Distribuované kódovanie MP3

BAKALÁRSKA PRÁCA

David Zachar

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Informatika 9.2.1

Vedúci záverečnej práce
Doc. RNDr. Rastislav Kráľovič, PhD.

BRATISLAVA 2008

Čestne prehlasujem, že som túto bakalársku prácu
vypracoval samostatne s použitím
citovaných zdrojov

.....

Poďakovanie

Chcem sa poďakovať vedúcemu mojej bakalárskej práce Doc. RNDr. Rastislavovi Kráľovičovi, PhD. za inšpiratívne rady, vďaka ktorým som sa v mojej práci vždy mohol posunúť ďalej.

Abstrakt

Vzhľadom na veľkú popularitu mp3 formátu a zložitosť kódovania zvuku do tohto formátu má zmysel zaoberať sa zrýchlením tohto procesu. Cieľom tejto práce je vytvoriť encoder do mp3 formátu ktorý by využíval distribuované programovanie, t.j. rozdelenie práce na viac výpočtových jednotiek.

Program je implementáciou princípov distribuovaného programovania použitých pri podobnom projekte Parallel BladeEnc na BladeEnc novšej verzie. Projekt Parallel BladeEnc v najnovšej dostupnej verzii 0.92.1b5 je založený na BladeEncu 0.92.1. Táto verzia obsahovala chyby, ktoré boli opravené v novších verziách BladeEncu, avšak vývoj Parallel BladeEncu neprebiehal ďalej a chyby z BladeEncu v ňom zostali. Môj projekt teda implementuje spôsob akým Parallel BladeEnc distribuje prácu na BladeEncu verzie 0.94.1. Keďže BladeEnc prešiel netriviálnymi zmenami bolo treba tieto zmeny zohľadniť.

Obsah

Úvod

Táto práca sa snaží spojiť dva fenomény dnešnej doby. Prvý je momentálne najpopulárnejší formát MP3. Aj keď kvalitou bol možno prekonaný inými formátmi rozhodne však nie je prekonaný čo sa týka popularity. K tomu prispieva aj to, že alternatívnych formátov je veľa a žiaden si nevie získať jednoznačné vedenie. Taktiež k tomu napomáha aj to, že do rôznych prehrávačov sa podpora ďalších formátov dostáva len pomaly.

Druhým fenoménom je zvyšovanie výkonu, pridávaním jadier do procesorov. Programy ktoré sú naprogramované sériovo však dokážu využívať len jedno jadro. A tak mnohokrát zvýšenie výkonu nevidíme hneď. Najskôr sa prejaví v tom, že viacero aplikácií beží súčasne a nespomalujú sa. Neskôr, ak programátori vytvoria programy, ktoré využívaju viac výpočtových jednotiek (napr. jadier procesora) uvidíme výkon aj v zrýchlení samotnej jednej aplikácie.

Mojou prácou som chcel docieliť práve zrýchlenie procesu kódovania zvuku z formátu WAV na formát MP3. Zrýchlenie som docielil rozdelením práce na viac výpočtových jednotiek (počítačov, procesorov a procesorových jadier). Môj program môže pracovať na počítači, ktorý má niekoľko jadier, procesrov, ale aj na niekoľkých počítačoch zapojených v sieti.

Tento dokument je rozdelený na 3 kapitoly.

V prvej kapitole sa venujem ukladaniu zvuku v počítači a to najmä formátom WAV a MP3, ako aj spôsobom konverzie z WAV do MP3.

V druhej kapitole sa venujem distribuovanému programovaniu. Vysvetľujem, čo to je a aký je rozdiel medzi paralelným a distribuovaným programovaním. Stručne rozoberám aj štandard MPI (Message Passing Interface), ktorý som využíval v svojej práci.

V tretej kapitole sa venujem samotnému programu. Rozoberám tu spôsob fungovania. Ako som implementoval tento spôsob. Nachádzajú sa tu aj štatistiky, ktoré ukazujú reálne zrýchlenie procesu.

Súčasťou tejto práce je aj CD so samotným programom.

Digitalizácia zvuku

„Zvuk je teda časť spektra mechanického vlnenia vzduchu, ktorú je schopný vnímať človek, v širšom ponímaní, ktorú je schopný vnímať živočích.“ – Wikipedia

Aby sme toto vlnenie mohli uložiť do počítača potrebujeme ho odmerať a zdigitalizovať. Potom sa väčšinou používajú rôzne druhy kompresie týchto dát. Najjednoduchší, spôsob ktorým sa ukladá zvuk v počítači je nekomprimovaný formát WAV. Digitalizácia zvuku do formátu WAV vyzerá nasledovne. Niekoľkokrát za sekundu sa zmeria veľkosť amplitúdy a tá sa uloží. Dôležité faktory sú koľko krát sa za sekundu zmeria veľkosť amplitúdy a do akej veľkej premennej sa uloží. Frekvencia je väčšinou 44 100Hz t.j. 44100 krát za sekundu je uložená veľkosť amplitúdy. A ukladá sa do 8, 16, alebo 24 bitov. Stavba WAV súboru je potom nasledovná. Na začiatku súboru je hlavička obsahujúca podstatné informácie ako je frekvencia, do koľkých bitov sa ukladá veľkosť amplitúdy, ale aj to, či je to mono, stereo zvuk a dĺžka súboru. Za hlavičkou sa nachádzajú už len dáta. Výhodou tohto formátu je jeho jednoduchosť pri nahrávaní, ale aj pri prehrávaní skladby. Pri nahrávaní stačí len v každom časovom intervale odmerať a uložiť. Pri prehrávaní si potrebujem prečítať hlavičku a potom keď sa potrebujeme presunúť do istej časti stopy, napríklad podľa sekúnd je veľmi jednoduché zrátať, koľko bytov treba preskočiť.

Iné je to pri formáte MP3. Formát MP3 je stratový kompresný formát, ktorého veľkou výhodou je práve veľmi malá veľkosť výsledného súboru. Túto kompresiu sa snaží dosiahnúť odstránením nadbytočnej (angl. redundant) informácie na základe psychoakustického modelu. Psychoakustika je štúdia subjektívneho vnímania zvuku človekom. Je to tiež štúdia psychologickej súvislosti s fyzikálnymi vlastnosťami zvuku. Za počuteľné pre človeka sa považujú frekvencie 20Hz – 20 000Hz. Avšak táto horná hranica sa časom stráca. Veľa dospelých ľudí nedokáže počuť zvuk nad 16kHz. Preto aj väčšina MP3 úplne odreže frekvencie nad 18 kHz a MP3 s dátovým tokom (angl. bitrate) 128kbit/s odreže frekvencie nad 16kHz. Tiež počuteľnosť spodnej frekvencie môže závisieť od hlasitosti. Ďalej ľudské ucho vníma aj rozlične zvuky v závislosti od kontextu ostatných zvukov. Rozdielne vnímame zvuk tlesknutia v prázdnej tichej miestnosti a rozdielne keď zaznie hneď po náraze dvoch aut. Tak isto zvuky ktoré zdieľajú frekvenčné pásmo, napríklad dva sínusové tóny s frekvenciou 440 Hz a 450Hz sú vnímateľne separátne, avšak nedajú sa vnímať súčasne. Všetky tieto „nedostatky“ vnímania ľudského ucha sa snaží využiť MP3, ale aj iné kompresie. Dokonca aj rôzne enkódery do MP3 formátu používajú rôzne sofistikované metódy ako dosiahnuť čo najlepšiu kompresiu pri zachovaní kvality zvuku.

Štruktúra MP3 súboru je nasledovná. MP3 súbor by sa dal rozdeliť na rámce (angl. frames) . Rámec sa skladá z hlavičky (angl. header) a samotných dát. V hlavičke sa nachádzajú

informáciu o dátach v danom rámci. Jedným z údajov je dátový tok (angl. bitrate). Tento určuje pomer bitov na sekundu skladby. Každý frame môže mať tento dátový tok rozdielny aj keď tvoria jeden MP3 súbor. V takom prípade hovoríme o premennom dátovom toku (angl. variable bitrate = VBR). Naopak ak majú všetky rámce v MP3 súbore rovnaký dátový tok hovoríme o konštantnom dátovom toku (angl. constant bitrate = CBR). V takomto prípade sa stačí pozrieť na dátový tok prvého rámca a ten si zapamätať pri prehrávaní. VBR však poskytuje enkóderu väčšie možnosti, napríklad isté časti skladby dokáže komprimovať viac, pretože je tam menej zvuku vnímateľného ľudským ušiam. Zatiaľ čo iné časti skomprimuje s väčším dátovým tokom. Výsledná dobre spracovaná VBR MP3 môže mať lepšiu kvalitu ako väčšia CBR MP3.

Toto využívanie VBR ako aj spôsoby komprimácie zvuku na základe vnímania zvuku ľudským uchom sa líšia od enkódera k enkóderu. Na základe toho sú niektoré kvalitnejšie ako iné a niektoré svoje algoritmy dokonca chránia. Pre všetky kvalitné MP3 enkódery však väčšinou platí, že sú pomerne výpočtovo náročné. Tieto algoritmy môžu byť často veľmi zložité. Vidieť to aj na základe zaťaženia procesora pri komprimácii do MP3 formátu.

Takže oproti formátu WAV získavame súbor, ktorý je omnoho menší, jeho tvorba je však pre procesor zložitejšia. Napriek zrýchľovaniu procesorov počítačov má zmysel zaoberať sa zrýchlovaním tohto procesu. A aj cieľom tejto práce je tento proces zrýchliť, ak máme viac výpočtových jednotiek ktoré nám môžu pomôcť.

Distribuované programovanie

Ako som spomenul v úvode zažívame fenomén zvyšovania výkonu počítačov zväčšovaním počtu jadier. Preto sa dnes stretávame s pojmami ako paralelné a distribuované programovanie. Aj autor Parallel BladeEncu, ktorého myšlienky som implementoval, nazval tento program paralelným. To vnáša trochu nepresností do týchto pojmov. Preto v tejto časti vysvetlím, alebo definujem, ako tieto pojmy používam ja a prečo nepoužívam pojem paralelizmus ale distribuovanosť.

Definície

Distribuované programovanie - programovanie programov, ktoré dokážu niektoré časti výpočtu (najlepšie tie výpočtovo najťažšie) distribuovať medzi ostatné výpočtové jednotky. Do problémov, ktoré treba riešiť patrí aj to, ako budú tieto jednotky koordinovať svoju prácu, aby pracovali optimálne. Berie sa do úvahy aj komunikácia medzi jednotlivými výpočtovými jednotkami, ako aj to, že nemusia mať rovnakú výpočtovú silu.

Paralelné programovanie- programovanie programov, ktoré dokážu niektoré časti výpočtu (najlepšie tie výpočtovo najťažšie) počítať paralelne na viacerých výpočtových jednotkách. Zväčša sa predpokladá, že s komunikáciou nie sú problémy a tak sa neberie do úvahy a taktiež sa niekedy predpokladá, že výpočtové jednotky sú rovnaké.

Rozdiel medzi distribuovaným a paralelným programovaním

Z definícií je vidieť niekoľko rozdielov. Ale najlepšie bude ukázať si príklad. Príkladom paralelného programu môže byť program ktorý je písaný tak, aby bežal v istom bode paralelne na viacerých vláknach (procesoch). Toto je typická predstava paralelného programu, podľa definícií spomenutej vyššie. Paralelný program teda beží na viacerých vláknach. Je jasné, že beží buď simulovane paralelne, alebo aj reálne ak máme viac procesorový systém, alebo viacjadrový procesor, alebo súčasne. Program však beží na tom istom počítači. Môžme predpokladať, že obe vlákna (procesy) majú približne rovnakú výpočtovú silu. Taktiež majú tieto dve vlákna väčšinou zdieľanú pamäť. To je dôvod prečo pri paralelnom programovaní neriešime komunikáciu. Vlákno zapíše niečo do pamäte a druhé vlákno to prečíta. Poprípade obe môžu narábať s tými

istými dátami. Avšak treba riešiť prístup.

Naproti tomu distribuované programy, ako napríklad Distribuovaný BladeEnc, môžu fungovať na jednom, alebo viacerých počítačoch v sieti. Každá výpočtová jednotka je samostatná. Každá má svoju pamäť. Neexistuje niečo ako zdieľaná pamäť. Výpočtová jednotka vie o ostatných len toľko, koľko sa dozvie zo správ, ktoré dostane. Keďže výpočtové jednotky nemusia byť na rovnakom stroji, treba brať ohľad na niektoré veci ktoré paralelné programovanie do úvahy neberie. Napríklad:

- výpočtová sila jednotlivých výpočtových jednotiek, môže byť odlišná
- posielanie správ nemusí byť lacné
- architektúra výpočtových jednotiek môže byť odlišná
- správy sa môžu po ceste stratiť
- výpočtové jednotky môžu používať iné kódovanie

Distribuované programovanie, teda predstavuje viac práce pre programátora. Avšak použitľnosť je väčšia. Zatiaľ, čo paralelné programy dosahujú väčší výkon len na počítačoch s viac procesormi, alebo jadrami, distribuované programy môžu dosiahnúť zvýšenie výkonu aj na niekoľkých slabších strojoch zapojených v sieti.

Message Passing Interface (MPI)

Je vidieť, že podstata distribuovaného programovania je založená na správach. Preto dôležitá opora programu Distribuovaného BladeEncu je práve Message Passing Interface (MPI) (rozhranie na výmenu správ).

MPI je štandard na posielanie správ. Je to komunikačný protokol nezávislí na programovacom jazyku. Existuje niekoľko implementácií MPI:

- OpenMPI
- LAM/MPI
- FT-MPI
- LA-MPI
- MPICH
- MPICH2

V súčasnosti sú 2 oblúbené verzie tohto štandardu:

- verzia 1.2 (skrátene MPI-1)
- verzia 2.1 (skrátene MPI-2)

Implementácie MPI, štadnardu sa líšia väčšinou len v tom ktorú verziu podporuju. Avšak MPI-2 je zväčša len rozšírenie MPI-1 o ďalšiu funkcionalitu, takže aj programy písane pre MPI-1 by mali fungovať na implementácii MPI-2. To je príklad Distribuovaného BladeEncu. Ja osobne som ho spúšťal na OpenMPI, ktorý podporuje štandard MPI-2, ale Distribuovaný BladeEnc používa funkcionalitu MPI-1. Keďže implementácie MPI implementujú ten istý štandard MPI,

Distribuovaný BladeEnc by mal fungovať aj na inej implementácií, ktorá implementuje MPI-1, s prihliadnutím na rozdiely v implemnáciach, ktorých zväčša nebýva veľa a bývajú dobre zdokumentované.

OpenMPI som si vybral, pretože je otvorený (angl. open) a bol pomerne jednoduchý na inštaláciu a konfiguráciu na jednom počítači oproti ostatným.

MPI z pohľadu užívateľa

Užívateľ potrebuje nainštalovať na počítač, respektíve počítače niektorú z implementácií MPI. Potom nakonfigurovať MPI a to hlavne v prípade, ak chce použiť viac počítačov. MPI dokáže spúšťať na ostatných strojoch svoje ďalšie procesy napríklad cez ssh, alebo rsh. Prístupy na tieto stroje je potom nutné uviesť do konfiguračného skriptu ako aj to koľko procesorov, resp. jadier procesorov na ktorom stroji je. Program využívajúci MPI je potom treba skompilovať s podporou MPI. Pre Distribuovaný BladeEnc to väčšinou znamená spustiť štandardné:

```
./configure
make
```

V správnom výpise z príkazu configure, ktorý rozpoznal MPI implementáciu, by mal byť tento riadok:

```
checking for MPI... yes
```

Ak to tak nie je treba skontrolovať, či je správne nainštalovaná niektorá implementácia MPI. Malo by sa to prejaviť existenciou príkazov mpicc, mpirun.

V prípade, že všetko prebehlo bez problémov užívateľ môže namiesto bežného zadania programu príkazom

```
program [argumenty]
```

zadať príkaz

```
mpirun -np 4 program [argumenty]
```

Tento príkaz spustí 4 procesy programu program. Argument -np určuje koľko procesov sa má spustiť. Ako a kde tieto procesy MPI spustí, záleží od implementácie. Väčšinou však implementácie prechádzajú zoznam strojov z konfiguračného súboru a tam sa jeden po druhom spúšťajú. Navyše niektoré implementácie podporujú spustenie toľko procesov koľko je počítačov, alebo toľko koľko je výpočtových jednotiek. Napríklad ak máme dva obyčajné PC jedným jedno-jadrovým procesorom a jeden 4-jadrový, tak sa program spustí 6-krát a to po jednom na obyčajných a 4-krát na štvorjadrovom. Toto však treba nastaviť do konfiguračného súboru, teda pri 4 jadrovom počítači treba uviesť, že sa tam dajú pustiť 4 procesy. Túto funkcionalitu veľa implementácií podporuje avšak líšia sa v tom ako ju nakonfigurovať.

Takže aby sme Distribuovaný BladeEnc spustili na 4 procesoch a aby skonvertoval súbor hudba.wav do hudba.mp3 treba spustiť:

```
mpirun -np 4 ./bladeenc hudba.wav hudba.mp3
```

MPI z pohľadu programátora

Popis niektorých funkcií MPI štandardu, ktoré využíva Distribuovaný BladeEnc.

*int MPI_Init(int *argc, char ***argv)*
Inicializuje prostredie MPI. Táto funkcia by mala byť zavolaná pred akoukoľvek inou MPI funkciou.

*int MPI_Comm_size (MPI_Comm comm, int *size)*
Zistí veľkosť skupiny asociovanej komunikátorom comm. Komunikátor je skupina procesov. Každý proces má v tomto komunikátore svoje jednoznačné číslo. Väčšinou sa používa jeden globálny komunikátor do ktorého patria všetky procesy. Tak je to aj v Distribuovanom BladeEncu. Veľkosť vráti v premennej size.

*int MPI_Comm_rank (MPI_Comm comm, int *rank)*
Zistí číslo (rank) pridelené súčasnému procesu v komunikátore comm. Vráti v premennej rank.

*int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)*Blokované posielanie správ. Pošle správu procesu číslo (rank) dest v komunikátore comm.
buf - buffer, smerník na pamäť s údajmi, ktoré chceme poslať
count - koľko premenných chceme poslať
datatype - akého typu sú jednotlivé premenné
tag - typ správy

*int MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag,*
*MPI_Comm comm, MPI_Status *status)*
Blokované prijímanie správ. Príjme správu od source na komunikátore comm.
buf - buffer, smerník na pamäť s údajmi ktoré chceme poslať
count - koľko premenných chceme poslať
datatype - akého typu sú jednotlivé premenné
tag - typ správy
status - stav

int MPI_Finalize()
Ukončí prostredie MPI. Táto funkcia by sa mala zavolať po skončení práce s MPI funkciami.

Tieto funkcie obsahuje takmer každá aplikácia využávajúca MPI, založená na blokovanom posielaní a prijímaní správ.

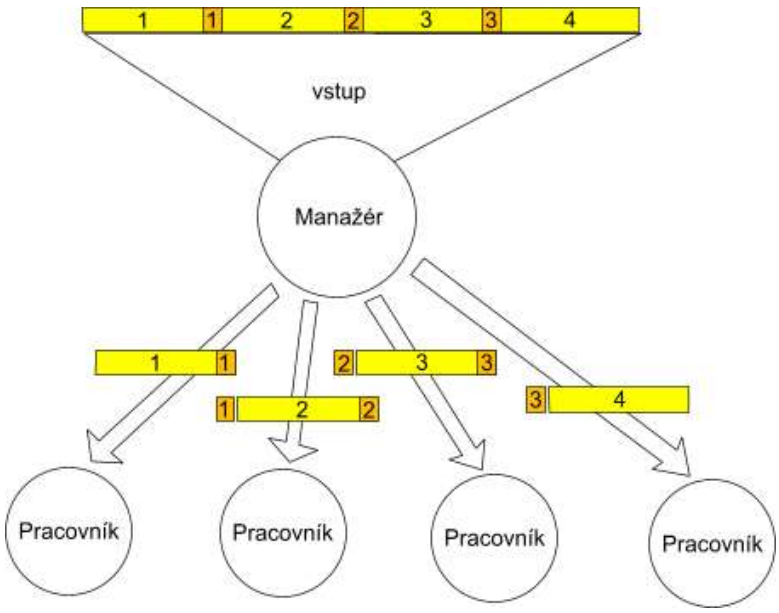
Okrem týchto funkcií sú v programe použité aj funkcie na tvorbu špeciálnych premenných MPI_Datatype, ktoré popisujú typ objektu, ktorý sa posiela. Pre bežné premenné, ktoré chceme posielať sa v MPI vyskytujú preddefinované typy, napríklad pre int je to MPI_INT, pre char MPI_CHAR a pod. Avšak ak chceme posielať premennú zložitejšieho typu napríklad štruktúru potrebujeme ju najskôr popísať a vytvoriť pre ňu špeciálnu premennú typu MPI_Datatype. Našťastie v MPI štandarde sú pre toto vytvorené funkcie, ktoré sa väčšinou začínajú prefixom MPI_Type. V programe Distribuovaný BladeEnc sú použíté tieto funkcie:
MPI_Type_vector
MPI_Type_struct
MPI_Type_commit
MPI_Type_free

Distribuovaný BladeEnc

Princíp fungovania

Ak Distribuovaný BladeEnc spustíme na n+1 procesoch, môžeme tieto procesy rozdeliť na 2 kategórie:

- Manažér (alebo tiež Pán z anglického Master)
- Pracovník (alebo tiež Otrok z anglického Slave)

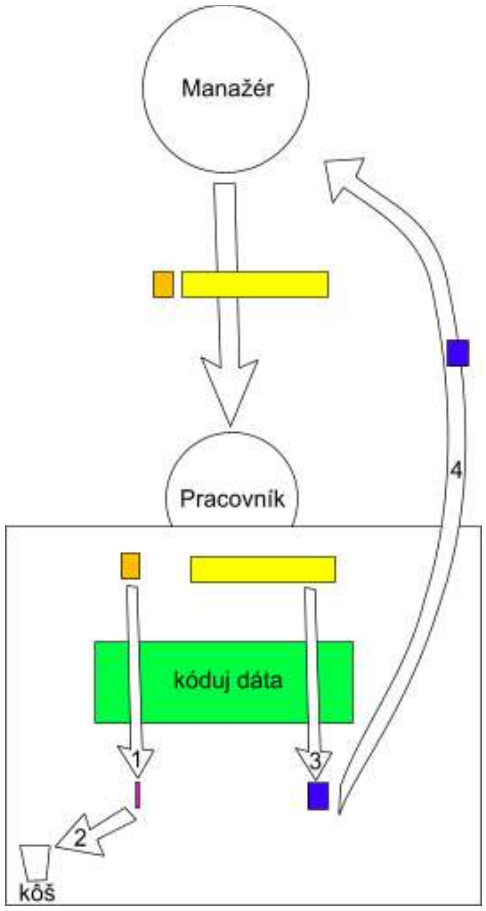


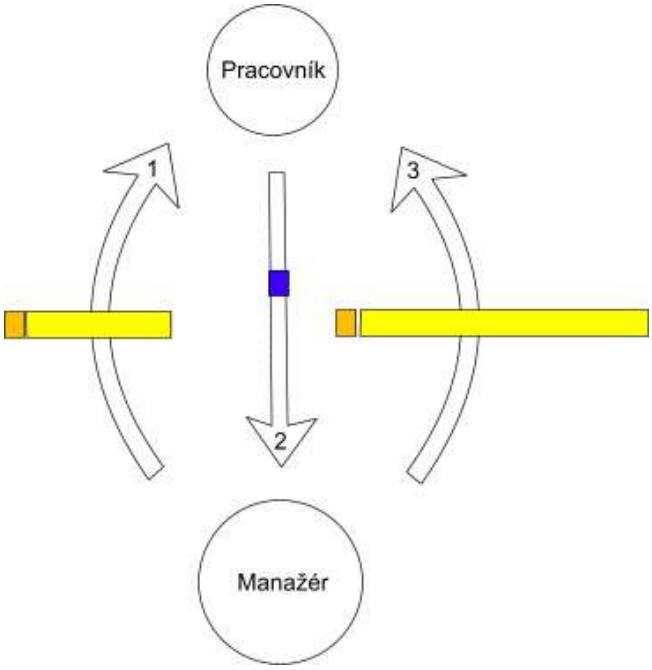
Jeden proces pracuje ako Manažér a ostatných n procesov ako Pracovníci. Manažér sa nachádza na stroji kde bol spustený celý Distribuovaný BladeEnc systém aby mal prístup k súborom. Manažér teraz časť vstupu rozdelí medzi pracovníkov a pošle kažfému na spracovanie. Spolu s dátami okrem prvého pošle každému aj časť z konca dát predchádzajúceho (na obrázku oranžové dáta na začiatku). Tieto nadbytočné (angl. redundant) dáta ktoré sa prekrývajú bude manažér posielať s každými dátami ktoré, od teraz pošle. Potom bude Manažér už len čakať na odpoveď niektorého Pracovníka.

Čo robia pracovníci?

Pracovník funguje stále rovnako. Najskôr čaká na správu. Môže prísť správa aby skončil, alebo dáta. Okrem prvého pracovníka prídu vždy najprv nadbytočné dáta z prekryvu. Tieto pracovník spracuje encoderom a zahodí. Prečo? Encoding prebieha tak, že encoder si svoje hodnoty mení podľa kódovania. Avšak my nechceme aby Encoder pracoval sériovo. Namiesto aby prechádzal každý pracovník celú predchádzajúcu časť súboru, alebo aby čakal na druhého pracovníka aby mu tieto informácie poslal spraví to tak, že spracuje časť informácie pred danou časťou a tak sa pokúsi nastaviť keď nie do totožnej aspoň do približnej podoby v akej by bol pri sériovom spracovaní. Takže preto zakóduje tieto nadbytočné dáta a zakódované dáta zahodí (šípky 1 a 2) a potom prejde pracovník k danej práci ktorú ma vykonať.

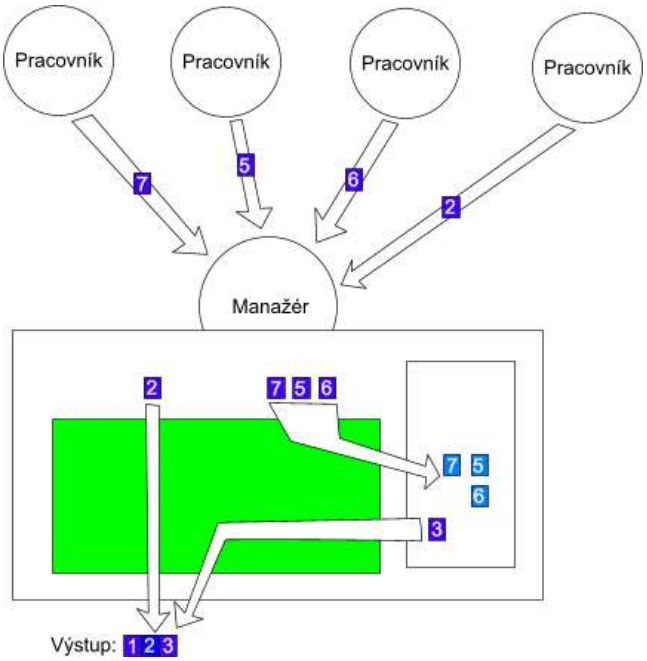
Zakóduje tú časť dát, ktorú potrebuje a zakódované dáta pošle späť Manažérovi (šípky 3 a 4).





Manažér po odoslaní n prvých častí pracuje v cykle nasledovne. Kým nepríde na koniec súboru, čaká na správu od pracovníka. Ako spravuje správu vysvetlím neskôr, avšak pracovníkovi pošle znova ďalšiu časť súboru. Znova mu pošle aj nadbytočné prekrývajúce sa dáta a za nimi dáta na zakódovanie. Rozdiel je však v tom, že týchto dát na zakódovanie pošle 2x viac. Samozrejme až kým sa nedostane po vopred určenú konštantu. Toto zväčšovanie dát pri posielaní je dôležité pri rôznorodom prostredí. Ak by rýchlosť pracovníkov nebola rovnaká, tak pracovník ktorý bude prvý hotový (potencionálne rýchlejší ako ostatní) dostane 2x viac práce. Tak isto sa aj postupne zmenšuje počet nadbytočných dát ako aj rozdelení celého súboru.

Spracávanie dát Manažérom do výstupu.



Manažérovi nemusia časti súboru prichádzať postupne. A je dosť pravdepodobné, že nebudú pri rôznorodom prostredí. Jeden z pracovníkov môže svoju časť spracúvávať veľmi dlho, pretože je pomalší ako ostatní a tak zatiaľ, čo ostaní už odovzdali svoju prácu možno aj viac než raz, tento pracovník ešte stále pracuje. Preto nemôže Manažér zapisovať do výstupu dáta tak ako prichádzajú od Pracovníkov. Robí to nasledovne pri každom dieli si pamätá jeho poradie. Keď príde správa od Pracovníka zavolá funkciu, ktorá sa pozrie, či tento diel patrí za už zapísaný do súboru, ak nie uloží ho na spracovanie neskôr (na obrázku časti očíslované 5,6,7). Ak dáta, ktoré prídu patria za už zapísané (na obrázku časť s číslom 2) tak ich zapíše a pozrie sa, či časti v dočasnej pamäti nepatrie za tieto práve zapísane, ak áno zapíše ich (na obrázku časť s číslom 3).

Implementácia a jej problémy

Hore uvedním spôsobom fungoval aj projekt pod názvom Parallel BladeEnc. Tento bol založený na BladeEncu verzie 0.92.1. Moje prvé spustenie tohto programu skončilo s nečakanou chybou počas komprimácie a program spadol. Keď som na danom súbore skúsil spustiť obyčajný BladeEnc verzie 0.92.1 skončil rovnako chybou. Keďže autor projektu Parallel BladeEnc na svojej stránke nenaznačoval nič také, že jeho program je len kus zbytočného kódu ktorý zatiaľ nezbehol, predpokladal som, že niekedy ho už spustil a fungoval mu. BladeEnc novšej verzie (0.94.1) fungoval bez problémov. Tak som usúdil, že chyba ktorou trpel projekt Parallel BladeEnc je chybou spôsobenou v BladeEncu 0.92.1 a, že princíp distribúcie kódovania na viacej výpočtových jednotiek je naprogramovaný dobre.

Moja práca teraz spočívala v implementácií tohto rozdistribuovania a menežovania práce na viacerých výpočtových jednotkách, ktorá bola použitá v Parallel BladeEncu 0.92.1b5 na BladeEnc 0.94.1. Aby som dosiahol tento cieľ potreboval som zistiť ako z BladeEncu 0.92.1 bol vyrobený Parallel BladeEnc 0.92.1b5. Taktiež aké sú zmeny BledeEncov verzií 0.92.1 a 0.94.1 a tieto zmeny potom aplikovať do distribúcie enkódingu.

Jednou zo zmien v BladeEncu, ktoré som potreboval zohľadniť bolo, že za niektorých podmienok sa načítalo 2x viac dát a tak dáta, ktoré sa potom posielali na spracovanie sa násobili za istých podmienok dvomi. Túto zmenu som riešil v Manažérovi. Ten za tých istých podmienok

poslal 2x viac dát na spracovanie. Rozdiel medzi sekvenčným a distribuovaným prístupom bol ten, že v sekvenčnom program používal pamäť konštanej veľkosti, ktorá bola dosť veľká aby sa tam vždy dáta zmestili, ale distribuovaný program sa snažil mať len toľko, koľko potrebuje. Preto som potreboval, ešte aj toto dvojnásobne zväčšanie zohľadniť pri alokácií pamäte pre dva smerníky.

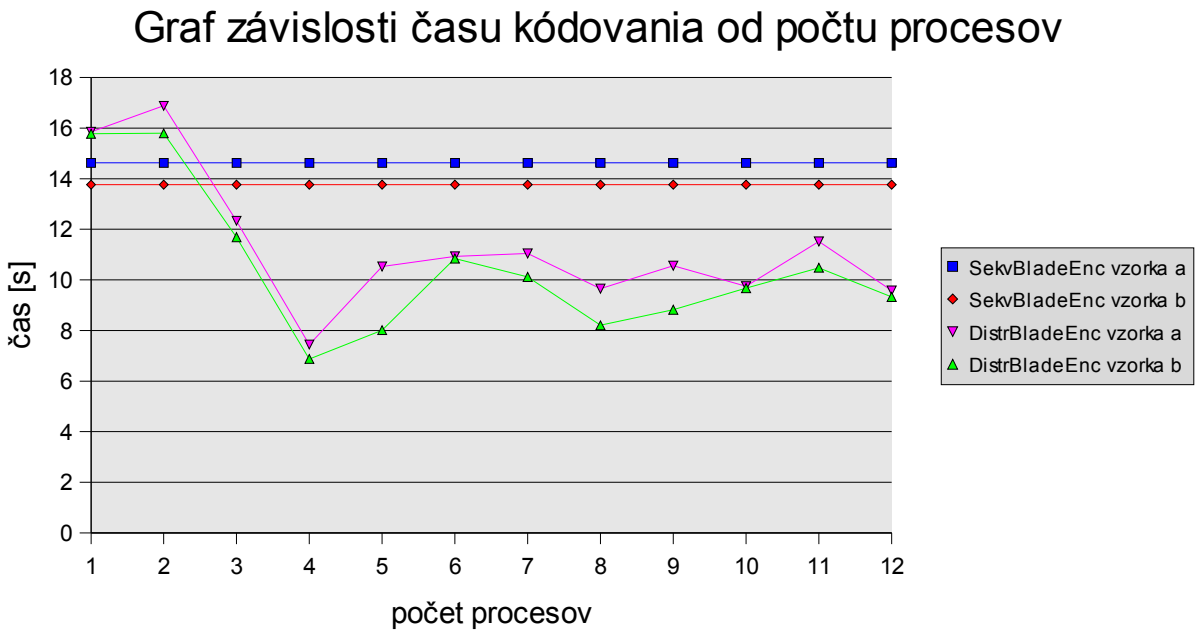
Ďaľšou dôležitou zmenou bola zmena niektorých štrukturálnych premenných. Napríklad typ SplIn sa zmenil na SI_Stream a trochu zmenil aj vnútornú štruktúru, alebo aj typ Job ktorý tiež zmenil vnútornú štruktúru. Problém v zmene týchto štruktúr bol v tom, že premenné týchto typov sa posielajú medzi jednotlivými procesmi. Ako som písal, na podrobné popísanie štruktúralnych typov je v MPI štandarde aparát funkcií začínajúcich prefixom MPI_Type, ktoré podobný popis týchto štadnardov zapíšu do premennej typu MPI_Datatype. Tieto funkcie bolo treba prispôsobiť novým štrukturálnym typom.

Štatistiky

Nakoľko sa táto paralelizácia vyplatí? Spravil som niekoľko testov na 4 jadrovom počítači, ktoré ukazujú nakoľko sa tento proces komprimácie dá zrýchliť.

Presná konfigurácia:
Procesor: AMD Phenom 9600 Quad-Core
Frekvencia procesora: 2300MHz
RAM: 4GB

Merania boli robené za pomoci programu time a čas bol meraný v sekundách. Výsledky meraní zobrazuje nasledujúci graf. Výška údaju v grafe je čas v sekundách. Vzdialenosť údaju v riadku predstavuje počet procesov. Merania boli uskutočnené na dvoch vzorkách vstupných súborov označných ako vzorka a a vzorka b. Pre porovnanie boli urobené merania aj na sekvenčnom t.j. klasickom BladeEncu verzie 0.94.1 označenom SekvBladeEnc. Napriek tomu, že tento program sa nedá spustiť na viacerých procesoch ako jeden, uviedol som tento údaj aj pre viac ako jeden proces pre lepšie vizuálne porovnanie. Pri viac procesoch je pre sekvenčný BladeEnc údaj skopírovaný z údaju pre jeden proces. Distribuovaný BladeEnc, ktorý je výsledkom tejto práce je v grafe uvedený ako DistrBladeEnc.



Aké závery môžme z grafu usúdiť?

Vidíme, že minimum dosahuje pri 4 procesoch. Dalo sa to predpokladať, pretože máme štvorjadrový procesor a tak na každom jadre pracuje jeden proces. Ak dáme procesov menej, tak niektoré jadro, alebo jadrá nepracujú vôbec. Keď pustíme procesov viac, tak strácame zbytočne výkon na menežment procesov. Avšak vzhľadom na to, ako pracuje Distribuovaný BladeEnc, keď jeden proces je iba menežér a rozdeľuje prácu ostatným a pracujú len tí ostatní, dalo by sa predpokladať, že menežér nepotrebuje toľko procesorového výkonu a tým by sme minimum dosiahli na 5 procesoch. Pretože každý pracovník by išiel na jednom jadre a menežér by raz za čas niekomu ukradol pár cyklov. Vidíme však, že to tak nefunguje. Vzhľadom na ďalšie pozorovanie predpokladám, že aj menšia nevýváženosť rýchlosti pracovníkov môže viesť k celkovému spomaleniu. Takže idálne je, ak aj menežér ma svoje jadro, ktoré nie je obsadené niekým ďalším.

Ďalšou zaujímavosťou sú body, v ktorých nastáva lokálne minimum. Odhliadnúc od vzorky a pri 10 procesoch je to ak je počet procesov deliteľný 4. Môj predpoklad bol, že po 4 procesoch bude každé zvýšnie počtu procesov viesť k miernemu zvýšeniu času. Toto by malo byť spôsobené nárastom potreby menežmentu procesov. Prekvapilo ma preto, keď čas poklesol pri 8 a 12 procesoch a to na obidvoch vzorkách. To vedie k zaveru, že ideálne je, ak sú výpočtove jednotky rovnako výpočtovo silné a rovnako zaťažené. Aj keď spôsob distribúcie a celkového princípu fungovania sa snaží fungovať optimálne aj keď výpočtové jednotky nie sú rovnako rýchle, ako vidíme to neznamená, že je lepšie ak sú rovnaké a rovnako vyvážené.

Prečo 2 procesy sú rovnako rýchle a možno aj pomalšie ako jeden? Mali by byť rovnako rýchle. Princíp Distribuovaného BladeEncu je taký, že z n+1 procesov je jeden menžer a n pracovníkov. Pre n+1 = 2 je to 1 menežér a 1 pracovník. Keďže je zbytočné aby jeden menežér

rozdeľoval prácu jednému pracovníkovi a menežoval ho, Distribuovaný BladeEnc pracuje tak ako by bol sputstený na jednom procese a druhý proces je nefunkčný. Takže pracuje sekvenčne.

Záver

Najdôležitejšia je otázka použiteľnosti. Je vidieť, že distribuovaným prístupom je možné dosiahnuť, aj keď nie ideálne, tak aspoň celkom pekné zrýchlenie. Optimálnosť strácame napríklad tým, že je potrenbé dosť výpočtovej sily na menežovanie práce. Ďalej kôli nastaveniu enkódera počítame niektoré časti 2 krát. Pochybná je tiež kvalita enkódera. Takže človek, ktorý potrebuje len občas konvertovať do formátu MP3 kvalitne, tak si radšej počká pár sekúnd. Avšak projekt nie je úplne na zahodenie. Pretože pri dnešnom trende počítačov, keď sú dostupné servery s 4, 8, 16, ale aj 32 jadrami a je možné, že rýchlosť prekonvertovania je dôležitejšia ako kvalita výsledku ak je počúvateľný je veľmi použiteľné toto riešenie.

Táto práca ukázala, že zrýchlenie za pomoci rozdelenia práce na viac procesov je dosiahnuteľné. Ako bude postupovať trend zvyšovania jadier na procesoroch bude potrenbé využívať tento výkon čo najlepšie. To dosiahneme paralelným, alebo distribuovaným programovaním a zlepšovaním kvality takto vytvorených programov. Aj tento program by sa dal vylepšiť možno iným prístupom, ktorý by hlavne nepotreboval konvertovať nadbytočné (angl. redundant) prekrývajúce sa časti. Otazne je či existuje prístup, ktorým by sa to dalo docieliť. Alebo, či existuje hudobný formát, pri ktorom by sa to dalo dosiahnúť.

Literatúra

1. **http://www.osl.iu.edu/~jsquyres/bladeenc** - Stránka projektu Parallel BladeEnc
2. **http://www.mpi-forum.org**- MPI Fórum, obsahuje oficiálnu dokumentáciu MPI štandardu
3. **http://www.lam-mpi.org** - Stránka LAM/MPI implementácie MPI štandardu, taktiež obsahuje dobrú dokumentáciu MPI štandardu
4. **http://www.open-mpi.org** – Stránka OpenMPI implementácie MPI štandardu
5. **http://www.wikipedia.org** – Slobodná encyklopédia, informácie napríklad z týchto podstránok:
   a) **http://sk.wikipedia.org/wiki/Zvuk**
   b) **http://en.wikipedia.org/wiki/MP3**
   c) **http://cs.wikipedia.org/wiki/MP3**
6. **http://www.bcr.org/cdp/best/digital-audio-bp.pdf** - Detailný popis digitálnych audio formátov (angl.)