SD-MARC: A New Multi-Processor Architecture

A. Somdip Dey
Department of Computer Science,
St. Xavier's College [Autonomous]
Kolkata, India.

Abstract *- In modern day, HPC (High Performance Computing) is applied to do massive scientific or huge computational works, but HPC architecture mostly consist of computer clusters (series of computers connected to each other to solve a single problem / task). Thus it is problematic for many institutions or organizations to maintain a computer cluster for doing huge computation. So, with the advancement of technology and efficient multiprocessor architecture, it is possible to integrate thousands of processors in one computer system and apply that system for doing the huge computation. This paper basically provides the idea of a new model to apply multiple processors (in order of* n^2 *(n* x *n) processors) in one computer system Architecture and how to implement the model to compute different problems faster than they can compute in reality; provided: n>=4 and n=2m, where m=0, 1, 2, 3, 4 ……. N.*

Keywords: Multiprocessor; High Performance Computing; Architecture; Matrix; Computer model;

1 Introduction

With the rising demand of Computational power, new architecture in multiprocessor system and technology advancement in multicore computers have been seen. This gave rise to High Performance Computing and parallel processor computer systems. Now a day huge processing power is required to compute and process a huge amount of data. So there is indeed an urgent need in advanced multiprocessor computer architecture to compute these huge amount of data within a short span of time.

The objective of this paper is to provide an overview concept of a new architecture in Multiprocessor System and how this architecture, SD-MARC, is beneficial in terms of computation and how to implement it.

2 The Architecture

This section covers the details regarding the architecture of SD-MARC.

Before moving on to SD-MARC, first let us see the various techniques and architectures used in multiprocessing systems.

Multiprocessor architecture can be basically classified into:

1.Symmetric Multiprocessing (SMP)
2.Asymmetric Multiprocessing (AMP)
3.NUMA (Non Uniform Memory Access Multiprocessing)
4.Cluster Multiprocessing (CM) [Concept of Distributed Computing]

SMP (Symmetric Multiprocessing):

In a multiprocessing system, when all CPUs / processors are treated equally, then the system is called Symmetric multiprocessing system (SMP). In SMP two or more identical processors are connected to a single shared main memory (computer memory) and are connected by a single Operating System instance, i.e. monolithic kernel type of OS (Operating System) is used for this type of system to utilize the resources. Now a day most of the multiprocessor architecture uses SMP. In SMP the processors are either connected to each other buses or crossbar switches or on-chip mesh networks.

The advantages of SMP include a large global memory and better performance per power consumption by the system. SMP also provides simple node-to-node (processor to processor) communication. The main disadvantages of SMP include the fact that the memory latency and bandwidth of a given node can be affected by other nodes, and cache "thrashing" may occur in some applications.

AMP (Asymmetric Multiprocessing):

AMP (Asymmetric Multiprocessing) designs uses SMP hardware architecture where a common global memory is shared between the various processors. In AMP designs, application tasks are sent to the system's separate processors. These processors may all be located on different boards or collocated on the same board, but each is essentially a separate computing system with its own OS and memory partition within the common global memory. One advantage of an AMP design is that asymmetric memory partitions can be assigned

from one large global memory, making more efficient use of memory resources and potentially reducing system cost.

SMP architectures differ from AMP in that a single block of memory is shared by the multiple processors or by multiple cores on a single multi-core processor. A single OS (Operating System) image runs across all the cores enabling truly parallel processing.

NUMA (Non Uniform Memory Access Multiprocessing):

In NUMA (Non Uniform Memory Access) multiprocessing the memory access time depends on the memory location relative to a processor. In recent time processors work faster than the memories used by them, so there is a big gap in the speed of a processor and a memory. So in a multiprocessor architecture to get high performance, one person have to install high-speed cache memory and use advanced algorithm to reduce the cache-'miss'. NUMA tries to solve this problem by providing separate memory for each processor, avoiding the performance hit when several processors attempt to address the same memory.

CM (Clustered Multiprocessing):

In Clustered Multiprocessing, many computers are loosely connected to each other, which forms computer cluster, and they work together so that in many respects they can be viewed as a single system. Usually the computers in a cluster are connected via high speed local area networks and this concept evolved from the concept of Distributed Computing, where different computers are connected to do one common task or achieve one same goal. In CM architecture we can see the use of master node and computing node, where the master node controls and distribute work (processes) to the compute node.

In SD-MARC, we combine the logic and ideas of most of these multiprocessor architecture, so that a new powerful architecture can be made out of it and can be implemented to compute faster and save cost of implementation relatively.

2.1 SD-MARC:

A. The Basic Design of SD-MARC

The basic design of SD-MARC consist of n^2 ('n x n') number of processors. Here, in this architecture the arrangement of the processors can be thought of square matrix formation of 'n x n', where 'n x n' signifies the square matrix system in which the number of columns and the number of rows are equal to 'n' and, n>= 4 and n is even number, i.e. n=2m , where m= 1,2,3,4,5,……..

So, from the above statement we get to know that the system consist of 'n x n' number of processors, as shown in the Fig 1.1, where the processors or the CPU (Central Processing Unit) are arranged in '4 x 4' formation and the total number of processors in this figure is 16 processors / CPU:

N.B.: In Fig 1.1 the number of processors in each row and each column are 4 and so it forms 4 x 4 multiprocessor system. From the figure Fig 1.1 we can see that 16 processors are arranged in 4 x 4 matrix formation and just like this 'n x n' number of processors can be arranged in the 'n x n' square matrix formation, where each row and each column contains 'n' number of processors.

Now this 'n x n' processor formation can be further divided into four divisions: Division 1, 2, 3, 4, where each division will have ((n x n) / 4) number of processors. So for example, if a system consists of 16 (4 x 4) processors then each Division will have 4 processors in it, or, for example in a system of 36 (6 x 6) processors there will be 9 processors in each division. This concept of Division system can be clear from the Fig 1.2, where the Division System of the Processors in 4 x 4 processor system has been shown.

Now along with this Division System we can use the concept of Computer Clusters. In Computer Clusters there is a concept of Master Node and Compute Node, where the Master node distribute the workload to different Compute node. Just like this concept, in the Division System of multiprocessor architecture, the concept of 'Master Processors' are implemented. Each division of processors has a master processor, which distribute the workload of that division to the different other processors, namely called Compute Processors, in that division. So basically there are 4 master processors in this multiprocessor architecture system.

Each Division has the processors numbered in order, like, CPU 1, CPU 2,…… and out of these processors, one is a master processor of that division, which is denoted as M CPU, as shown in Fig 1.3. So in the first division, i.e. Division 1, there will be CPU 1 to CPU j, where 'j' is the last processor number in that division and the starting number of processor in the next division will be CPU j+1, which goes up to CPU k, where 'k' is the last processor number in the Division 2, and so on.

In this system the 4 master processors are placed in side by side fashion, so that they can communicate with each other easily and very fast. The 4 master processors of 4 divisions communicate with each other time to time and synchronize among themselves the works / processes they are coordinating. The above mentioned concept can also be figured out from the Fig 1.3.

B. Processor Architecture and Relation between each Processor

Each processor will have high speed register memories of their own along with L1 (Level 1) Cache memory. Two adjacent processors / CPUs will be having or sharing a L2 (Level 2) Cache memory and then four adjacent processors will share L3 (Level 3) Cache memory. In Fig 2.1 we can visualize the concept clearly, where the relation between processors/CPU in terms of memory has been shown.

In modern world, the main computer memory can be classified basically in three ways:

a. Distributed Memory

b. Shared Memory

c. *Distributed Shared Memory*

Mostly Distributed Memory and Distributed Shared Memory concepts are used in case of computer architecture, where each CPU has a private memory block of its own.

But in this architecture a concept of '**Hybrid Memory Distribution**' is used. The 'Hybrid Memory Distribution' concept is: A memory block will not be private only to one CPU but each memory block will be shared by four CPUs / Processors, as seen in the Fig 2.1.

Since, four CPUs will be sharing only one memory, so there can be possibility of resource holding, which may give rise to Dead-lock situation. So to do away with that, each memory block will have memory address of its own and that memory address will not be same in any way in any of the other memory blocks. For example, in Memory 1the starting address is 1000 (Hex Address) and the end address in the memory is F000 (Hex Address), then the starting address in Memory 2 can be F001 (Hex Address) and the address in memory2 will never be the same as the addresses in Memory 1.

So from Fig 2.1 we can see that one memory is shared by four CPUs and these four CPUs form a 'Block'. In Fig 2.1 we can see two blocks of CPUs, Block 1 and Block 2. Basically in this architecture the whole multiprocessor system is divided in 'matrix' system, then each matrix system is divided in four 'Divisions' and at last each division has several Blocks.

Now each block has a memory of its own which is not shared with other blocks or other CPUs of other blocks, and the addresses in the memory never coincide with the address of other memories of other blocks.

If a matrix system of multiprocessor is of such a form that ((n x n) / 4) = ans, where ans is not perfectly divisible by 4 again then that system must have a sinlge CPU / processor left out of the block formation. For example there are 36 processors in a system then it is of form '6 x 6' CPU system. Now if we form four Divisions then there will be 9 CPUs in each Division. And if we try to form Blocks out of each Division then we can see that only two blocks can be made in each Division and there will be one CPU left out of the block in that Division. Then in that case we will consider the last CPU as the Master CPU and will attach that CPU to the last block formed in that Division. This concept can be clear from the Fig 2.2.

N.B.: In Fig 2.2 The 'BUS' means the Bus system through which each of the interaction of CPUs and memory system takes place.

Since '6 x 6' multiprocessor system is taken as an example so as from Fig 2.2 we can see that these two Blocks are in each Division and so there are 8 Computer Memory in the architecture.

C. Interaction Between the Processors

In Fig 3.1 the interaction and communication between the processors are shown. In this multiprocessor architecture, all the Compute processors in a Division are connected to the Master processor. The Master processor of that Division is again connected to other Master processors adjacent to it. The Master processor of a Division controls and distribute workloads to other Compute processors, and the Master processors then communicate with each other to synchronize the tasks they have performed. When a Master Processor is assigned a work (process) to perform, it first breaks that work into several small processes (can also be denoted as 'threads') and assign these small processes to different Compute processors of that Division along with the instructions of fetching different memory addresses needed to perform the small processes. The Master processor keeps track of each task (thread) assigned to each Compute processor and the memory addresses accessed by those Compute processors. After the completion of each task (thread), each Compute processor synchronize their tasks (threads) with the master processor and free the memory addresses used for the threads, which are again being tracked by the Master processor. And then that Master processor synchronize the process completed by it with the other Master processors of other Divisions.

D. Operating System to this Architecture

To utilize the computational capacities of the hardware of a computer an Operating System is most important. A kernel is a central component of an operating system. It acts as an interface between the user applications and the hardware. In most of the Computers in the world, either Micro Kernel or Monolithic Kernel type of Operating Systems is used. Monolithic Kernel can perform all the operations and consist of only one layer, i.e. in PCs (personal Computers) monolithic kernel is used to perform computation. And Micro Kernel can perform mainly few operations along with one global operation, i.e. it can perform one global process and other small low-level processes. Micro kernels are mainly used in Distributed Systems or Multiprocessor Systems.

Since, in this Architecture multiple processors are integrated in one single computer, so use of either of the two Operating Systems only will not be feasible to perform the processes. To solve this problem, a new concept of Operating System is introduced along with this architecture.

The new concept is that the system will have a basic monolithic kernel. This monolithic kernel will again have many small kernels (just like the micro kernels) inbuilt within it. These micro kernels will perform small tasks along with a global task and will be assigned to each Master processor. Since micro kernel can perform only one global process at a time, the Master processor can utilize this fact and perform that one global process at a time along with small other processes. This type of a Kernel is called a 'Hybrid Kernel'.

3 List of Figures:

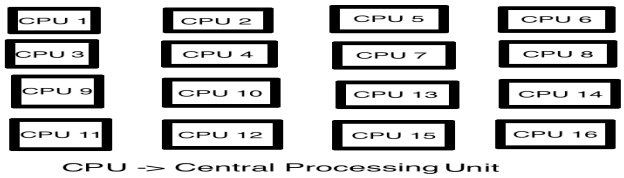

Fig 1.1: Figure shows 16 processors arranged in 4 x 4 matrix pattern.

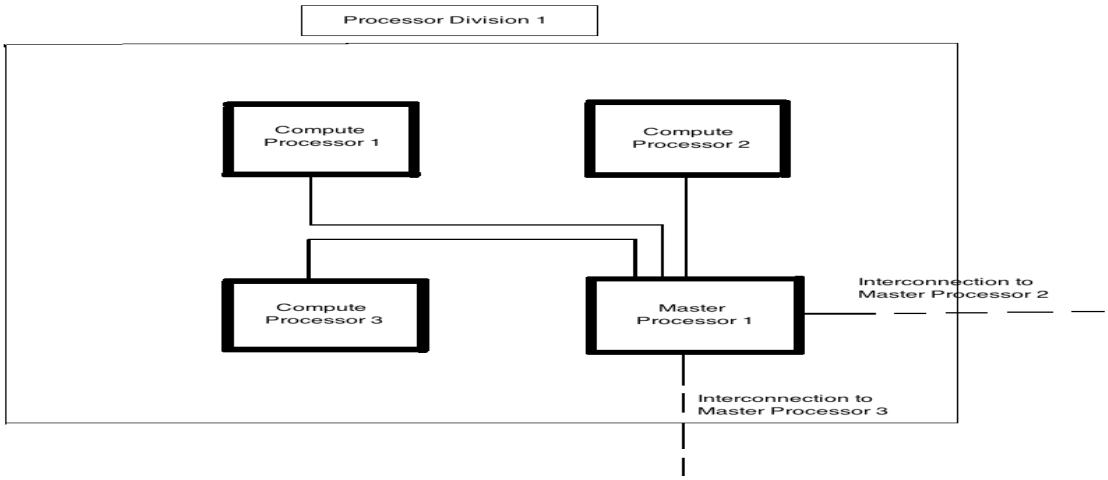

Fig 3.1: The Interaction between CPUs / Processors in '4 x 4' Processor System

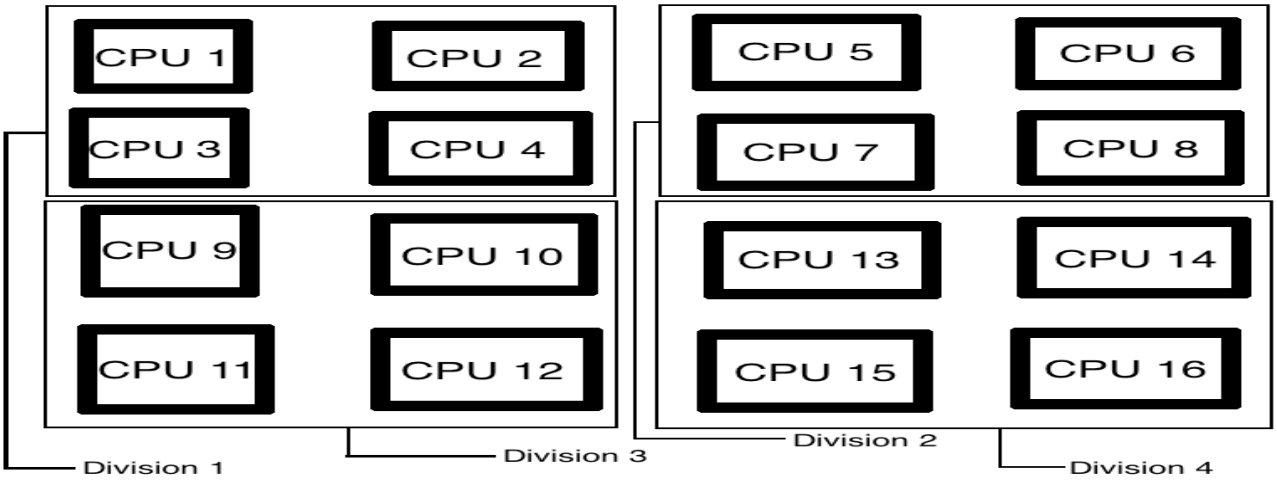

Fig 1.2: Division System in 4 x 4 Processor System

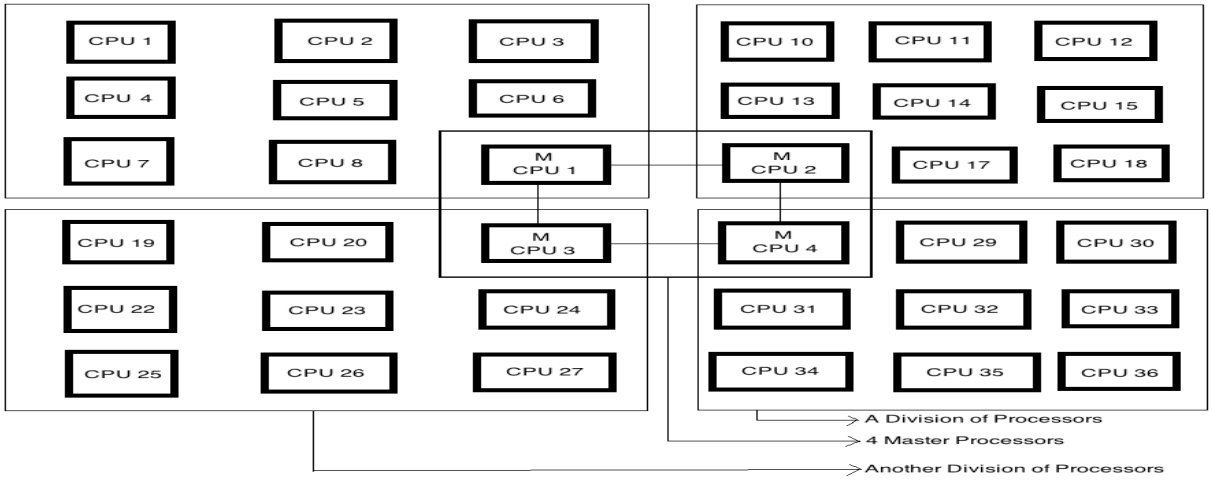

Fig 1.3: Division System with the concept of Master Processors and Compute Processors in (6 x 6) processor System

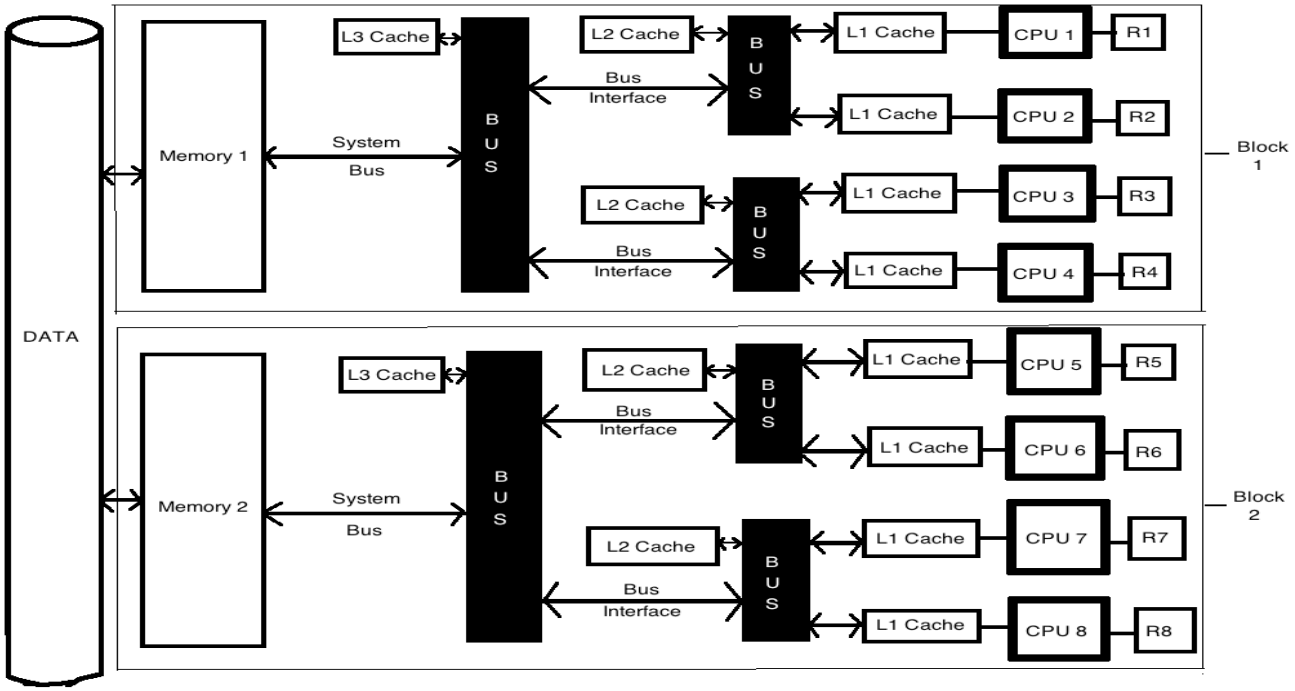

Fig 2.1: Relation between CPUs in terms of memory
N.B.: R1 – R8 are Register Memories of the CPU 1 – CPU 8

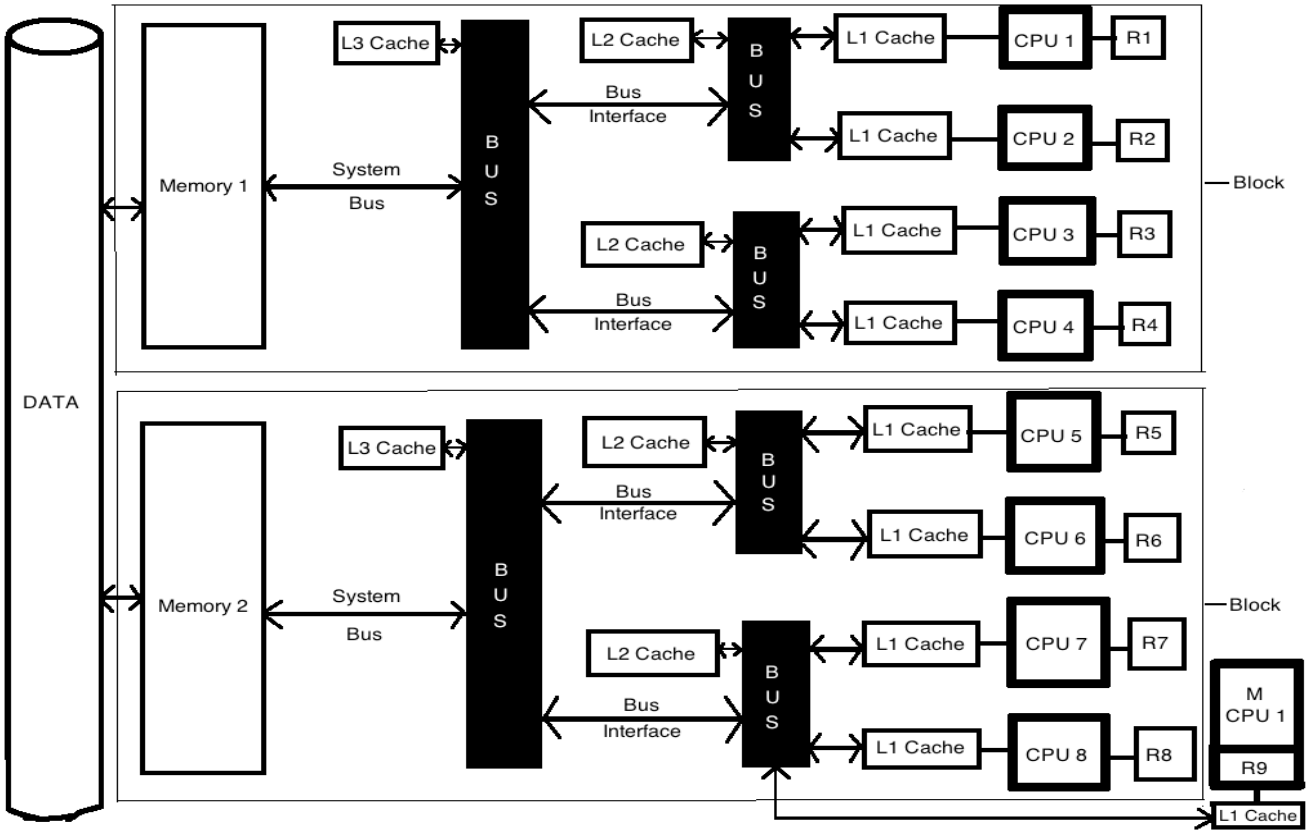

Fig. 2.2: Attachment of the M CPU (Master CPU) to the last Block in a division

N.B.: R1 – R8 are Register Memories of CPU 1 – CPU 8; R9 is the Register Memory of M CPU

4 Few More Details

A. *Cooling of the System*

Since multiple processors are integrated within a single computer, it is obvious that the system will get very hot and will generate a lot of heat. So to deal with this problem high level of cooling system is needed to be installed within the system. Both Air Cooling and Liquid Cooling systems can be used to keep the computer cool and ventilate out the hot air.

B. *Need of Multiple Processors in one System*

In HPC (High Performance Computing) or in Super Computers mainly the concept of Computer Clusters are used to compute a task, where many computers are connected to each other via a network. But there is an ever growing demand of fast computers in todays world. If Computer Clusters are used then the system become huge to maintain and are also very costly economically. So if multiple (in order of hundreds or thousands) processors are integrated in one computer using nano-technology or recent technological advancement then it will be easier to manufacture smaller (in size) super computers and will be easier to maintain. If multiple processors are used in one system then it will be easier to compute and finish a task faster than it could be, and to make the system easier to be maintained.

C. *Use of A Single Computer Memory for 4 CPUs*

In this Architecture four CPUs (processors) share a single computer memory (RAM or Random Access Memory in general). If the concept of Distributed Memory System or Distributed Shared Memory System is used then the number of processors will have equal number of computer memory, and in this architecture it is of the order 'n x n' (n^2) which is a very large number. So the number of computer memories used would have been n^2 for 'n x n' number of processors and it would have made the system pretty large to maintain. To deal with this problem four CPUs share a single computer memory and for a system with 'n x n' number of processors, the number of computer memories needed in this architecture is (abs[(n x n) / 4] * 4), where abs() denotes the absolute value or the integer value of a number, for example: abs(2.25)= 2.

D. *Architecture Usage and Future Scope*

Since, SD-MARC uses all the advantages of the different types of multiprocessor architecture system, it is beneficial to use and implement it, but still there's a drawback of this architecture. There's no Operating System till date to support this architecture. If this architecture is used in small systems or very few number of processors are used to built a multiprocessor system using this architecture, then there may be lack of performance, as it will be in a system with thousands of processors.

5 Conclusion

Since, this multiprocessor architecture, SD-MARC, is used in a single computer system, it can be used in any field of application, where intense computational power is needed in one single system. For example, this architecture can be used by hospitals to compute different problems and chemical structures of medicines, can solve and diagnose different diseases in very less amount of time. Even computer enthusiast can use this architecture to build their own Personal Super Computers and use those to compute time-consuming problems in really less amount of time.

6 References

[1] W. Anderson, F. J. Sparacio, and R. M. Tomasulo, ''The IBM System/360 Model 91: Machine Philosophy and Instruction-Handling,'' *IBM Journal of Research and Development,* Vol. 11, No. 1, pp. 8-24, January 1967.

[2] Proceedings. Supercomputing '88 (IEEE Cat. No.88CH2617-9) ,November, 1988.

[3] Norman P. Jouppi, ''The Nonuniform Distribution of Instruction-Level and Machine Parallelism and Its Effect on Performance,'' *IEEE Transactions on Computers,* Vol. 38, No. 12, pp. 1645-1658, December 1989.

[4] Youngjin Kwon, Changdae Kim, Seungryoul Maeng, Jaehyuk Huh, "Virtualizing performance asymmetric multi-core systems", International Symposium on Computer Architecture, pp. 45-56.

[5] Rajkumar Buyya (editor): *High Performance Cluster Computing: Architectures and Systems*, Volume 1, ISBN 0-13-013784-7, and Volume 2, ISBN 0-13-013785-5, Prentice Hall, NJ, USA, 1999.

[6] Internet Source: http://www.intel.com/pressroom/archive/reference/whitepaper_QuickPath.pdf [last visited 10/03/2012]

[7] Internet Source: http://users.ece.utexas.edu/~bevans/papers/2009/multicore/MulticoreDSPsForIEEESPMFinal.pdf [last visited 11/03/2012]