On the Concept of Simultaneous Execution of Multiple Applications on Hierarchically Based Cluster and the Silicon Operating System

N.Venkateswaran[§], Vinoth Krishnan Elangovan[♭], Karthik Ganesan[♭]
TP Ramnath Sai Sagar[†], Sriram Aananthakrishanan[†], Shreyas Ramalingam[†]
Shyamsundar Gopalakrishnan[♭], Madhavan Manivannan[♭], Deepak Srinivasan[♭]
Viswanath Krishnamurthy[♭], Karthik Chandrasekar[♭], Vishwanath Venkatesan[♭]
Balaji Subramaniam[†], Vidya Sangkar L[†], Aravind Vasudevan[†]
Shrikanth Ganapathy[†], Sriram Murali[†], Murali Thyagarajan[†]

Abstract

In this paper we present a novel cluster paradigm and silicon operating system. Our approach in developing the competent cluster design revolves around an execution model to aid the execution of multiple independent applications simultaneously on the cluster, leading to cost sharing across applications. The execution model should envisage simultaneous execution of multiple applications (running traces of multiple independent applications in the same node at an instant, without time sharing) and on all the partitions(nodes) of a single cluster, without sacrificing the performance of individual application, unlike in the current cluster models. Performance scalability is achieved as we increase the number of nodes, the problem size of the individual independent applications, due to non-dependency across applications and hence increase in the number of non-dependent operations(as the problem sizes of the applications get increased) and this leads to better utilization of the unused resources within the node. This execution model is very much dependent on the node architecture for performance scalability. This would be a major initiative towards achieving performance Cost-Effective Supercomputing.

1 Introduction

High performance monolithic clusters, having good performance and scalability are becoming increasingly popular in the research community for their ability to cater to specific application requirements. The level of performance is characterized by the node architecture, network topology, compiler, parallel programming paradigm and operating system. Making better design choices would improve the execution time of large scale applications, which is currently predicted to be in Teraflop years. In this paper, we discuss the impact of these design choices on the application's performance and provide insights into a supercomputing model which would cater to the demands of the next generation grand challenge applications. Future generation applications might require close coupling of previously independent application models, as highlighted in NASA's report on Earth Science Vision 2030[1].

Performance scalable and cost effective supercomputing call for simultaneous execution of independent applications. There is hence a need, to develop an execution model for cost effective supercomputing which will envisage simultaneous execution of multiple applications (running traces of multiple independent applications at an instant, without time sharing) in the same node, which is not the case in [10] and on all the partitions (nodes) of a single cluster, without sacrificing the performance of individual applications, unlike the current models in which different applications are executed in independent partitions of the cluster or in the same node, but time shared. This time-sharing cannot lead to effective performance scaling as the problem size and the cluster size increases. To achieve effective performance scaling up at the cluster level for the proposed execution model, the node architecture[2,8] also plays a major role and this has not been addressed in[10]. This execution model introduces new challenges in the node and cluster architecture including an operating system to enable it to handle the increased mapping complexity and tracking, during the execution of simultaneous multiple applications. However, the support for execution of such diverse workloads encountered during simultaneous multiple application execution lies in the design philosophy of the node architec-

*§N.Venkateswaran Founder Director, Waran Research Foundation (WARFT), Chennai, India. Email : - waran@warftindia.org
††- WARFT Research Trainee, 2005-2007
‡♭—Former WARFT Research Trainees

978-1-4244-1694-3/08/$25.00 ©2008 IEEE

ture. In [3] the capability of MIP(Memory In Processor)-paradigm[5,8] based homogeneously structured Heterogeneous Multi-Functional Core Node Architectures to handle SMAG(Simultaneous Multiple AlGorithms) execution and large number of general purpose operations in parallel aiding the proposed execution model by running traces of multiple applications simultaneously in the same node. Performance scalability is achieved as we increase the number of nodes, the problem size of the individual independent applications, due to non-dependency across applications and increase in the number of non-dependent operations (as the problem sizes of the applications get increased) leading to better utilization of the resources within the node and the unused nodes(as a worst case). Arriving at the proper mix is important.

The paper is organized into 4 sections. Section 2 discusses the scope for improvement in the design features of current generation clusters in order to meet the requirements of performance greedy applications, also taking into consideration the operating cost factor. Section 3 highlights a cluster model that incorporates all the architectural concepts proposed in Section 2 and investigates its potential for cost effective execution of multiple applications. Section 4 addresses the ramification of this model on performance, resource utilization profile and their influence on the performance/cost relation.

2 Impact of Operating Systems on High Performance Clusters

Performance modeling has come a long way in helping researchers characterize cluster designs to achieve expected performance. Different methodologies have been evolved to accurately compare, analyze and predict the performance of various designs and features of high performance clusters[4]. In the current execution model, the workload of a single application is mapped on to a set of nodes, which does the work of load balancing across the nodes of a cluster. The node is usually empowered with a stripped kernel, which performs the core OS functionalities such as Memory management, Process scheduling, I/O handling and Interrupt handling. But in the context of the proposed execution model(SMAPP), a new OS paradigm is required for handling the complexities associated with parallel mapping and data tracking of the huge amount of data associated with the different independent applications. In this scenario, allocation of the thousand processes belonging to different applications among large number of nodes, keeping track of their execution, handling exemptions, interrupts and reliability of the operating system are of paramount importance as the integrity of IO data sequencing corresponding to different applications is critical, particularly when dealing with million node clusters. Thus the capability of the cluster to stomach the complexities involved in multiple applications' execution relies totally on an efficient OS design.

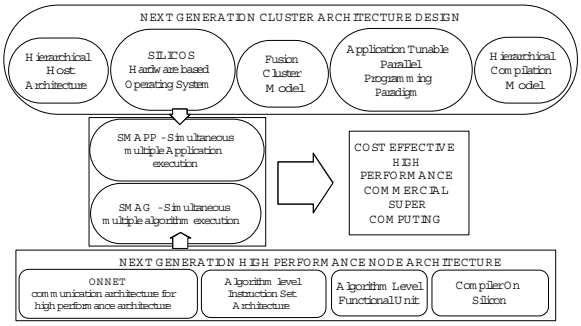


Figure 1. Model for next generation Supercomputers

3 Model for Next Generation Supercomputers

In order to create a design space for supercomputers, the focus should also be on aspects like power, performance, cost and their related tradeoffs. In this section we present a conceptual model (fig.1) for cluster design taking into consideration all the design issues discussed in section 2. The cluster model comprises of MIP-paradigm based nodes[5,8] which are capable of handling Simultaneous Multiple AlGorithm (SMAG) execution and parallel execution of a large number of general purpose operations. In our discussion, we primarily attempt to give a conceptual overview of the proposed cluster model.

3.1 Simultaneous Multiple APPlication (SMAPP) Execution & Cluster/ Host Architecture

Due to the ever increasing demand posed by scientific and engineering applications, it becomes mandatory to evolve supercomputing clusters whose node architecture is highly tuned towards these applications. The design of the node[2,8], while improving the capabilities to handle increased complexity of the application, should also enable the node to support the intended SMAPP execution at the cluster level. The details of this cluster architecture and the execution flow of SMAPP are discussed in section 3.2. The fig. 2 is an abstraction of the SMAPP flow in the multiple host hierarchical cluster. A major challenge for the host system lies in tackling the complexity of mapping and sequencing the thousands of terabytes of data resulting from the simultaneous execution of several applications, which otherwise might lead to a potential virtual I/O bottleneck.

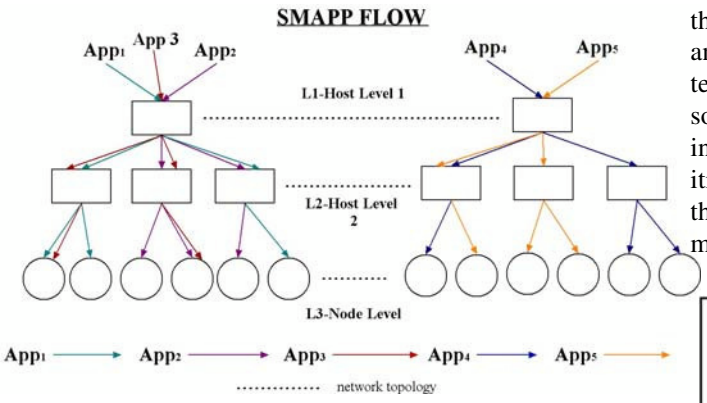


Figure 2. SMAPP concept

Moreover, considering the computational strength of the MIP-paradigm based node architecture, the host system should be efficient in meeting the feed rate required by the nodes. Preprocessing at the cluster reduces the compiling complexity used in the node thereby aiding efficient execution and reducing the compiling load on the node, leaving only the scheduling. Especially when the cluster size becomes very large, these issues have a huge impact. Investigation on these issues lead us to the idea of developing a hierarchical based host system. The architecture and functionalities of the host system will be presented in 3.3 and 3.4. In accordance with the above mentioned school of thought, we have developed a hybrid, pyramid structured cluster design as depicted in fig 3, wherein two stages of problem decomposition occurs at the primary and secondary hosts, to efficiently partition and map the applications onto the cluster[6,7] as discussed in sections 3.3 and 3.4.

3.2 SILICon Operating System (SILICOS)

In order to handle the complexity involved in simultaneous execution of multiple applications and to manage scheduling, memory, preprocessing of multiple application and I/O operations, we had highlighted the need for an efficient Operating System (OS) design. A software OS may not be proficient enough to exploit the power of the underlying MIP-paradigm based nodes as well as to meet the requirements of SMAPP. Due to computational speeds of such nodes, and large number of processes involved in multiple applications the operating system needs to perform the varied process allocation faster and efficiently and also to enhance memory management process. Parallel mapping of multiple applications across very large cluster will be a big burden for the soft OS. In view, we have resorted to a hardware-based operating system termed as SILICOS [7], the functionality of which is distributed across the primary and the secondary host planes. By designing suitable architectures for the primary and secondary host plane's processors to incorporate the functionalities of the cluster operating system. While the above mentioned core OS functionalities are implemented on a hardware platform in SILICOS, the rest are formulated as software libraries residing in primary and secondary hosts.

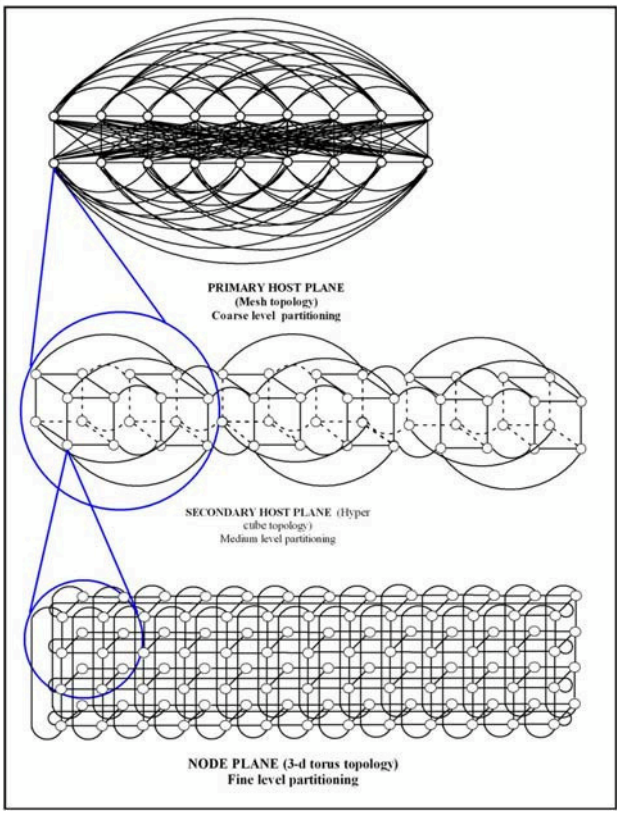


Figure 3. MIP Cluster architecture

Besides managing the complexities associated with SMAPP, the primary and secondary hosts architectures will make the hardware based operating system more reliable and immune to software aging. Fault tolerance in SILICOS is realized through on-demand network reconfiguration among the primary and secondary host planes. In the event that a specific resource of a particular host system fails, the load is transferred to the healthy sections of the host plane. System maintenance is undertaken with great ease when such a hierarchical host system is adopted.

When multiple independent applications are executed simultaneously, one of the main overheads on the OS is the ability to keep track of the traces of every application executed across all partitions of the cluster. To facilitate tracking, the Primary Host generates a generic tracking format

which is interpreted and translated at various levels of the cluster. The control, data and instruction flow in the MIP cluster is portrayed in (fig. 8) along with the various format definitions, based on which the multiple applications are tracked and the data consistencies maintained and the corresponding field definitions is given in (fig. 4). This way the node resident OS tasks are relieved from the nodes to a great extent thereby improving their processing capability.

3.3 Primary Host Functional Architecture

The primary host has several hardware modules that forms the core kernel functionalities, thereby keeping the software complexity low. The primary host performs parallel decomposition of the multiple applications simultaneously into their constituent sub-problems. This coarse-grain partitioning of multiple applications at the primary host plane requires close interaction among the host processors for uniform exchange of workload (sub-problems belonging to multiple applications). The interaction will facilitate distribution of a multiple application load (application mix) across a set of secondary host processors under a primary host. The exchange of workloads across the host processors is achieved through a Self Mapping process among the primary hosts. This necessitates that the primary hosts be completely connected. The primary host architecture is shown in (fig. 5).

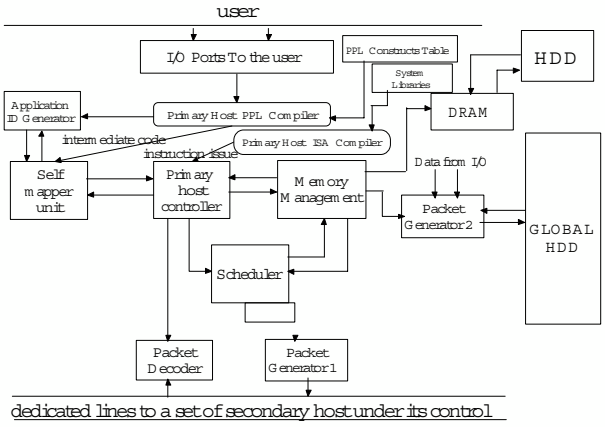


Figure 4. Functional Architecture of Primary host

The primary host PPL compiler generates the intermediate code, which is used by the self mapper unit to exchange the workloads associated with the different applications. An application ID generator in the host processor embeds the application ID onto the intermediate code format during load balancing and exchange across the primary hosts, which helps to keep track of the various application workloads assigned to each primary host. The primary host scheduler balances the workload across its underlying secondary host processors and assigns them to the process scheduler, forwarded in the form of packets through the packet generator unit1. The data corresponding to the application mixes is formed into packets by the packet generator unit2 that embeds header information which includes application ids, library and sub-library ids. These packets are stored in the global hard disk, and are accessed later by the secondary host processor. The local compiler generates and issues the instructions to activate the primary host controller which triggers the corresponding functional modules. The design of memory management unit and packet generator are along conventional lines and the design details on self mapper and scheduler are available in [7]. The hardware aspects of SILICOS would contribute its fraction to increasing the power consumption of the cluster on the whole. This fraction of power consumption is reducible by selectively shutting down certain functional modules after the completion of associated tasks, such as the scheduler and self-mapper in the primary host processors.

3.4 Secondary Host Functional Architecture

This forms the intermediate plane of the MIP cluster and majority of the Operating System functionalities are attributed to these hosts. The sub-problems issued from the primary host are further divided into libraries that the node handles by using the secondary host compiler. The functionalities of the secondary hosts include scheduling of libraries uniformly across the underlying MIP nodes and the corresponding data transfer, I/O handling, Interrupts and Exception handling, Data Packet Generation, Performance and Resources monitoring, facilitating data communication across the nodes when necessary and most importantly gathering the outputs and reforming data packets for issuing if necessary.

The secondary plane performs the majority of the operating system functionalities for the cluster concerning the SMAPP execution. Considering multiple applications being executed simultaneously across the nodes, the amount of interactions between the nodes might be expected to be higher. For such interactions between distant nodes, latency becomes a huge issue and hence data from the remote node is routed through the secondary hosts. This way the number of hops is drastically reduced thereby reducing latency. Also, making use of a hypercube topology, which accommodates a greater number of nearest neighbors for connecting the secondary hosts, would further improve the latency issue.

The secondary host architecture is shown in (fig. 6). The process scheduler present in the secondary host processor receives format2, which contains the details of the applica-

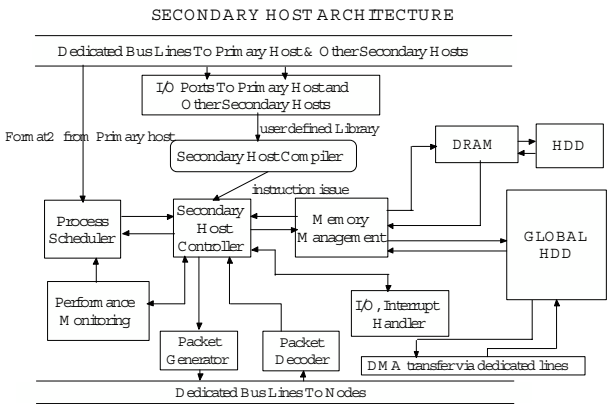


Figure 5. Functional Architecture of secondary host

tion mix (assigned to it by the primary host scheduler) and gives the association between the instruction packet (application mix) and the data packet. The secondary host process scheduler now schedules these application mixes across the nodes in the form of packets (generated by packet generator) addressed by format1 to the node plane. While selective shutdown of the scheduler module is undertaken at appropriate instants of time in order to save on the power consumption, it is re-instantiated as and when a process requirement occurs.

The memory management unit assists the physical packet transfer from the global hard disk to respective onboard DRAM through a DMA (Direct Memory Access) transfer mechanism via a dedicated set of lines. The secondary host also governs the maintenance aspects of the cluster through the performance monitoring unit. The IO exception/interrupt module handles the interrupts and exceptions received from the distributed controller of the respective node[2]. Communication between the node and secondary host is facilitated by dedicated lines spawning from the node, to support high volumes of data transfer and maintenance information between the secondary host processors and the nodes.

The local compiler is responsible for generating the instructions to trigger the various modules discussed above. Performance Monitoring Architecture models the workload characteristics of real-time high end applications. This pseudo-application model is developed[9] taking into consideration both the computational and communication complexity associated with applications, to generate different workload traits of the applications that are intensive in terms of communicational or computational complexity or even strike a balance between the two.

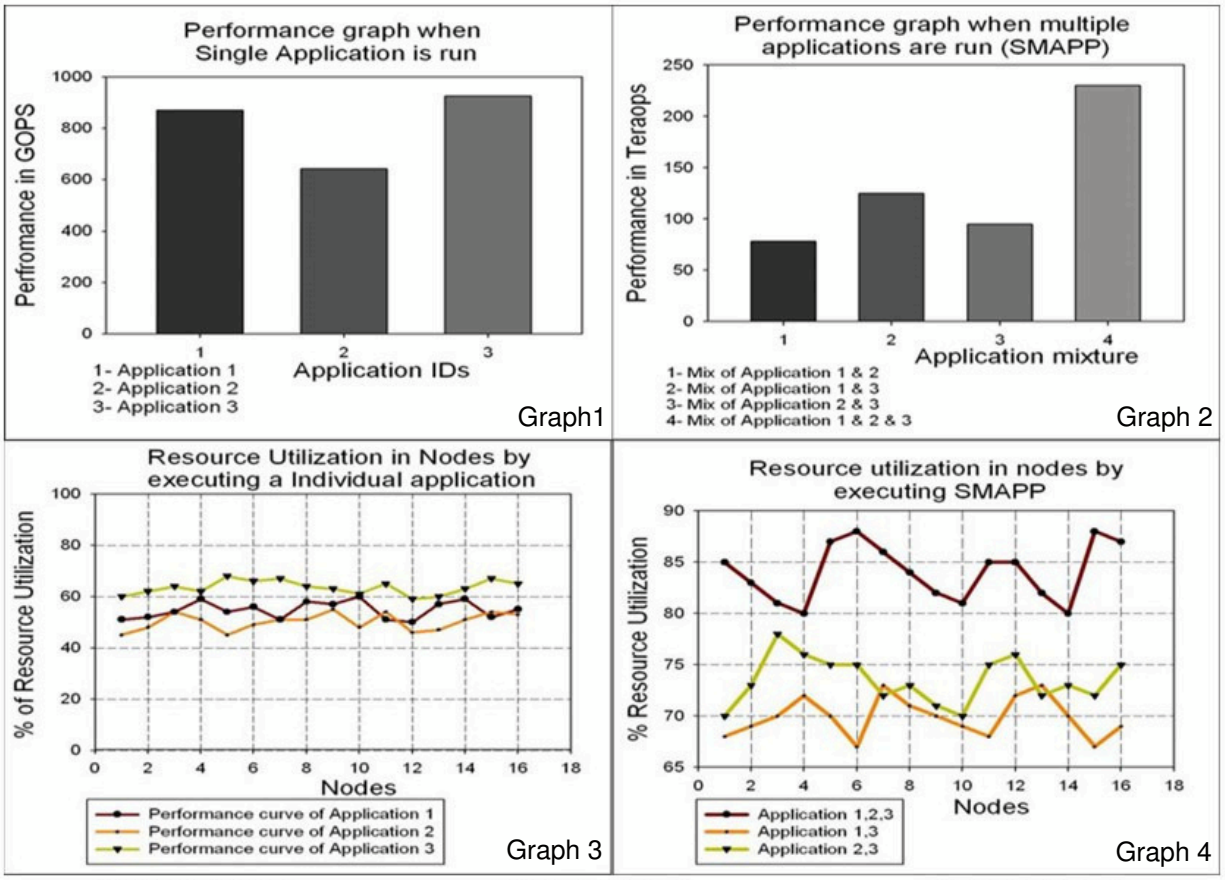
Here in this paper, a detailed conceptual view of SILICOS is presented. The architectural details of the individual functional units like Self Mapper, Scheduler, MMU and Data Packet generator of the primary and secondary hosts are available in [7].

4 Simulation Analysis

The simulation results presented in this section are extracted by running multiple pseudo-applications [9] comprising a workload from three distinct application traces, on a 16 node (2-D torus topology) emulated MIP cluster. The performance and resource utilization across the cluster have been measured and a detailed analysis is presented to show the effectiveness of executing SMAPP in MIP cluster. The simulation is done by mapping computations from the BENSIM to the emulated MIP cluster. The BENchmark SIMulator is capable of generating workload characteristics, both computation and communication reflecting the complexity of real time applications. The details of this benchmark simulator tool are available in [9].

Graph 1 and graph 2 depicts performance profile for individual pseudo-applications and different pseudo-application mixes respectively. Graph 3 shows a resource utilization profile of the 16 node MIP cluster, for the three pseudo-applications which governs different workload sets, as shown in the table 1, 2 and 3 and they are mapped individually. Such variation in resource utilization for the three pseudo-applications are prevalent mainly due to the distinct workload characteristic that is associated with it, mapping strategies adopted by the hosts and the architectural characteristics of the MIP paradigm based node architecture. Graph 4 depicts the resource utilization profile when the three pseudo-applications are simultaneously mapped for execution on the same cluster. The resource utilization of each node directly reflects in the performance of the cluster (graph 2).

The variation in resource utilization during SMAPP execution (graph 2) is mainly attributed to the simultaneous multiple algorithm execution at the MIP paradigm based node architecture [2], the cluster operating system and the flexibility provided by the cluster interconnection. The cluster level performance improvement for the SMAPP execution model is primarily due to MIP paradigm, characterizing a node architecture involving homogeneously structured heterogeneous functional cores [8]. This node architecture leverages the performance when workloads of multiple applications (multiple algorithms) and general purpose operations are executed simultaneously without time-sharing. Extensive clock level simulations of the execution of algorithm mixes are available in [8]. These simulations clearly show that as the number of operations increases (as a consequence of increase in number of independent applications) the number of clock cycles increase by a marginal scale ow-



BENSIM APPLICATION 1

LIBRARY NAME	SIZE	TYPE
FFT	24576	SCALAR
CONVEX HULL	1500	SCALAR
DFT	24480	VECTOR
QRD	256 X 256 & 3 X (64 X 64)	MATRIX & VECTOR

BENSIM APPLICATION 2

LIBRARY NAME	SIZE	TYPE
DFS	30000	GRAPH THEORETIC
BITONIC	3000	VECTOR
MATRIX MULTIPLICATION	3 X (256 X 256)	MATRIX
GLUE	3000	SCALAR

BENSIM APPLICATION 3

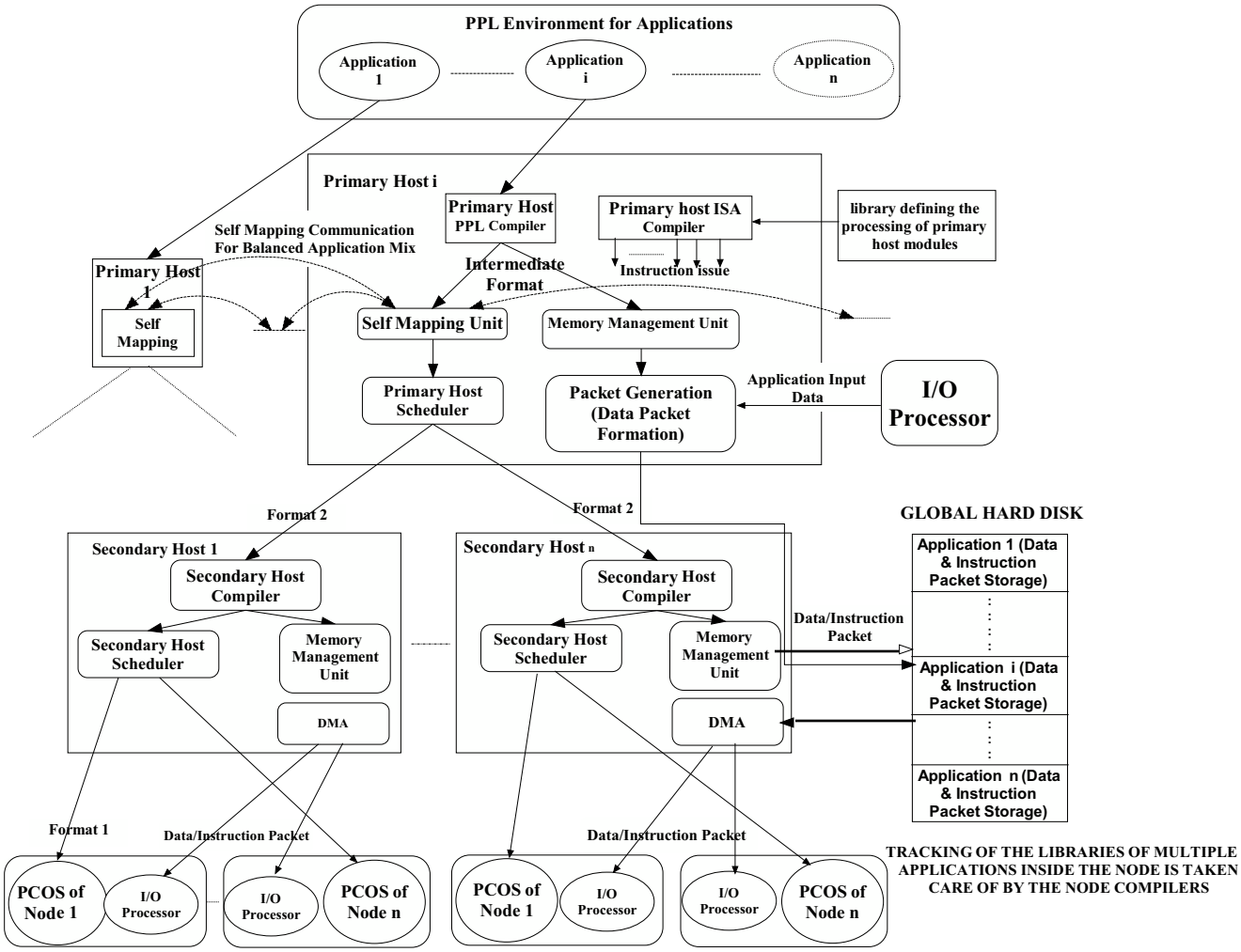
LIBRARY NAME	SIZE	TYPE
LUD	512 X 512	MATRIX
SVD	300 X 300	SCALAR
FFT	12288	MATRIX & SCALAR

MULTIPLE APPLICATION WORKLOAD DISTRIBUTED ACROSS 16 NODE MIP CLUSTER SNAPSHOT OF EXECUTION TRACE DURING SMAPP EXECUTION

NODE ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
BENSIM 1 (LIB IDs)	FFT 4096	FFT 4096 DFT 2048 QRD 256	CONVEX HULL 1000	DFT 2048	FFT 4096	CONVEX HULL 1000	FFT 4096	DFT 2048	DFT 2048	FFT 4096 CONVEX HULL 1000	QRD 256	-------	FFT 4096 DFT 2048	DFT 2048 QRD 256	DFT 2048	DFT 2048 QRD 256
BENSIM 2 (LIB IDS)	DFS 3000 GLUE 1000	DFS 3000	BITONIC 1000 GLUE 1000	GLUE 1000	DFS 3000	----	BITONIC 1000 GLUE 1000	------	DFS 3000	DFS 3000 BITONIC 1000	---	-------	DFS 3000	BITONIC 1000	DFS 3000	GLUE 1000
BENSIM 3 (LIB IDs)	SVD 52 X 52	------	SVD 52 X 52	QRD 64 X 64	GLUE 2000 MAT ADD 64 X 64	MATMUL 64 X 64 SVD 52 X 52 MATSUB 64 X 64	------	MATINV 64 X 64 CHAINMAT ADD 64 X 64 QRD 64 X 64	---	---	---	MATMUL 64 X 64 MATINVR 64 X 64 CHAINMAT ADD 64 X 64 CROUT 64 X 64	------	------	MATMUL 64 X 64 MATINVR 64 X 64	SVD 52 X 52

Figure 6. Simulation Results

TRACKING OF DATA/INSTRUCTIONS TILL THE NODE PLANE DURING SIMULTANEOUS EXECUTION OF MULTIPLE APPLICATIONS IN MIPSCOC CLUSTER



FORMAT FOR TRACKING MULTIPLE APPLICATIONS EXECUTION ACROSS CLUSTER

INTERMEDIATE FORMAT

APPL ID	LIB ID	SUB-LIB ID	COMPUTATION INVOLVED	COMMUNICATION COMPLEXITY	LIB TYPE	PRIMARY HOST ID	LOGICAL ADDRESS 1	LOGICAL ADDRESS 2

FORMAT 2

APPL ID	LIB ID	SUB-LIB ID	COMPUTATION INVOLVED	COMMUNICATION COMPLEXITY	LIB TYPE	PRIMARY HOST ID	SECONDARY HOST ID	LOGICAL ADDRESS 1	LOGICAL ADDRESS 2

FORMAT 1

APPL ID	LIB ID	SUB-LIB ID	COMPUTATION INVOLVED	COMMUNICATION COMPLEXITY	LIB TYPE	PRIMARY HOST ID	SECONDARY HOST ID	NODE ID	LOGICAL ADDRESS 1	LOGICAL ADDRESS 2

DATA PACKET FORMAT

APPL ID	LIB ID	SUB-LIB ID	NODE ID	DATA PCK ID	DATA TYPE	DATA FIELD	FOOTER

INSTRUCTION PACKET FORMAT

LIB ID	SUB-LIB ID	TOTAL INS	PCK SIZE	DEPENDENT INS	PARALLEL INS	INSTRUCTION FIELD	FOOTER

Figure 8. Data and Control Flow in the MIP Cluster

ing to the heterogeneity characteristics of the MIP node able to execute diversified workloads. This fact decreases the runtime sacrifice at the cluster level when the SMAPP model is used.

5 Conclusion

In this paper, we have proposed a novel architectural model for high performance clusters. In the beginning we had indicated the need to develop a more efficient model for future high performance application execution to pave the way for cost sharing. We primarily discussed a model for multiple applications simultaneous execution on all partitions of the cluster, unlike in the conventional. We had delegated the cluster operating system role to a hierarchical host system developed to tackle the complexities (ranging from parallel mapping of the multiple applications onto the cluster to track the execution of these applications on the node) posed by this execution strategy. A cluster design having a strong node architecture based on MIP paradigm would help realize performance scalable Cost-Effective Supercomputing in its entirety while delivering enhanced cluster performance.

References

[1] PETER HILDEBRAND (CHAIR), WARREN WISCOMBE (SCIENCE LEAD) ET. AL,"Earth Science Vision 2030 Predictive Pathways for a Sustainable Future" *NASA Working Group Report*

[2] VENKATESWARAN et. al , "Homogeneously Structured Heterogeneous Very Large Functional Cores Node Architecture", *WARFT Internal Report* www.warftindia.org/papers08/node.pdf

[3] VENKATESWARAN et. al , "Impact Of Higher Level Functional Units In Heterogeneous Node Architecture:Performance And Power", *WARFT Internal Report* www.warftindia.org/papers08/hlfu.pdf

[4] DAVID H. BAILEY, ALLAN SNAVELY, " Performance Modeling: Understanding the Present and Predicting the Future" in *Lawrence Berkeley National Laboratory (University of California) Year 2005*

[5] NIRANJAN KUMAR SOUNDARARAJAN, "Hardware Compilation concept for the MIP S.C.O.C and Hierarchically-based Multiple Host System for the MIP cluster", *A Thesis Submitted to Waran Research Foundation 2002* www.warftindia.org/thesis/niranjan.pdf

[6] KARTHIK GANESAN, "Hierarchical Multihost based operating System for simultaneous Multiple application Execution on MIP SCOC Cluster" *A Thesis Submitted to Waran Research Foundation 2006* www.warftindia.org/thesis/gk.pdf

[7] VINOTHKRISHNAN ELANGOVAN, "Architecture of SILICon Operating System (SILICOS) for MIP SCOC Cluster" *A Thesis Submitted to Waran Research Foundation 2006* www.warftindia.org/thesis/evk.pdf

[8] SHYAMSUNDAR GOPALAKRISHNAN, "Memory In Processor Based Heterogeneous Multi-Core Node Architecture For Supercomputing Clusters" *A Thesis Submitted to Waran Research Foundation 2007* www.warftindia.org/thesis/shyam.pdf

[9] A BENCHMARK SIMULATOR"A benchmarking tool developed by the research trainees at WARFT, " www.warftindia.org/tools/bensim.zip

[10] FABRIZIO PETRINI AND WU-CHUN FENG "Buffered Coscheduling: A New Methodology for Multitasking Parallel Jobs on Distributed Systems," IPDPS, p. 439, 14th International Parallel and Distributed Processing Symposium (IPDPS'00), 2000