Západočeská univerzita v Plzni
Fakulta aplikovanch věd
Katedra informatiky a výpočetní techniky

Základy operačních systémů

Práce s tabulkou pseudoFAT

Plzeň, 2016 David Pivovar

Obsah

1. Zadání

Tématem semestrální práce je práce s tabulkou pseudoFAT. Vstupní data budou představovat FAT tabulky a odpovídající datové bloky. Protože se bude pracovat s velkým objemem dat, je potřeba danou úlohu paralelizovat - zpracovávat ve více vláknech. Výstupem pak budou časy, kolik zpracování dané úlohy trvalo při běhu v 1, 2, 3, 4, 5, 6, a více vláknech.

Práce má 2 části:

- Kontrola, zda každý řetěz FAT má správnou délku (odpovídá velikosti souboru v adresáři).
- Setřesení volného místa (volné místo na konci, ale bloky souborů nemusí jít za sebou).

Práce bude vytvořena v C/C++ jazyce. Lze využít základní synchronizační konstrukce, které daný jazyk poskytuje.

2. Spuštění

Program je psaný v C/C++. Je přeložen kompilátorem Microsoft (R) C/C++ Optimizing Compiler Version 18 pro platformu windows.

Program má 4 argumenty při spuštění. První argument je režim programu, druhý počet vláken, které se budou spouštět. Režim a počet vláken jsou vždy povinné. Třetím je název vstupního souboru, pokud nebude zadán, bude se načítat soubor output.fat. Čtvrtým je název výstupního souboru, pokud nebude zadán, výstup se uloží do souboru output.out.fat. Do výstupního souboru se zapisuje pouze nová pseudoFAT se setřeseným volným místem.

Režimy:

- 1 - Kontrola, zda každý řetěz FAT má správnou délku (odpovídá velikosti souboru v adresáři).
- 2 - Setřesení volného místa (volné místo na konci, ale bloky souborů nemusí jít za sebou).
- 3 - Setřesení volného místa - zachová se relativní pořadí clusterů.

Program je možné spustit bash scriptem FAT.bat, který program spustí několikrát s různým počtem vláken.

Výsledné časy běhu programu jsou v souboru *out.txt*.

3. Použité struktury

3.1 fat

Struktura cele pseudoFAT.

- *p_boot_record - struktura boot record
- *fat_table - tabulka fat (pole unsigned int).
  Velikost unsigned int * počet clusterů.
- *root_directory_table - tabulka root directory (pole struct root_directory).
  Velikost struct root_directory * počet souborů.
- *cluster_table - tabulka clusterů (pole char).
  Velikost počet clusterů * velikost clusterů.

3.2 boot_record

Viz zadání: PopisFAT-verze2015-12-09-poledne.docx

3.3 root_directory

Viz zadání: PopisFAT-verze2015-12-09-poledne.docx

4. Načtení/uložení dat z/do souboru

Nejprve se načte boot record, který obsahuje informace o pseudoFAT. Poté následují tabulky FAT, které se uloží do pole uint. Za FAT tabulkami je root directory s informacemi o jednotlivých souborech. Poslední se načítají jednotlivé clustery. Ty se ukládají do pole char.

Na závěr se vytvoří instance struktury fat, která odkazuje na načtená data v paměti.

Ukládání pseudoFAT je opačný proces.

5. Kontrola, zda každý řetěz FAT má správnou délku

V tomto režimu se porovnává velikost souborů v root directory a skutečná velikost souborů.

Funkce *check()* zařídí načtení pseudoFAT a spuštění danného počtu vláken. Funkce *compare()* je volána při spuštění vlákna.

Vláknům jsou postupně ve for cyklu podle indexu přidělovány soubory z tabulky root directory. Přidělení souboru vláknu je kritická sekce programu. Z toho důvodu je vytvořen mutex, který se zamkne před čtením root directory na indexu a následnou imkrementací indexu. Dále již kritická sekce není.

Velikosti souboru se počítá včetně znaků '\0'. Podle tabulky FAT se počítají clustery náležející souboru. Velikost souboru je počet clusterů souboru bez posledního * velikost clusteru v boot record + velikost posledního clusteru. Velikost posledního clusteru je počet znáků do '\0' včetně.

6. Setřesení volného místa

V tomto režimu se setřese volné místo v pseudoFAT. Režim má dvě možnosti. Prázdné clustery se hledají od začátku struktury. Poté se zaplňují plnými buď od konce, nebo následujícími za volným místem. Druhá varianta je podstatně pomalejší, jelikož se musí provést více přesunů.

Funkce *shrink()* zařídí načtení pseudoFAT, spuštění danného počtu vláken a následné uložení nové pseudoFAT. Funkce *make_shrink()* a *make_shrink_ordered()* jsou volány při spuštění vlákna.

Nejprve se nalezne podle FAT tabulky volný cluster, poté dle zvolené strategie neprázdný. Clustery se prohodí a aktualizuje se tabulka FAT. Na závěr se musí zkontrolovat zda na právě přesunutý cluster není odkaz v root directory a případně root directory aktualizovat.

V programu je několik kritických sekcí. Aby se vlákna navzájem neblokovala, zvolil jsem zamykání nad polem mutexů, kde každý mutex odpovídá jednomu clusteru. Pokud je nalezen vhodný cluster, vlákno se pokusí zamknout mutex se stejným indexem jako má cluster. Pokud se to nepovede, pokračuje vyhledáváním dalšího vhodného clusteru.

Další mutex je vytvořen pro root directory. Pokud je v root directory odkaz na právě přesunutý cluster, zamkne se a root directory se obnoví.

7. Závěr

Největší nevýhodou programu je, že se pseudoFAT načítá do paměti celá. Při použití na větším objemu dat by bylo vhodné načítat clustery do paměti po částech. Také vyhledávání clusterů při setřesení volného místa není optimální. V jedné variantě jsou data hodně nekonzistentní, druhá zase trvá příliš dlouho. K ošetření kritických sekcí by se pak dalo přistoupit jinak, aby se v paměti nedrželo pole mutexů.