MPI Örnekleri

Eray Özkural

Sunum Çerçevesi

- MPI kütüphanesini kullanma
- İki nokta arası iletişim
- Toplu iletişim
- Paralel vektör iç çarpımı
- Örnek uygulama: paralel quicksort

MPI kütüphanesini kullanma

- Init / Finalize
  - mutlaka gerekli
- MPI_COMM_WORLD:
  - iletişim "handle"ı
  - bütün işlemciler için
- Comm_size: kaç işlemci?
- Comm_rank: ben kaçıncıyım?

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

int main(int argc, char **argv)
{
  int id, numprocs;

  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD,&id);

  /*printf("Merhaba! Ben islemci %d.\n", id);*/
  if (id==0)
    printf("Merhaba! Ben islemci 0. Toplam %d
islemci var.\n", numprocs);

  MPI_Finalize();

  return 0;
}
```

İki nokta arası iletişim

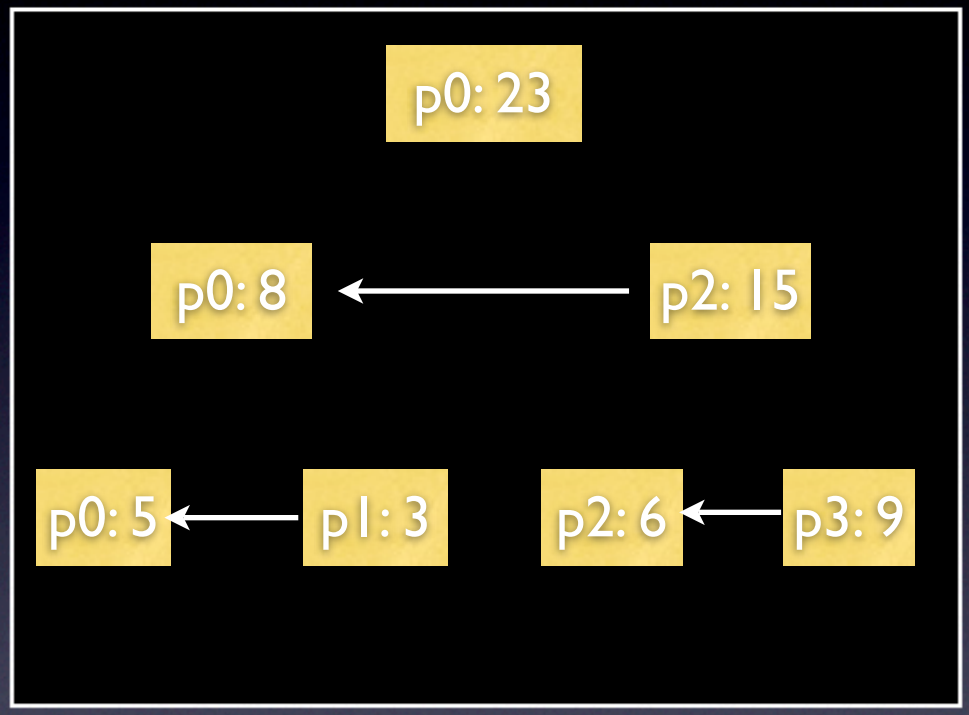
- MPI_Recv ve MPI_Send komutlari
  - iki işlemci arasında gerçekleşir
  - blocking mode'da veri gönderip alır
  - MPI_IRecv ve MPI_ISend nonblocking versiyonları. Hesaplama/iletişim aynı anda.
  - İletişim sıralarına dikkat etmek gerekir
  - Birçok paralel algoritma bunlarla yazılabilir

İki nokta arası iletişim

- İki işlemcide 10 elemanlık birer rastgele tamsayı vektörü
- Birinci işlemci ikinciye elindeki vektörü gönderir
- İkinci işlemci iki vektörü çarpıp sonucu yazar

```
  MPI::Init(argc,argv);
  MPI::Intracomm comm = MPI_COMM_WORLD;
  int numprocs = comm.Get_size();
  int id = comm.Get_rank();
  if (numprocs!=2) {
    cout << "Bu program sadece iki islemciyle calisir."
<< endl;
    return 1; // error
  }
  int a[10];
  rand_array(a, 10, 100); // 10 random ints in 0..99
  if (id==0)
    comm.Send(a, 10, MPI::INT, 1, 13);
  else {
    int buf[10];
    comm.Recv(buf, 10, MPI::INT, 0, 13);
    int dot_prod = dot_product(a, buf, 10);
    cout << "Iki tamsayi vektorun carpimi = " <<
dot_prod << endl;

   MPI::Finalize();
```



Hatırlatma: MPI_Reduce

Toplu iletişim

```
int a[10];
rand_array(a, 10, 100); // 10 random ints in 0..99
int buf[10];
comm.Reduce(a, buf, 10, MPI::INT, MPI::SUM, 0);
if (id==0)
  print_vector(buf, 10);
```

- Her işlemcide 10'ar elemanlık birer rastgele tamsayı vektörü
- Tek bir iletişim komutuyla bütün vektörler toplanır
- İlk işlemci toplam vektörünü yazar

Vektör iç çarpımı

```
  float a[10];
  rand_array(a, 10, 1.0); // 10 random numbers in (0,1)
  float b[10];
  rand_array(b, 10, 1.0); // 10 random numbers in (0,1)
  float localdotp = dot_product(a, b, 10);
  float globaldotp;
  comm.Reduce(&localdotp, &globaldotp, 1,
MPI::FLOAT, MPI::SUM, 0);
  if (id==0)
    cout << "a.b = " << globaldotp << endl;
```

- a ve b vektörleri işlemcilere dağıtılmıştır
- her işlemci elindeki vektör parçalarıyla yerel iç çarpım değerleri hesaplar
- Reduce işlemiyle global iç çarpım değeri hesaplanır

Paralel Quicksort

- Hiperküp topolojisi üzerinde iletişim
- Temel mantık:
  - Her işlemcide bir sayı listesi vardır
  - İşlemciler arası pivotlar tespit edilir ve "partition" alt rutini çalıştırılır (logp adım)
  - Sonuçta ilk işlemcide en küçük sayılar, ikincide sonrakiler... son işlemcide en büyükler kalır.
  - Her işlemci elindekileri sıralar