Passive Identification of Under-Utilized CPUs in High Performance Cluster Grid Networks

Lanier Watkins
Department of Computer Science
Georgia State University
Atlanta, GA, USA

Raheem Beyah
Department of Computer Science
Georgia State University
Atlanta, GA, USA

Cherita Corbett
Sandia National Laboratories
Livermore, CA, USA

Abstract—In this paper we propose a passive approach to using network traffic to discover the availability of resources in local distributed networks (e.g., Cluster Grids, Campus Desktop Grids, etc.). To our knowledge, this is the first approach of its kind. The ability to quickly identify resource availability is critical because the presence of available resources directly affects the job execution time of a distributed environment. The proposed method creates a delay sensitive profile generated by the analysis of monitored network traffic, which emulates high performance UDP based grid services such as file transfer applications (FOBS, Tsunami, UDT, SABUL, etc.), message passing platforms (MPICHG2/Score, etc.), and others. An energy value is derived from the delay sensitive profile, which represents the state (over-utilized CPU or under-utilized CPU) of the resource of interest. Then a simple threshold (derived from initial calibrations on the over-utilized resources.) is applied to the energy value to identify the state of the resource. This method could be used to enhance existing resource discovery algorithms used in local distributed networks because this approach is capable of passively determining a major dynamic resource attribute - CPU utilization. The main benefits are the reduction in the necessary complexity associated with the use of non-passive algorithms (e.g., flooding algorithm, name-dropper algorithm, distinctive awareness algorithm, etc.) and the reduction in the extra network traffic that results from the continual need to determine the availability of dynamic resources. Since this method is passive in nature, there is no need to query potential resources directly to determine their availability to complete distributed computing related jobs. Results suggest that once the CPU utilization approaches 70% (unavailable) the network traffic produced by that node exhibits different behavior than when the CPU utilization is less than 70% (available).

Keywords—**grid computing, dynamic resource discovery, wavelet transform, passive resource discovery, CPU scheduling**

I. INTRODUCTION

A cluster is a collection of independent and cheap machines used together as a supercomputer to provide a solution [16]. In practice, this is implemented by applying grid middleware to clusters of computers to form a high-performance platform for distributed applications or simply a Cluster Grid. Also in [17], a local Desktop Grid is defined as a harvester of idle computing resources of desktop PCs. One issue central to distributed systems is how to locate resources [1]. Another equally important issue is determining a resource's availability. CPU utilization ratios between 60% and 70% are desirable to avoid damage to the node after prolonged computing use [2]. Accordingly, it may not be in a user's best interest to allow resources to be borrowed from their machine past this range. The high end of this range, less than 70% utilization, will be used to define availability throughout this paper, and unavailability will be defined as 70% or greater. The execution time of a CPU bound task on a host is tightly related to the CPU load on the host [3]. Thus, CPU availability is critical in job scheduling since scheduling jobs on unavailable nodes lead to longer execution durations which is undesirable in high performance distributed networks. In a distributed environment, applications compete for resources with unknown workloads. This contention for resources by users causes CPU utilization to greatly vary over time [12]. Thus the problem, the continual need to identify available resources in distributed networks. Leading methods either actively measure CPU utilization continuously or forecast CPU utilization based on previous active measurements. The method detailed in this paper is a different approach to determining CPU utilization. It could be used to enhance existing prediction and intrusive methods.

In this paper we focus on resource discovery in high performance cluster environments confined to low latency LANs that heavily utilize UDP for different services. One such service is file transfer via FOBS, Tsunami, UDT, SABUL, etc. [18]. Another is message passing via MPICH-G2/Score [19]. The method proposed in this paper is passive in nature and determines the availability of a node based upon the network traffic generated by that node. This non-intrusive method could enhance existing resource discovery algorithms, because there is no need to query potential resources directly for their CPU utilization information. Also, forecast-based methods could be enhanced by using the proposed non-intrusive method as a basis for its forecast, thus no longer needing to intrusively gather measurements. This would render the forecast-based methods passive, a characteristic not presently available. Section II discusses several intrusive and forecast-based resource discovery implementations that could be enhanced by adding a passive component.

This method treats CPU utilization as a binary variable. Using this design, the node is either available (CPU utilization less than 70%) or not available (CPU utilization of 70% or greater). This design can help facilitate dynamic load balancing across available resources, which protects the resources from becoming over-loaded or under-utilized. One limitation of this passive method is that its use is limited to periods of time when there is traffic on the network from the node of interest. This limitation can be offset by pinging the node of interest and using the proposed method to analyze the ICMP replies from the node. This does not require any query response software to be placed on the node given that the ping application is implemented in most operating systems.

The remainder of this paper is organized as follows. Section II discusses related work. Section III presents an introduction to signal processing by using the Discrete Wavelet Transform. In Section IV a closer look is taken at the inner-workings of the node to attempt to characterize the source of the delays that distinguish available nodes from unavailable nodes. Section V explains the experimental setup and the experimental procedure. Section VI discusses the simple identification scheme used to discern available nodes from unavailable nodes. Section VII concludes the paper and discusses future work.

II. RELATED WORK

A. *Intrusive Resource Discovery Methods*

A survey of resource discovery algorithms is discussed in [4]. The flooding algorithm, which is similar to the method used by network routers to advertise routes, assigns to each node a fixed set of neighboring nodes, and each node contacts its fixed set of neighbors and transmits the updates to them. The swamping algorithm is very similar to the flooding algorithm except that each node may open connections with all of their neighbors, not just a fixed set of neighboring nodes. Another algorithm, the random pointer jump algorithm should only be used if there exists a path between every pair of machines. This algorithm mandates that each node contacts a random neighbor and the chosen neighbor then sends resource information to the contacting node. Finally, the name-dropper algorithm works as follows: each node sends information to one, randomly chosen neighbor then every time a pointer jumps, a back edge is added. For example, when node A chooses node B and node B passes to A all of its neighbors, node B also obtains a pointer back to A.

In [13] the proposed resource discovery system is based on the peer-to-peer (P2P) model and provides a complex query interface. It supports rich resource descriptions and complex queries by encoding resources and queries with the Resource Description Framework (RDF). To avoid flooding queries to irrelevant nodes, a semantics-based routing scheme is proposed to only route queries to related nodes.

A data dissemination strategy called distinctive awareness is introduced in [5]. In this algorithm, the nodes with distinct attributes are more significant and thus their status information gets propagated accordingly. To implement this algorithm, the concept of Grid Potential is used to encapsulate the relative processing capabilities of different machines and networks that constitute the grid and only data corresponding to nodes with the highest grid potential gets disseminated.

These intrusive methods gather static resource information (physical memory size, CPU speed, total hard disk space, etc.) as well as dynamic resource information (CPU utilization, available disk space, available memory, etc.). The dynamic resource information changes frequently, therefore requiring continuous update. The requirement for continuous update of the dynamic resource information produces additional traffic on the network and additional CPU load on the resource. This is highly undesirable in a high performance networking environment.

B. *Resource Prediction Methods*

Network Weather Service (NWS) is a performance monitoring and prediction service that is based on the concept of performance sensors, which actively gather information about the available resources and use them to predict performance. Specifically, past measurement values are treated as time series that are analyzed by simple statistical techniques to make short-term forecasts [14].

In [12] the CPU load for computational grids is forecasted using a one-step-ahead load prediction strategy. This strategy is based upon the tendency (to increase or decrease in several previous measurements) several steps backward, and uses polynomial fitting to produce the prediction value. Another interesting predictive method proposed by [15] uses homeostatic (assumes the average of time series remains constant) and tendency-based methods (assumes the time series retains its increase or decrease tendency). In this method, the predicted value is adjusted according to the magnitude of the last load measurement and the last prediction error. The method outperforms [14] but is inferior to [12] due to the use of the tendency several steps backwards versus one-step backwards and the use of polynomial fitting versus linear increment for predictive value creation.

In the above-mentioned prediction methods, CPU availability is derived from recent intrusion upon the resource of interest. Therefore, they are not passive in nature.

Many of the aforementioned resource discovery related mechanisms are very powerful and feasible methods. However, they all rely upon direct interaction with nodes. The approach detailed in this paper introduces a different type of resource discovery algorithm that is passive in nature; thus no nodes must be probed to obtain resource information.

III. SIGNAL PROCESSING

Similar to the Fourier Transform, which uses the sine and cosine functions to express the original signal, the DWT expresses the original signal in terms of the chosen wavelet, and the inverse DWT reproduces the original signal exactly. There are many wavelets in existence and custom wavelets can be created by ensuring that the proposed wavelet meets the admissibility condition [6], which essentially says that the proposed wavelet is of zero mean. The Haar Wavelet is the simplest wavelet, which only calls for the convolution of the original signal with a low pass filter to produce the Approximated Coefficients and a convolution of the original signal with a high pass filter to produce the Detailed Coefficients. The Approximated Coefficients contain all of the low frequency information of the original signal, and the Detailed Coefficients contain all of the high frequency information of the original signal. The Haar DWT of a signal is given in (1), (1a) are the Approximated Coefficients and (1b) are the Detailed Coefficients. The subscripts j and k refer to the scale of the DWT and index of the signal respectively [7]. In this paper, the first level (j=1) Haar DWT is used solely.

$$a_{j,k} \equiv \frac{1}{\sqrt{2}}(s_{2k-1} + s_{2k}) \qquad (1a)$$

$$d_{j,k} \equiv \frac{1}{\sqrt{2}}(s_{2k-1} - s_{2k}) \qquad (1b)$$

Given a time series s(t), constructed using the process discussed in Section III, the majority of the energy in the wavelet domain is captured by computing the magnitude squared of the Approximated Coefficients taken from the DWT of the time series. There is conservation of energy from Parseval's Energy Theorem for Wavelets [8] such that the sum of the energy of the Approximated Coefficients and the Detailed Coefficients equal the energy of the input signal s(t). This is given in equation (2), where L and t are the length and index of the input signal respectively and k is the index of the wavelet coefficients.

$$\sum_{k=1}^{L/2} \left|a_{1,k}\right|^2 + \sum_{k=1}^{L/2} \left|d_{1,k}\right|^2 \equiv \sum_{t=1}^{L} \left|s(t)\right|^2 \qquad (2)$$

The Approximated Coefficients are the average of the original signal, except they are one half of its length. The Detailed Coefficients are also one half the length of the original signal, and contain information about transients (sharp changes) in the original signal. By using the DWT a delay sensitive profile can be created. The Approximated Coefficients can be used to obtain an energy value for each time series. Also the Detailed Coefficients can be used to obtain transient information about each time series. Together these two pieces of information comprise a delay sensitive profile that is used to classify time series created by nodes of either available or unavailable nodes. Further elaboration on the use of this delay sensitive profile and the experimental setup is given in Section V.

IV. COMPUTER ARCHITECTURE DISCUSSION

To better understand the source of the delays induced in the network traffic produced by nodes subjected to CPU utilization, the high level process of creating a packet in a Pentium based architecture is explored. This process involves many internal parts of the node working together; the CPU, the physical memory, the front side bus, the PCI bus, the network card, etc. The operating system along with the CPU creates a buffer descriptor in main memory, which contains the starting memory address and length of the packet that is to be sent. If the packet consists of multiple discontiguous regions of memory, multiple buffer descriptors are created. This communication to main memory is done via the front side bus. The CPU then writes to a memory-mapped register on the Network Interface Card (NIC) with information about the new buffer descriptors. This data traverses the front side bus through the Northbridge to the PCI bus. The NIC initiates one or more direct memory access (DMA) transfers to retrieve the descriptors. Then, the NIC initiates one or more DMA transfers to move the actual packet data from the main memory into its transmit buffer using the address and length information in the buffer descriptors. This data again leaves the front side bus, travels through the Northbridge to the PCI bus into the NIC. The NIC informs the operating system and CPU that the descriptor has been processed. After the packet is transferred, the NIC sends the packet out onto the network through its medium access control (MAC) unit [9].

As stated above, the creation of network traffic involves many internal processes of a node working together. This drives the process of paging, buffering and CPU use, which induce very small but detectable delays. These delays form a unique signature relative to the CPU load on the node.

V. EXPERIMENTAL SETUP AND PROCEDURE

A. Experimental Setup

The experimental setup is composed of three nodes (see Figure 1). Two nodes are IBM NetVista 2 GHz Pentium IVs (Monitor node and Node 1) and one node is a Dell OptiPlex GX200 866 MHz Pentium III (Node 2). The monitor node uses tcpdump to capture the time stamp of packets as they leave the node of interest toward the sink node. Node 1 and Node 2 interchange as the node of interest to produce UDP and ICMP results. These nodes emulate UDP traffic on a grid network. All nodes are installed with Linux Red Hat 7.3 and have no applications running on them unless the experimental procedure requires the addition of a CPU load. For ease of packet capture a hub is used; however, in practice the monitor node would be placed on a mirror port of a switch.

B. Experimental Procedure

The purpose of the experimental setup is to emulate a high performance, low latency cluster environment that heavily utilizes UDP for different services. To do this, the sock program [10] is used to generate UDP network traffic. The timestamps captured and stored by the monitor node are used to build a time series that describes the departure times in seconds for the UDP packets that leave the node of interest.
Considering situations where nodes are less chatty and little or no network traffic is available for observation, a node can be pinged to induce traffic. To emulate this case, the monitor node uses the standard ping application to generate source traffic (ICMP replies) from the node of interest.Once the necessary traffic has been observed and the corresponding time series has been created, the next step is to use the Discrete Wavelet Transform (DWT) to create a delay sensitive profile. A threshold is applied to the energy value

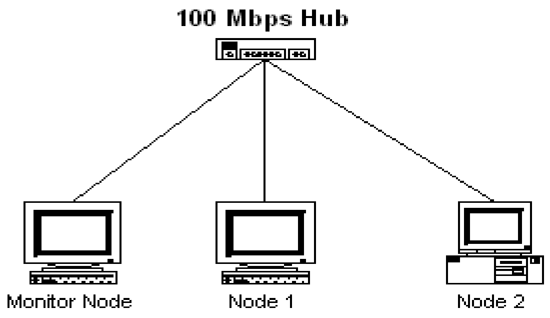

Figure 1. Experimental Setup.

taken from this delay sensitive profile, which is used to numerically classify the corresponding time series. We investigate the effect of CPU utilization on network traffic. Traffic is sent from the node of interest to the sink node varying the CPU load between 0% and 100% in steps of 20%. These tests are repeated 100 times.

The desired amount of CPU utilization is achieved by executing multiple instances of a script that continuously compiles a small program using the gnu complier. The CPU utilization is verified via the Linux vmstat application. The CPU utilization was observed to fluctuate about 10% during each experiment; therefore the specific ranges that were tested are: 0%-10%, 20%-30%, 40%-50%, 60%-70%, 80%-90% and 100%. Note that all of the experiments labeled 60%-70% CPU load are defined to not include 70%, thus all of these experiments are classified as available. Therefore, the energy threshold is chosen such that available nodes are below this threshold value and unavailable nodes would have an energy equal to or above this value.

C. Passive Resource Characterization Algorithm (PRC)

The diagram in Figure 2 provides a high level flow of the PRC Algorithm. In our previous work, the PRC was used to detect memory utilization using network traffic [20].

Network Traffic Analysis- The tcpdump packet sniffer application is used to capture the network packets produced by the node of interest. The time series is created using the timestamp from each packet that departs the node of interest. The algorithm works with very few packets (520 packets for ICMP and 1024 for UDP).

Pre-Processing- The wavelet transform separates the time series into Approximated Coefficients and Detailed Coefficients. Together, these two sets of coefficients comprise the delay sensitive profile. The Detailed Coefficients contain very important information; however for the purposes of this paper, they are discarded and only the Approximated Coefficients are used. The Detailed Coefficients may prove helpful in conjunction with the use of Fuzzy Logic or Neural Networks for classification and detection.

Feature Extraction- An energy value is calculated from the Approximated Coefficients (see Section III). This energy value is used to represent the delay strength of the corresponding time series.

Detection and Decision- In Section VI, the PRC algorithm is used along with a simple threshold on the energy value to decide between under-utilized and over-utilized nodes. The

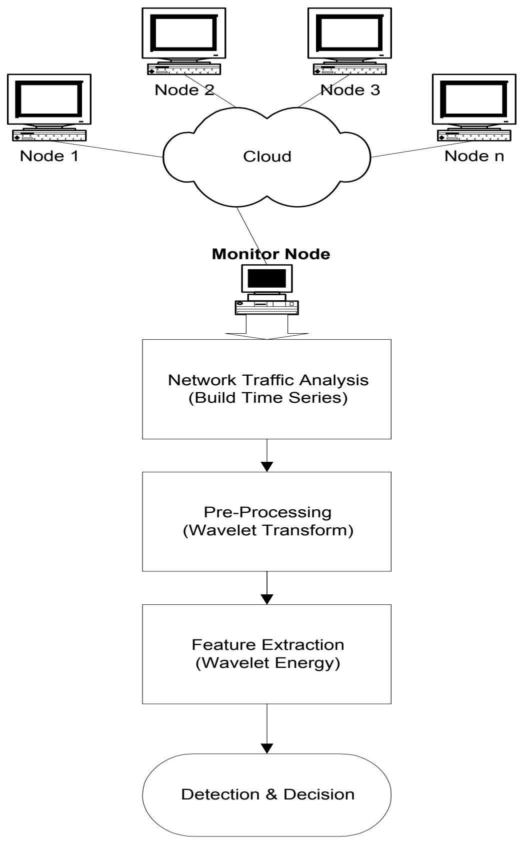
Node 1
Node 2
Node 3
Node n
Cloud
Monitor Node
Network Traffic Analysis (Build Time Series)
Pre-Processing (Wavelet Transform)
Feature Extraction (Wavelet Energy)
Detection & Decision

Figure 2. PRC Algorithm.

threshold energy value is derived by initial calibrations on over-utilized resources. This algorithm can be used to enhance existing intrusive resource discovery algorithms in the following manner: a Dedicated Resource Evaluator (DRE) node containing a hybrid resource discovery method composed of the proposed method and one of the methods mentioned in Section II. The DRE could be placed on the mirror port of a switch connected to a local Cluster Grid. Upon the registration of each node on the Grid, static resource information (e.g. number of CPUs, CPU speed, disk space, operating system type, etc.) could be documented for the node. This static resource information need only be documented once. Prior to dynamic resource information (CPU utilization) being needed, the DRE would passively extract resource information from existing network traffic produced by each node on the grid using the PRC algorithm. As data is analyzed, one Resource Availability Table (RAT) would be computed and continuously updated. Only under-utilized nodes would be listed. All other nodes would have been eliminated by the threshold energy value. Whenever nodes on the Grid require grid resources, the DRE would be queried and the under-utilized resources on the Grid would be made available to the requesting node.

VI. DISCUSSION

The DWT is used as an analysis tool to create a delay sensitive profile, which contains an energy value that is used to discern available nodes from unavailable nodes. Further, by using the Haar Wavelet this method reduces to a very simple

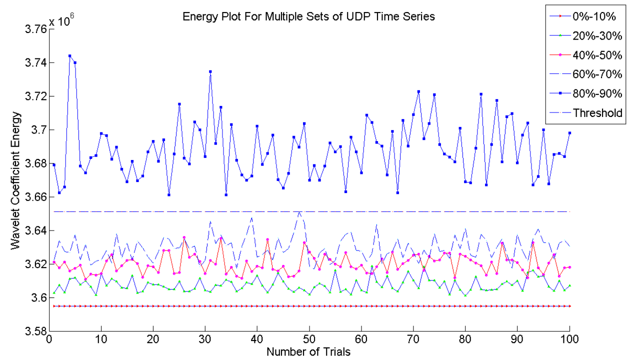

Figure 3. Energy Plot for 100 UDP Trials Using 2 GHz P-IV Nodes[1].

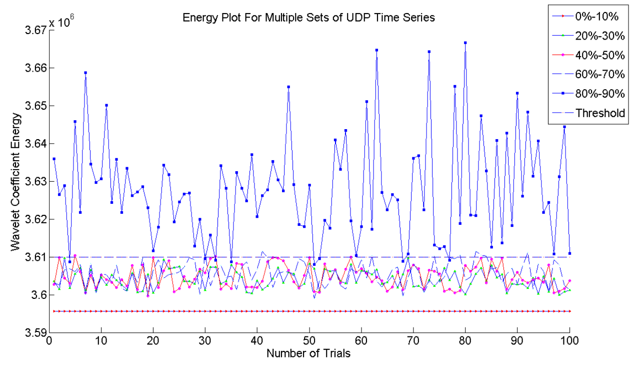

Figure 4. Energy Plot for 100 UDP Trials Using 866 MHz P-III Nodes[1].

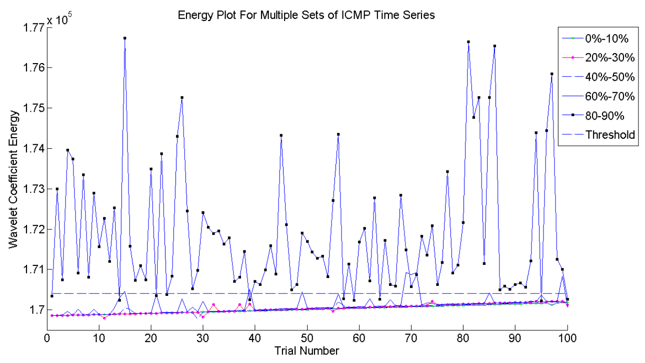

Figure 5. Energy Plot for 100 ICMP Trials Using 866 MHz P-III Nodes[1].

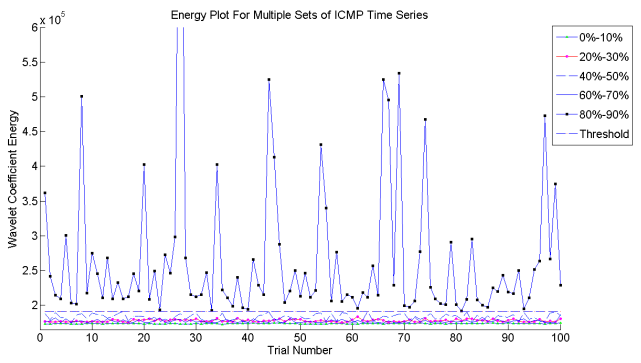

Figure 6. Energy Plot for 100 ICMP Trials Using 2 GHz P-IV Nodes[1].

Table 1. Classification Results for UDP Traffic.

CPU Speed	% Correct Classification
2 GHz P IV	100%
866 MHz P III	95%

Table 2. Classification Results for ICMP Traffic.

CPU Speed	% Correct Classification
2 GHz P IV	100%
866 MHz P III	97%

algorithm that can be easily implemented in hardware or software [11].

As mentioned in Section IV, the delays detected by the DWT correspond to various events that occur within a node. In our design, once the CPU utilization approaches 70% (unavailable) the network traffic produced by that node exhibits different behavior than when the CPU utilization is less than 70% (available).

Figures 3-4 (UDP network traffic) and 5-6 (ICMP network traffic) display the results of two sets of experiments performed at: 0%-10%, 20%-30%, 40%-50%, 60%-70%, 80%-90% and 100% CPU utilization (the experiments for 100% CPU utilization are not shown, because the magnitude of the results are large in wavelet energy thus causing scaling issues in the figures) for Pentium III and Pentium IV nodes. The data in Tables 1 and 2 display how accurately the simple threshold is able to separate the experiments into available or unavailable nodes. In Figures 3-6, the energy thresholds are chosen such that the two groups of trials (available and unavailable) can best be correctly separated.

The method works well for as few as 1024 packets and a send rate as low as 117 kbps constant bit rate (CBR) for UDP traffic. Similarly, the method works well for 520 packets for ICMP traffic. Note, the energy scale in Figures 3-4 are higher than the energy scale in Figures 5-6 primarily because fewer packets were used in the ICMP results than in the UDP results. The results in Figures 3-6 support the premise that once the CPU utilization approaches 70% (unavailable) the network traffic produced by that node exhibits different behavior than when the CPU utilization is less than 70% (available). The method as presented in this paper (a threshold on the energy value of a time series) can only be used to identify available nodes in homogenous networks (all nodes have the same architecture). The proposed method could be applied in heterogeneous networks (nodes have mixed architecture) by using Fuzzy Logic instead of a threshold.

Further, the results suggest that even older Pentium III machines are a suitable platform to employ the proposed method. This is important because traditionally, large numbers of older, cheap machines are used to create powerful local cluster and desktop grid networks capable of solving very complicated problems. This method used in conjunction with forecast or intrusive based schemes would further enhance the performance of these cluster and desktop grids by alleviating the need to make active CPU utilization measurements, thus reducing the amount of network traffic needed for a functional grid network.

[1] For ease of viewing, 100% CPU utilization Trials are not shown.

VII. CONCLUSION

This paper introduces a different type of resource discovery algorithm, a passive method designed to identify available and unavailable nodes on a low latency high performance cluster or desktop grid using network traffic.

This method could enhance existing resource discovery algorithms because there is no need to query potential resources directly for their CPU utilization information. Also, forecast-based methods could be enhanced by using the proposed non-intrusive method as a basis for its forecast, thus no longer needing to intrusively gather measurements. This would render the forecast-based methods passive, a characteristic not presently available.

We are extending our scheme to include the use of TCP network traffic to identify available nodes on a low latency high performance cluster or desktop grid. Further, we are in the process of determining a classification algorithm (includes threshold and Fuzzy Logic methods) that takes in account the amount of data needed versus accuracy of the scheme. Also, we plan to implement this method in the Georgia State University (GSU) Grid Network so further study of this method can be performed. Finally, we look to apply our method to security applications such as remote malware detection. It may be possible to detect that a node has been compromised by merely examining the network traffic it generates.

REFERENCES

[1] Vassilios V. Dimakopoulos and Evaggelia Pitora. “On the Performance of Flooding-Based Resource Discovery”,IEEE Transactions on Parallel and Distributed Systems, vol 17, no. 11, 11/2006.

[2] Yan Yang, Liansheng Tan, and Naixue Xiong. “A Resource-based Admission Control Algorithm for Grid Computing Systems”, IEEE CIT, 2004.

[3] Peter Dinda and David O’Hallaron .“An Evaluation of Linear Models for Host Load Prediction”, IEEE 1999.

[4] Sivadon Chaisiri, Putchong Uthayopas. “Survey of Resource Discovery in Grid Environments”, https://hpcnc.cpe.ku.ac.th/Members/sivadonc/RS_GRID_SURVEY.

[5] Maheswaran, M. and Krauter, K. “A Parameter-Based Approach to Resource Discovery in Grid Computing System.”,Proceedings of the First IEEE/ACM international Workshop on Grid Computing, December 17, 2000.

[6] John Sadowsky. “Investigation of Signal Characteristics Using the Continuous Wavelet Transform”, John Hopkins APL Technical Digest Volume 17, Number 3 (1996).

[7] Gilbert Strang and Truong Nguyen. “Wavelets and Filter Banks”, Wellesley-Cambridge Press, 1996, pp. 28-34 and 183-193.

[8] I. W. Selesnick, R. G. Baraniuk, and N. Kingsbury. “The dual-tree complex wavelet transform - A coherent framework for multiscale signal and image processing.”. IEEE Signal Processing Magazine, 22(6):123-151, Nov. 2005.

[9] H. Kim, V. Pai, S. Rixner. “Exploiting Task-Level Concurrency in a Programmable Network Interface”, ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming (PPoPP), San Diego, CA, (June 2003).

[10] W. Richard Stevens, Bill Fenner, Andrew M. Rudoff.“Unix Network Programming, Vol. 1: The Sockets Networking API”, Third Edition, Addison-Wesley Professional, 22 October, 2003.

[11] Lanier Watkins, Kenneth R. Perry, John S. Hurley, B.Olson and B. Pain. “Wavelet Transform Image Compression Prototype”, 1999 Technical Proceedings of the International Conference on Modeling and Simulation of Microsystems.

[12] Yuanyuan Zhang, Wei Sun and Yasushi Inoguchi. “CPU Load Prediction on the Computational Grid”, IEEE CCGRID2006.

[13] Juan Li and Son Vuong. “A Semantics-based Routing Scheme for Grid Resource Discovery”, Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing, eScience2005 , December, 2005, Melbourne, Australia.

[14] Wolski, R.; Spring, N.; Hayes, J. “Predicting the CPU Availability of Time-shared Unix Systems on the Computational Grid”, Proceedings of The Eighth International Symposium on High Performance Distributed Computing, 1999. 3-6 Aug. 1999 Page(s):105 – 112.

[15] L. Yang, I. Foster, and J.M. Schopf. “Homeostatic and Tendency-based CPU Load Predictions”, International Parallel and Distributed Processing Symp. (IPDPS 2003), pp. 42-50, 2003.

[16] Chao-Tung Yang; Chuan-Lin Lai.“Apply cluster and grid computing on parallel 3D rendering”,2004. ICME '04. 2004 IEEE International Conference on Multimedia and Expo,Volume 2, 27-30 June 2004 Page(s):859 - 862 Vol.2.

[17] Mustafee, N.; Taylor, S.J.E. “Using a desktop grid to support simulation modeling”,.; International Conference on Information Technology Interfaces, 2006. 28th, 2006 Page(s):557 – 562.

[18] Anglano, C.; Canonico, M.“A Comparative Evaluation of High-Performance File Transfer Systems for Data-intensive Grid Applications”; 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WET ICE 2004, 14-16 June 2004 Page(s):283 – 288.

[19] Matsuda, M.; Kudoh, T.; Ishikawa, Y. “Evaluation of MPI implementations on grid-connected clusters using an emulated WAN environment”, 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003 Proceedings. CCGrid 2003. 12-15 May 2003 Page(s):10 – 17.

[20] Lanier Watkins, Raheem Beyah, and Cherita Corbett “Using Network Traffic to Passively Detect Under Utilized Resources in High-Performance Cluster Grid Computing Environments.”, To appear in the Proceedings of ACM International Conference on Networks for Grid Applications (GRIDNETS), October 2007.