Paralelno programiranje

Praktikum iz operativnih sistema

Filip Brčić

9. juni 2006

Osnove paralelnog programiranja

Tradicionalno programiranje

- Jedan kompjuter
- Jedna memorija
- Jedan program
- Jedan podatak

Osnove paralelnog programiranja

Flinovo grananje

JIJP **J**edna **I**nstrukcija – **J**edan **P**odatak

JIVP **J**edna **I**nstrukcija – **V**iše **P**odataka

VIJP **V**iše **I**nstrukcija – **J**edan **P**odatak

VIVP **V**iše **I**nstrukcija – **V**iše **P**odatak

Osnove paralelnog programiranja

Razlike između tradicionalnog i paralelnog programiranja

- Organizacija taskova
- Faze inicijalizacije pre računanja
- Izbor granularnosti
- Heterogenost sistema

Osnove paralelnog programiranja

Standardne paradigme paralelnog programiranja

- „Gospodar – rob” odnos (master – slave)
- Komunistički model (svi su jednaki)
- Model drveta
- Hibridni modeli

Osnove paralelnog programiranja

Model grupe

- „Gospodar – rob” i komunistički modeli su suštinski isti
- Jedna instanca prikazuje rezultate (i) računa
- Ostale instance samo računaju
- Pogodno za algoritme gde se zna broj instanci unapred (množenja matrica)

Osnove paralelnog programiranja

Model drveta

- Najprirodniji model za paralelne algoritme
- Nepoznat broj instanci unapred
- Prva instanca prikazuje rezultate,
- ostale instance samo vraćaju rezultate „roditeljskoj” instanci.
- Pogodno za rekurzivne algoritme (pretrage, sortiranja, „zavadi-pa-vladaj”, . . .)

Model grupe

Faze algoritma

1. Inicijalizacija procesa grupe (u zavisnosti od parametara problema)
2. Računanje (svi u isto vreme)
3. Sklapanje finalnog rezultata i prikaz rezultata („gospodar”)

Ilustracija modela grupe – Mandelbrot

Program gospodara – Inicijalizacija

```
for i := 0 to BrojRadnika - 1
        Startuj radnika i
        Posalji zadatak radniku i
endfor
```

Ilustracija modela grupe – Mandelbrot

Program gospodara – Rad

```
while (ImaPosla)
        Primi rezultat
        Posalji zadatak sledecem slobodnom radniku
        Prikazi rezultat
endwhile
```

Ilustracija modela grupe – Mandelbrot

Program gospodara –Kraj

```
for i := 0 to BrojRadnika - 1
        Primi rezultat
        Ubij radnika i
        Prikazi rezultat
endfor
```

Ilustracija modela grupe – Mandelbrot

Program radnika

```
while (true)
        Primi zadatak
        Izracunaj rezultat
        Posalji rezultat gospodaru
endwhile
```

Model drveta

Faze algoritma

1. Podela podataka
2. Slanje podataka čvorovima ispod (deci)
3. Skupljanje podataka od čvorova ispod (dece)
4. Sklapanje podataka u celinu
5. Slanje podataka roditeljskom čvoru (ili ispis)

Šta je MPI?

- Specifikacija biblioteke
- Implementacije na Fortran-u, C-u, C++-u
- Za paralelne računare, klastere i heterogene mreže

Šta je MPI?

Osobine MPI-a

- Modularnost
- Portabilnost
- Heterogenost
- Sigurna komunikacija
- Podgrupe
- Topologije
- Alati za merenje performansi

Koliko je komplikovan MPI?

- MPI je veliki: MPI-1 ima 128 funkcija, a MPI-2 152 funkcije
- MPI je mali: sa samo 6 funkcija može 99% stvari da se postigne.

Osnovnih 6 funkcija MPI-a

1. Inicijalizacija *MPI_Init ()*
2. Kraj rada *MPI_Finalize ()*
3. Koliko ima računara u klasteru? *MPI_Comm_size ()*
4. Koji je moj rang? *MPI_Comm_rank ()*
5. Pošalji podatke *MPI_Send ()*
6. Primi podatke *MPI_Recv ()*

Najmanji MPI program

Inicijalizacija i kraj rada

```
#include <stdio.h>
#include "mpi.h"
int main (int argc, char **argv)
{
  MPI_Init (&argc, &argv);
  printf ("Zdravo svima!\n");
  MPI_Finalize ();
  return 0;
}
```

Najmanji MPI program

Moj rang i veličina klastera

```
#include <stdio.h>
#include "mpi.h"
int main (int argc, char **argv)
{
  int rank, size;
  MPI_Init (&argc, &argv);
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
  MPI_Comm_size (MPI_COMM_WORLD, &size);
  printf ("Zdravo svima! Ja sam %d od %d\n",rank,size)
  MPI_Finalize ();
  return 0;
}
```

Pocetak rada

```
#include <stdio.h>
#include <mpi.h>
int main (int argc, char **argv)
{
  int i, rank, size, dest;
  int to, src, from, count, tag;
  int st_count, st_source, st_tag;
  double data[100];
  MPI_Status status;
  MPI_Init (&argc, &argv);
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
  MPI_Comm_size (MPI_COMM_WORLD, &size);
  printf ("Proces %d od %d se javlja.\n", rank, size);
  dest = size - 1;
  src = 0;
```

Slanje podataka

```
if (rank == src) {
  to    = dest;
  count = 100;
  tag   = 2001;
  for (i = 0; i < 100; i++)
    data[i] = i;
  MPI_Send (data, count, MPI_DOUBLE, to, tag,
            MPI_COMM_WORLD);
}
```

Primanje podataka

```
else if (rank == dest) {
  tag     = MPI_ANY_TAG;
  count = 100;
  from = MPI_ANY_SOURCE;
  MPI_Recv (data, count, MPI_DOUBLE, from, tag,
            MPI_COMM_WORLD, &status);
  MPI_Get_count (&status, MPI_DOUBLE, &st_count);
  st_source= status.MPI_SOURCE;
  st_tag= status.MPI_TAG;
  printf ("Informacije: izvor = %d, oznaka = %d,"
          " velicina = %d\n",
           st_source, st_tag, st_count);
  printf (" %d primio: ", rank);
  for (i = 0; i < st_count; i++)
    printf ("%lf ", data[i]);
  printf ("\n");
}
```

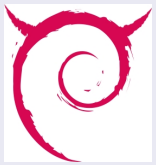
Kraj rada

```
    MPI_Finalize();
    return 0;
}
```

Ilustracija modela drveta – Bucket sort

Primer sa adrese http://csce.uark.edu/~aapon/courses/concurrent/-mpiexamples/16parallel_bucket_sort.c

Klaster

-  Gentoo GNU/Linux, AMD Athlon64 3000+, 3GB RAM, 200GB HD, IP: 192.168.1.1, LAM/MPI 7.1.1
-  FreeBSD 6.1, AMD Athlon64 3000+, 128MB RAM, 5GB HD, IP: 192.168.4.2, LAM/MPI 7.1.1
-  Debian GNU/kFreeBSD 20060331, AMD Athlon64 3000+, 128MB RAM, 5GB HD, IP: 192.168.6.2, LAM/MPI 7.1.1

Kraj predavanja

Literatura

- http://www-unix.mcs.anl.gov/mpi/
- http://www.lam-mpi.org/
- http://www.mpi-forum.org/
- http://csce.uark.edu/~aapon/courses/concurrent/schedule.html

Kontakt

- Filip Brčić <brcha@users.sourceforge.net>
- Web: http://purl.org/NET/brcha/home/documents/MPI