) no cluster, aproxima tempos de comunicação aos de computação, acrescentando importância à minimização do "tempo consumido por decisão de encaminhamento", em detrimento da minimização do "número médio de saltos de encaminhamento".

4.2.4 Encaminhamento Agregado

O conjunto de todas as tabelas de encaminhamento disponíveis num SE, para uma certa DHT, permite a realização de *encaminhamento individual* e de *encaminhamento agregado*.

No encaminhamento individual (ou convencional), uma decisão de encaminhamento é alimentada apenas pela tabela de encaminhamento da partição actualmente visitada. No encaminhamento agregado, uma decisão de encaminhamento leva em conta parte das (ou mesmo todas as) tabelas de encaminhamento, das partições da mesma DHT, mantidas pelo SE visitado.

Em termos topológicos, o encaminhamento agregado permite, habitualmente, "encurtar caminho", à custa porém de um maior tempo consumido por cada decisão de encaminhamento, que implica agora a análise de mais informação de encaminhamento. Assim, para que os algoritmos de encaminhamento agregado sejam executados da forma mais efciciente possível, a estrutura de dados que suporta o conjunto das tabelas de encaminhamento deve ser escolhida convenientemente.

Por exemplo, no caso da topologia *Chord*, o encaminhamento agregado beneficia de uma estrutura onde a pesquisa por uma certa partição retorna automaticamente, na sua ausência, uma sucessora/antecessora directa, sob uma determinada ordenação. O módulo de árvores binárias balanceadas do tipo *Red–Black*, que usámos para validar e simular algoritmos de encaminhamento agregado para a topologia *Chord*, oferece essa funcionalidade. Já no caso da topologia *de Bruijn*, foi necessário desenvolver um módulo específico de *tries compactas* para efectuar a sua simulação.

Cache de Encaminhamento Num SE, a estrutura que suporta a cache de encaminhamento de uma DHT, é semelhante à que suporta o conjunto das tabelas de encaminhamento locais da DHT. Tal permite efectuar encaminhamento agregado com os registos da cache, de forma que numa situação de *cache–miss* a cache ainda contribui para a decisão de encaminhamento. Na presença da cache, o *next–hop* resultará então da escolha da melhor das "propostas" das tabelas e da cache, ou seja, daquela que mais encurta a distância topológica até ao fim da cadeia de encaminhamento.

4.3 Subsistema de Armazenamento

Através do seu *Subsistema de Armazenamento* (SA), um serviço DHT efectua, para uma ou mais DHTs, a gestão de armazenamento de um conjunto de partições. Essa gestão consiste, para cada uma daquelas DHTs, em manter e facultar o acesso a um *repositório*, que "armazena" partições da DHT (ou melhor, que armazena os dados que nelas se mapeiam, por via da função de hash).

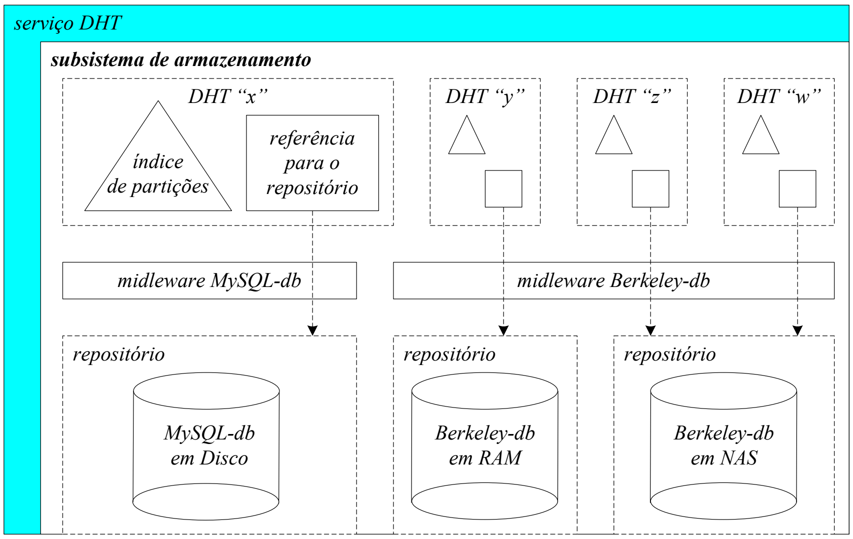

Figura 6: Subsistema de Armazenamento - entidades e relações.

4.3.1 Informação de Armazenamento

A gestão de armazenamento socorre-se de *informação de armazenamento* diversa (ver figura 6). Assim, para cada DHT mantém-se um *índice de partições* e uma *referência para o repositório* respectivo. O índice identifica todas as partições da DHT localmente armazenadas, mantendo ainda uma referência para o serviço de endereçamento de cada uma (*referência de endereçamento*).

O mapeamento "reverso" *serviço de armazenamento* → *serviço de endereçamento* é fundamental para se poder actualizar, no serviço de endereçamento de uma partição, a sua *referência de armazenamento*, para um novo serviço de armazenamento. Essa necessidade surge, por exemplo, na eventualidade de a partição ser migrada, no âmbito do balanceamento da utilização do espaço de armazenamento global do cluster.

Adicionalmente, a inspecção de índices de partições no SA companheiro de um SE, durante uma cadeia de encaminhamento, poderá economizar alguns saltos de encaminhamento, ao possibilitar a determinação prematura do serviço de armazenamento de uma partição.

4.3.2 Repositórios

O termo "repositório" refere-se a qualquer sistema de armazenamento capaz de guardar *dicionários*.

Os repositórios geridos por um SA podem ser partilhados em modalidades diversas. Assim, um repositório poderá ser *dedicado* a uma só DHT (como acontece para os repositórios das DHTs "x" e "y" na figura 6) ou *partilhado* por duas ou mais DHTs (como é o caso do repositório das DHTs "z" e "w" na mesma figura). No limite, um SA poderá manter um único repositório *unificado* para todas as DHTs. A partilha de um repositório por diversas DHTs implica a composição das chaves dos registos, antes do seu armazenamento, com informação suficiente para se distinguirem registos com a mesma chave, mas relativos a DHTs diferentes.

A arquitectura de um SA é bastante flexível, permitindo adoptar diferentes tecnologias para os repositórios, com diferentes qualidades de serviço (*e.g.*, persistência, transaccionalidade, etc.), apropriadas a diferentes cenários de aplicação. A interacção com diferentes tipos de repositórios faz-se através de *midleware* apropriado.

4.4 Subsistema de Balanceamento

A atribuição de funções de endereçamento e de armazenamento aos diversos serviços DHT é regulada por um mecanismo global de balanceamento dinâmico, no qual todos os serviços DHT participam, através do seu *Subsistema de Balanceamento* (SB).

O desempenho de funções de balanceamento é então obrigatório para qualquer serviço DHT, ao contrário das funções de endereçamento e armazenamento, que são facultativas[4]. Neste contexto, podem definir-se dois níveis de participação de um serviço DHT na realização de DHTs:

1. *participação efectiva*: o serviço DHT desempenha funções de balanceamento e pelo menos uma das funções de endereçamento ou armazenamento;

2. *participação potencial*: o serviço DHT desempenha apenas funções de balanceamento, permanecendo disponível para assumir funções de endereçamento e/ou armazenamento.

Um SB pode assumir mais ou menos funcionalidade, dependendo do tipo de participação do serviço DHT correspondente. Assim, no âmbito de uma participação potencial, o SB limita-se a responder a interrogações acerca da disponibilidade dos recursos locais. No contexto de uma participação real, para além dessa funcionalidade, o SB monitoriza periodicamente a utilização dos recursos locais (do nó hospedeiro) relevantes para o desempenho dos outros subsistemas (SE e SA). Se a disponibilidade desses recursos se reduzir abaixo de um certo limiar, o SB desencadeia um *evento de balanceamento*, do qual deverá resultar a) a descoberta de um serviço DHT com disponibilidade suficiente dos recursos em falta e b) a transferência, para esse serviço, da quota parte necessária das responsabilidades em sobrecarga; essa transferência diz respeito a *informação*

[4]Com a restrição de que, para qualquer DHT, deverá haver pelo menos um SE e um SA empenhados na gestão do seu endereçamento e do seu armazenamento.

de encaminhamento ou a blocos de dados de uma ou mais partições, c.f. se pretenda aliviar *carga de endereçamento* ou *carga de armazenamento*.

O despoletamento de eventos de balanceamento *por necessidade* traduz a opção por uma política de *load-shedding*, alternativamente a uma política de *load-stealling*. A ideia base é evitar as perturbações inerente à redistribuição (seja da carga de endereçamento, seja da carga de armazenamento) de uma DHT, até que seja realmente necessário, dado que essa redistribuição interfere com o acesso das aplicações à DHT.

Embora diversos eventos de balanceamento possam ser desencadeados individualmente por cada SB, de forma assíncrona, na prática o seu tratamento é *serializado* através de um *SB coordenador*. Esta entidade dispõe de (ou é capaz de reunir), para cada DHT, informação total sobre a carga de endereçamento/armazenamento dos serviços que a suportam. Essa informação permite tomar decisões de balanceamento à partida mais justas do que seria possível com esquemas descentralizados, tipicamente baseados num conhecimento parcial da distribuição de carga.

A função coordenadora pode ser transferida entre SBs, por forma a aliviar a carga do nó hospedeiro. Essa transferência pode ser comunicada por difusão[5] a todos os serviços DHT. Alternativamente, a descoberta do novo SB coordenador pode ser feita por redireccionamento, a partir do antigo SB coordenador. Outra hipótese é o registo dessa informação num Serviço de Directoria que, como verificamos, faz parte do pacote de serviços mínimos exigidos pelos serviços DHT.

Por fim, refira-se que a exigência da presença do SB em qualquer serviço DHT apenas se justifica numa óptica de perseguição permanente do objectivo da rentabilização de recursos. Efectivamente, uma DHT poderá sobreviver à ausência de re-balanceamentos, à custa porém de condições de operação possivelmente degradadas, que ditam um desempenho subóptimo.

4.5 Flexibilidade e Dinamismo Funcional

Para além da obrigatoriedade da execução de funções de balanceamento, um serviço DHT goza de uma *flexibilidade funcional* apreciável, podendo assumir qualquer combinação de funções de endereçamento e de armazenamento, aplicadas a qualquer DHT

Da operação do mecanismo de balanceamento decorre *dinamismo funcional*, dado que aquelas combinações podem variar ao longo do tempo. Essa variação pode mesmo conduzir, no limite, à oscilação do nível de participação de um serviço DHT entre os estados *efectivo* e *potencial*.

Assim, enquanto que noutras abordagens a DHTs, as entidades que as sustentam gozam de paridade funcional *permanente* (ou seja, desempenham todas, e sempre, o mesmo tipo de funções), a necessidade de lidar com a heterogeneidade das condições em que as aplicações executam resulta, na abordagem CoM_{dht}, numa paridade funcional que, embora *potencial*, raramente será *efectiva*.

5 Aplicações DHT

Genericamente, uma aplicação DHT será qualquer uma que interactua com serviços DHT, procurando explorar as suas funcionalidades. Todavia, nesta definição bastante abrangente, caberiam

[5] Relativamente eficiente de realizar num ambiente do tipo cluster.

também os próprios serviços DHT, uma vez que actuam como clientes uns dos outros, no decurso das suas interacções. Assim, refinamos a definição no sentido de reposicionar a aplicação DHT num contexto (lógico) exterior aos serviços, ou seja, uma aplicação DHT é uma entidade de software cliente de serviços DHT mas não é seu par, precisamente a visão veiculada pela figura 1.

5.1 Classes de Aplicações DHT

Tomando como critério de classificação o objectivo funcional das aplicações DHT, é desde logo possível classificá-las em duas categorias básicas: 1) aplicações DHT de *nível supervisor*; 2) aplicações DHT de *nível utilizador*.

As aplicações DHT de nível supervisor encaram as DHTs como objectos de gestão sendo, portanto, aplicações de carácter administrativo. Um exemplo de uma aplicação desta categoria é uma *consola de administração*, a partir da qual se efectua a gestão corrente (activação, expansão/contracção, monitorização e desactivação) das DHTs suportadas pelos serviços DHT.

Na perspectiva das aplicações DHT de nível utilizador, as DHTs são vistas como grandes repositórios, sobre os quais (por intermédio dos serviços DHT), interessa essencialmente efectuar operações de salvaguarda/escrita ou recuperação/leitura de dados. À partida, espera-se que este tipo de aplicações DHT seja o mais vulgar.

5.2 Biblioteca de Acesso aos Serviços DHT

O acesso de uma aplicação DHT aos serviços DHT efectua-se por intermédio de uma biblioteca apropriada, que designamos por *Biblioteca de Acesso aos Serviços DHT* (BAS). O uso da biblioteca por uma aplicação DHT pressupõe a disponibilidade dos serviços mínimos no nó computacional hospedeiro da aplicação DHT. A figura 7 apresenta os subsistemas da biblioteca e as suas relações, que passamos a descrever, de forma breve.

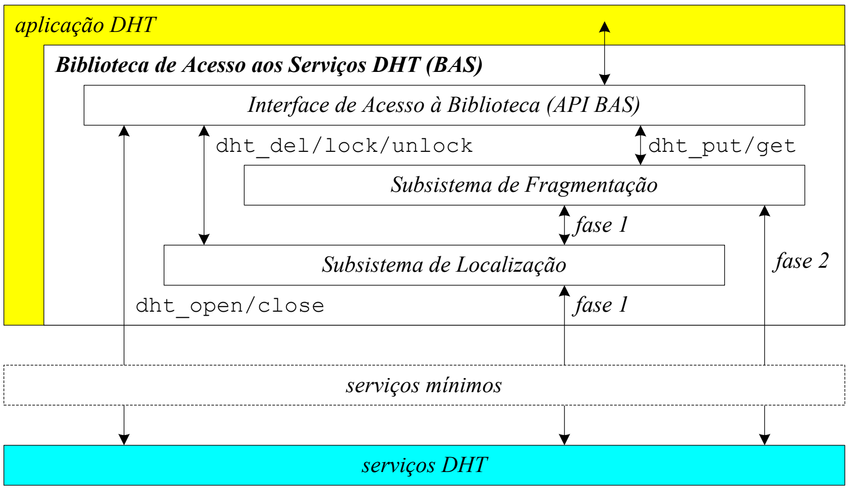

Figura 7: Biblioteca de Acesso aos Serviços DHT - entidades e relações.

5.2.1 Interface de Acesso à Biblioteca

A biblioteca BAS publica um conjunto de funções, de interface normalizado, que constituem a API BAS. Essas funções compreendem funcionalidades de *acesso corrente*, como

- `dht_handler=dht_open(dht_name)`:
  obter uma referência para uma DHT;
- `dht_status=dht_close(dht_handler)`:
  prescindir de uma referência para uma DHT;
- `dht_status=dht_put(dht_handler, dht_key, dht_data)`:
  escrever um registo numa DHT;
- `dht_status=dht_del(dht_handler, dht_key)`:
  remover um registo de uma DHT;
- `dht_status=dht_get(dht_handler, dht_key, dht_data)`:
  ler um registo de uma DHT;
- `dht_status=dht_lock(dht_handler, dht_key)`:
  bloquear (o acesso a) um registo de uma DHT;
- `dht_status=dht_unlock(dht_handler, dht_key)`:
  desbloquear (o acesso a) um registo de uma DHT.

Destas funcionalidades, as parametrizadas por uma chave `dht_key` necessitam da intervenção de um *Subsistema de Localização* e as parametrizadas por dados `dht_data` necessitam eventualmente dos préstimos de um *Subsistema de Fragmentação*. A relação de cada uma das funcionalidades de acesso corrente com estes subsistemas é evidenciada na figura 7.

Estão também disponíveis funcionalidades de *acesso administrativo*, como

- `dht_handler=dht_create(dht_name, dht_properties)`:
  criação de uma DHT;
- `dht_status=dht_destroy(dht_handler)`:
  destruição de uma DHT;
- `dht_handler=dht_boot(dht_name)`:
  reconstituição de uma DHT a partir de repositórios persistentes;
- `dht_status=dht_shutdown(dht_handler)`:
  salvaguarda de uma DHT em repositórios persistentes, seguida de desactivação;
- `dht_status=dht_change_properties(dht_handler, dht_properties)`:
  modificação de propriedades de uma DHT;
- `dht_status=dht_get_properties(dht_handler, dht_properties)`:
  inspecção de propriedades de uma DHT.

As funções anteriores acedem directamente aos serviços DHT e exploram também funcionalidades dos serviços mínimos (como por exemplo a salvaguarda das propriedades de cada DHT que é efectuada ao nível de um Serviço de Directoria).

5.2.2 Subsistema de Fragmentação

O *Subsistema de Fragmentação* (SF) efectua a fragmentação/reconstrução, para efeitos de armazenamento/recuperação, de *registos* (pares <dht_key, dht_data>) em/a partir de *blocos* (unidade mínima de armazenamento definida para cada DHT).

A fragmentação dos registos em blocos é opcional, sendo especificada a sua realização, para cada DHT, no acto da sua criação (através do parâmetro dht_properties, da função dht_open, que também possibilita a definição da dimensão do bloco).

Sem fragmentação, um registo será armazenado, na sua totalidade, pelo *serviço de armazenamento* da partição na qual se mapeia a chave do registo. Com fragmentação, cada bloco será armazenado de forma independente, levando em conta uma chave própria, deduzida da combinação do índice do bloco e da chave do registo.

A subdivisão dos registos em blocos tem, entre outras, as seguintes vantagens: a) uma utilização mais equilibrada do espaço de armazenamento dos repositórios, uma vez que se previnem situações em que um número reduzido de registos, de grandes dimensões, açambarca a maior parte do espaço disponível; b) eventuais ganhos de desempenho, derivados da possibilidade de localizar e aceder, de forma paralela, aos diversos blocos de um registo, situados provavelmente em repositórios sitos em nós computacionais diferentes.

5.2.3 Subsistema de Localização

O acesso a um registo (ou bloco) cuja chave pertence a uma determinada partição ocorre em duas fases (ver figura 7), a primeira das quais implica a intervenção do *Subsistema de Localização* (SL):

- *fase 1*: descobrir o *serviço de endereçamento* da partição, recebendo de volta uma referência para o correspondente *serviço de armazenamento*;
- *fase 2*: submeter o pedido de acesso ao *serviço de armazenamento*.

Para que se inicie a descoberta do serviço de endereçamento de qualquer partição de uma DHT, basta submeter o pedido de localização a qualquer serviço de endereçamento da DHT. Após a "abertura" de uma DHT, via dht_open, é garantido o conhecimento, ao nível do SL da biblioteca BAS, de pelo menos um dos serviços de endereçamento da DHT.

Complementarmente, à medida que o SL resolve pedidos de localização, pode adicionar a uma cache local o resultado da resolução. Essa cache segue a mesma estrutura da cache mantida pelo SE de um serviço DHT e é usada para acelerar a localização, recorrendo aos mesmos algoritmos de encaminhamento agregado usados no âmbito daqueles serviços.

Referências

[1] J. Rufino, A. Alves, A. Pina, and J. Exposto. Toward a dynamically balanced cluster oriented DHT. In *Proceedings of PDCN '04*, 2004.

[2] J. Rufino, A. Alves, A. Pina, and J. Exposto. A cluster oriented model for dynamically balanced DHTs. In *Proceedings of IPDPS '04*, 2004.

[3] A. Alves, A. Pina, J. Rufino, and J. Exposto. RoCL: A Resource oriented Communication Library. In *Proceedings of Euro-Par 2003*, 2003.

[4] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balkrishnan. Chord: A Scalable P2P Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM'01*, 2001.

[5] J-C. Bermond, Z. Liu, and M. Syska. Mean Eccentricities of de Bruijn Networks. Technical report, Université de Nice-Sophia Antipolis, 1993.