CS 5334/4390 Spring 2017 Shirley Moore, Instructor Exam 2 Part 2 Retake 40 points Name Sheikh Tenveer Hossein

You may use notes for this exam. You may not use any books, computers, PDAs, cellphones, or other electronic aids. Please show all work and explain your answers fully. Each problem is worth 20 points. See page 3 for MPI syntax.

- 1. Assume you have 16 processes in MPI_COMM_WORLD and arrays A and B each containing 64 double precision floating point numbers.
- a) Write the MPI statements that create a 2D Cartesian communicator comm2d for the 16 int indims, reorder; MPI_Comm Comm2d; int dims[2] reviod[2]; indims=2; reorder=0; dims[0]=4; dims[1]=4; period[0]=0; reviod[1]=0; MPI-Cert-create (MPI-Comm-world, ndims, dims, periords, reorder Comm 2d;);
 - b) Write the MPI statements to create 1D row and column subcommunicators named rowcomm and colcomm, respectively. int remain-dims []; remain_dims [6]=0; remain-dims [1]=1; MPI-Cert sub (comm 2d, remain-dims; row comm); remain dims [0]=1, remain-dims [1]=0; MPI_Cort_sub (comm 2d, remain_dims; colcomm);
 - c) Assume that A and B have been distributed in a 2D block distribution to the processes in comm2d. Write the MPI collective communication statements that will distribute the blocks of A in each row of comm2d to all the processes in that row and the blocks of B in

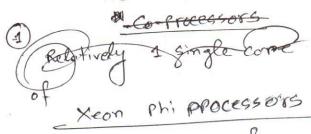
* Consider that the processors kept their respective value of A and B in double precision veriable named Double *record * record, * recv B; recv = melloc (recv, 4* size of (double)); MPI - All gether (a, 1, MPI - Double, recvA, 1, MPI - Double, row comm)

-2 Each process has a 2x2 block of A and a 2x2 block of B, so the MPI_Allgathers need to send data of length 4, not 1.

MPI - Allgother (b, 1, MPI - Double, recvB, 1, MPI - Double, col comm)

2. Both NVIDIA GPUs and Intel Xeon Phi processors can be thought of as accelerators or coprocessors. Compare and contrast the two in terms of the programming model, execution model, and memory system. and contrest

compercisonais given below:



- (1) One single core of Keon phi processors REE more powerfull then AGPU
- 2) The number of comes is ground 72
- (3) Good for Multiple date Multiple instruction.
- (A) Here acess to 16 GB of high - band width memory,
- (5) Can store relatively large private memory
 - 6) 2D mech interconnect system 6 No such option Aveilable. helps to do foster communication
 - (7) Vectorization system available
 - (8) For moderate complex calculation in large number

NVIDIA GPUS

- A single Gpu thread is relatively less power full.
- 2) Thousands of cores available.
- 3) Good for Multiple data single.
- (a) No such option is evailable
- (5) Relatively low private memory,
- Tot evallable.
 - 8 Good for doing simple operation in large number.

(P.T.O)

- The code is not needed to be a changed very much.
- be run on xxeon-phi processor.
- (11) No seperate library is needed.
- (12) We can run the mpI, open_mpI codes

- The part we want to run in app is needed to be edited rigorously.
- (10) only the app a part of the code is ren on GPU.
- to use the GPU
- 12) The code must be written with CUDA commands
 NO MPI JOPEN MPI allowed.

MPI function C syntax:

- int MPI Comm size(MPI Comm comm, int *size)
- int MPI Comm rank(MPI Comm comm, int *rank)
- int MPI_Send(const 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(const void *buf, int count, MPI_Datatype datatype, int dest,
 int tag, MPI Comm comm, MPI_Request *request)
- int MPI_Sendrecv(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
 int dest, int sendtag, void *recvbuf, int recvcount,
 MPI_Datatype recvtype, int source, int recvtag,
 MPI Comm comm, MPI Status *status)
- int MPI_Sendrecv_replace(void *buf, int count, MPI_Datatype datatype,
 int dest, int sendtag, int source, int recvtag, MPI_Comm comm,
 MPI_Status *status)
- int MPI Wait(MPI Request *request, MPI Status *status)
- int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype,
 int root, MPI Comm comm)

- int MPI_Gather(const void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
- int MPI_Scatter(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
- int MPI_Cart_create(MPI_Comm comm_old, int ndims, const int dims[],
 const int periods[], int reorder, MPI_Comm *comm_cart)
- int MPI Cart sub(MPI Comm comm, const int remain dims[], MPI_Comm *comm_new)
- int MPI Cart coords(MPI Comm comm, int rank, int maxdims, int coords[])
- int MPI_Cart_rank(MPI_Comm comm, int coords[], int *rank)