

The Use of Cluster Performance Matrix For Evaluating Computing Clusters

Sarmad Alshawi, Zahir Irani and Ayad Jassim¹

Information Systems Evaluation and Integration Group
Department of Information Systems and Computing
Brunel University, Uxbridge,
Middlesex, UB8 3PH, UK

¹Silicon Graphics Inc.

ABSTRACT:

Many organisations are increasing their expenditure on Information Technology (IT) to obtain or even sustain a competitive advantage in their respective marketplaces. The inability of managers to select appropriate computing hardware are considered attributable to a lack of appropriate decision aids that might act as a framework culminating knowledge and understanding about the true performance of the IT. In developing a broader understanding of the need for a decision aid to support the evaluation of computing hardware clusters, the authors of this paper introduce a novel model for presenting performance data. In doing so, supporting IT specialists in the evaluation of computer clusters. The need for such a novel decision aid is justified as computing clusters can vary significantly in terms of their hardware and software configuration. Such diversity makes the process of comparing cluster performance a very complex procedure. The model presented in this paper is based upon the notion of a 'Cluster Performance Matrix', and its derived metrics.

Keywords: Cluster; IT Hardware Evaluation; Cluster Performance Matrix.

Introduction

In recent years there has been a growing interest in clustering small/medium sized computers to build inexpensive parallel computing networks. However, as the world of computing evolves, business perception of Information Technology (IT) must change to incorporate new ideas, techniques, and tools. In doing so, supporting the uptake of this innovative approach to hardware configuration. Clustering is a popular strategy for implementing parallel processing applications. The reason for this is that it enables companies to leverage the investment already made in Personal Computers (PCs) and workstations. In addition, it is relatively easy to add new processors simply by adding a new PC to the network. Clearly, the advent of networking, high performance computers, and with freely available cluster computing tools all being widely available, most industrial institutions have the opportunity to incorporate into their business processes high quality IT performance utilising cluster computing.

Regardless, there remain many organisations that need direction in *what* and *how* to utilise and acquire cluster computing, which involves IT evaluation at a tactical level [1]. So far, the process of cluster selection has been based on ad-hoc decision making processes, influenced to a large extent by the vendor's sales strategy. Such strategies involve the presentation of a problem supplied by a possible customer, with a particular cluster performance database. The use of such an approach often catches the customer's interest by showing that their product would deal with the customer's job more efficiently than their competitor products.

Changes in technology, such as cluster computing are constantly affecting businesses; from the products they produce to the management of technology that is used to manufacture them. Technology is also changing the way that organisations use IT/IS. Therefore, there is a clear need for an appropriate decision making aid that acts as a framework for management thus, supporting the evaluation of new technology. As a result, this paper proposes to demonstrate the need for a decision aid to support the evaluation of computing hardware clusters, which will be presented through a novel model. The need for such a novel decision aid is justified as computing clusters can vary significantly in terms of their hardware and software configuration. In addition, there may be several combinations where a parallel job can allocate its threads among cluster nodes, and the number of nodes used. Such diversity makes the process of comparing cluster performance a very complex procedure.

The model presented by the authors of this paper is based upon the notion of a 'Cluster Performance Matrix', and its derived metrics. A numerical example based on a computational fluid dynamics application is used to demonstrate the applicability of the underlying model when using two different hardware clusters.

Business IT investment strategies

The adoption of IT is one of the most important issues facing manufacturing and service industries, with its responsive development and implementation for example, supporting the penetration of supply chains. Yet, most management executives are not comfortable with the available set of tools and techniques used to justify their investments in IT [2]. Such techniques lack the preciseness in definition and results that management expect. In support of this, Irani and Love [3] have found that management tends to be myopic when approaching IT investment decisions, primarily because they have no robust framework to evaluate their IT investments at strategic, tactical and operational levels.

Regardless, substantial amounts of money continue to be spent on the purchase of IT, in the anticipation that in some way the deployment of new technology will offer itself as a panacea. However, the evaluation of IT is an integral part of an information systems life-cycle [4] but remains subjective in approach. Yet, by adopting a structured evaluation process companies can better improve their business performance [strategic, tactical and operational] and develop a culture of knowledge/information management and sharing.

As more companies adopt IT, many are increasingly expressing their difficulty in its evaluation, and identifying the complexity associated with its decision making. For example, investments in cluster computing offer the opportunity to reap a wide variety of benefits and savings in terms of increased computing performance yet, with the nature and size of such benefits causing much controversy as there remains no consistent frame of reference.

Cluster Computing

The basic definition of a cluster is connecting two or more computers together in such a way that they behave like a single computer. Clustering is used for parallel processing, for load balancing and for fault tolerance [5,6]. In the past decade there has been a dramatic shift from mainframe or 'host-centric' computing to a distributed 'client-server' approach. In the next few years the authors suggest that this trend is likely to continue with further shifts towards 'network-centric' computing becoming apparent.

Clustering is a popular strategy for implementing parallel processing applications because it enables companies to leverage the investment already made in PCs and workstations. In addition, it is relatively easy to add new processors simply by adding a new PC to the network.

Clustering essentially creates a quick-response, intelligent backup system in case one of the servers, or one of its components, fails. Under clustering, two or more servers are tied together through communications lines. If one fails or goes down, other servers in the cluster pick up the computing load. Users, as a result, suffer far fewer network outages. Cluster computing is a rapidly maturing technology that seems certain to play an important part in the 'network-centric' computing future.

Cluster Performance Assessment

To reduce development cost, vendors are generally not releasing complete new systems for server clustering. Rather, they are modifying existing hardware to accommodate clustering. Helped by availability and affordability, clusters have become increasingly popular for high performance computing. Consequently, performance assessment became an important subject when comparing different clusters. Clusters can vary significantly in terms of their hardware and software configuration. In addition, there may be several combinations in which a parallel job may allocate its threads among cluster nodes and number of nodes used [6,7,8]. Such diversity makes the process of comparing clusters performance a very complex procedure even when presented in the form of a benchmark performance table. This complexity will be highlighted in subsequent sections of this paper.

Technical Background

Traditionally clusters were known to be a notion that referred to a number of workstations networked together to run an application using some message passing library such as PVM [9] and MPI [10]. The main objective was to enable the application to solve large inputs whose minimum memory requirements could only be met by using the whole memory on a particular number of hosts/nodes in a cluster. Originally, clusters were networked via older slower forms of networking hardware, such as 10bT Ethernet, which meant that clusters performance could not be achievable in a scalable form that is competitive to traditional shared memory multi processing architectures [11]. In addition, message-passing software, e.g. PVM and MPI, in early releases, were not efficient and reliable enough so as to make clusters performance an attractive feature. In recent years, computer technology has improved in three ways that have made clusters performance a desirable corporate objective. Namely,

- The production of multi processor systems at the medium to low end hardware
- Vast improvements and innovations in network communications hardware (such as 100bT Ethernet, Hippi, FDDI, Myrinet, Gigaset, Dolphin, etc.) and,
- Improvements in message-passing software, mostly MPI based, customised and tuned for various hardware and network types.

The result is that clusters are now able to show scalable performance good enough to justify their price/performance ratio [11, 12]. At the high-end, such clusters became capable of meeting time to solution requirements for very large problems by inter-connecting multiprocessor supercomputers to form cluster nodes. In the past few years a form of multiprocessor supercomputers, known as CC-Numa has appeared. CC-Numa stands for Cache-Coherent Non-uniform memory access and refers to multi-processor architectures that are physically distributed but globally addressed as a one shared memory system [ref, ref]. CC-Numa was first introduced by in 1996 under the Origin2000 (used in our example) product family and later evolved under Origin3000 architecture [13]. Each Origin2000 node contains one, two or four processors, memory, directory memory for cache coherence, I/O interface, and an interface that links it to system nodes/routers. The interconnection fabric is a mesh of one or more routers with multiple links that link the nodes so that many processors can communicate simultaneously.

The Origin2000 product family produced performance for many parallel applications, scalable on average to a large number of processors using, for example, PVM, MPI and OPENMP. Many applications were able to scale to a factor of 64X or more. At the low end of the Origin family, another form of CC-Numa systems emerged as the Origin2000. Such CC-Numa systems were designed to have a maximum of 4 to 8 processors per system. The result was that such low end CC-Numa designs became suitable and affordable candidates for a cluster computing environment.

As a result both CC-Numa and Non CC-Numa type clusters became an attractive solution for cost effective high performance computing. Accordingly clusters performance assessment becomes an important factor when deciding on selection of a cluster from what is being offered from different vendors. However, such selection is not a straightforward process as different clusters (that may be similarly priced) can vary significantly in terms of their hardware and software configuration. The most common observation about CC-Numa and Non-CC-Numa clusters is that both types can have more than one processor on each of their individual nodes. In the light of cluster performance evaluation, this can give rise to more than one performance curve for the single cluster depending on how many threads per node an application is likely to use for a particular run [11, 12]. Furthermore, there may be several combinations in which a parallel job may choose to allocate its threads among cluster nodes

and the number of nodes it uses. This makes the process of clusters performance evaluation to become even more complex.

Example Case

As an application example we chose a data intensive application in the area of Computational Fluids Dynamics (CFD). Using STAR-CD [13] as the underlying application software. In this case it is used to solve the flow field equations for the airflow in a tunnel and its effects on the aerodynamics around a train body. The equations are solved to calculate the velocity vectors, pressure, turbulence and energy dissipation. The case has been run on two clusters as listed in table 1. These clusters were connected using Ethernet, Myrinet and Gigaset interconnects for the Linux cluster and Hippi for the Origin2100 cluster. The example setting assumes a homogeneous cluster (all nodes are of the same hardware and software configuration), even number of processors, equal number of threads per cluster node and one performance observation per cluster.

ia32 1200 Linux Cluster: 8 nodes - "A Non CC-NUMA Cluster" 2 X Pentium III 700Mhz w 256KB L2 1GB Memory per node linked using Ethernet, Myrinet and Gigaset.
Origin 2100: " A CC-NUMA Cluster" 8X 350Mhz R12000 MIPS 4MB L2 4 Nodes: 1GB Memory per node linked using Ethernet and Hippi.

Table 1: Hardware platforms used to run the CFD experiment

Table 2 presents elapsed times (in seconds) recorded after running the model on both hardware settings described. This table represents a traditional standard format that most cluster vendors provide to present performance benchmark information.

	ia32 Linux Cluster				Origin2100 Cluster		
	Pentium III, 700Mhz 512Kb L2				MIPS R12K,350Mhz 4MB L2		
#CPU	Same Host	Myrinet	Giganet	Ethernet	Same Host	Hippi	
1	75028	-	-	-	66837	-	-
2	49118e	-	-	-	32323	42253	32518
4	-	25614e	25483e	25872e	15775	20396	20712
8	-	13201e	12542e	19114e	-	10262e	9858
16	-	7615e	6755e	slow!	-	-	4827
32	-	-	-	-	-	-	2928

Table2 : Elapsed times (seconds) for STARCD CFD Example running on an ia32 Linux Cluster and Origin2100 cluster.

From the users point of view, the job timing information provided by table 2, although appear complete, it does not reflect the whole of the allocated thread/node combinations of the parallel job that generated the data shown. For the full data to be shown in this format the resultant table will be too complicated to present and describe. Moreover, vendors tend to hide weaknesses [e.g. low bandwidth time but fast communications] in their products and present only final performance figures as depicted in the table.

For the Linux cluster, the 2-processor job was made on one host, i.e. not distributed. The three 4cpu Linux jobs (i.e. over Ethernet, Myrinet and Gigaset) were made on two hosts with two threads in each. The 8-processor jobs were made on four hosts with two threads in each. Similarly the 16-processor jobs were made on eight hosts, each host executing two threads. For the Origin2100, three types of timings are shown, namely on same host made as 1pN and 2pN runs, and clustered runs using Hippi link. The 2-processor over Hippi job was made using two threads, each thread ran on a different node. The 4-processor over Hippi job was made on two hosts with two threads in each. The 8-processor over Hippi job were made on two hosts with four threads in each. The 16-processor over Hippi job was made on four hosts, four threads in each. For the Origin2000, both for the 300Mhz and 400Mhz types, all runs were made such that all threads ran internally on the same host, with all runs also made as 2pN (two threads per node) jobs.

Examining the performance numbers of table 2, it becomes evident that comparing the performance of applications on different clusters may not be a straightforward comparison and can present a complex procedure. This complexity is caused by the cluster design fact that a cluster node may itself have more than one processor. For example, the Linux clusters involved in table 2 had two processors in each while the Origin2100 cluster had eight processors per cluster node. Any performance comparison (s) in order to be broadly useful will have to take into account the node/processor/thread combination facts. Clearly table 2, or any similar time only based structure will not be able to provide the performance data with relation to the required node/processor/thread information.

Cluster Performance Matrix

In this section we introduce a generic definition for a presentation model that will properly define our concept of "Cluster Performance Matrix". This definition is based on the number of threads per cluster node and the number of nodes a parallel job requires. A cluster performance matrix is one whose elements define all possible node, thread and processor combinations.

The main point is to present the resulting run times in a manner that makes performance comparison more representative of all possible combinations and at the same time remains reasonably informative.

Now let define the Cluster Performance Matrix, CPM, as a matrix of order $(K \times N_c)$ and for a set of a job elapsed times E_{ij} ; $i=1, \dots, K$; $j=1, \dots, N_c$.

$$\text{CPM} = \begin{matrix}
 & E_{11} & E_{12} & \dots & E_{1j} & \dots & E_{1N_c} \\
 E_{21} & E_{22} & \dots & E_{2j} & \dots & E_{2N_c} \\
 \cdot & \cdot & \dots & \cdot & \dots & \cdot \\
 \cdot & \cdot & \dots & \cdot & \dots & \cdot \\
 E_{i1} & E_{i2} & \dots & E_{ij} & \dots & E_{iN_c} \\
 \cdot & \cdot & \dots & \cdot & \dots & \cdot \\
 E_{K1} & E_{K2} & \dots & E_{Kj} & \dots & E_{KN_c}
 \end{matrix}$$

Where:

K is the number of processors per cluster node and N_c is the number of cluster nodes.

The generic CPM shows:

All elements lying on any diagonal represent elapsed times for jobs using the same number of processors.

All elements lying on the j th column ($j=1, \dots, N_c$) represent elapsed times for jobs running using the same number of cluster nodes.

All elements lying on the i th row ($i=1, \dots, K$) represent elapsed times for jobs using the same number of threads per cluster node.

Cluster Performance Matrix for Example case

Going back to our example we first utilize a cluster of several Origin2100 nodes. Because an Origin2100 node can have up to 8 processors, there are a larger number of combinations for a particular run in terms of the possible ways it can distribute its threads among the nodes in the cluster. For example, an 8-processor job in a cluster of 8 Origin2100 nodes might have the following major combinations:

- 1 thread on each of the 8 nodes
- 2 threads on each of any 4 nodes
- 4 threads on each of any 2 nodes

Plus other minor combinations, e.g. 2 threads on each of any 2 nodes and 4 threads on a 3rd node.

Table 3 defines a Cluster Performance Matrix for an Origin2100 cluster in which we include only the major combinations as its elements. In this matrix the ij -th element represents the elapsed time for a cluster based run using a major combination of the i th row number multiplied by j th column number as the total overall number of processors used by that run.

	1	2	4
1	66837	32518	16224
2	42253	20712	9994
4	20396	9858	4827
8	10262	4905	2928

Table 3: Origin2100 Cluster Performance Matrix

For example, the (4,4) element is the elapsed time for a 16-processor job made using 4 nodes (the column index) and using 4 threads each (the row index). The (8,2) element is also a wall elapsed time for a 16 processor job made using 2 nodes and using 8 threads on each of the two nodes.

Results from running the same example job using an ia32 1200 Linux cluster are shown in tables 4a 4b and 4c. Each cluster node had two processors on board. For this cluster we may have the following three Cluster Performance Matrices depending on the networking hardware used to connect the cluster. In this case Myrinet, Giganet. And Ethernet communication hardware (4a, 4b and 4c respectively).

	1	2	4	8
1	75028	39782	18967	9277
2	49118	25065	13201	7615

Table 4a: ia32 Linux/Myrinet Performance Matrix

	1	2	4	8
1	75028	-	-	-
2	49118	25483	12542	6755

Table 4b: ia32 Linux/Giganet Cluster Performance Matrix

	1	2	4	8
1	75028			
2	49118	25872	19114	too slow

Table 4c: ia32 Linux/Ethernet Cluster Performance Matrix

Example Discussion

Examining the Cluster performance Matrices of tables 3 and 4 enable us to compare the performance of the two types of clusters using a set of data that represent several possible combinations in which this experiment may run on these clusters. This comparison is primarily based on comparing the corresponding elements for the matrices involved. This shows how a Cluster Performance Matrix can provide performance information in a manner that may be very difficult to present otherwise. For example in table 3, the (4,1) and (1,4) elements both represent the elapsed time for 4 processor jobs, where the former element resulted from using four threads running on the same host and the latter resulted from using

four hosts in the cluster. At first it may appear surprising that element (4,1) is larger than (1,4). This is because in the former element there is no network communication overhead as all four threads are running on the same host. Element (1,4) however, although ran distributed over four cluster nodes, may be seen as one thread executing in each of the four nodes. This means it ran fully standalone within each cluster node giving it the advantage of enjoying the full Origin200 processor-to-hub bandwidth. This turns out to be more advantageous as long as cluster communication overhead is small.

Derived Metrics

The interpretation for the elements of the Cluster Performance Matrix may help derive several related performance metrics that can aid in the selection process by supplying the performance data in a graphical way. In this paper we introduce the following metrics:

Cluster Performance Surface is a 3D plot of a surface where the X-axis represents the number of threads, the Y-axis represents the number of Cluster nodes and the Z-axis represent Jobs Elapsed Times. The coordinates for a point on this surface are:

x: Number of Threads used Per Cluster Node (i.e. Y_pCN), which is the row index for the Cluster Performance Matrix.

y: Number Cluster Nodes used, which the column index for the Cluster Performance Matrix.

z: The job's elapsed time corresponding to the particular x and y ordinates which is the Cluster Performance Matrix ij-th element. Fig 1 shows the Cluster Performance Surfaces for Origin2100 and Linux clusters.

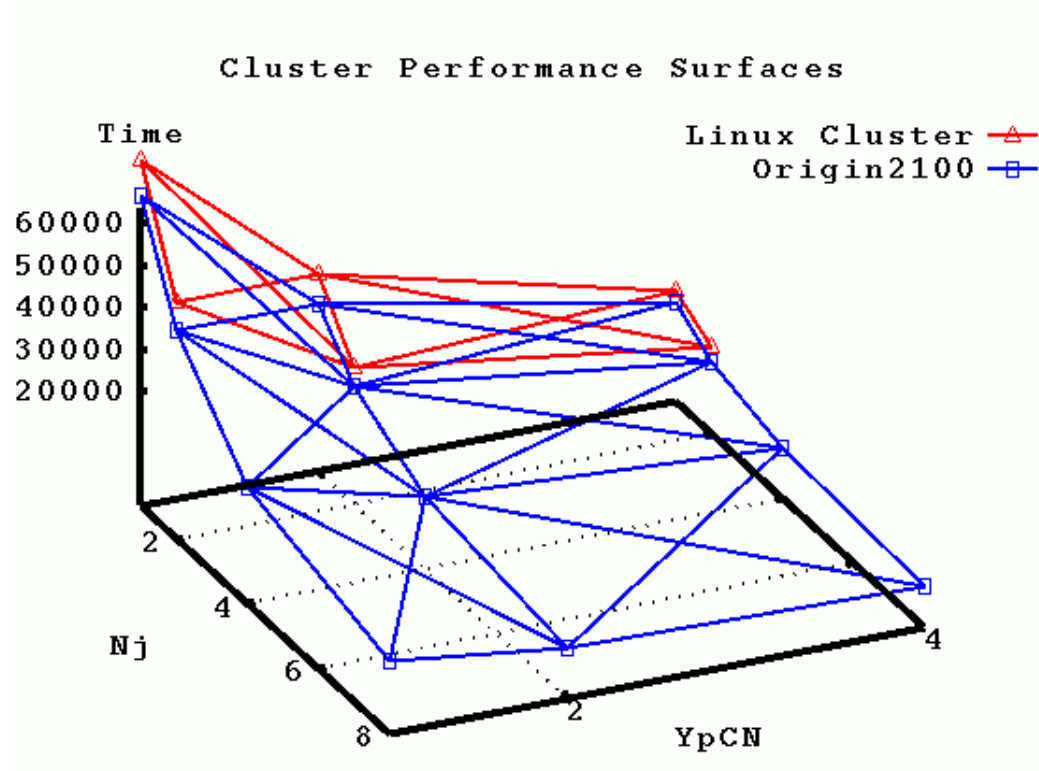


Figure 1: Cluster Performance Surfaces for Origin2100 & ia32 Linux clusters

Cluster Performance Scatter diagram is a 2D plot where the X-axis represent the total number of processors (i.e. T_p) used by a job and the Y-axis being the Elapsed Time. Each point on this plot corresponds to an element in the corresponding Cluster Performance Matrix.

Cluster Performance Average Curve is a curve where each point on it is the average of elapsed times for jobs using the same number of processors. That is the average of all elements lying on the same diagonal in the Cluster Performance Matrix.

Other statistical indicators may also be derived (e.g. ratios, logs and 3D surface diagrams), however, for the purpose of this paper, only the above two diagrams are produced.

Using information from tables 3 and 4, figures 2a and 2b show the Cluster Performance Scatter and Average Curves diagrams for the Origin2100 and ia32 Linux clusters respectively.

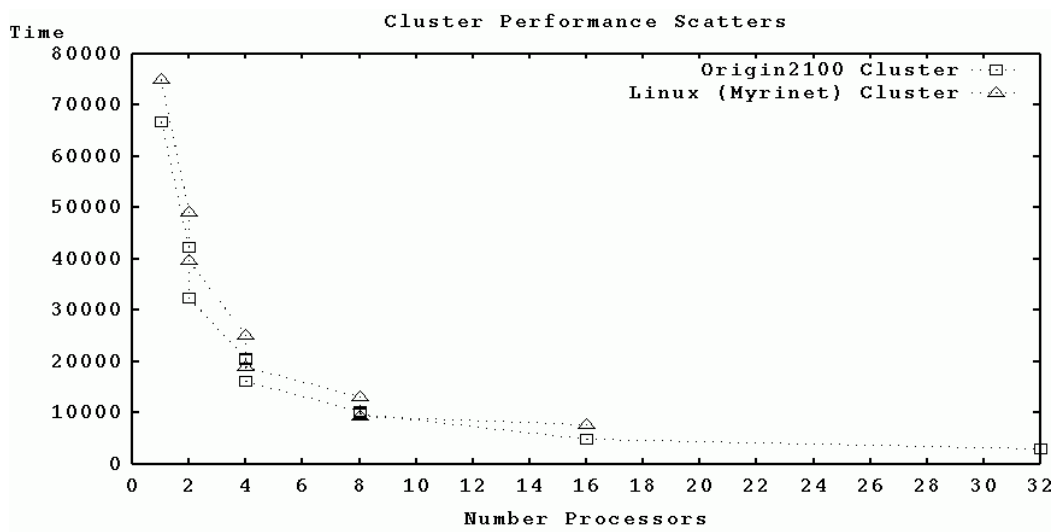


Figure 2a: Cluster Performance Scatters for Origin2100 & ia32 Linux clusters

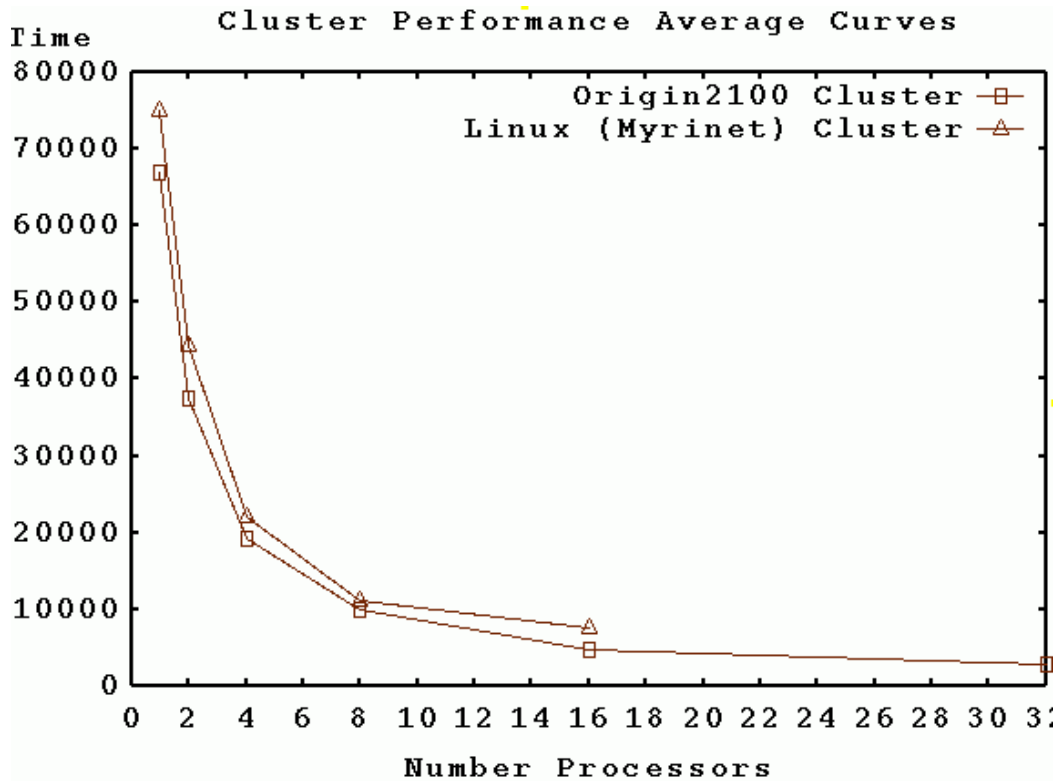


Figure 2b: Cluster Performance Averages for Origin2100 & ia32 Linux clusters

These figures show the amount of inference that can be obtained using the data provided by the Cluster Performance Matrix, thus providing a more detailed comparison for the performance of the two clusters. Note that the objective of this paper is not to provide a type of comparison upon which one cluster is favored versus another, but is only using the two named clusters as an example aid for demonstrating the underlying representation model. In fact although the results of tables 3 and 4 and Figure 2 (a, b), indicate that the Origin2100 cluster outperforms the Linux cluster by a certain amount, the latter remains an attractive solution, and is likely to become more scalable using future revisions of cluster communications hardware.

Concluding Comments

The authors of this paper have presented a case for using clustering, which remains a popular strategy for implementing parallel processing applications. The reason for this is that clustering enables companies to leverage the investment already made in PCs and workstations. In addition, it is relatively easy to add new processors simply by adding a new PC to the network. However, the evaluation of clustering remains an important and yet difficult process that complicates IT evaluation. The reason for this is that time based cluster performance tables are inappropriate because they do not have the ability to present performance data with relation to the cluster node/processor/thread combination. Knowledge of the combination data is vital for any cluster performance evaluator trying to reach an investment decision.

In addressing this, the authors define a method for presenting and comparing multiple combination performance data for various cluster types and architectures. This definition is based on the number of threads per cluster node and the number of nodes a parallel job requires. The method primarily presents itself as a table structure, which we call the Cluster Performance Matrix. The elements of the Cluster Performance Matrix present performance data in an informative and standard form suitable to most clusters including those with CC-Numa node type architectures. Using data from the Cluster Performance Matrix, several more detailed performance metrics such as the 'Cluster Performance Surface', 'Cluster Performance Scatter' and 'Cluster Performance Average Curve', can be derived for further performance comparison. Other performance metrics may also be derived depending on the evaluation requirements.

References:

- [1] Irani Z, Ezingear J-N, Grieve R.J and Race P. 1999. 'Investment justification of information technology in manufacturing'. *The International Journal of Computer Applications in Technology*, **12**(2): 90-101.
- [2] Parker M.M and Benson R.J. 1989. 'Enterprise-wide information economics: Latest concepts'. *Journal of Information Systems Management*, **6**(4): 7-13.
- [3] Irani Z and Love P.E.D. 2000. 'The propagation of technology management taxonomies for evaluating investments in information systems' *Journal of Management Information System*, **17**(3): 159-175.
- [4] Kumar K. 1990. 'Post implementation evaluation of computer-based information systems', *Communications of the ACM*, **33**(2): 203-212.
- [5] Wilkinson B. and Allen C.M. 1998. 'Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers', Prentice Hall.
- [6] Arpaci-Dusseau A. C., Arpaci-Dusseau R. H, Culler D. E., Hellerstein J. M., and Patterson D. A. 1997. 'High-performance sorting on Networks of Workstations', In Proceedings of International Conference on Management of Data.
- [7] Carter R. and Laroco J. 1995. 'Commodity clusters: Performance comparison between PC's and workstations', In Proceedings of International Symposium on High Performance Distributed Computing.
- [8] Gets A., Beguile A., Dungaree J., Jingo W., Manteca R. and Sunderam V. 1994. 'PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing', MIT Press, USA.
- [9] Pacheco P. 1990. 'Parallel Programming with MPI', Morgan Kaufmann Publishers Inc, USA.
- [10] Pfister G.F. 1998. 'In Search of Clusters', Prentice Hall.
- [11] Buyya R. (ed.) 1999. 'High Performance Cluster Computing: Architectures and Systems', Prentice Hall.

- [12] Buyya R. (ed.) 1999. 'High Performance Cluster Computing: Programming and Applications', Prentice Hall.
- [13] Bronwell D. and Young D. 2000. 'SGI Origin 2000 Series Technical Configuration Guide', Document Number 007-4311-001, Silicon Graphics Inc, UK.
- [14] STARCD User Guide, Version 3.1.1999. 'Computational Fluid Dynamics Limited', London, UK.