Parallel Computing: Assignment 1

Sayed Ahmed, 6794063

Submitted To

Dr. Primula Thulasiram
Asst. Professor, CS Department
University of Manitoba
Winnipeg, Canada

October 2, 2003

1. Compile and run the sample programs
a. SendReceive.c
b. Broadcast.c

Find the output of the programs

1a. Node=4
command : mpirun -np 10 SendReceive.o

data of file:
5 10 15 20 25

Output:
Process 1 got 5 from 0
Process 1 got 10 from 0
Process 1 got 15 from 0
Process 1 got 20 from 0
Process 1 got 25 from 0
Process 1 got -1 from
Process 2 got 5 from 1
Process 2 got 10 from 1
Process 2 got 15 from 1
Process 2 got 20 from 1
Process 2 got 25 from 1
Process 2 got -1 from 1
Process 3 got 5 from 2
Process 3 got 10 from 2
Process 3 got 15 from 2
Process 3 got 20 from 2
Process 3 got 25 from 2
Process 3 got -1 from 2
Process 4 got 5 from 3
Process 4 got 10 from 3
Process 4 got 15 from 3
Process 4 got 20 from 3
Process 4 got 25 from 3
Process 4 got -1 from 3
Process 5 got 5 from 4
Process 5 got 10 from 4
Process 5 got 15 from 4
Process 5 got 20 from 4
Process 5 got 25 from 4
Process 5 got -1 from 4
Process 6 got 5 from 5
Process 6 got 10 from 5
Process 6 got 15 from 5

Process 6 got 20 from 5
Process 6 got 25 from 5
Process 6 got -1 from 5
Process 7 got 5 from 6
Process 7 got 10 from 6
Process 7 got 15 from 6
Process 7 got 20 from 6
Process 7 got 25 from 6
Process 7 got -1 from 6
Process 9 got 5 from 8
Process 9 got 10 from 8
Process 9 got 15 from 8
Process 9 got 20 from 8
Process 9 got 25 from 8
Process 9 got -1 from 8
Process 8 got 5 from 7
Process 8 got 10 from 7
Process 8 got 15 from 7
Process 8 got 20 from 7
Process 8 got 25 from 7
Process 8 got -1 from 7

Observations:

- All processes got all data from the file
- A process gets data just from its predecessor i.e whose rank is -1 from it's rank
- Also all process gets -1 from its previous process
- After getting -1 a process terminates
- Processes start and terminate sequence may vary
- A process gets data as order written in the file but this may not be always true
- A process sends data to it's successor process i.e. rank¿1
- 0th process only sends and the last process only receives others both send and receive

1b.Node=10
Command: mpirun -np 10 Broadcast.o

Output:
Process 0 got 5

Process 0 got 10
Process 0 got 15
Process 0 got 20
Process 0 got -1
Elapsed Time = 771 microsecond
Process 2 got 5
Process 2 got 10
Process 2 got 15
Process 2 got 20
Process 2 got -1
Process 8 got 5
Process 8 got 10
Process 8 got 15
Process 8 got 20
Process 8 got -1
Process 4 got 5
Process 4 got 10
Process 4 got 15
Process 4 got 20
Process 4 got -1
Process 5 got 5
Process 5 got 10
Process 5 got 15
Process 5 got 20
Process 5 got -1
Process 6 got 5
Process 6 got 10
Process 6 got 15
Process 6 got 20
Process 6 got -1
Process 3 got 5
Process 3 got 10
Process 3 got 15
Process 3 got 20
Process 3 got -1
Process 9 got 5
Process 9 got 10
Process 9 got 15
Process 9 got 20
Process 9 got -1
Process 7 got 5
Process 7 got 10
Process 7 got 15
Process 7 got 20
Process 7 got -1

Process 1 got 5
Process 1 got 10
Process 1 got 15
Process 1 got 20
Process 1 got -1

Observations:

- Here process 0 broadcasts data and every one receives that
- Elapsed time for process 0 is also shown
- mentionable process 0 sends data also to itself
- After receiving -1 a process terminates
- After termination of process 0 also after showing his elapsed time other processes may continue to receive data by running **1b.Broadcast1.c** we can observe it
- process creation and termination sequence varies
- A process gets data as the sequence of the file
- The program in example doesn't send the last file data

2a. Consider, add.c, that adds the numbers in an array. Let N denote the size of the array; P, the number of processors and Q, the number of processes. Do the following: Compile and run the program. Output the result.

Node=9
command: mpirun -np 5 add.o

Data in the file:
5 10 15 20 25 30 35 40 45 50

Output:
process 0 chunk 2 low 0 high 2
The result from process 0 is 15
process 2 chunk 2 low 4 high 6
The result from process 2 is 55
process 4 chunk 2 low 8 high 10
The result from process 4 is 95
process 1 chunk 2 low 2 high 4
The result from process 1 is 35
process 3 chunk 2 low 6 high 8
The result from process 3 is 75

Observations:

- No of Data=10, No of processes=5 so each process gets 2 data item
- Each process adds and gives the sum as output

Another Run:

Node=9, command: mpirun -np 3 add.o

Data in the file:
5 10 15 20 25 30 35 40 45 50

Output:
process 1 chunk 3 low 3 high 6
The result from process 1 is 75
process 0 chunk 3 low 0 high 3
The result from process 0 is 30
process 2 chunk 3 low 6 high 9
The result from process 2 is 120

Observations:

- No of Data=10 No of processes=3 so each process gets 3 data item
- Each process adds their 3 data and gives their sum as output
- The last item has no contribution in addition as 10/3=3.333 so chunk=3
- We could write ceil(10/3)=4 then First two could get 4 the last could get 2 data item

2b. Modify the program so that data is generated by a random number generator and store it in the array, of size N, provided by the user

For program please see the file **2b.randomaddarg.pdf** or **2b.randomaddarg.c**

Note:

- No of input is passed as command line parameter
- In original add.c if (dataitem%no of process)!=0 then last item/items was/were ignored but here this will not happen

- Here chunk=(index/num_procs)+1 if index%num_procs!=0. so no data miss
- To create random number srand() and random() functions were used

Node=9
Command: mpirun -np 3 randomaddarg.o 10

Output:
Random Datas are as follows
index=0 Data=40
index=1 Data=1
index=2 Data=79
index=3 Data=84
index=4 Data=56
index=5 Data=23 index=6 Data=5
index=7 Data=70
index=8 Data=81
index=9 Data=71
process 1 chunk 4 :::low 4 :::high 7
The result from process 1 is 154
process 0 chunk 4 :::low 0 :::high 3
The result from process 0 is 204
process 2 chunk 4 :::low 8 :::high 9
The result from process 2 is 152

node=9
command: mpirun -np 5 randomaddarg.o 10

Output:
Random Datas are as follows
index=0 Data=40
index=1 Data=1
index=2 Data=79
index=3 Data=84
index=4 Data=56
index=5 Data=23
index=6 Data=5
index=7 Data=70
index=8 Data=81
index=9 Data=71
process 0 chunk 2 :::low 0 :::high 1
The result from process 0 is 41

process 2 chunk 2 :::low 4 :::high 5
The result from process 2 is 79
process 4 chunk 2 :::low 8 :::high 9
The result from process 4 is 152
process 3 chunk 2 :::low 6 :::high 7
The result from process 3 is 75
process 1 chunk 2 :::low 2 :::high 3
The result from process 1 is 163

2c. Modify the program to find the maximum element in the array and output the result

For the program please see the file **2c.maximum.pdf** or **2c.maximum.c**

Note:

- We broadcast all data to all processes
- But a process finds the maximum of his local chunk of data
- Then we use MPI_Reduce to find the maximum of all these local maximums

node=9
command: mpirun -np 3 maximum.o

Output:
Datas are as follows
5 10 15 20 25 30 35 40 45 50

process 1 chunk 4 low 4 high 7
process 2 chunk 4 low 8 high 9
process 0 chunk 4 low 0 high 3
max=50

Node=9
command: mpirun -np 5 maximum.o

Output:
Datas are as follows
5 10 15 20 25 30 35 40 45 50

process 1 chunk 2 low 2 high 3
process 4 chunk 2 low 8 high 9
process 3 chunk 2 low 6 high 7
process 2 chunk 2 low 4 high 5
process 0 chunk 2 low 0 high 1
max=50

2d. Now, perform the following experiments: i.Compile and run the program for N =1024 and vary, P (P = 2, 4 and 8). Get the timing for these processors. What is the relation between P and Q?

For the related data please see **2di.relation2di.pdf**

i.Relation:

- To find the relation we observed the total run time of all processes related program is **2d.addtiming.c** or **2d.addtiming.pdf**
- So as the no of node(P) increases processing time reduces in significant amount if Q is fixed specially when Q is high
- Q=50 or 30 then P increased from 2 to 8 processing time reduced in great amount
- Q fixed at 10 processing time reduces with the increase of P though not much specially when P=4 or P=8 the processing time is similar(Q=10)
- Q increases processing time also increases for fixed P

2d.ii. for add.c, Let N = 1024, P = 4 and vary Q(Q = 1, 2, 4, 8,16). Get the timings. What is the relation between P and Q?

For the related data see the file **2dii.relation2dii.pdf** also program **2d.addtiming.c**

Relation:

- Here P and N fixed so as Q increases Processing time is increased
- Q=2 it needs two times processing time than Q=1
- Q=3 it needs near two times processing time than Q=2
- Q=4 it needs near three times processing time than Q=2
- Q=8 it needs two times processing time than Q=4
- Q=16 it needs near three times processing time than Q=8
- i.e processing time is around 2 times if no of processes increase to double

2d.iii: for add.c Fix P = 4, Q = 4 and vary the data size N (N=512,1024,2048). Get the timing results

For the related data see the file **2diii.relation2diii.pdf** also program **2d.addtiming.c**

Relation:

- here P and Q fixed so as N increases Processing time also increases
- when N=512 it needs around 5500 micro sec to terminate all the tasks
- when N=1024 it needs around 8500 micro sec just 3000 micro sec more than N=512
- when N=2048 it needs around 11500 micro sec just 3000 micro sec more than N=1024

3.Modify the program, add.c, as follows:
a)Distribute the process's portion of data by using the scatter operation.
b)Compute the sum of the numbers and find the maximum number and output the results

For the program please see **3ab.summaximum.c** or **3ab.summaximum.pdf**
Command: mpirun -np 3 summaximum.o

Output:
rank=1 rdata=25
rank=1 rdata=30
rank=1 rdata=35
rank=1 rdata=40
Rank=1 SUM=130
rank=2 rdata=45
rank=2 rdata=50
rank=2 rdata=0
rank=2 rdata=0
Rank=2 SUM=95
rank=0 rdata=5
rank=0 rdata=10
rank=0 rdata=15
rank=0 rdata=20
Rank=0 SUM=50
MAXIMUM=130

Command: mpirun -np 5 summaximum.o
rank=1 rdata=15

rank=1 rdata=20
Rank=1 SUM=35
rank=3 rdata=35
rank=3 rdata=40
Rank=3 SUM=75
rank=2 rdata=25
rank=2 rdata=30
Rank=2 SUM=55
rank=4 rdata=45
rank=4 rdata=50
Rank=4 SUM=95
rank=0 rdata=5
rank=0 rdata=10
Rank=0 SUM=15
MAXIMUM=95

c. Perform the same experiments as described above in question 2d

i.Relation:
For the related data please see the file **3ci.summaxrelation1.pdf** also the file **3ci.summaximumtiming.c**

- N and Q Fixed so if P increases processing time reduces
- P=2 for Q=64 total process time around 4017687 but P=8 then Processing time=1577978. so greatly reduced
- For a Fixed P with the increase of Q processing time increases for P=4 if Q=32 then time=446565 if Q=64 then time =2638937 so drastically increased
- P,N fixed then it may happen for a number of Q the system may not work for N=1024, P=4 then if Q=200 we get no output due to resource limitation

ii.Relation:
For the related data please see **3cii.summaxrelation2.pdf** also the file **3cii.summaximumtiming.c**

- Now P,N fixed so with increase of Q processing time increase
- This increase is in a significant amount if Q increases double then total processing time increase Double/Triple like that
- Q=2 time=3300,Q=4 time=7000, Q=8 time=21,000
- P=4 Q=180 works Q=200 doesn't work, P=2 Q=80 works but Q=90 doesn't work, P=8 Q=360 works but Q=370 doesn't work, Resource limitation

iii.Relation:

For the related data please see **3ciii.summaxrelation3.pdf** also the file **3ciii.summaximumtiming.c**

- P,Q fixed so with increase of Data amount processing time also increases significantly
- But this increase rate is not as significant like the previous 2
- data=400 time=4300, data=800 time=6100, data=2000 time=9000

3d. What is the difference between the modified add.c and original add.c in terms of performance? Which is more efficient?

- In the original add.c we broadcasted data but in modified program we scattered data from process 0
- Modified program is more efficient because here the amount of data passed to all processes is a fraction but not all
- Process to process data flow less for modified version
- Also storage required is less for modified version
- Modified version is more efficient

4. Compute the matrix multiplication of two mxm matrices. Use the random number generator to generate a 16x16 matrix. Assume the numbers are integers a)Create m2 processes. Experiment with different number of processors (P = 2,4,8). Show the output of the resulting matrix.

For the program please see **4a.matrixfinal.c** or **4a.matrixfinal.pdf**
command: mpirun -np 9 matrixfinal.o 3
Output:
A array row wise ***** B Array Column wise

14 ***** 2

13 ***** 11

9 ***** 11

2 ***** 13

10 ***** 2

3 ***** 8

6 ***** 5

9 ***** 2

1 ***** 7
rank=0 row=0 col=0 component=270
rank=2 row=0 col=2 component=159
rank=1 row=0 col=1 component=280
rank=3 row=1 col=0 component=147
rank=4 row=1 col=1 component=70
rank=7 row=2 col=1 component=104
rank=8 row=2 col=2 component=55
rank=5 row=1 col=2 component=51
rank=6 row=2 col=0 component=122

Observations

- With increased no of P process runtime reduces specially when no of process is much
- In our lab as many P, as greater process can work due to availability of more resources
- The program when P=4 could compute 13*13 matrix but failed to compute 14*14 matrix due to resource limitation
- The program when P=8 could compute 18*18 matrix but failed to compute 19*19 matrix due to resource limitation
- So P increases no of processes support becomes higher

4b.Is this an efficient way of partitioning the matrices? Can you think of any other way? If so, explain your algorithm. Code it, run and explain the results

No this(4a) is not a efficient way of partitioning the matrix because:

- It broadcasted the 2 complete input matrixes to all the processes
- But a process works only with a single row and column
- It would be better if we could send only the needed data with minimum interprocess communication
- Besides it needs much more ram storage per process that could be used by other programs or by other purposes

For the code please see the file **4b.matrixalternate.pdf** or **4b.matrixalternate.c**

Alternative Algorithm:

1. Send only one row from first matrix and only one column from the second matrix to the process who will work with it
   Pseudo Code:
   for (i=0 to m*m)
   begin
   //i treated as the rank of the process
   //m=dimension of the process
   //r=row of 1st matrix to send c=col of second matrix to send
   r=i/m
   c=i

   MPI_Send(A[r],m,MPI_INT,i,0,MPI_COMM_WORLD)
   MPI_Send(B[c],m,MPI_INT,i,1,MPI_COMM_WORLD)

   end

2. A process multiplies the row with the column and adds the results to give output
   Pseudocode:
   for(i=0 to m)
   begin
   sum=sum+rA[i]*rB[i]
   end

3. It is assumed that for A*B, A is in row wise and B is in column wise i.o 0th row of B in our array = 0th column of B practically

Output:

A array row wise ********** B array colwise shown
2*2 Array
4 ******************************7
8 ******************************16
4 ******************************16
7 ******************************13
rank=0 row=0 col=0 component=156
rank=2 row=1 col=0 component=140
rank=3 row=1 col=1 component=155
rank=1 row=0 col =1 component=168

Performance Comparison:
for the data please see the appendix file "4bcomparison.pdf" and "4bmatrixalternatetiming.c"

- According to the timing measurement it is seen that 4a algorithm takes much lower time
- For 10*10 4a takes 3000/4000 microsecs but 4b takes about 10,000 microseconds
- Because 4b needs much interprocess communication to send data
- Though 4b's data partitioning is better as it sends only necessary data to the appropriate process
- But where the amount of computation is huge i.e for very large matrixes also where communication latency ,propagation delay in the network is negligible then 4b may be more efficient

5a. What is fine-grain parallelism and coarse-grain parallelism?

Processes and granularity:
In case of parallel computing a program is first divided into several tasks or processes to run them simultaneously for faster operation and better efficiency. The size of a process is described by its granularity. Like

Fine-grain:
Here the process size is too small usually contains a few instructions even can contain one instruction.

Medium-grain:
Medium granularity describes the middle ground between fine-grain and coarse grain.

Coarse-grain:
Here the process size is much big and usually consists of a large number of instructions that run simultaneously and takes a substantial amount of time to execute. Notable, sometimes the size of the computation between communication or synchronization points is termed as granularity. We increase granularity to avoid cost of process creation and interprocess communication but it reduces the number of parallel processes so also reduces the amount of parallelism. A program with variable granularity is expected because it will be scalable.

5b. What is grid computing?

Grid Computing:
Grid is one kind of parallel and distributed computing infrastructure.It is built on the Internet and the World Wide Web that enables dynamic sharing, selection and aggregation of geographically distributed resources. This sharing depends on the availability , performance, cost, capability and users' qos requirements. Grid provides a way for discovering and negotiating access to remote resources. It's scalable , secure and highly efficient. The Grid makes it possible for scientific collaborations like share resources and let geographically distributed research groups to work together

in ways that were previously impossible. Grids share and aggregate distributed computational capabilities and deliver them as service.
In a Grid, each node has its own resource manager and doesn't aim for providing a single system view that clusters do. Grid tools are concerned with resource discovery, data management, scheduling of computation, security, and so forth.

5c. What is P2P computing?

In case of client/server model data from client is moved through a server totally controlled by the servers but with P2P server is removed. So the PCs/clients can communicate directly to share files and other data directly. P2P communicates directly using virtual namespaces, it defines a set of computing nodes that treat each other as peers and supply processing power, content or applications to other nodes in a distributed manner, with no assumption of a hierarchy of control/server.

P2P allows decentralized application design, here each peer, independent of software and hardware platforms, can benefit and profit from being connected to millions of other peers.

Here, clients and servers have a horizontal relationship rather than vertical relationship of traditional networks, giving the whole peer group tremendous processing power and storage space. Mentionable P2P computing needs to overcome some complex issues such as security, network bandwidth, and architecture designs as yet they are not so much developed.

How it works?
The P2P model is like communal living. We can imagine a group of people who are living together and sharing certain belongings. They have one public cabinet where they keep everything they intend to share with others But they can also maintain a locked cabinet where they store their private belongings. P2P works like this. One decides which files/folders/programs/service from his hard drive he wants to share it others and makes it public also he can use the public property of others.

P2P Application:

- It should be able to locate other peers in the network
- After locating, it should be able to communicate with them using messages
- After the communication is established with other peers, the application should be able to receive and provide information, such as content

P2P and World Wide Web:

They both provide support for publishing and discovery across networks. P2P computing is decentralized and it's main aim is to supply processing power, content, or applications to peers in a distributed fashion and it is less concerned of semantics of messaging formats and communication protocols. Where World wide web is Centralized and primarily based on standard messaging formats and communication protocols.

5d. Where do you think the future is heading and why?

In future The scenario may happen like follows:

- Clusters like Beowulf will be dominant architecture
- GRID and P2P technology will emerge as high performance computing platform.They provide significant advamtage for parallel problems and also sharing. The computational Grid can go far beyond the computing power of any current supercomputer
- Because Internet and LAN speed will be much like multigigabit/s. So a LAN can be used as a high performance parallel computing platform. Also in WAN it will be possible to compute with a sufficient high speed. Because then message passing latency will be too low for both LAN and WAN. So then all LAN based PCs will act like Beowulf and together they will create the GRID
- Beowulf and like products are also to emerge. Building a strongly coupled high performance supercomputer costs much. The software plus hardware costing plus operating costing also significant. Beowulf's economies is a key reason for its emerge because hardware/software is much less expensive Besides its Sociology is also important because users have access to both generic and profession specific program. Besides it's a convergent architecture that runs over multiple computer generations and hence protects application investment.
- For BEOWULF like computers it will happen that a great number of processors will be found in a single chip that highly reduce the interprocessor communication latency also provides good locality. Chips with multiple processor of combined speed of 20 Gflops is available now. With the continued advances in electronics and VLSI fabrication this is to happen. These chips will be used to create the nodes of a cluster
- Also for shared memory systems huge cache only memory architectures might emerge. A processor in memory or multisystem on a chip with good locality may also be a part of high performance computing

There is a little possibility to happen a revolutionary change in near future because there are no laboratory research results for near term revolution. So it is more likely that today's most efficient technologies will remain in near future with some enhancements.

6. Go to top500.org. Indicate the top 5 supercomputers with the flops ratings and the architecture (shared memory, distributed memory, etc.)

Top 1:
Model: The Earth Simulator (ES)
By: A project of Japanese agencies NASDA , JAERI and JAMSTEC for climate modeling.
Flops: 40 Tflops
Architecture:

- Inside a node shared memory so openmp is used inside a node

- But distributed memory as a whole so works with message passing from node to node
- An optimized code will need to employ MPI-2 at the subdomain level, OpenMP at the loop level, and vectorization directives at the instruction level all at once

Top 2:
Model: ASCI Q at LANL
By: A collaboration between the U. S. Department of Energy's National Nuclear Security Administration and the Sandia, Lawrence Livermore, and Los Alamos national laboratories.
Flops: 13.88 Tflops
Architecture: Shared memory

Top 3:
Model: MCR Linux Cluster
By: LLNL, Livermore
Flops: 11.2 Tflops Linux cluster
Architecture: Distributed Memory

Top 4:
Model: ASCI White, IBM SP Power3
By: LLNL, Livermore
Flops: 7.634 Tflops
Architecture: Shared Memory

Top 5:
Model: Seaborg, IBM SP Power3
By: NERSC/LBNL, Berkeley
Flops: 7.304 Tflops
Architecture: Distributed Memory

7. Write in your own words, a paragraph on the kind of research conducted at SETI?

A very interesting research has been and is being conducted at SETI. The project provides a very interesting distributed computing over the Internet whose processing power is 13 Tflops which exceeds the combined processing power of the top three of the top500 supercomputers at that time and nearly equivalent to the Top2 computer of 2001 Top500 list. Mentionable this research does not belong to the research program of SETI institute itself but it is a part of the project SERENDIP and uses data collected with the Arecibo Radio Telescope by SERENDIP in Puerto Rico. The telescope downloads 35 GB of data everyday. The research project uses thousands of Internet connected PCs to search for extraterrestrial intelligence and provides a way to use the unused processing powers of personal computers connected to the Internet. The interested person to participate in SETI@HOME project should first download a software freely available at SETI website. When a user is idle for a somewhat long time the software begins to work and downloads 300 KBs chunk of data The users are also able to see the data on their own screen. User's PC Analyzes the data and sends the result to the SERENDIP team. The SERENDIP team uses this

data along with all the other participant's data to search for extraterrestrial signals. This system has some drawbacks like it analyzes only the 2.5 MHz piece of the observed spectrum. And as the data processing does not occur in real time so interesting signals are must be followed up at a later date/time. The advantages are it allows looking for a variety of signal types that current SERENDIP processing cannot yet uncover. Undoubtly SETI participants will increase more and more folks will join

8. Write a one page report on the article "Why is PVM and MPI so different"? Has there been any improvements in MPI and PVM in the recent years? Explain.

This paper mainly adverts to some key differences and convergences between MPI and PVM standard mainly in implementation as well as in specification. The topics include areas like implementation , group concept, inter process communication, inter group communication, dynamic process, message transfer, deadlock etc. Also it discusses some recent improvements and convergences.
Some of the main goals of MPI are implementation as a libray, thread support, scalability, modularity, extensibility, heterogeneous computing support, deadlock prevention etc.
MPI and PVM both supports error handling and recovery.
MPI sources and destinations are relative to a group but in PVM they are always absolute in terms of the task ids. Both MPI and PVM provide way to connect parallel programs where both programs must belong to the same PVM virtual Machine to keep task ids unique. Also different PVMs can not be merged into a single PVM for the same reason.
In PVM pvm_reg_tasker routine allows to start tasks. A powerful hook to allow extension by special applications like debugger servers though Some PVM implementations for MPPs do not provide this support. Standard MPI does not but MPICH have this service used by the totalview debugger. Both MPI-2 and PVM support creation and attachment to processes but there is a great difference in the handling of resource information used to determine where to create the new process. This is also the difference how they support operating in distributed operating system. PVM calls pvm_config to collect information then pvm_spawn to explicitly create the process but between there may happen some other calls(pvm_delhosts) by other routines(so info by pvm_config becomes wrong) and the new process creation may fall into a problem. But in MPI_Comm_spawn call simultaneously see for information also creates process if possible simultaneously to avoid race condition. MPI_Comm_spawn_multiple, MPI context are also uses same strategy to avoid race to create multiple processes and to create context respectively. But pvm_newcontext may fall in a race. Also to avoid race in MPI multiple parents are supported in group communication where a parent is a must.
MPI supports two way for reliable programming. Buffered send with guaranteed amount of buffering and non blocking operations. So correct choosing and using are able to avoid race. Where PVM pvm_psend and pvm_precv may fall in race if data size is too large.
MPI-2 has developed remote memory operations like put, get, accumulate compatible in heterogeneous environment using default MPI data type and a new MPI object MPI_Win. Introduces a number of ways to synchronize access to the shared data areas. MPI-2 supports for the BSP. PVM provides no such functionality

Parallel I/O operation is highly supported in MPI-2 usable in heterogeneous environment also choosing among different forms optimized for a particular system is possible. But in PVM no integrated parallel I/O capability.

IMPROVEMENTS:

Dynamic Processes in MPICH. Static groups and Message Contexts in PVM besides the features discussed with MPICH and MPI-2 is also improvement over standard MPI

Appendix

Program Codes and Data Files: