Evaluation of Virtual and Physical Parallelism Environments

Mustapha M. Baua'a[1], Jehan K. Shareef [2], Aqeel M. Hamad[3]
[1]South Oil Company Dhi-qar Field
Financial Dept., Nasiriya, Iraq
[2]Dhi-Qar University
College of Digital Media Faculty, Iraq
[3]Dhi-Qar University
Head of Computer Center, Iraq
mustafa.mohamed46@yahoo.com, jihansh@yahoo.com, aqeel_mh80@yahoo.com

ABSTRACT: *Parallelism is the concept that associates with participating more than one computer, process, or thread in one problem execution. The concept is used for increasing multitasks performance execution effectively. Many algorithms, techniques, architectures, frameworks, and environments are found to enhance this concept and finally enhance the performance. However, two basic environments are exists to execute parallel tasks; these two environments are Virtual and Physical parallelism. Both of these two runtime environments have their features, advantages and disadvantages. The paper focuses on analyze, evaluate and clarify some differences between the two environments. Results depend on real case study executed via Java-Thread, Open-MP, MPI in virtual environment and an MPI in physical environment. Implementation of the case study and validation results are shown and they clearly clarify the advantages, disadvantages and limits of using any environment of them.*

Keywords: Parallelism in Virtual Environment, Parallelism in Physical Env ironment, JavaThreads, OpenMP, MPI

Received: 13 March 2015, Revised 19 April 2015, Accepted 25 April 2015

1. Introduction

The technology of parallelism is mean participating more than one processor in executing one program in order to gain high performance execution. For instance, certain program may takes 30 minutes when it sequentially executed, but with parallelism perhaps take half of that time according to the number of participated processors. However, many technologies were attempted to support parallelism for that some of them trying to introduce new parallel compiler, other introduce new parallel language library like Open-MP, Java-Thread, MPI, etc. Some of these technologies carry out their programs only in virtual environment and other in both physical and virtual environment like MPI.

The term virtual has meaning of dividing a process into more than one thread during execution. Where, each thread has its own copy of the same program and its space of memory that can enable it to run its own copy independently. The basic benefits of virtual technologies are manageability, reliability, and usability. Also, the virtual methodology was established in order to

maximize the utilization of hardware resources. In which its trying to utilize the available resources efficiently due to the facilities it introduced such as [1] ease of maintenance, lowering hardware costs, and other benefits. In addition it reduces the time needed for communication between processes connected to each other physically. Furthermore, virtualization works with shared memory model in order to maximize memory utilization.

However, virtualization has a disadvantage of, for instance, when two processes trying to access the same memory address space in this case developer needs to apply synchronization. And if the number of processes works inside the virtual environment high then they will cause performance degradation (e.g. the overload on the whole system will be very high). Also, virtualization not all time leads to good program structured and high performance, but the unexpected behavior approves in some situations the disadvantage of virtualization.

There are many implementations that support virtual parallelism. Open-MP, Java-Thread, and MPI each of them have its own technique to implement parallel virtualization. Compiler directive is used in Open-MP, while, Java-Thread extends class Thread in order to implement virtual parallelism, whilst, message-passing model used with MPI between communicated processes [2]. MPI can works with two kinds of parallelism environments where it can be implemented by virtual and physical parallelism environments.

In addition to all features mentioned above, parallelism now and with today substantial advances in technology it is implemented by two methods, one is virtual parallelism and second is physical parallelism. Each method has its advantages and disadvantages in term of performance, flexibility, reliability, communication time, processes interleaved, processes dependent, and other challenges may face in parallelism implementation.

While the implicit mean of physical parallelism is executing a parallel program on multi-computer physically. Each computer or processor has its own memory and its own resources, where these computers connected to each other across LAN network, WAN network, or other kinds of networks. Each computer partially participates in a problem solving according to the algorithm that is used to divide the problem among participators. The type of computers, their speeds, their versions, and the topology of computers arranged with have the potential impacts on the performance of the cluster network. Although, MPI addresses the computer topology (across using the MPI function MPI_CART_CREATE) still this challenge have impact on the performance. The challenge comes from the available network topologies. However, physical parallelism exposes many challenges such as time spent for communication between processes, costs of buying more new hardware, flexibility, manageability, and others. In addition to these challenges one may think when enterprise increases the number of computers; the execution performance time for parallel program will decrease. For the first glance, increasing the number of computers in enterprise is good for availability of the system to solve many problems at one time. But increasing the number of computers to participate in solve one problem will cause performance degradation. Because this increasing will result in interleaved processes, increase time communication, waiting time... Etc. And according to Amdahl's law every parallel program have sequential and parallel fraction, if the sequential fraction increase, the behavior of the program performance will be affected.

$$Speedup = \frac{1}{\frac{P}{N} + S}$$

Where $\boldsymbol{P}$ represents the parallel fraction, $\boldsymbol{N}$ represents the number of processes, and $\boldsymbol{S}$ the sequential fraction. As one can see with increasing the sequential part, speed up will be effected. For example, if $\boldsymbol{S} = 0$, $\boldsymbol{P} = 1$, $\boldsymbol{N} = 2$, then speedup will doubled, and if $\boldsymbol{S} = 0.5$, $\boldsymbol{P} = 0.5$, $\boldsymbol{N} = 2$, then speedup equal to 0.666. Then according to Amdahl's law adding more computers not all time yielding to more waiting time.

2. Virtual Parallelism

Virtualization is a high level of covering the implicit implementation details. While virtual parallelism is established to work with shared memory in terms of threads, where each thread is executed in its own space address inside a process. This is what happens with Open-MP, Java-Threads, and MPI when they are execute their programs on single computer. Each of these technologies executes their programs on single computer and use threads technology in order to parallelize the programs [3]. At the start of program execution the master thread create a team of threads called worker threads. This team of threads in virtual environment runs the tasks requested in parallelism. The execution of tasks happened on one computer and all threads works

inside the space of the master thread.

There are many technologies that support virtual parallelism such as Open-MP, Java-Threads, and MPI. Open-MP is a new novelty industry standard for shared memory model, in which it implements a fork\join model. Open-MP is a technology that can extend the sequential program written in C\C++, or FORTRAN into parallel program through using compiler directives. This technology can achieve very significant results for shared memory parallelism [4]. In addition, Open-MP apply data parallelism model and their threads are more autonomous; ideally they are use course-grain parallelism. Also, it can deal with Multiple-Instruction Multiple Data computation. However, an Open-MP program suffers from less scalable, and less portable in comparison with MPI programs [5].

Threads [6] are semi-processes executed inside a process where the process creates a team of threads at the start of the program execution. Multi-threading model was established in order to get full benefit of available resources. Also, in order to use memory in efficient way. Java provides a class Thread in order to implement multi-threading model. However, this model works with virtual parallelism or concurrent processes. In this multi-threading model developer have opportunity of increasing the number of threads till limits of Java Thread class but still this model suffer from unexpected behavior of threads. Threads have ability of starting new threads during execution but this ability bounded by some limits such as process memory limit, limits of thread functionality, etc.

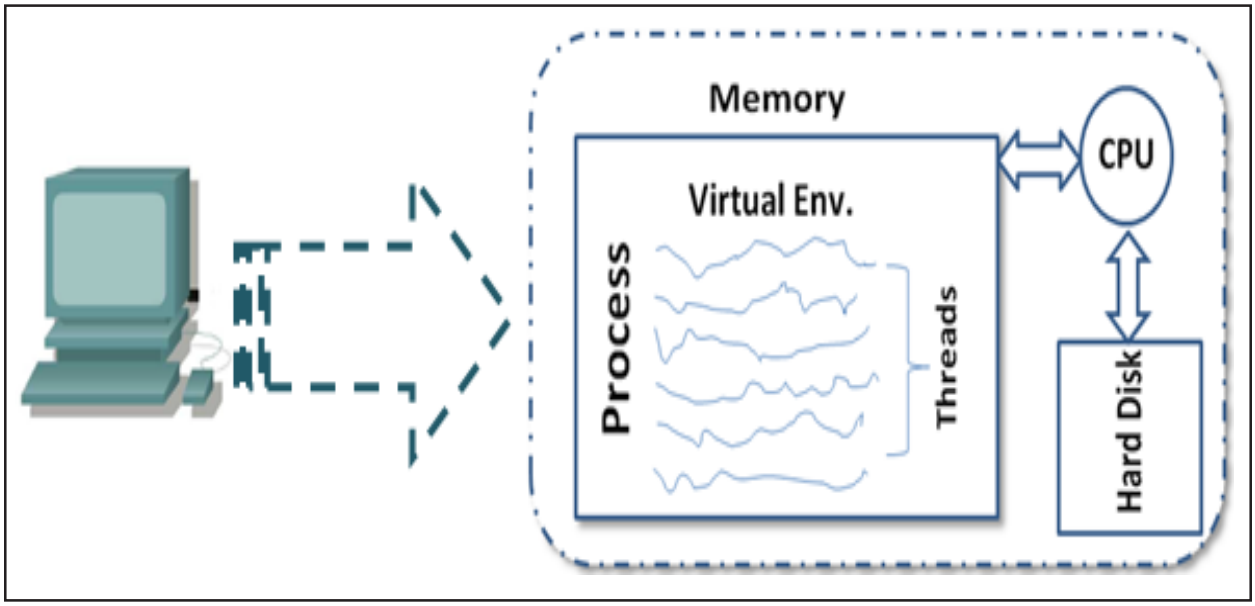


Figure 1. Virtual Environment

3. Physical Parallelism

As mentioned above parallelism can be executed in virtual and physical environment and even virtual parallelism saves hardware costs, but still physical parallelism can do better performance. Physical parallelism achieves good performance when it implement by full independent processes. There are many implementations that apply parallelism physically such as MPI. In MPI the program at the beginning of execution distribute the tasks on the processes participates in problem solving. In physical parallelism each process has its own hardware resources because processes running on many computers physically. In a few details in physical parallelism the program create at the start of execution a team of processes that are executed on multicomputer connected to each other physically. Where, in MPI for example the number of processes is applied before execution. Also if MPI-2 is used the number of processes can be changed during execution.

Many technologies are supporting physical parallelism like MPI, PVM, and others. Message-Passing Interface is a de facto standard that established to switch from sequential programs to parallel programs. MPI is a set of methods that are combined in a certain way in order to construct parallelism environment. It contains many features such as Communicator, Point-to-Point communication, Collective communications, and others. Also it exposes set of challenges that can be addressed like flexibility, reliability, time communication … etc. [7].

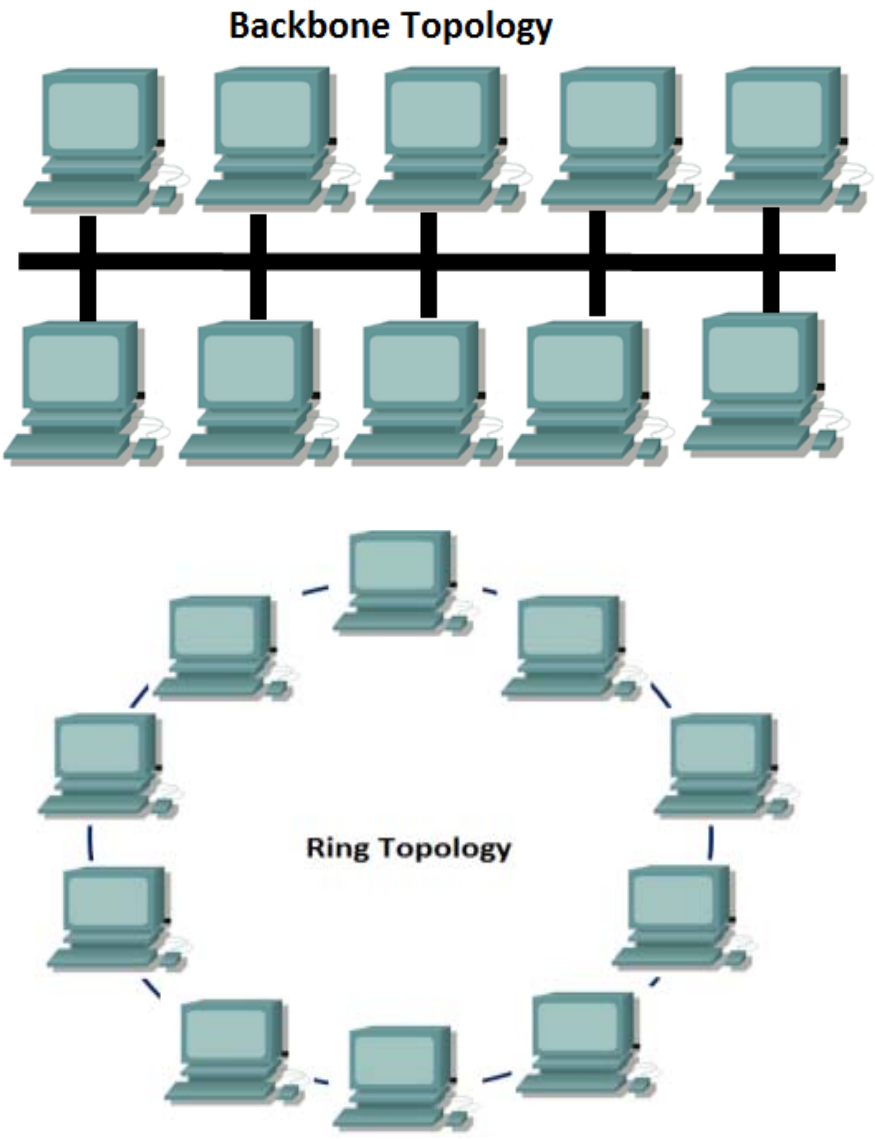


Figure 2. Physical environment

4. Case Study

The paper case study is about execution a program in virtual and physical environment in a parallel way. The program involves a method of computing square and a matrix contains the values needs to be computed. Then these values divided on available processes or threads where each process responsible for computing part of them according to the following:

```
Matrix[n]
iStart = (ProcessID*n)/Nprocs
iEnd = (ProcessID+1)*n/Nprocs
for(i=iStart; i< iEnd;i++)
matrix [i] = Sqr(matrix[i]);
```

Where matrix represent the values needed to be squares, iStart, iEnd represent the limits of space work for each process on matrix. In this case study we are trying to clarify the differences between performance of virtual and physical parallel environment. The intended of this algorithm is to implement it by the technologies of virtual and physical parallelism.

5. Implementation and Results

Two tables below shows the performance of implementations with different technologies (e.g. Java-Threads, OpenMP, and MPI). The implementation consists of two parts one is executed with variable number of processes and fixed number of operations. Second is running with fixed number of processes and variable number of operations.

Operations	Performance table with fixed processes and Variable No. of Operations in µsec				
	Processes	*Open-MP*	*Java-Thread*	*V_MPI*	*P_MPI*
10000	3	2	1	0.000119	0.000174
20000	3	2	1	0.000126	0.000233
30000	3	2	1	0.000135	0.000257
40000	3	4	1	0.000135	0.000233
50000	3	4	2	0.000157	0.000283

Table 1. Performance table part_1

Operations	Performance table with Variable processes and fixed No. of Operations in µsec				
	Processes	*Open-MP*	*Java-Thread*	*V_MPI*	*P_MPI*
10000	3	5	1	0.000108	0.000174
10000	6	7	2	0.000175	0.000348
10000	9	8	2	0.000217	0.000522
10000	12	31	5	0.000301	0.000696
10000	15	84	7	0.000386	0.00087

Table 2. Performance table part_2

6. Results Analysis

Table 1 and 2 are clearly clarifies the results of our implementations of the case study in section 4. For results in table part_1 the Laptop is used with Intel ® Core (MT) i5-2320M CPU @ 2.60 GHz, RAM 6.00 GB usable, and 64-bit Operating System x64-based processor. The Laptop used as virtual environment in order to execute the implementation of OpenMP, Java Thread, and MPI.

For last columns of Physical MPI implementation execution result, we set up a real multicomputer network Lab in Thi-Qar University\Computer and Mathematical Science College in Nasiriya, Iraq. These computers have been used to carry out the implementation and for different number of operations. The computers have the same features where, CPU speed is 2.80 GHz; RAM 1.00GB, and 32-bit Operation System. The network of MPICH2 was configured and installed by Mustapha M. Baua'a.
In Table 1 Part_1 the number of processes was fixed and number of operations or requests variable. The last two columns explain the important of MPI performance in both two cases virtual and physical executions. MPI in compared with OpenMP and Java Thread do better performance especially virtual implementation because it executed on the same environment. While the Physical MPI is also do better performance but it executed on another environment as we explain above.

MPI as shown in our experiment do better in virtual environment and this perhaps because there is no time communication and

also the CPU of Laptop have higher speed than those in Cluster network. According to our experiment the virtual parallelism environment may use for small data processing and small number of processes participating in problem solving. While, the physical parallelism environment is better to be used with those applications having big data processing because time communication will become not big deal in comparison with processing time.

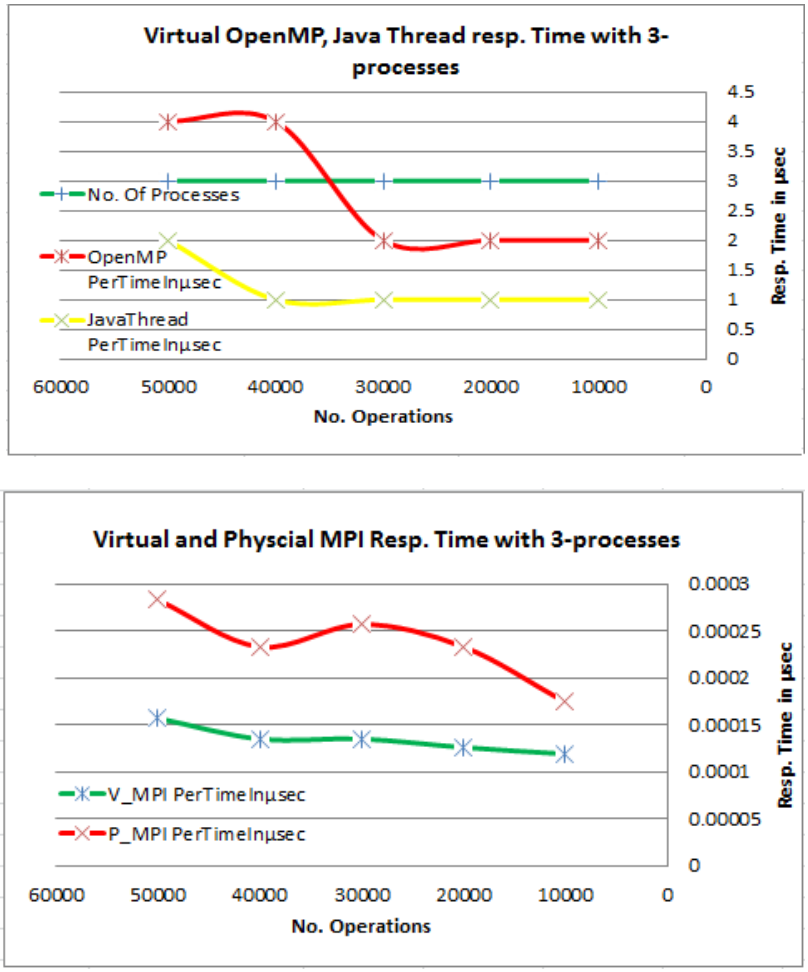
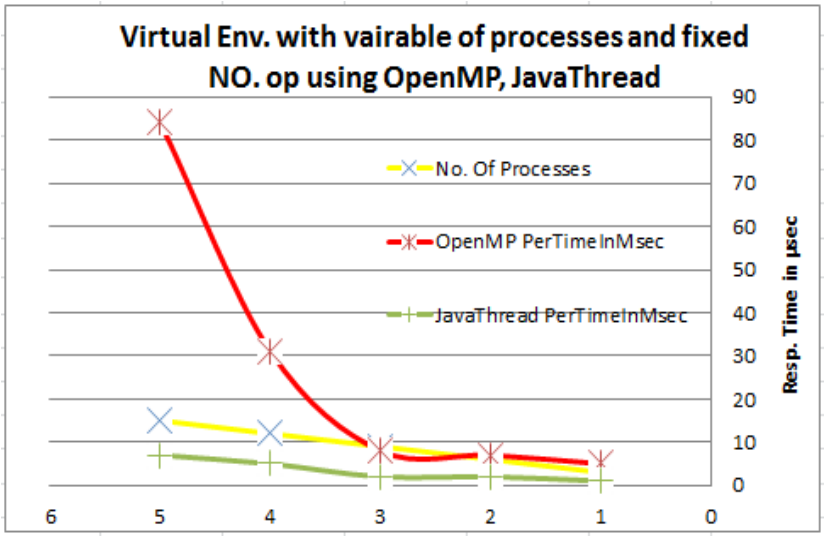


Figure 3. Fixed number of processes and Variable number of requests



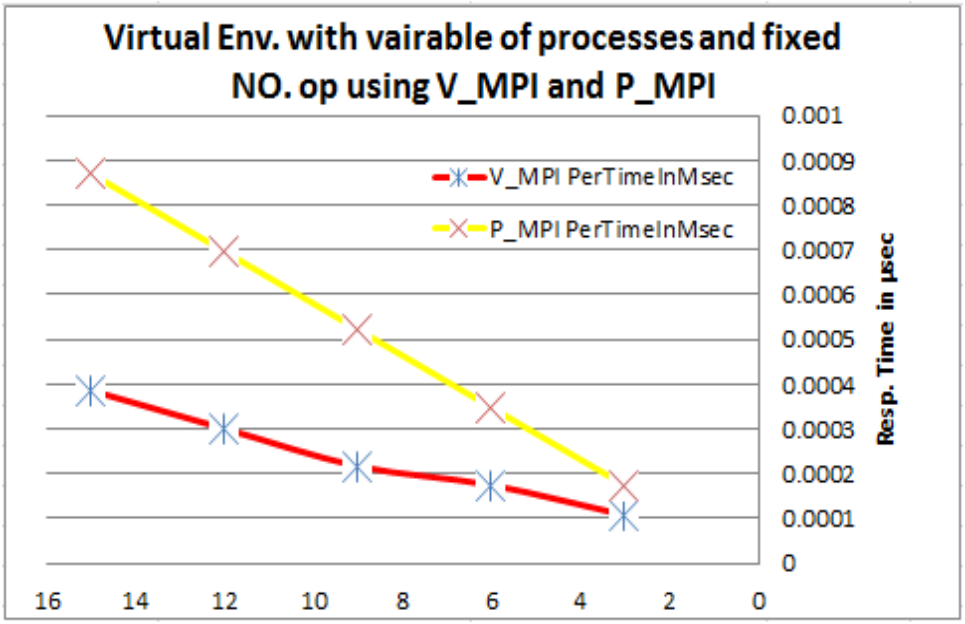


Figure 4. Variable number of processes and fixed number of requests

The basic problem of MPI programs implemented virtually are the manageability and the time needed by each thread to wait its turn. Also the leak of resources will be a problem for those MPI applications that need huge processing time. In addition, there is an unbalancing problem for loads will appear due to the limits of resources available. Whilst, the MPI programs implemented physically have a problem with time spending on communication, the different speeds of computers used in problem solving, the topology computers arranged with, etc.

7. Conclusion

Two environments that have been used to implement parallelism are introduced. These two environments are virtual and physical environments. Also the technologies of using these environments have been analyzed. Case study and its implementation are represented. The implementation results explain some differences between two different environments for parallelism implementation. At appendix A the paper represents some significant points between the two environments in order to clarify advantages, disadvantages, and limits for each one of the environments.

In future work we will try to apply our evaluation on big algorithm (like rendering or other algorithms) where it needs huge computation. Also, we will work on SOA in order to parallelize it by using new technique of parallelization.

References

[1] Rich Ubling., Gil Neiger., Dion Rodgers., Amy L. Santoni., Fernando, C. M. Martins., Andrew V. Anderson., Steven M. Bennett., Alain Kagi., Felix H. Leung., Larry Smith. (2005). Intel Virtualization Technology, Intel Corporation, Published by the IEEE Computer Society, IEEE.

[2] Seyong Lee., Seung-Jai Min., Rudolf Eigenmann., (2009). OpenMP to GPGPU: A Compiler Framework for Automatic Translation and Optimization, School of ECE, Purdue University, West Lafayette, IN, 47907, USA, {lee222,smin,eigenman}@purdue.edu, February 14—18, Raleigh, North Carolina, USA.

[3] Thomas Anderson[1], Larry Peterson[2], Scott Shenker[3], Jonathan Turner[1]. (2004). Overcoming the Internet Impasse Through Virtualization, [1]University of Washington, [2]Princeton University, [3]UC Berkeley & ICSI, GDD-05-01, Appears as IEEE Computer (April 2005) and HotNets-III (Noveber).

[4] Jin, H., Frumkin, M., Yan, J. (1999). The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance, NAS

Technical Report NAS-99-011 October, {hjin,frumkin,yan}@nas.nasa.gov, NAS System Division, NASA Ames Research Center, Mail Stop T27A-2, Moffett Field, CA 94035-1000.

[5] Frank Cappelo, Daniel Etiemble. (2000). MPIversus MPI+OpenMP on the IBM SP for the NAS Benchmarks, LRI, Universite Paris-Sud, 91405, Orsay, France, Email: fci@lri.fr, 0-7803-9802-5/2000/$10.00© 2000 IEEE.

[6] Gerald Hilderink, Jan Broenink, Wiek Vervoort, Andre Bakkers. (1997). Communicating Java Threads, University of Twente, dept. EE, Control Laboratory, P.O.Box 217, 7500 AE Enschede, The Netherlands, (G.H.Hilderink, J.F. Broenink, A.W.P. Bakkers)@el.utwente.nl, W.Vervoort@cs.utwente.nl, year of publishing.

[7] Greg Burns, Raja Daoud, James Vaigl, (1994). *In*: Proceedings Of Supercomputing Symposium: LAM: An Open Cluster Environment for MPI, Ohio Supercomputer Center Columbus.

Appendex A:

Power Points	
Virtual Parallelism	**Physical Parallelism**
Can be applied for shared memory systems.Its number of threads can be unlimited but it limited by HW limits.	Can be applied with distributed memory systems.
Its performance affected by increasing number of threads executed.	Its number of processes limited to available network of computers linked to each other.Its performance affected by communication between processes.
Basic benefits are manag eability, flexibility, reliability, ease to maintenance…etc.	Hard challenge of extending network of computers connected due to the costs of buying new hardware.Basic challenges scalability, flexibility, maintenance.
Reducing the costs for buying new hardware.	More overhead on messages passed between processes due to communication information needed.

Table 3. Advantages and disadvantages of Virtual and Physical Parallelism environment