Relatório de Sistemas Operativos

KOS: Key value store of the OS course – Parte 2

4 de Dezembro de 2013

Grupo 48
74262, Gonçalo Filipe Lampreia Vicente
68603, Margarida Alexandra Quitério Flores
74248, João Miguel Loureiro Velez

Índice

1. Concorrência

Número de Clientes ≠ Número de Servidores

O grupo tentou implementar uma concorrência entre o número de clientes, o número de servidores e o número de posições no buffer, mas durante tentativa de execução obteve-se diversos erros de sincronização. Devido aos diversos erros durante a implementação, conseguiu-se apenas concretizar a concorrência entre o número de clientes e o número de servidores(Apêndice A).

Paralelismo no acesso ao KOS

Permitir acessos paralelos a partições diferentes:

Implementou-se o acesso paralelo a partições (shards) diferentes logo na primeira parte do projecto, e manteve-se esta esta estrutura para as funções get e getAllKeys, as funções de leitura, estas funções esperam que a partição não esteja em utilização para usarem a partição e depois de a usarem avisam que já acabaram.

Permitir acessos em leitura paralelos numa partição:

No entanto todas as funções verificam se a partição está a ser usada, por um get ou um getAllKeys, e caso estejam esperam que elas acabem, avisando logo de seguida que já acabaram, para que não haja conflito entre o get, getAllKeys e as outras funções do KoS.

Permitir acessos paralelos a listas distintas numa partição:

Conseguiu-se, também, implementar o acesso paralelo a listas distintas na mesma partição para as funções put e remove.

Permitir sequências de pesquisa e acessos em leitura paralelos num partição:

Fez-se do ponto 1 ao 3, mas como não o grupo não compreendeu, nem houve tempo para implementar o ponto 4 (Permitir sequências de pesquisa [varrimento de uma lista antes de atingir o local do put ou do remove] e acessos em leitura paralelos numa partição.).

2. Persistência dos dados do KOS

Para guardar as partições, mantendo os dados do KOS mesmo depois do fecho do programa usou-se as funções open(2), close(2), read(2) e write(2).

A inicialização de cada partição em memória é feita a partir do ficheiro correspondente na fase de inicialização do sistema.

Gravou-se os ficheiros com o nome f(shardId); para a partição 0 o ficheiro é o f0, para a partição 1 o ficheiro é f1, para a partição 2 é f2, e assim sucessivamente.

Se não existir este ficheiro assume-se que a partição não tem dados, devendo ser inicializada vazia em memória e criado e inicializado o ficheiro correspondente.

Caso o ficheiro exista, importa-se os dados e insere-se na partição correspondente. Infelizmente existe um erro ao escrever no ficheiro e ao importar o ficheiro, a função Read_File() insere na partição pares a mais e a função Write_file() escreve no ficheiro dois valores como um par em vez de um par com uma chave e um valor. (o grupo continua a tentar compreender este erro).

A escrita é atómica, isto é, a operação de escrita decorrente da execução de um pedido put ou remove é realizada na partição em memória e é imediatamente propagada para o ficheiro-partição.

Sempre que o servidor executa um put escreve-se no ficheiro o par, a chave tem SIZE(20) caracteres e o value tem mais SIZE(20) caracteres, mesmo que tenha apenas x caracteres o resto é preenchido por caracteres ‘\00’. Sempre que ele remove nós substituímos a linha por uma linha de caracteres ‘\00’(Apêndice B).

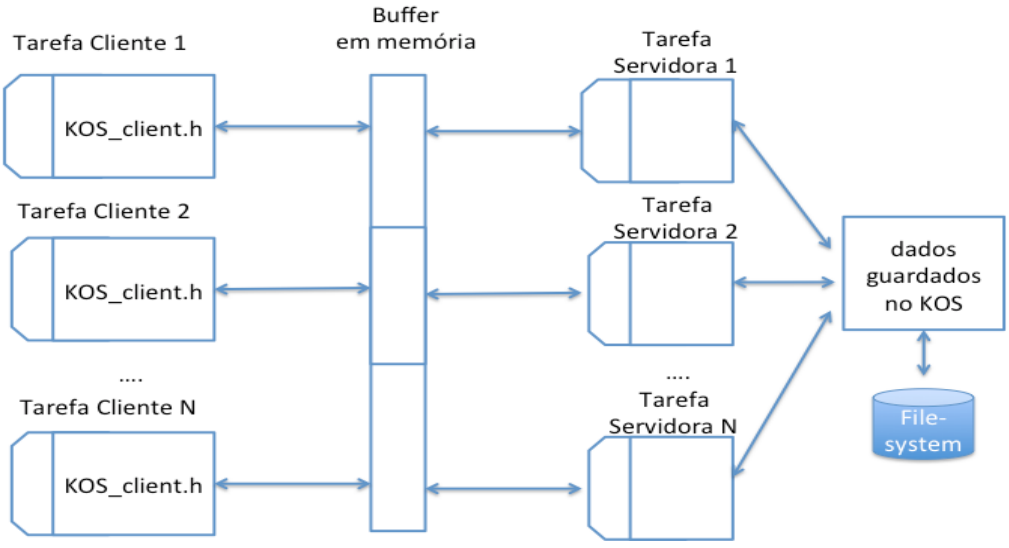
Uma operação que implique uma escrita (put, remove) só se considera concluída após a escrita da informação em memória e no ficheiro. Considera-se que a escrita no ficheiro foi realizada após o retorno da chamada à função da API do sistema de ficheiros UNIX que efetua a escrita.

Usou-se uma flag na abertura do ficheiro, O_SYNC, que faz com que o servidor não possa prosseguir sem que acabe de escrever no ficheiro, para evitar que haja uma interrupção sem que a informação seja escrita.

O ficheiro gravado fica com apenas uma linha em que a cada 40 caracteres se encontra o novo par, 20 da chave e 20 do valor sem '\n'. Implementou-se assim porque achou-se mais fácil a manipulação dos pares no ficheiro.

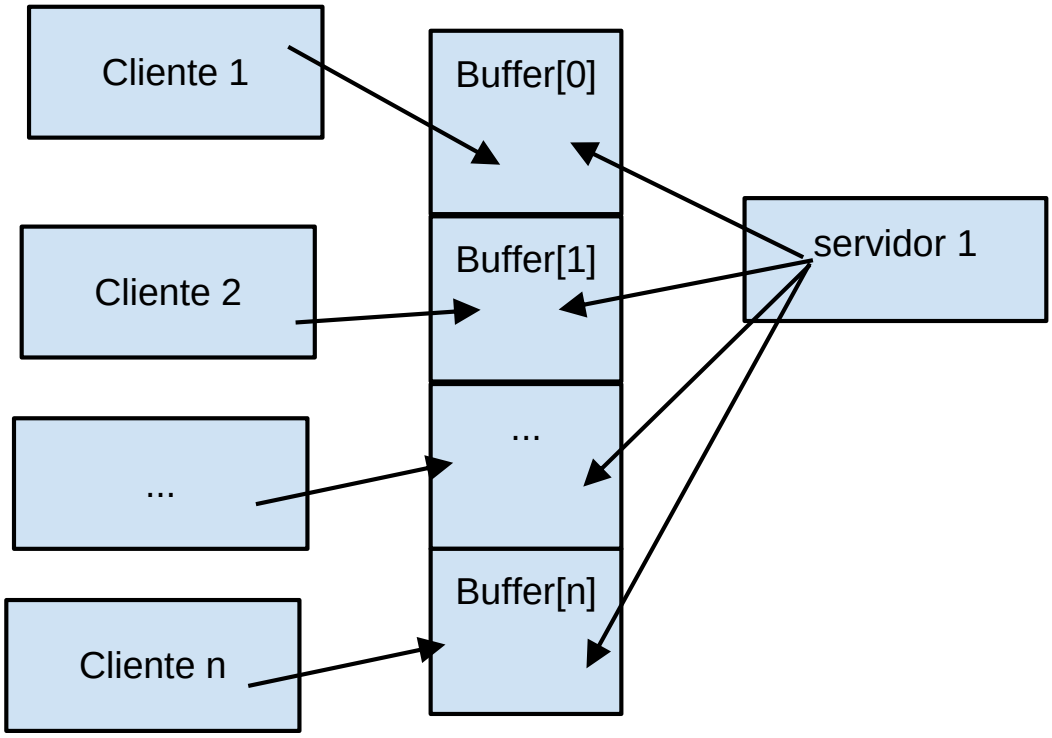
Número de pares Key-Value armazenados no ficheiro – int NPKV – seguido da sequência dos pares Key-Value guardados sem nenhuma ordem particular (opção P0):

Não se implementou esta opção visto que já se guardou o número de pares na partição numa variável da partição. Esta variável é incrementada cada vez que se insere um par. Consideramos um ficheiro vazio a partir do momento em que não se consegue ler nenhuma linha, o read(2) devolve 0 ou seja encontrou logo o caracter EOF. Não se implementou nenhuma das opções, P0 a P3 devido a falta de tempo.

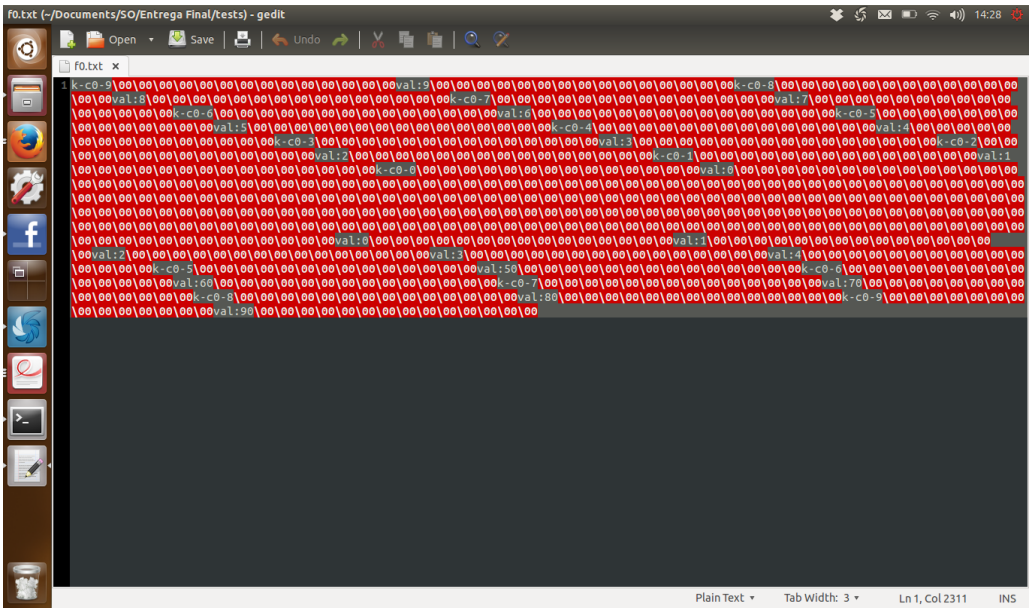


Apêndice A

Implementado na 1ª Parte do projecto. O buffer em que a cada posição corresponde um servidor e um cliente.



Implementado na segunda parte do Projecto podendo ter um servidor para vários clientes e várias posições do buffer.



Apêndice B

O que é impresso no ficheiro pela função Write_File()