Hacettepe Üniversitesi
Bilgisayar Mühendisliği Bölümü
BBM442 Koşut İşlem
Genel Sınav 04-06-2014

NOT: Sadece iki soruyu yanıtlayınız. Soruların ağırlıkları eşittir.

Soru 1. "Paylaşımlı Bellekli MIMD" mimaride, MPI işlevlerini kullanan ve "Dağıtılmış Bellekli MIMD" için geliştirilmiş yazılımları çalıştırmak için bir gereksinim olduğunu varsayınız. "MPI_Gather" işlevini "Paylaşımlı Bellekli MIMD" mimaride gerçekleştirmek için:

a) Tasarım yapınız,

b) İşlevleri C Programlama Diliyle kodlayınız.

(Not: Gerçekleştiriminizde **MPI_Barrier(MPI_Comm comm)** işlevini kullanabilirsiniz.)

```
int MPI_Gather(void *sendbuf, int sendcount,
        MPI_Datatype senddatatype, void *recvbuf, int recvcount,
        MPI_Datatype recvdatatype, int target, MPI_Comm comm)
```

Each process, including the target process, sends the data stored in the array sendbuf to the target process. As a result, if p is the number of processors in the communication comm, the target process receives a total of p buffers. The data is stored in the array recvbuf of the target process, in a rank order. That is, the data from process with rank i are stored in the recvbuf starting at location i * sendcount (assuming that the array recvbuf is of the same type as recvdatatype).

The data sent by each process must be of the same size and type. That is, MPI_Gather must be called with the sendcount and senddatatype arguments having the same values at each process. The information about the receive buffer, its length and type applies only for the target process and is ignored for all the other processes. The argument recvcount specifies the number of elements received by each process and not the total number of elements it receives. So, recvcount must be the same as sendcount and their datatypes must be matching.

Soru 2. Aşağıda CUDA uyumlu bir grafik işlemcinin çok sayıda çekirdeğini koşut olarak kullanan bir kod kesimi verilmiştir (NVIDIA'nın CUDA teknolojisi). Bu örnekte C dizisinin her elemanı bir izlek (thread) tarafından hesaplanmaktadır. VecAdd işlevini (kernel) C dizisinin toplamını da bulacak şekilde yeniden kodlayınız.

```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
int i = threadIdx.x;
C[i] = A[i] + B[i];
}

int main()
{
...
// Kernel invocation with N threads
VecAdd<<<1, N>>>(A, B, C);
...
}
```

Soru 3. İki boyutlu Torus topolojisiyle bağlanmış Dağıtılmış Bellekli MIMD türü bir bilgisayarda yatay eksen boyunca görevlerdeki en büyük değeri bulan algoritmayı tasarlayınız ve MPI kitaplığını kullanarak kodlayınız. (İşlem sonunda her görev bulunduğu satırdaki en büyük değeri yerel değişkeninde elde etmiş olacaktır.)

Kullanabileceğiniz MPI işlevlerinden bazıları:

```
int MPI_Init(int *argc, char ***argv)

int MPI_Finalize()

int MPI_Comm_size(MPI_Comm comm, int *size)

int MPI_Comm_rank(MPI_Comm comm, int *rank)

int MPI_Send(void *buf, int count, MPI_Datatype datatype,int dest, int
                tag, MPI_Comm comm)

int MPI_Recv(void *buf, int count, MPI_Datatype datatype,int source, int
                tag, MPI_Comm comm, MPI_Status *status)

int MPI_Isend(void *buf, int count, MPI_Datatype datatype,
        int dest, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Irecv(void *buf, int count, MPI_Datatype datatype,
        int source, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status)

int MPI_Wait(MPI_Request *request, MPI_Status *status)

int MPI_Sendrecv(void *sendbuf, int sendcount,MPI_Datatype senddatatype,
                int dest, int sendtag,void *recvbuf, int recvcount,
                MPI_Datatype recvdatatype,int source, int recvtag,
                MPI_Comm comm,MPI_Status *status)
```

Kullanabileceğiniz değişmezler: **`MPI_COMM_WORLD,MPI_ANY_SOURCE,MPI_ANY_TAG`**