Dynamic Threshold for Monitor Systems on Grid Service Environments

E.N. Huh

Seoul Women's University, Division of Information and Communication, Seoul, Korea
huh@swu.ac.kr

Abstract. Grid technology requires use of geographically distributed multiple domain's resources. Resource monitoring services or tools consisting sensors or agents will run on many systems to find static resource information (such as architecture vendor, OS name and version, MIPS rate, memory size, CPU capacity, disk size, and NIC information) and dynamic resource information (CPU usage, network usage(bandwidth, latency), memory usage, etc.). Thus monitoring itself may cause system overhead. This paper proposes push based resource notification architecture on OGSI (Open Grid Service Infrastructure) and the dynamic threshold to measure monitoring events in accurate and with less overhead. By employing the new feature (dynamic threshold), we find out unnecessary system overhead is significantly reduced and accuracy of events is still acquired.

1 Introduction

In these days, the high-performance network technology enables geographically localized clusters to gather remote domain's clusters replacing supercomputing power with small cost. Remotely networked computers need to be cooperated to process huge amount of data in the area of bio-information technology (BT), space-technology (ST), environment-technology (ET), and nano-technology (NT) interfacing with electronic equipments. To enable information technology (IT) under the above areas, computational Grid is announced as a middleware system, which considers complex systems as a single system in point of a user. Thus, Grid is a virtual supercomputing infrastructure interacting with widely dispersed computing powers [1].

In order to effectively use the heterogeneous computing resources, such as storage devices, and various research equipments, these should be efficiently managed in Grid environments. Hence Grid is consisting of resource information service, resource brokering service, security service, resource management service, monitoring service, and data access service. The resource information service (GAIS-Grid Advanced Information Service) provides discovery and registration of resources. The brokering service functions an agent to provide higher level abstraction of resource information to user applications to be executed. The resource management service allocates and reserves resources. The security service using single sign on (SSO) based on public key infrastructure allows grid middleware components to access signed resources remotely. The monitoring service detects system changes and basic information, and delivers them to GRIS. The data access service manages distributed huge amount of

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 1162–1169, 2004.

data using meta-catalog for searching and replicas for duplicating data, respectively [2].

The monitoring system needs to collect the system information correctly while resource changes occur dynamically. The dynamic characteristics make resource allocator hard as delivered monitoring events are old under the situation. Thus the monitoring agents or sensors should run in real-time to provide accurate resource information of system events.

This paper presents a model of the dynamic update of Grid resource information service on GTK 3.0 using notification mechanism to measure monitoring events accurately during run-time. This paper is organized as follows: related studies are discussed in section 2; dynamic system change analysis is presented in section 3; and experiments are illustrated in section 4; and section 5 concludes this study.

2 Related Work

This section depicts Grid monitoring architecture from Grid Monitoring Architecture Working Group (GMA-WG) draft [3] and other monitoring tools such as iperf [4], ntop [5], tcpdump [6], topomon[8], and pingER [7] currently used for different purposes.

There are three components in monitoring architecture, directory service, consumer and producer. In order to describe and discover performance data on the Grid, a distributed directory service for publishing and searching must be available. The GMA directory service stores information about producers and consumers that accept requests. Functions of the directory service are Add, Update, Remove, and Search. A producer is any component that uses the producer interface to register its identification and basic information to the directory service and send events to a consumer. Functions of a producer defined in [3] are Locate Event, Locate Consumer, Register, Accept Query, Accept Subscribe, Accept Unsubscribe, Initiate Query, Initiate Subscribe, and Initiate Unsubscribe. A consumer locates resources from the directory service and requests events by query to the provider. Functions of a consumer are Locate Event, Locate Producer, Initiate Query, Initiate Subscribe, Initiate Unsubscribe, Register, Accept Query, Accept Subscribe, and Accept Unsubscribe. The overall mechanism among components is illustrated in Figure 1 [3].

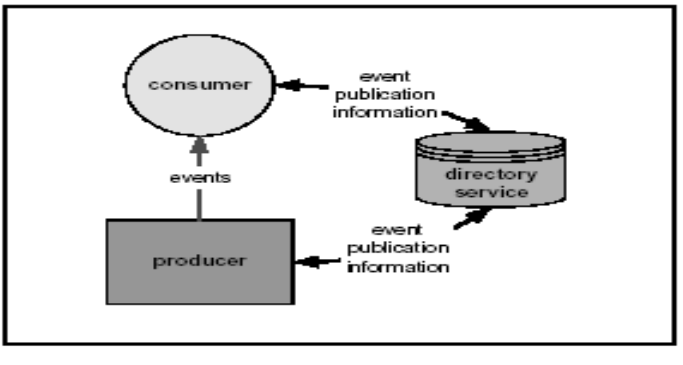

Fig. 1. Grid Monitoring Architecture

This architecture is designed for transmission of performance data, time-stamped events, and must define standard schema of events, which is extensible.

In [9], the monitoring mechanism for Information Power Grid (IPG) by NASA is explained shown in Figure 2. The service based monitoring consists of components, sensor, actuator and event service interface. It is designed for detection of the system behavior and control of QoS.

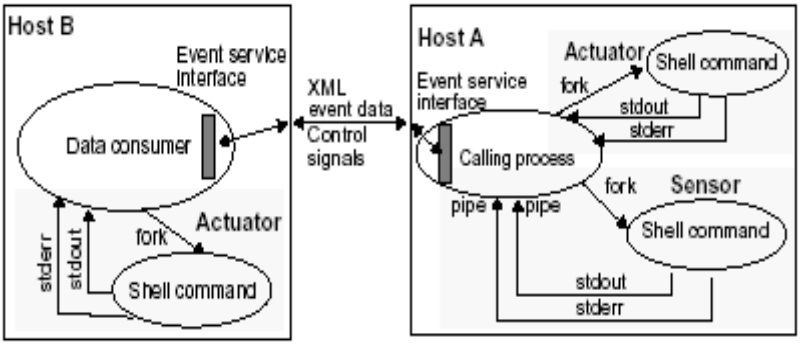

Fig. 2. IPG monitoring mechanism

The sensor is primary collecting tool and the actuator is pre-defined specific shell command (to collect events) by developer. And the event service provides standard interface to the requestor for transmission of inquires and results.

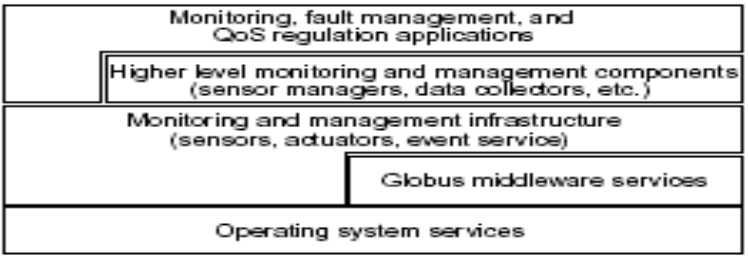

Fig. 3. Monitoring S/W layer

The monitoring S/W structure is designed on Globus middleware service [10] for security and monitoring event transmission defined in [11]. As shown in Figure 3, this focuses on the run-time environment to detect faults and QoS metrics on the external request (actuation). There is neither periodic real-time collection technique nor even overhead management in Grid monitoring tools.

Thus, this study supports real-time collection on Grid environments and proposes a model of the optimal and dynamic monitoring interval. Our sensors or agents are initiated to collect events by external and self requests.

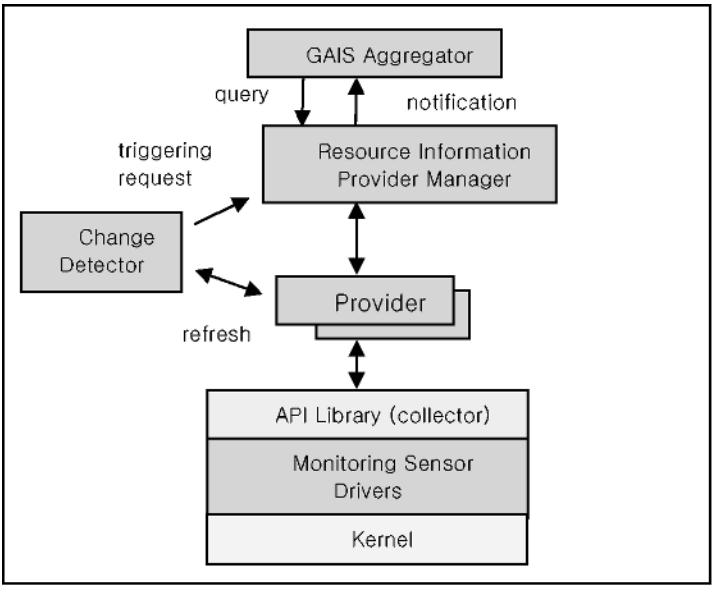

Fig. 4. Monitoring Architecture on OGSI using Notification

3 Proposed Monitoring Architecture and Dynamic Threshold

The resource of computing systems is changed very often during run-time as general processes and system processes (e.g., daemon) can be executed at any time. That is, we need a scheme for a threshold to determine if the significant system condition changes occur. For example, if the CPU is changed by 5% from the previous interval, then is it significant system change? Thus, we are proposing here additional technique, a ***dynamic threshold***, to decide significant system changes at run-time to support precise resource information for future resource allocation.

Mainly, processes use CPU, memory, disk, and network resources, which are main types of resources that Grid resource allocation service uses. Analyzing client/server program's resource usage (DynBench [12] is used for benchmarking application set, which consumes resources dynamically.), a strong correlation among resource types is assessed that other resources are depending on CPU changes. Thus, we collect initially only CPU usage percentage event in real-time to decide overall system changes. If CPU change is significant, then collects other resource events. According to the step 5 at the previous section, the dynamic optimal interval for the next cycle is longer than the current optimal interval if CPU change ratio is not. Otherwise, the optimal interval becomes shorter.

Thus, it is hard to determine the significance of CPU change. For example, CPU is changed by 1% at every cycle for long periods, and suddenly 2% is changed at any cycle. Then can you say it is significant change? Assume another case that CPU change ratio by 5% at every cycle occurs for long periods, and 7% CPU change is detected at sudden. Then is it significant change? In order to find out the significance

of condition change, we need to consider history of changes, current value, and difference from the previous.

The Figure 4 is the overall architecture proposed to update dynamically GAIS aggregator which is a global resource holder (Xindice XML DB is used). . The proc file system in Unix is widely used (e.g., NETSTAT , VMSTAT, TOP) to collect kernel and system information as well as process information. However, the event in proc file system is updated when process state is changed. Hence the accurate information of running processes is not guaranteed at the collecting time. Therefore, the device driver called "sensor" is employed in kernel space to collect more accurate events than other methods using directly reading the kernel variables. There exist 4 different types of providers we implement as follows:

- ✓ HostStaticInfoProvider : OS name, Vendor, clock speed, memory size, cache size, disk size, MIPS rate.
- ✓ HostDynamicInfoProvider: current CPU, CPU load info during 1, 5, and 15min., memory usage(buffered, swapped), uptime, disk usage, context switching time
- ✓ ProcessInfoProvider: CPU, memory usage, network usage, bandwidth, utilization
- ✓ NetworkInfoProvider: TCP parameters (loss, error, retransmit), utilization, network device usage (sent bytes, received. bytes, collision)
- ✓ PBSInfoProvider: Qinformation (name, queue-length) and Job status information

All of above events are formatted to XML for delivery to GAIS corresponding to the suggestion of Discovery and Monitoring Event Description (DAMED) Working Group [11].

Now, remain of this section describes how the dynamic threshold is derived for monitoring system, which requires les overhead and accuracy. Fortunately, Proportional, Integrated, Differential (PID) controller is designed to solve similar problems such as Quasar scheduler [13]. The PID controller is commonly used in water tank to keep proper water level automatically. Especially, in [14], PID controller is explained well as follows:

- The Proportional term causes a larger control action to be taken for a larger
- error
- The Integral term is added to the control action if the error has persisted for
- some time
- The Derivative term supplements the control action if the error is changing
- rapidly with time.

The formula for the desired level (denoted to ***u(k)***) calculation by PID controller is described below:

$$u(k) = k_p \times e(k) + k_i \times \left(e(k) - e(k-1)\right)$$

where, Kp, Ki, and Kd are parameters (tunable).

We consider that the steady state implies no changes of system resources. That is, $|e(k)-e(k-1)| = 0$, where $e(k) = R_{Max}-R_{obs}(k)$, R_{Max} is the maximum amount resource (e.g., for CPU, it is 100%), and $R_{obs}(k)$ is resource usage collected in current k^{th} interval. With steady state condition, we simplify PID controller formula by subtracting two states as follows:

$$
\begin{aligned}
&\left|u(k)-u(k-1)\right| \\
&= (k_p \times e(k) + k_i \times \sum_{j=1}^{k} e(j) + k_d \times (e(k)-e(k-1)) \\
&- (k_p \times e(k-1) + k_i \times \sum_{j=1}^{k-1} e(j) + k_d \times (e(k)-e(k-1)) \\
&= k_p \times (e(k)-e(k-1)) + k_i (\sum_{j=1}^{k} e(j) - \sum_{j=1}^{k-1} e(j)) \\
&\cong k_i \times e(k)
\end{aligned}
$$

From the above simplification, |u(k)-u(k-1)| is less than equal to Ki * e(k) under steady state. Thus, if |u(k)-u(k-1)| is greater than Ki * e(k), the current state is not steady. From this fact, we bring the dynamic threshold, Ki * e(k). Therefore, the significant change of resource is determined if |u(k)-u(k-1)| > Ki * e(k).

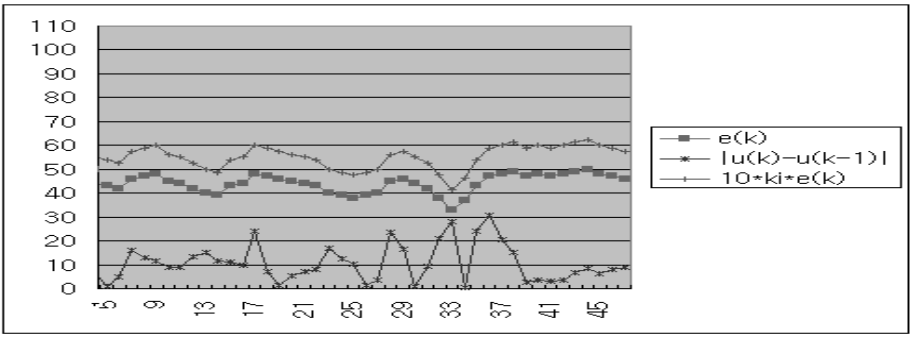

Fig. 6. Smooth system change scenario experiment

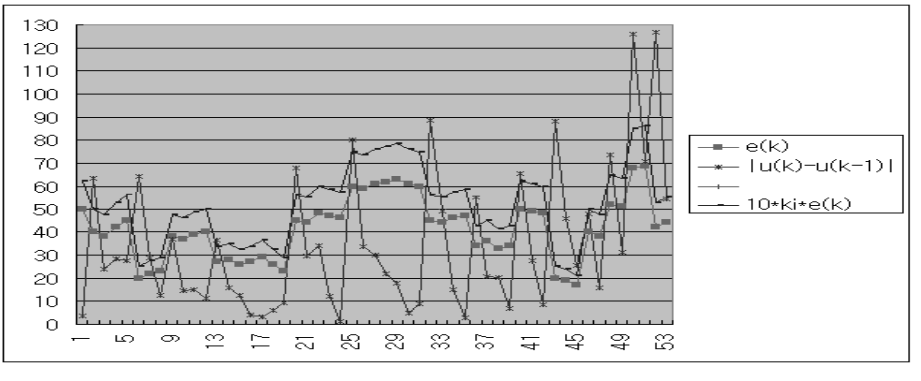

Fig. 7. The significant changes detection at the dynamic scenario

4 Experiments

In this section, two types of experiments are examined. The first experiment supports is done under the stable system. From the Figure 6, no significant change is detected. We change (add and drop) only by two percentage of CPU resource in this experiment.

The second experiment scenario for the dynamic system change is designed. There are CPU changes by 1% up to 15%. From the scenario, interesting result is illustrated in Figure 7.

There looks many significant changes expected but only 10 times out of 30 samples are significant from the Figure 7. When we look at the Table 1 carefully, the proposed scheme takes an influence of history trend and current changes.

From the two experiments, the results are very effective and our novel approach works well for the real-time Grid monitoring system on Grid Service based .

5 Conclusion

Grid will use many resources geographically dispersed. Grid will work together around world with a number of computers and equipments. Monitoring the resource is an essential component to enable better middleware services to users. The overhead of real-time monitoring tools might be tremendous without overhead management. This study provides many interesting factors in monitoring itself to researchers.

Our proposed scheme, the dynamic threshold, performs well in Grid Service environments, which reduces lots overhead and obtained accuracy of events from the experiments. And the complexity of our schemes is very small so that developers can implement easily.

References

[1] I. Foster., “The Grid: A New Infrastructure for 21st Century Science.”, *Physics Today*, 55(2):42-47 2002

[2] Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B, Schwartzkopf, H, Stockinger, K. Stockinger, B. Tierney, November 2002, “Giggle: A Framework for Constructing Scalable Replica Location Services.”, *Proceedings of Supercomputing 2002 (SC2002)*

[3] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, R. Wolski, “A Grid Monitoring Architecture”, March 2000, *Global Grid Forum Performance Working Group*

[4] Chung-Hsing Hsu, Ulrich Kremen, “ IPERF : A Framework for Automatic Construction of Performance performance models”, *Workshop on Profile and Feedback-Directed Compilation* (PFDC), Paris, France. October 1998.

[5] Luca Deri, Stefano Suin, “Improving Network Security Using N top events”, *raid 2000 Materials.*

[6] tcpdump, http://www.tcpdump.org.

[7] pingER, http://www.slac.stanford.edu/comp/net/wanmon/tutorial.html

[8] Mathijs den Burger, Thilo Kielmann, and Henri E. Bal, TopoMon: A Monitoring Tool for Grid Network Topology, *Proceedings of ICCS 02*.
[9] Tierney, B., B. Crowley, D. Gunter, M. Holding, J. Lee, M. Thompson "A Monitoring Sensor Management System for Grid Environments" *Proceedings of the IEEE High Performance Distributed Computing conference* (HPDC-9), August 2000, LBNL-45260
[10] A. Waheed, W. Smith, J. George, J. Yan. , "An Infrastructure for Monitoring and Management in Computational Grids", In Proceedings of the 2000 Conference on Languages, Compilers, and Runtime Systems.
[11] DAMED working group, https://forge.gridforum.org/projects/damed-wg
[12] Welch, L. R. and B. Shirazi., "DynBench: A Dynamic Real-Time Benchmark for Assessment of QoS and Resource Management Technology". *IEEE Real- time Application System*, 1999.
[13] Ashvin Goel, David Steere, Calton Pu, and Jonathan Walpole, "Adaptive Resource Management Via Modular Feedback Control", Submitted to HOTOS 1999
[14] PID Tutorial, http://www.engin.umich.edu /group /ctm /PID/PID.html