Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

(Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan)

Prof. Dr. Célio V. N. Albuquerque

Etienne César R. de Oliveira
Doutorando em Computação

Objetivo e Motivações

Aplicações e sistemas *peer-to-peer* são sistemas distribuídos que não dispõe de um controle centralizado, nem de uma organização hierárquica, conseqüentemente, cada nó dispõe das mesmas funcionalidades de todos os demais nós. Uma revisão das características de aplicações *peer-to-peer* aponta para uma lista extensa: armazenamento redundante, seleção do servidor mais próximo, anonimato, localização, autenticação etc. No entanto, a principal operação de sistemas *peer-to-peer* é a localização eficiente de itens de dados. Ou seja, identificar em que nó da rede encontram-se os dados solicitados.

O artigo apresenta *Chord*, um protocolo distribuído de pesquisa que se propõe a resolver esta questão. *Chord* realiza apenas uma função: dado uma chave qualquer, *Chord* mapeia esta chave em um nó. A localização de dados pode ser facilmente implementada através da associação de uma chave a um dado específico, e armazenando o par chave/dado em um nó mapeado pela chave. Os autores afirmam que o algoritmo proposto adapta-se eficientemente às mudanças de topologia, ou seja, as inclusões e exclusões de nós. Resultados obtidos através de análises teóricas, simulações e experimentos demonstram que *Chord* é escalável, com custo de comunicação e o estado mantido por cada nó escalando de forma logaritma ao número de nós.

Proposta

O balanceamento de carga entre os diversos nós da rede é alcançado através de uma variante do método *hashing* consistente [ver referência 11 do artigo], proporcionando uma distribuição de chaves bastante uniforme entre os nós. O identificador do nó é calculado a partir do seu endereço IP e o identificador da chave a partir da própria chave, gerando-se um identificador de *m-bits* baseado em alguma função de *hash*, como SHA-1. Além do balanceamento de carga, os autores descrevem outras funcionalidades, tais como:

- Descentralização – *Chord* é totalmente distribuído: todos os nós têm o mesmo grau de importância, o que torna esta proposta robusta e indicada para aplicações *peer-to-peer* livremente organizadas;

- Escalabilidade – Como o custo para localização cresce a uma escala *O(logN)*, mesmo para sistemas com uma grande quantidade de nós, a localização é factível;
- Disponibilidade – *Chord* é capaz de ajustar automaticamente suas tabelas internas, refletindo tanto as inserções quanto as remoções de nós. Desconsiderando-se a existência de problemas nas camadas de rede inferiores, o nó responsável por uma chave poderá ser sempre localizado.

De forma a manter o processo de localização de chaves escalável, os nós mantêm uma tabela de roteamento com, no máximo, *m* entradas, sendo *m* a quantidade de bits do identificador da chave/nó. Os nós armazenam informações sobre uma quantidade pequena de outros nós e mantêm mais informações sobre os nós que encontram-se próximos ao seu círculo identificador.

A inserção de um nó pode afetar a pesquisa de alguma chave que tenha sido solicitada antes da estabilização do círculo (anel). Três situações distintas podem ocorrer. As tabelas mantidas pelos nós envolvidos na pesquisa encontram-se atualizadas, possibilitando a identificação do sucessor desejado em *O(logN)*. Na segunda situação os ponteiros para os sucessores encontram-se corretos, no entanto os *fingers* contêm informações imprecisas. Neste caso, a pesquisa será mais lenta. Na terceira situação, os nós da região afetada pela inserção apresentam ponteiros para sucessores incorretos ou as chaves ainda foram migradas para os novos nós inseridos. Conseqüentemente, a pesquisa irá falhar. Cabe, então, a camada superior solicitar uma nova pesquisa após um pequeno intervalo de tempo.

O esquema utilizado pelo *Chord* garante a inserção de novos nós de forma a preservar a localização dos nós existentes, mesmo em função de reordenações, perdas ou inserções de nós de forma concorrente. De acordo com os autores, problemas com o surgimento de topologias com múltiplos círculos desconexos ou círculos em *loop* não podem ser ocasionados pelas ações de inserção, remoção ou re-ordenamento. Não se pode afirmar que estes problemas podem surgir em função do particionamento e recuperação de redes ou falhas intermitentes.

Se um nó *n* falhar, os nós cujas tabelas incluem *n* devem localizar o sucessor de *n*. Além disso, a falha do nó *n* não deve causar a interrupção da localização em andamento e, enquanto isso, o sistema se estabiliza novamente.

Vantagens

O algoritmo *Chord* pode ser implementado de forma iterativa ou recursiva. Em relação ao percurso necessário para se alcançar uma chave, os resultados apresentaram uma resposta eficiente, com a maioria das localizações resolvidas em $\frac{1}{2} \log_2 N$.

Desvantagens e Limitações

Os resultados obtidos no processo de simulação foram baseados em uma versão antiga do algoritmo de estabilização. Os autores não conseguiram identificar e nem descrever situações que poderiam causar o particionamento da rede.

Em todas simulações realizadas, com quantidades diferentes de nós, alguns nós não armazenavam chaves, indicando que a forma de distribuição poderia ser melhorada. A solução foi o uso de nós virtuais, o que provocaria um aumento da tabela de roteamento e no tempo de recuperação das chaves.

A quantidade de nós utilizada no experimento não reflete a realidade da proposta de uso.