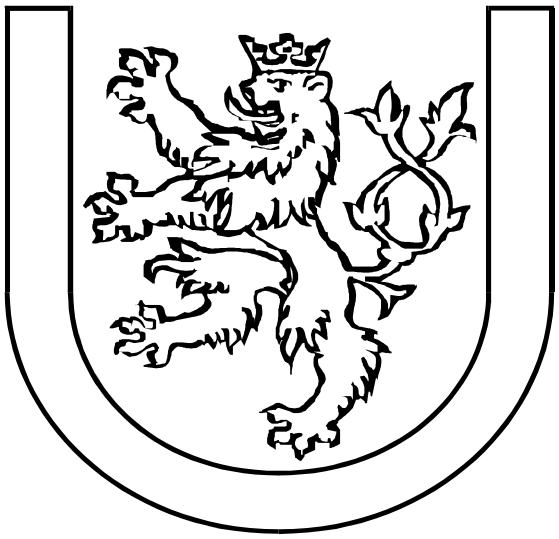
Z Á P A D O Č E S K Á
U N I V E R Z I T A



Katedra informatiky a výpočetní techniky

Paralelní programování

Výpočet k-mocniny matice

KLABUSAY Dušan FAV
leden 2003
klabusay@kiv.zcu.cz

OBSAH:

1. Zadání

Výpočet *k*-té mocniny čtvercové matice (n,n) s reálnými prvky.

2. Řešení problému

2.1 Programové řešení

Tento problém byl řešen pro dva systémy:

1) Paralelní program pro systém se sdílenou pamětí řešený pomocí programovacího jazyka JAVA.

   Použité řešení:
   Program nejprve vygeneruje *matici* se zadaným rozměrem. Poté se staticky spustí jednotlivá vlákna a každému vláknu je předána původní matice, řádka, kterou má spočítat a mocnina matice. Vlákno spočítá k-mocninu dané řádky (vlastně vektor (řádky*matice)*K) a výsledný vektor (řádka matice) předá zpět do původní matice. Po doběhu všech vláken se výsledek vytiskne na obrazovku.

2) Paralelní program pro systém s distribuovanou pamětí řešený pomocí programovacího jazyka C a systému PVM.
   Použité řešení:
   Nejprve se načte ze vstupního souboru matice do paměti, poté se pospouští procesy, počet procesů je dán rozměrem matice (např. pro matici 3x3 je počet vláken=3). V každém procesu je proveden výpočet *k*-té mocniny pro danou řádku matice a tento výsledek je vrácen farmerovi, kde je pak vyhodnocen a vypsán na obrazovku a uložen do souboru.

2.2 Řešení vstupních/výstupních dat programu

Jako vstupní data slouží data ze souboru *matice.txt*. V tomto souboru je uložena matice ve formátu:

121.5 31 33

456 654 64

12.12 25 25.65

Výstup programu je proveden jednak na obrazovku, a déle do výstupního souboru vystup.txt. Do tohoto souboru se ukládá jenom umocněná matice.

3. Popis řešení v Javě

Zdrojový program v Javě se jmenuje **Matice.java** a skládá ze dvou tříd: ***Matrix, RadekSpocti*** a ***Matice*** obsahující hlavní metodu *main()*.

Matrix

Třída implementující objekt, do kterého se ukládá matice.

Atributy objektu:

```
private double [][] matice; // Vlastni matice
private double [][] kmatice; // pomocna matice
```

```
public static int RAND_MAX = 10;   //timto intem lze menit maximalni
                        hodnoty v generovane matici
```

Konstruktor objektu:

```
Matrix(int rozmer){
      this.rozmer = rozmer;
      generateMatice(); // vygeruje matici nahodnych cisel
}
```

Metody:

```
double[] dejRadek(int sloupec)
```

- každé vlákno si sáhne pro svoje data (řádek matice)

```
double[][] prekopirujMatici(int kook)
```

- překopíruje obsah jedné matice do druhé

```
void nastavRadek(int sloupec,double[] naco)
```

-každé vlákno vrátí svá data zpět do matice

```
void generateMatice()
```

- Vygeneruje matici náhodných čísel

```
void printOutMatrix()
```

- Vytiskne matici na monitor

RadekSpocti

Třída definující vlákno, které počítá pro daný řádek matice k-mocninu.

Atributy:

```
private int       mytag;      //id vlakna
private Matrix out;          //objekt pro ukladani vysledku
private double[] mradek;     //vstupni radek
private double[] vradek;     //vystupni radek
private double[][] matice;    //matice, kterou budu mocnit
private int N;                //Rozmer marice
private double pomoc[];       //pomocny vektor pro vypocet
private int kmocnina;        //mocnina, na kterou mam mocnit matici
```

Konstruktor:

```
RadekSpocti(int   tag,   double[]   radek,int   pocetDelniku,double[][]
vmatice,int lmocnina,Matrix result)
```

- zde se načtou potřebné parametry
- Předá se řádek matice, mocnina, no kterou mám mocnit
- Rozměr matice
- kam se dávají výsledky result

Metody:

```
public void run()
```

- zde se spočítá vlastní mocnina matice

Matice

Popis hlavního programu main()

Proměnné:

```
int pocetDelniku = 3;        //pocet delniku, udava zaroven rozmer matice
int mocnina = 4;         // Mocnina matice, kterou chceme spocitat
double maticeZaklad[][];  //pomocna matice, kterou predavame vlaknum

Matice matice = new Matice(); // kam se budou ukládat prvky matice
```

```
RadekSpocti[] poleVlaken = new RadekSpocti[pocetDelniku];   //inicializace
pole vlaken
```

Popis programu:

Program nejprve vygeneruje *matici* se zadaným rozměrem. Poté se staticky spustí jednotlivá vlákna a každému vláknu je předána původní matice, řádka, kterou má spočítat a mocnina matice. Toto je implementováno metodou run() daného vlákna.

Vlákno spočítá k-mocninu dané řádky (vlastně vektor (řádky*matice)*K) a výsledný vektor (řádka matice) předá zpět do původní matice. Po doběhu všech vláken se výsledek vytiskne na obrazovku.

4. Práce s programem v Javě

Zdrojový program se jmenuje **Matice.java**. Tento soubor přeložíme příkazem **javac Matice.java** a spustíme příkazem **java dusan/Matice**(Musíme mít zdrojové texty v adresáři dusan)**.** Pokud chceme změnit rozměr matice (a tím i počet vláken), tak musíme upravit zdrojový kód a přeložit znovu.

5. Popis řešení v C pro PVM

Program v jazyce C se skládá ze dvou zdrojových programů: **farmer.c** a **worker.c** a pomocného modulu **help.c**.

farmer.c - Implementuje hlavní program, který vytváří jednotlivé tasky worker. Po zkontrolování správných vstupních argumentů se načte celá matice do dynamické struktury a farmer spustí daný počet workerů se vstupními argumenty: Název souboru s daty, mocnina matice a řádek, který má worker zpracovat.

Po spuštění tasků, čeká na příjem zpráv od prvního workera., který zpracovával první řádku matice. Výsledek je reprezentován polem doublu, počet prvků pole je dán rozměrem matice. Toto pole je vypsáno na obrazovku a pak do souboru. Farmer dále čeká na zprávu od dalšího workera v pořadí. Takto pokračuje až do skončení všech workeru, pak farmer skončí.

worker.c – Program implementuje jednotlivý task programu. Nadřízený task (tedy farmer) spouští tento task. Worker nejprve zkontroluje vstupní argumenty, poté si načte matici ze souboru, vypočítá její *k*-tou mocninu a výsledek (pole doublu) pošle farmerovy a skončí.

help.c – V tomto modulu jsou některé funkce, které používají oba programy (farmer i worker). Jsou to procedura pro alokaci paměťového místa pro matici a pro string, procedura pro načtení matice ze souboru a procedury pro otevření a zavření souboru a procedura pro vypsání matice na obrazovku.

6. Práce s programem v C

Pomocí skriptu **Makefile** se program přeloží příkazem *make*. Pro spuštění programu si nejprve spustíme konzolu PVM příkazem **pvm**, poté pomocí příkazu **add** *síťové_jméno_stroje* přidáme jmenovaný stroj do konfigurace virtuálního multiprocesoru, potom příkazem **farmer** matice.txt výpočet spustíme. Vstupní argumentem souboru farmer je soubor s maticí (“farmer <soubor_s_matici>”). Poté se spustí daný počet workeru a výsledek výpočtu je vypsán na obrazovku a uložen do souboru *vystup.txt.*

7. Výpis programu v programovacím jazyku Java

```
/*
* Semestralni prace z PPR
* Vypracoval KLABUSAY Dusan
* Vypocet k-mocniny matice
* leden 2003
*/

package dusan;

import java.io.*;
import java.util.*;

class Matrix implements Serializable{
      private double [][] matice; // Vlastni matice
      private double [][] kmatice; // pomocna matice
      private int rozmer = 6;       //rozmer matice
      public static int RAND_MAX = 10;    //timto intem lze menit maximalni
hodnoty v generovane matici
      /**
       * konstructor pro Matrix s rozmerem
       */
      Matrix(int rozmer){
            this.rozmer = rozmer;
            generateMatice();
      }

  double[] dejRadek(int sloupec){  // kazde vlakno si sahne pro svoje data
    return matice[sloupec];
  }
/*
* Prekopiruje obsah matice do druhe
*/
double[][] prekopirujMatici(int kook){
      kmatice = new double[this.rozmer][this.rozmer];
      for(int i = 0; i < this.rozmer; i++){
                  for (int j = 0; j < this.rozmer; j++)
                        kmatice[i][j] = matice[i][j] ;
            }
      return kmatice;
}

  void nastavRadek(int sloupec,double[] naco){ //nastavi kazde vlakno
vypocetena data
      matice[sloupec]=naco;
  }
      /*
      * Vygeneruje nahodnou matici rozmeru rozmer
      */
      void generateMatice(){
            Random generator = new Random();
            matice = new double[this.rozmer][this.rozmer];
            for(int i = 0; i < this.rozmer; i++){
                  for (int j = 0; j < this.rozmer; j++)
                        matice[i][j] = (double)
generator.nextInt(this.RAND_MAX);
            }
      }
      /**
       * Vytiskne matici na monitor
       */
      void printOutMatrix(){
      for(int i = 0; i < this.rozmer; i++){
            System.out.println();
```

```
                for (int j = 0; j < this.rozmer; j++)
                if (matice[i][j] < 10)
                      System.out.print(" "+matice[i][j]+" ");
                else System.out.print(""+matice[i][j]+" ");
            }
            System.out.println();
}
}


class RadekSpocti extends Thread{
      private int       mytag;      //id vlakna
      private Matrix out;           //objekt pro ukladani vysledku
      private double[] mradek;    //vstupni radek
      private double[] vradek;    //vystupni radek
      private double[][] matice;    //matice, kterou budu mocnit
      private int N;                //Rozmer marice
      private double pomoc[];       //pomocny vektor pro vypocet
      private int kmocnina;         //mocnina, na kterou mam mocnit matici

      RadekSpocti(int tag, double[] radek,int pocetDelniku,double[][]
vmatice,int lmocnina,Matrix result){
      mytag = tag;            // toto je identifikator vlakna
      out = result;           // vysledek
      N = pocetDelniku; //rozmer matice
      mradek = radek;         // vstupni radek
      pomoc=radek;            // pomocny radek
      vradek=radek;           // radek pro ukladani vysledku
      kmocnina=lmocnina;      // mocnina matice
      matice = vmatice; // vlastni matice
      }

    public void run(){ //Vlastni implementace RUNu()
      int pom_j,n_i,n_j,i,pomocna=0;
      System.out.println("Pocitam "+mytag+" radek matice!");
      pomoc = new double[N];
      //Vlastni vypocet matice, nesnazte se pochopit indexy ve for cyklech
      for(i=0;i<N;i++){
            pomoc[i]=mradek[i];
      }
      for (i =1;i<kmocnina;i++){
            for (pom_j=0,n_i=0;pom_j<N;pom_j++,n_i++){
                  pomocna = 0;
                  for (n_j=0;n_j<N;n_j++){
                        pomocna += pomoc[n_j]*matice[n_j][n_i];
                  }
                  mradek[pom_j]=pomocna;
            }
            for (int k=0;k<N;k++)
                  pomoc[k]=mradek[k];
      }
      for (int k=0;k<N;k++)
            vradek[k]=mradek[k];
    out.nastavRadek(mytag,vradek);
    //mytag znamena cislo vlakna 0,1,2.....
  }

}




public class Matice {
```

```
  public Matice() {
  }

  public static void main(String[] args) {
    int pocetDelniku = 3; //pocet delniku, udava zaroven rozmer matice
    int mocnina = 2;      // Mocnina matice, kterou chceme spocitat
    double maticeZaklad[][]; //pomocna matice, kterou predavame vlaknum

    Matice matice = new Matice();
    matice.invokedStandalone = true;
    RadekSpocti[] poleVlaken = new RadekSpocti[pocetDelniku];
//inicializace pole vlaken
    Matrix result = new Matrix(pocetDelniku);  //Vygenerujeme si nahodnou
matici
    maticeZaklad=result.prekopirujMatici(pocetDelniku);    //Zkopirujeme si
matici do pomocne matice
    System.out.println("Pocatecni vygenerovana matice:");
    result.printOutMatrix();
    for (int i=0; i<pocetDelniku; i++ ){ //Spustime si dany pocat vlaken
      poleVlaken[i] = new
RadekSpocti(i,result.dejRadek(i),pocetDelniku,maticeZaklad,mocnina,result);
                  poleVlaken[i].start();
            }

    for (int p=0; p<pocetDelniku; p++){ // cekani na ukonceni vlaken
          try{
              poleVlaken[p].join();
        }   catch(InterruptedException e){
            System.out.println("Thread is interrupted!");
      }
    }
      System.out.println("Vypoctena matice");
      result.printOutMatrix(); // Vtiskneme vyslednou k-matici
  }
  private boolean invokedStandalone = false;
}
```

8. Výpis programů v programovacím jazyce C pro PVM

soubor farmer.*c*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "/opt/pvm3/include/pvm3.h"
#include "help.h"
#define WORK_NAME "worker" /* jmeno spustitelneho souboru workera*/
#define VYSTUP "vystup.txt" /*vystupni soubor*/

/* PVM - ukazka vypocetniho modelu Farmer - Workers           */
/* Sef zadava delnikum postupne cela licha cisla do hodnoty m */
/*   a delnik zkouma, zda je to prvocislo. Vysledkem je  pole */
/*   vsech prvocisel az do hodnoty m.                         */
/* Pouzivane kody zprav:                                      */
/*   4 ... broadcast (id sefa  -> delnici)                    */
/*   5 ... odpoved   (id delnika -> sef)                      */
/*   6 ... zadani ukolu (sef -> delnik),                      */
/*         zaporna hodnota cisla znamena konec cinnosti       */
/*   7 ... vysledek     (delnik -> sef),                      */
/*         zaporna hodnota cisla znamena "neni to prvocislo"  */

int main(int argc, char *argv[]) {
  int mytid;        /*  identifikator procesu sef  */
  int *wtids;       /*  identifikatory procesu delnici */
  int num_workers;  /*  pocet delniku */
  char *w_arg[3];   // pole argumentu pro workery
  double *vysledek; // pole pro vysledek
```

```
 int mocnina;
 int i,j;          /*  pomocne promenne */
 FILE *fr,*fw;     // soubory
 double **matice;  //vlastní matice

/* kontrola vstupnich parametru*/
if (argc !=2) {
       printf("Spatne zadane vstupni parametry!\n");
       printf(" pouziti: farmer <jmeno_souboru_s_matici> \n");
       exit(1);
}

      /*Otevreni souboru s datz a nacteni do pameti*/
      fr = otevri_soubor(argv[1],"r");
      matice = nacti_matici(fr);/* otevrit soubor se vstupnimi daty*/
      printf("Rozmer matice: %d\n",N);
      vypis_matice(matice);
      /*Ziskani mocniny matice*/
      printf("Zadejte mocninu teto matice:");
      scanf("%d", &mocnina);
      /*Počet workeru je dan velikosti matice*/
      num_workers=N;
      /* alokace pole pro tid workeru*/
      if((wtids = (int *) malloc(N*sizeof(int))) == NULL) {
            printf("Malo pameti!\n");
            exit(1);
      }

      if((vysledek = (double *) malloc(N*sizeof(double))) == NULL) {
            printf("Malo pameti!\n");
            exit(1);
      }

      /* alokace pameti pro parametry workeru*/
      w_arg[0]=aloc_string(30); /*jmeno souboru*/
      w_arg[1]=aloc_string(20); /*Ktery radek matice bude zpracovavat*/
      w_arg[2]=aloc_string(20); /*Mocnina, kterou je treba vypocitat*/
      w_arg[3]=0x00;


      mytid = pvm_mytid();  /* prihlaseni do PVM */
      strcpy(w_arg[0],argv[1]);/*Jmeno souboru s matici*/
      sprintf(w_arg[2],"%d",mocnina);

      /* vytvoreni delniku */
      for (i=0; i< num_workers; i++) {
            sprintf(w_arg[1], "%d", i);/*Ktery radek se bude zpracovavat*/
            if (pvm_spawn(WORK_NAME, w_arg, 0, "", 1, &(wtids[i])) != 1) {
                  fprintf(stderr,"pvm_spawn: Task nebyl vytvoren!\n");
                  pvm_exit();
                  exit(1);
            }
      fw = otevri_soubor(VYSTUP,"w");
      printf("Vytvoren delnik %d.\n", wtids[i]);
      }
      /*Ziskani vysledku od workeru*/
      for (i=0; i<num_workers; i++) {
            pvm_recv(wtids[i],5);           // blokujici -> ceka na prijem zpravy
            pvm_upkdouble(vysledek, N, 1);   //pole radky, kterou vypocital worker
            printf("Radka %d\n",i);
            for (j=0;j<N;j++)
                  printf("%lf\t",vysledek[j]);
                  fprintf(fw,"%lf\t",vysledek[j]);
            printf("\n");
            fprintf(fw,"\n");

      }

 zavri_soubor(fr);
 zavri_soubor(fw);
 free(w_arg[0]);
 free(w_arg[1]);
```

```
  free(w_arg[2]);
  free(w_arg[3]);
  /* ukonceni cinnosti v PVM */
  pvm_exit();
return 0;
}
```

soubor worker.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "help.h"
#include "/opt/pvm3/include/pvm3.h"


int main(int argc, char *argv[]) {
       int mytid;      /*  identifikator procesu delnik  */
       int chief_tid;  /*  identifikator procesu sef  */
       int i;          /*  pomocna promenna */
       int mocnina;    /*  pomocna promenna */
       int pom_j,n_i,n_j;
       int radka;
       double **matice;
       FILE *fr;
       double *vektor,*pomoc,pomocna; /*promenne pro vypocet*/

       mytid = pvm_mytid();  /* prihlaseni do PVM */
       chief_tid=pvm_parent(); /* zjisteni id farmare*/
        if (argc != 3) {
              printf("Chybne argumenty.\n");
              pvm_exit();
              exit(1);
       }
       radka = atoi(argv[2]);
       mocnina=atoi(argv[3]);
       fr=otevri_soubor(argv[1],"r");  /* otevrit soubor se vstupnimi daty*/
       matice = nacti_matici(fr);/* otevrit soubor se vstupnimi daty*/
       if((vektor = (double *) malloc(N*sizeof(double))) == NULL) {
         printf("Malo pameti!\n");
         exit(1);
       }
       if((pomoc = (double *) malloc(N*sizeof(double))) == NULL) {
              printf("Malo pameti!\n");
              exit(1);
       }

       for(i=0;i<N;i++){
              vektor[i]=matice[radka][i];
              pomoc[i]=vektor[i];
       }
       /*Vlasni vypocet mocniny radky*/
       for (i =1;i<mocnina;i++){
              for (pom_j=0,n_i=0;pom_j<N;pom_j++,n_i++){
                     pomocna = 0;
                     for (n_j=0;n_j<N;n_j++){
                            pomocna += pomoc[n_j]*matice[n_j][n_i];
                     }
                     vektor[pom_j]=pomocna;
              }
              for (i=0;i<N;i++)
                     pomoc[i]=vektor[i];
       }


       //inicializace vysilaciho bufferu
       if (pvm_initsend(PvmDataDefault) == 0) {
              fprintf(stderr,"Chyba pri inicializaci vysilaciho bufferu!\n");
       pvm_exit();
       }
       pvm_pkdouble(vektor,N,1);
       pvm_send(chief_tid, 5);

       zavri_soubor(fr);
```

```
	free(pomoc);
	free(vektor);
	/* ukonceni cinnosti */
	pvm_exit();
return 0;
}
```

Soubor help.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*Nacte vstupni matici ze souboru*/
double **nacti_matici(FILE *fr){
int i=0,j=0;
double **matice,buff;
	printf("Nacitam matici ze souboru :)\n");
	while (fscanf(fr,"%f",&buff) != EOF)
		i++;
	fseek(fr,SEEK_SET,0);
	N=(int)sqrt(i);
	/*Nemame presny rozmer matice */
	if (N*N != i) {
		printf("Chyba ve vstupnim souboru!!\n");
		if (fclose(fr) == EOF){
			printf("Soubor se nepodarilo uzavrit :( \n");
		}
		exit( 0);
	}
	printf("Rozmer matice:%d\n",N);
	matice=alokuj_matici();
	i=0;
	printf("Nacitam vlastni matici\n");
	while (fscanf(fr,"%f",&buff) != EOF) {
		/*printf("Nacitam prvek [%d][%d]=%f\n",i,j,buff);*/
		matice[i][j]=buff;
		j++;
		if (j==(N)) {
			j=0;
			i++;
		}
	}
	zavri_soubor(fr);

return matice;
}


/*Vypise matici*/
int vypis_matice(double **matice){
	int i,j;
	printf("Vypis matice:\n");
	for (i=0;i<N;i++){
		for (j=0;j<N;j++){
			printf("%f\t",matice[i][j]);
		}
	printf("\n");
	}
	return 1;
}

/*Naalokuje misto pro matici*/
float **alokuj_matici(void){
	double **matice;
	int i;
	printf("Alokuji misto pro matici\n");
	matice=(double **)malloc(N*sizeof(double *));
	for (i=0;i<N;i++)
		matice[i]=(double *)malloc(N*sizeof(double *));
	return (matice);
}
```

```
// funkce pro operace se soubory
FILE *otevri_soubor(char *name, const char *mod){
FILE *f;
  if ((f=fopen(name,mod)) == NULL) {
    fprintf(stderr,"fopen: Chyba pri praci se souborem : %s\n",name);
    exit(1);
  }
  return(f);
}

void zavri_soubor(FILE *file){
    if (fclose(file) == EOF) {
       fprintf(stderr,"fclose: Soubor nelze uzavrit.\n");
       exit(1);
    }
}


char *aloc_string(int delka)  {
 char *c;
   if ((c=(char *) malloc(delka*sizeof(char))) == NULL) {
         fprintf(stderr,"malloc: Malo pameti!\n");
        exit(1);
   }
   return(c);
}
```

9. Závěr

Program v PVM jsem nejprve zkoušel odladit na školním serveru jumbo, ale měl jsem s tím hodně problémů (program např. už po zavolání pvm_mytid(); ”zasegmentil”), tak jsem si PVM nainstaloval na svůj linux, na kterém jsem neměl problémy. Pouze předávání pole double mi činilo problémy.

Program v Javě byl vytvořen a testován na operačním systému Windows. Je to docela jednoduchý program, mohl být složitější, ale vzhledem k tomu, že toto je můj vůbec první program v Javě, co jsem kdy napsal (a je to asi poznat), bych poprosil o shovívavější hodnocení. Nemohl jsem například vyřešit vstup matice ze souboru, potřeboval bych více zkušeností s tímto jazykem, ale to snad doženu na předmětu OOP.

Tato práce mi trochu změnila pohled na paralelní programování, už to pro mě není takový “strašák” jako dřív, docela mě bavilo si „hrát“ s PVM. Ale jako hlavní přínos této práce bych ohodnotil, že jsem konečně naprogramoval něco v Javě, a musím říct, že se mi tento jazyk velice zalíbil.