SmallSync: A Methodology for Diagnosis & Visualization of Distributed Processes on the Web

Ming C. Hao, Deon Glajchen, Joseph S. Sventek
{(mhao, sventek)@hpl.hp.com}, deon-glajchen@hp.com
Hewlett-Packard Co., CA

Abstract

SmallSync, an internet event synchronizer, is intended to provide a monitoring and visualization methodology for permitting simultaneous analysis and control of multiple remote processes on the web. The current SmallSync includes: (1) a mechanism to multicast web window-based commands, message passing events and process execution events among processes; (2) an event synchronizer to allow concurrent execution of some functions on multiple machines; (3) a means to report when these events cause errors in the processes; and (4) ad hoc visualization of process states using existing visualizers.

1.0 Introduction

There is increasing interest in having software systems execute over the internet. The web presents tremendous opportunity to develop many different mechanisms[1, 3, 4] to support multi-process coordination and diagnostics. We anticipate that the future will see the development of web extensions for synchronous collaboration among processes. These facilities are developed by bring together existing diagnostic and visualization facilities to allow users to share process states, to share web pages and to invoke visualization for direct communication. In addition, the technique can be used to expand the diagnostic toolset available to the user for cases where 'a-priori' indications of likely sources of observed error conditions are not available.

SmallSync is an integrated visualization and diagnostic environment that has been developed at Hewlett-Packard Laboratories to meet these challenges of tool development and to facilitate the task of distributed processing. Its main objective is to use the strengths of network-based computing for sequential/distributed processing and to apply the existing technologies and tools to solve the problems of distributed process development. Due to the nature of the specific issues involved in developing distributed processes as opposed to the development of sequential processes, the manner in which these existing tools and technologies are applied is the key to solving several tool development problems. The design of SmallSync addresses these issues to enable the users to use their favorite diagnostic tools for analyzing their distributed processes. Similarly, it enables the users to add multiple visualization tools of their choice to the SmallSync environment to understand the dynamic execution behavior of their applications.

Tool development for distributed systems has been an active area of research but has largely fallen short of the user expectations [10]. In many cases users are not satisfied with the way a particular tool works and the learning curve associated with using it to accomplish a specific task. SmallSync allows the user to use his/her favorite tool by making it a part of the processing environment. In order to realize this environment, the following issues are involved:

1. Enabling the programmer's favorite, existing diagnostic and visualization tools to become a part of this process development environment. This has become an important issue in view of increasing indifference of users toward "novel" distributed process development tools.
2. Synchronizing and controlling the activities of diagnostic and visualization tools to provide a consistent view of process execution to the user.
3. Integrating heterogenous types of visualization tools, such as general-purpose and conventional performance visualization tools across different platforms. This is necessary to address a wide range of requirements of various types of process behavior visualizations, such as application performance, system performance, and process data visualization.
4. Reducing transmission bandwidth requirements to permit simultaneous visualization of the process flow of multiple machines.

We have addressed these issues in the design of SmallSync to assist the users in developing distributed processes

using a message-passing library for a large number of machines. We have enabled several commonly used diagnostic tools, Similarly, we have integrated popular distributed process visualization tools such as ParaGraph in our system, in addition to commercially available visualization tools, such as Matlab and Gnuplot for customized process performance and data visualization. This paper depicts the architecture and the main features of SmallSync.

2.0 Architecture of SmallSync

In a distributed programming environment, an application consists of multiple processes running on one or more physical nodes that are distributed on the web. SmallSync executes each of these application processes under the control of an available (and perhaps the user's favorite) analyzer. One analyzer executing a single process presents the same scenario as analyzing a single sequential application. The only difference is the message-passing among these otherwise independent processes. Visualization has been recognized as an appropriate technique to represent message-passing and process execution behavior [7]. Several tools have been developed and used to represent various aspects of concurrent process and system behavior [9]. SmallSync enables the use of these visualization tools online by providing three major capabilities:

1. multicasting message-passing events among multiple processes;
2. integrating visualization tools to represent multiple perspectives of application behavior; and
3. controlling and synchronizing the execution of application processes and visualization tools.

Figure 1 depicts the overall architecture of SmallSync and its functionality. Despite the distributed processes, the environment allows the user to control the configuration and actions of all the distributed application and tools. It is important to note that SmallSync is running locally, whereas the other application processes, analyzers, and visualization tools might be running locally or remotely. Therefore, SmallSync acts as a controller for the whole environment which is the key to resolving the problems involved in visualizing distributed applications. We present the major functions of SmallSync related with distributed process visualization in the next section.

2.1 Multicasting Process Events

Distributed process visualization tools have to rely on some mechanism for multicasting events among the concurrent processes, in order to represent this activity graphically. Process code is instrumented and linked with the available communication library to multicast these communication events. However, most of the instrumentation systems for distributed processes perturb the application behavior mainly due to the additional message-passing required for generating and communicating the trace data. SmallSync does not need such explicit message-passing. Instead, it relies on a Multiple Event Protocol (MEP [6]) to multicast message-passing and process execution events among processes. MEP uses inter-client communication primitives to receive these events. Additionally, there is no explicit binding between application and SmallSync's multiple event processing and synchronization activities. Whenever a process executes a particular message-passing function, it sends that event to the event queue of underlying system which can be triggered by SmallSync. Once SmallSync receives the event, it assigns the event a time-stamp and generates a corresponding trace record. This trace record can be further processed and passed on to the visualization tools to dynamically visualize the process behavior.

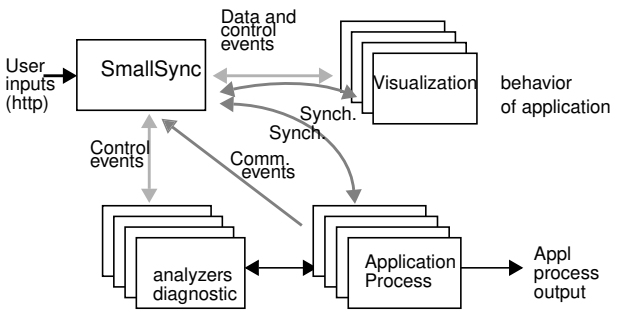


FIGURE 1. **Architecture of SmallSync distributed process development and visualization** environment.

2.2 Monitoring Multiple Tools

Tool integration is a well-known problem in software engineering and there is an on-going standardization effort to develop more practical frameworks for this purpose[2]. More recently, it has found its way into the design of tools for distributed processing because it is difficult for a single tool to satisfy all the requirements of all users [10]. Monitoring multiple tools will be useful in distributed process development; it meets two requirements:

1. There should be no dependence between the internal semantics of a tool and the rest of the environment, to ensure generality of the design and to avoid any problems related with the issues of overall performance and portability.

2. It should be possible for the environment to pass necessary performance data and the desired actions to be taken on that data by the tool.

A tool interface (TI) was designed to specifically meet the above two requirements. The functionality of the TI is depicted by Figure 2. As shown in the figure, each visualization tool which is to be integrated in the rest of the environment needs an interface. This interface is used for two specific purposes: (1) receiving data and control information from SmallSync; and (2) forwarding this data and control information to the particular tool that the interface is responsible for. The interface converts the control information into a form which is in accordance with the semantics of that particular tool. Optional bi-directional communication is supported by the interface for synchronization purposes.

SmallSync simplifies the issues involved in tool integration. It provides a common interface to obtain user input to control visualization tools as well as the rest of the environment. SmallSync sends the trace records to the visualizers as soon as they are generated to provide on-line visualization of dynamic process behavior.

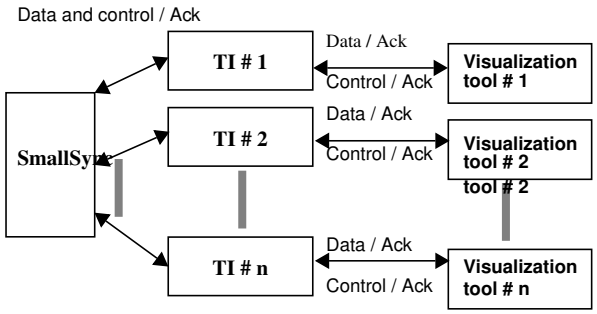


FIGURE 2. **Integrating visualization tools into SmallSync for interactive visualization and animation.**

2.3 Controlling and Synchronization

SmallSync control and synchronization permit diagnosis of multiple components as a single high-level process. A user can start or stop the execution of the entire distributed application using the control mechanism provided by SmallSync through its GUI. Application processes are executed under the control of analyzers and SmallSync sends the control commands to the multiple analyzers using the Event Sense Protocol (ESP [5]). ESP provides a mechanism to multicast window based commands from a single control window to some subset of analyzers and visualizers on various processes. It allows SmallSync to control the analyzers without any binding between the two. Therefore, the SmallSync environment is efficient, flexible and extensible. Application processes and visualization tools can be synchronized with SmallSync to ensure the consistency between application behavior and visualization displays.

3.0 Features of SmallSync

The design features of SmallSync presented in Section 2 have been used to provide several distributed process visualization features. This section briefly presents some of these features.

3.1 Global Concurrence Control and Update

SmallSync allows a user to control several clients (applications) simultaneously. A key feature of our system is that any existing application can be used with no modification of any kind. For example, SmallSync enables us to update multiple copies of a Lotus spreadsheet by entering the commands once. In fact, the applications being controlled need not be running on machines of the same architecture or even be identical applications. SmallSync allows us to control Lotus running on HP and Sun workstations and Excel on an IBM system by typing commands once. The only requirement is that the commands typed be meaningful to each system.

This kind of multiple process, multiple data repository collaboration has many uses. We could simultaneously update all the data servers, or print servers, or the like -within a particular account that we are managing.

Combining the conventional single client, multiple server collaborative environment with the multiple server, multiple client model of SmallSync completes the picture by providing for multiple processes monitoring. Application experts in different locations could simultaneously control various aspects of a distributed processes running on multiple machines.

3.2 Relative Remote Diagnostic

SmallSync is capable of handle several processes at the same time. This methodology could be used to run a diagnostic application on two or more remote systems, and to do a real-time intelligent comparison of the results. In this way multiple ‘known good’ systems could be compared to a ‘problem’ system. Comparisons can take into account differences in data representations of the reference process and the process being analyzed.

The diagnostic model presented in Figure 1 makes it necessary to use as many instances of the user’s selected analyzers as the number of concurrent processes. The user sets breakpoints in the code, interactively examines process variables, and verifies that these variables have expected values. This step-wise diagnosis-comparison can be single-stepped, and stopped based on any required set of ‘flag’ criteria - and so used as a non-invasive diagnostic tool.

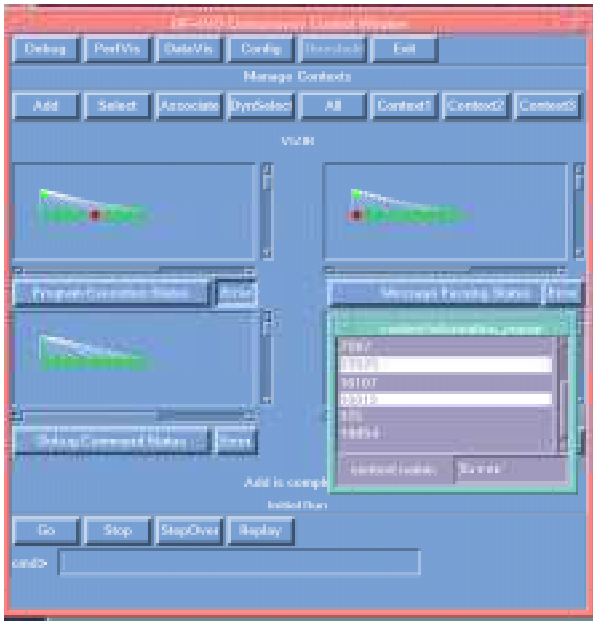
3.3 Deadlock and Error Notification

Usually deadlocks are hard to identify during the execution of a typically long-running process. In practice, deadlocks might occur if a process is in a blocked receiving state for a message type which was never sent to it by any other process.This is a common processing error and without a real-time display of the states of all the processes, it might be rather tedious to identify and analyze this error. SmallSync recognizes the states of individual processes belonging to one of, busy computing, sending a message, blocked for receiving a message, normal process enter or exit, analyzer-notified processing error, and meeting with a user defined condition (threshold). These states are represented by different colors in four status windows provided by SmallSync. The same result could be accomplished by using an appropriate display of one of the visualization tools integrated with this environment.

The user can monitor the states of all the processes throughout the execution of the process. If one or more processes are blocked, they can be identified by their PID and the user can click on the context to bring up its corresponding analyzer and source code that shows the line of code where it is blocked. The user can then figure out the cause of deadlock or any other process error resulting in undesirable behavior. Figure 3 illustrates an example where deadlock is detected and the corresponding process of the deadlocked process has been located.

3.4 On-the-Fly Visualization

SmallSync can send the trace records to visualization tools, immediately after they are generated by assigning time-stamps. Mostly, visualization tools[8] such as ParaGraph process one trace record at a time. As soon as a new trace record is received, all the selected displays are updated by the tool. This process is facilitated by the tool interfaces that were presented in Section 2.



SmallSync provides diagnostic and synchronization of the environment.

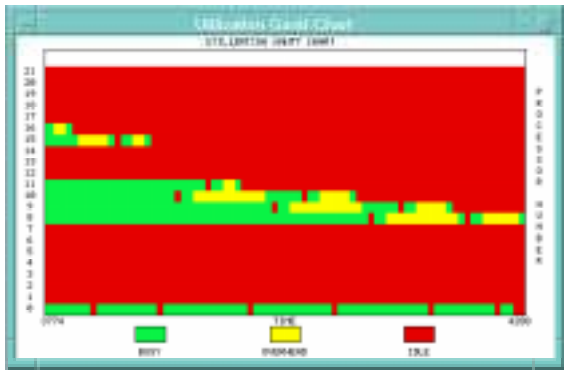
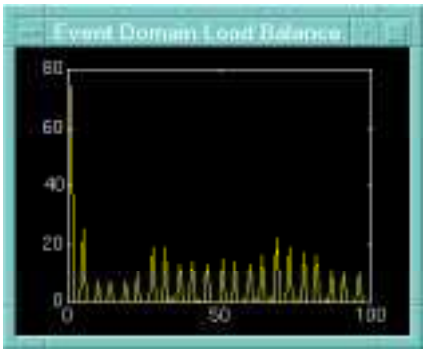


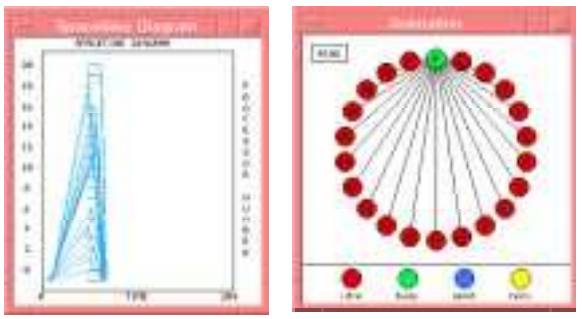
FIGURE 3. Multiple views from multiple domains and tools to visualize and analyze on-the-fly execution behavior of a distributed process.

3.5 Multiple Views using Multiple Tools

Visualizing the behavior and performance of distributed processes is always a multi-dimensional assessment. Not only does it take multiple tools but also multiple views and multiple-domains [11] are needed to represent a complete picture of process behavior to the user. SmallSync provides typical performance visualization and animation views that are implemented in ParaGraph. It can also allow the use of general-purpose data analysis tools such as AVS, Matlab, Gnuplot, Mathematica, and so on to represent multiple perspectives on application performance and behavior. Figure 3 represents some of these views.



Event domain views using Matlab as an analytic and visualization engine, integrated with SmallSync.



Typical application visualization displays from ParaGraph integrated with SmallSync

4.0 Conclusions

SmallSync is an on-going experiment in Hewlett-Packard Laboratories. SmallSync provides standard window interfaces to existing diagnostic and visualization tools with on-the-fly control and synchronization. Processes in the distributed application can be halted by an analyzer at the same point that performance and process state visualization is being done. In addition, for example, performance and process errors detected by an analyzer may automatically trigger the diagnostic processing to halt.

Although we have applied our mechanisms to a prototype diagnostic and visualization environment for distributed processing, they have much wider applicability. This approach can be used anytime we want to do the same thing on more than one machine on the web. Examples include installing and tuning loosely-coupled, heterogeneous software systems, and sharing large volumes of data on the web.

Acknowledgment & References

Thanks to Mary Loomis from the Software Technology Laboratory for her encouragement and suggestions. Umesh Dayal, Renee Jacowitz, Rick Bowers, and Adrian Pell provided valuable technical discussions.

[1] John F Patterson, Mark Day, Jakov Kucan, “Notification Servers for Synchronous Groupware”, CSCW 96, Boston, Massachusetts.

[2] Chen, Minder and Ronald J. Norman, “A Framework for Integrated CASE,” IEEE Software, March 1992, pp. 18–22.

[3] Stephen G. Eick, Michael C. Nelson, Jeffery D. Schmidt, “Graphical Analysis of Computer Log Files”, December, 1994, Communications of the ACM.

[4] P.Ciancarini, D. Rossi, F. Vitali, A.Knoche and R.Tplksdorf, “Coordination Technology for the WWW”, Workshops of Enabling Technologies: Infrastructure for Collaborative Enterprises, 1996, Ca.

[5] Hao, Ming C. Alan Karp, Daniel Garfinkel, “Collaborative Computing: A Multi-Client Multi-Server Environment”, ACM Organizational Computing Systems Conference, August, 1995

[6] Hao, Ming C. Alan Karp, Mehdi Jazayeri, “MESH: Sharing Multiple Events in Distributed Computing” HPL, Hewlett-Pack Laboratories, Palo Alto 7/94.

[7] Ramesh Jain, “Visual Information Management”, 1997, Communications of the ACM

[8] Kraemer, Eileen and John T. Stasko, “The Visualization of Parallel Systems: An Overview,” Journal of Parallel and Distributed Computing, 18(2), June 1993, pp. 105–117.

[9] Paul Robertson, “Integrating Legacy Systems with Modern Corporate Applications”, May, 1997, Communications of the ACM.

[10] Waheed, A., B. Kronmuller, Roomi Sinha, and D. T. Rover, “A Toolkit for Advanced Performance Analysis,” proceedings of International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS ‘94) Tools Fair, Durham, North Carolina, Jan. 31– Feb. 2, 1994.