Metody programowania równoległego

Sprawozdanie z mierzenia opóźnienia i przepustowości w klastrze

autor: Ilona Tomkowicz

Akademia Górniczo-Hutnicza
Wydział Informatyki, Elektroniki i Telekomunikacji,
Informatyka, II stopień, I semestr

08 marca 2020

Contents

1. Kody źródłowe

1.1 Komunikacja z użyciem Send i Recv

```
#include <stdio.h>
#include "mpi.h"

void get_device_info(int* rank, int*size)
{
    MPI_Comm_rank (MPI_COMM_WORLD, rank);  /* get current process id */
    MPI_Comm_size (MPI_COMM_WORLD, size);  /* get number of processes */
    //printf("Current process id %d, number of processes %d.\n", *rank, *size);
}

int main(int argc, char** argv) {
    int rank, size, i;
    int          n = 10000;
    MPI_Status  status;
    double       start, elapsed;

    MPI_Init(&argc, &argv);
    get_device_info(&rank, &size);

    int len = 2<<13; //

    /* opoznienie */
    if (rank == 0)
    {
          int k[len];
          start = MPI_Wtime();
          for (i = 0; i < n; ++i)
          {
              k[0] = i;
              MPI_Send(k, len, MPI_INT, 1, 123, MPI_COMM_WORLD);
              MPI_Recv(k, len, MPI_INT, 1, 123, MPI_COMM_WORLD, &status);
              if (k[0] != i+1) printf("error\n");
          }
        elapsed = MPI_Wtime() - start;
        printf("av message of length %d has time of %g sec\n", len,  elapsed/(2*n));
        fflush(stdout);
```

```
    }

    /* przepustowosc */
    if (rank == 1)
    {
            int k[len];
            start = MPI_Wtime();
            for (i = 0; i < n; ++i)
            {  k[0] = i;
               MPI_Send(k, len, MPI_INT, 0, 123, MPI_COMM_WORLD);
               MPI_Recv(k, len, MPI_INT, 0, 123, MPI_COMM_WORLD, &status);
               if (k[0] != i+1) printf("error\n");
            }
            elapsed = MPI_Wtime() - start;
            long int s = sizeof(k);
            printf("av message of size %lu has %g Mbit/s\n",
                   8*s, 8*s/ (1000000 * elapsed/(2*n)));
            fflush(stdout);
       }
     MPI_Finalize();
     return 0;
}
```

1.2 Komunikacja z użyciem Isend i Irecv

```
#include <stdio.h>
#include "mpi.h"

void get_device_info(int* rank, int*size)
{
    MPI_Comm_rank (MPI_COMM_WORLD, rank);  /* get current process id */
    MPI_Comm_size (MPI_COMM_WORLD, size);  /* get number of processes */
    //printf("Current process id %d, number of processes %d.\n", *rank, *size);
}

int main(int argc, char** argv) {
    int rank, size, i;
    int          n = 10000;
    MPI_Status  status;
    MPI_Request req1, req2;
    double       start_time, elapsed_time;

    MPI_Init(&argc, &argv);
    get_device_info(&rank, &size);

    int len = 2<<4;

    if (rank == 0)
    {
```

```
            int k[len];
            start_time = MPI_Wtime();
            for (i = 0; i < n; ++i)
            {
                k[0] = i;
                MPI_Isend(k, len, MPI_INT, 1, 123, MPI_COMM_WORLD,&req1);
                MPI_Irecv(k, len, MPI_INT, 1, 123, MPI_COMM_WORLD, &req2);
                MPI_Wait(&req1, &status);
                if (k[0] != i) printf("error\n");
            }
          elapsed_time = MPI_Wtime() - start_time;
          printf("av message of length %d has time of %g sec\n", len,  elapsed/(2*n));
          fflush(stdout);

    }

    if (rank == 1)
    {
            int k[len];
            start_time = MPI_Wtime();
            for (i = 0; i < n; ++i)
            {  k[0] = i;
               MPI_Isend(k, len, MPI_INT, 0, 123, MPI_COMM_WORLD, &req1);
               MPI_Irecv(k, len, MPI_INT, 0, 123, MPI_COMM_WORLD, &req2);
               MPI_Wait(&req2, &status);
               if (k[0] != i) printf("error\n");
            }
            elapsed_time = MPI_Wtime() - start_time;
            printf("av message of size %lu has %g Mbit/s\n",
                   8*s, 8*s/ (1000000 * elapsed/(2*n)));
            fflush(stdout);
       }
     MPI_Wait(&req1, &status);
     MPI_Wait(&req2, &status);
     MPI_Finalize();
     return 0;
}
```

1.3 Komunikacja z użyciem jednego węzła i pamięci współdzielonej

Konfiguracja pliku allnodes zawierała tylko adres tego węzła, a program uruchamiano na tym samym węźle.

```
vnode-01.dydaktyka.icsr.agh.edu.pl:4
```

1.4 Komunikacja z użyciem jednego węzła i połączenia sieciowego

Konfiguracja pliku allnodes zawierała tylko adres tego węzła, a program uruchamiano na innym węźle, w tym wypadku na 02.

1.5 Komunikacja z użyciem dwóch węzłów, będących fizycznie na tej samej maszynie i połączenia sieciowego

Konfiguracja pliku allnodes zawierała adresy używanych węzłów (01, 03), a program uruchamiano na węźle 02.

```
vnode-01.dydaktyka.icsr.agh.edu.pl:4
vnode-03.dydaktyka.icsr.agh.edu.pl:4
```

1.6 Komunikacja z użyciem dwóch węzłów, będących fizycznie na różnych maszynach i połączenia sieciowego

Konfiguracja pliku allnodes zawierała adresy używanych węzłów (05, 06), a program uruchamiano na węźle 02.

```
vnode-05.dydaktyka.icsr.agh.edu.pl
vnode-06.dydaktyka.icsr.agh.edu.pl
```

2. Dane pomiarowe

3. Wykresy dla różnych konfiguracji

4. Wnioski