Trabalho 1 de MO401 - Arquitetura de computadores I
Prof.: Paulo Centoducatte
Unicamp

Resumo do artigo:

"Bigtable: A Distributed Storage System for Structured Data"

(Citação bibliográfica: Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. 2008. Bigtable: A distributed storage systemfor structured data. ACM Trans. Comput. Syst. 26, 2, Article 4 (June 2008), 26 pages. DOI = 10.1145/1365815.1365816. http://doi.acm.org/10.1145/1365815.1365816.)

por **Heitor Lellis Nicoliello**
R.A. nº 089041

O **Bigtable** é um sistema distribuído de armazenamento de dados em larga escala usado por muitos projetos da Google. Custou sete homens-ano e entrou em produção em 2006. Não suporta um modelo relacional; provê ao usuário um modelo simples que suporta controle dinâmico sobre o formato dos dados. Os fatos mostram que este design funciona muito bem em prática, já que, apesar da interface pouco usual, os usuários, com demandas bem distintas, adaptaram-se bem e também pelo fato do sistema ser facilmente escalável, bastando adicionar máquinas ao sistema.

Modelo

Uma tabela é um mapa esparso, distribuído, persistente e multi-dimensional, organizado em três dimensões - linha, coluna e hora - formando uma célula. Linhas com chaves consecutivas são agrupadas em *tablets* - unidades de distribuição e balanceamento de carga. As colunas são agrupadas em famílias que formam a unidade de controle. Várias versões de uma mesma célula podem ser armazenadas, indexadas pelo horário. A unidade transacional é a linha (não oferece transacionamento além de uma única linha).

O Bigtable tem três componentes principais: uma biblioteca comum aos clientes, um servidor mestre e vários servidores de *tablets*. O servidor mestre é responsável por designar *tablets* para os servidores de *tablets* - balanceando cargas - e também por responder por mudanças nos esquemas. Cada servidor de *tablet* responde às requisições de leituras e escritas nas tabelas. O serviço de travas Chubby é usado para administrar os servidores.

Refinamentos

Alguns refinamentos foram necessários para atingir alto desempenho, dentre os quais: compactações e divisões de tabelas feitas quando tamanhos alcançam certos limites para manter rápidos os acessos; agrupamento locais (o usuário define a cada família de colunas um grupo de localização, assim informações que serão acessadas freqüentemente em conjunto ficam juntas); compressão local (esses grupos locais podem ser comprimidos ou não, assim não é necessário expandir um arquivo todo para acessar apenas um grupo); servidores de *tablets* usando dois níveis de *cache*; filtros *bloom*, com os quais o usuário pode criar um filtro para um grupo local para informar se um certo arquivo pode conter dados de um par específico de linha-coluna; *commit-log*, um arquivo com vários *commits* de várias *tablets* para agilizar a escrita de dados; agilização da recuperação de tabela, que consiste em compactar uma tabela antes de retirá-la do sistema para diminuir o tempo de recuperação desta tabela, se for o caso; e exploração da imutabilidade dos arquivos SSTable, que elimina problemas como sincronização de leitura a esses arquivos.

Discussões finais

Foram escolhidos seis benchmarks de operações (variando entre leitura e escrita) para analisar a quantidade de leituras e escritas por segundo variando-se o número de *tablets* por servidor. Alguns resultados interessantes: Leituras seqüenciais são mais rápidas que aleatórias; escritas são mais rápidas que leituras; e escritas aleatórias e seqüenciais têm desempenho similares. O *throughput* aumenta vertiginosamente conforme se aumenta o número de servidores, exceto entre uma e 50 máquinas, segmento no qual há uma queda no *throughput*, causada por falta de balanceamento entre o servidores na rede.

Eis as lições que os autores dizem ter aprendido: sistemas distribuídos são vulneráveis a muito mais falhas do que as padrão assumidas em protocolos; funcionalidades devem ser adicionadas somente quando ficar claro quando elas serão usadas, assim elas podem ser mais focadas; monitorar o sistema em todos os níveis; e manter o design simples (que não significa design mais fácil de elaborar).