LevelDB é um banco de dados NoSQL, de chave-valor de código aberto escrito em C++.

Origens do LevelDB

LevelDB foi criado por Jeff Dean e Sanjay Ghemawat, que trabalham no Google. Ele é baseado na implementação do Bigtable, outra ferramenta semelhante do Google que tem dependências em bibliotecas proprietárias internas do Google, então eles queriam construir algo com conceitos semelhantes do Bigtable que pode ser de código aberto.

Características

LevelDB tem três operações básicas:

Get

Put

Delete

Get recupera um valor de uma chave, Put grava um valor em uma chave, criando a chave se ela não existir, e Delete exclui a chave e seu valor.

Existem funções *open* (leva um argumento de nome de arquivo) e *close* para criar / carregar e descarregar um banco de dados, e funções que retornam iteradores sobre todas as chaves e valores.

As chaves e valores podem ser qualquer matriz de bytes e não apenas strings. Isso é útil se você tiver dados que deseja armazenar e não deseja codificar em uma string.

LevelDB oferece suporte a operações atômicas. Você pode executar várias operações ao mesmo tempo em uma única chamada ininterrupta.

Tudo isso contribui para um banco de dados extremamente simples que, ao mesmo tempo, tem uma implementação otimizada e extremamente eficiente.

Diferentes bibliotecas do LevelDB

Cada linguagem tem sua própria implementação da biblioteca LevelDB, a biblioteca principal é usada para C ++.

L[evelup](https://github.com/Level/levelup) é uma versão para NodeJS, que por si só é um wrapper para o pacote leveldown [de](https://github.com/Level/leveldown/) baixo nível.

Memdown é o mesmo que leveldown, mas armazena o banco de dados na memória, portanto, é descartado se o banco de dados for fechado ou o aplicativo sair, o banco de dados será excluído. E, finalmente, [level-js](https://github.com/Level/level-js) quando usado junto com Browserify permite que você use LevelDB em navegadores.

#### Controle de simultaneidade

Leveldb permite que apenas um processo seja aberto por vez. O sistema operacional usará o esquema de bloqueio para evitar o acesso simultâneo. Dentro de um processo, o Leveldb pode ser acessado por vários threads. Para multi-gravadores, só permitirá que o primeiro gravador grave no banco de dados e outros gravadores serão bloqueados. Para conflitos de leitura e gravação, os leitores podem recuperar dados do imutável, que é separado do processo de gravação. A versão atualizada entrará em vigor no processo de compactação.

Tamanho do bloco

leveldb agrupa chaves adjacentes no mesmo bloco e tal bloco é a unidade de transferência de e para o armazenamento persistente. O tamanho do bloco padrão é de aproximadamente 4096 bytes não compactados. Os aplicativos que geralmente fazem varreduras em massa sobre o conteúdo do banco de dados podem desejar aumentar esse tamanho. Os aplicativos que fazem muitas leituras de pontos de valores pequenos podem desejar mudar para um tamanho de bloco menor se as medições de desempenho indicarem uma melhoria. Não há muito benefício em usar blocos menores que um kilobyte ou maiores que alguns megabytes.

Compressão

Cada bloco é compactado individualmente antes de ser gravado no armazenamento persistente. A compactação está ativada por padrão, pois o método de compactação padrão é muito rápido e é automaticamente desativado para dados não compactáveis. Em casos raros, os aplicativos podem desejar desativar totalmente a compactação

Gravações síncronas

Por padrão, cada gravação no leveldb é assíncrona: ela retorna após enviar a gravação do processo para o sistema operacional. A transferência da memória do sistema operacional para o armazenamento persistente subjacente ocorre de forma assíncrona. O sinalizador de sincronização pode ser ativado para uma gravação específica para fazer com que a operação de gravação não retorne até que os dados sendo gravados tenham sido enviados para o armazenamento persistente.

Arquitetura do Sistema

No leveldb, os imutáveis ​​são armazenados no disco e podem ser compartilhados por diferentes nós do cluster. Existem no total 7 níveis mais no máximo duas tabelas na memória. O procedimento pode ser descrito como, em primeiro lugar, as operações de gravação dos buffers do sistema em uma tabela na memória chamada MEMTable que descarrega os dados no disco quando ele fica cheio. No disco, as tabelas são organizadas em níveis. Cada nível contém várias tabelas chamadas SSTable. O nível inferior mantém uma capacidade maior do que o nível superior. Quando o nível superior está cheio, o sistema precisa enviar os dados para o nível inferior, o que pode precisar ler e gravar vários SSTables.

A biblioteca Memdown