On-line WWW-Monitor

DMITRY VAZHENIN, ALEXANDER VAZHENIN
Computer Software Department
University of Aizu
Tsuruga, Ikki-mach, AizuWakamatsu, Fukushima, 965-8580, JAPAN

Abstract: One of the main problems of parallel and GRID technologies is to obtain information about execution of remote programs. In this paper we propose the on-line WWW-monitoring scheme for real-time watching of remote MPI-processes. The key-point of this scheme is that each process can actively report its state to the special WWW-server, which is responsible for correct and on-time collecting of process data. The user can specify the monitoring operations by means of the special C-library designed. It is extremely useful for large-scale jobs, which may require the execution time about several hours, days or more. The design was implemented in the framework of the WWW-based Workplace of Applied Programmer (W4AP). From the user's viewpoint, this workplace represents an application-oriented virtual machine being accessible via standard WEB-browsers.

Key-Words: - Parallel Processing, Distributed and GRID-computing, Client/Server Technology, On-line WWW Monitoring, Message Passing Interface

1 Introduction

Internet computing and WWW allow now utilizing the aggregate resources of many networked computers to solve a single problem. Grid technologies seek [1,2] to make this possible, by providing the protocols, services and software development kits needed enabling flexible, controlled remote resource sharing on a large scale. This allows coupling geographically distributed resources and offers consistent and inexpensive access to resources irrespective of their physical location or access point. The key point of these technologies is in providing so-called "remote access" to the computational resources. This means that we have a situation, in which the user and computing systems are separated, and their cooperation is impossible without special communications channels. This requires special attention to the software design as well as human-computer interface/protocols.

Developing and debugging parallel programs has proven to be one of the most cumbersome tasks of program development. In the process of running a parallel program, many things can go wrong in many places. Deadlock situations can be caused by bugs in the message procedures or by poorly controlled access to the shared data. There exist also some hardware crashes for which the programmer may or may not be responsible. For effectively debugging parallel programs, we should know what went wrong in each of these cases. This confirms necessity to have effective tools for on-line monitoring system remote applications supporting simple and convenient integration of data as well as representation these results on the client computer.

Begole et al. [3] proposed an approach to access data resources external to the application in replicated synchronous collaborative applications. However, the introduced solution of Begole focuses on synchronous access to data and files from universal networking. It does not discuss about programming developing and management.

JXTA technology is a network programming and computing platform [4]. JXTA uses XML as the encoding format, mainly for its convenience in parsing and for its extensibility. The JXTA Shell is similar to the Unix Shell for writing scripts. It is not a convenient interface for applied users.

Zhou and Yang [5] propose an approach to manage the distributed resources and provide a platform for users to develop their applications with
the available resources. However, the proposed platform provides a set of specific application tools on Client site. It means it does not have transparency of Server.

Neophytou and Evripidou present a tool, Net-dbx, that utilizes Java and other World Wide Web tools for debugging of MPI programs from anywhere in the Internet [6]. The Net-dbx is a source-level interactive debugger with the full power of gdb (the GNU Debugger) augmented with the debug functionality of the public-domain MPI implementation environments. The advantages of this tool are building a parallel debugging architecture for MPI programs and accessing by WWW-interactive interface over Internet. The general facilities of programming are not their purpose.

Computational Portal [7] represents a proxy server placed between cluster and the user's workstation. It controls remote data processing operations. The server has a rich set of tools allowing access from users and their applied programs via http- and email protocols. System has also unified remote access library, good portability and safety. However, it is also necessary to support special high-performance computing server and to install special software on the cluster nodes providing communications with proxy-server.

WebSubmit is a Web-based utility providing access to applications on a collection of heterogeneous computing resources. Its goal is to make it easy to use remote computing resources via the Web without requiring knowledge of the specifics of unfamiliar operating systems and dynamic application environments. Each application module is presented as an HTML form filled out and submitted to the server, which executes the desired tasks on the specified remote system using another secure protocol. The advantages of this system are in specifying applied problems and simplicity human-computer interaction. Unfortunately, system does not allow implementing on-line analysis of results and debugging programs.

The paper describes an approach to the on-line monitoring for remote parallel applications. The proposed monitoring scheme allows to the user to watch the computational processes via Internet in real time. During execution, each application process actively reports its state to the user. This allows controlling the big amount of remote applications. It is also extremely useful for large-scale jobs, which may require about several hours, days or more of processors time. The design was implemented in the framework of the WWW-based Workplace of Applied Programmer (W4AP) [9,10]. From the user's viewpoint, this workplace represents an application-oriented virtual machine being accessible via standard WEB-browsers. In the presented work, we extend possibilities of the W4AP system supporting on-line monitoring of remote applications.

In the presented paper, section 2 shows the model and basic features of the W4AP system. In the section 4, an approach is proposed of the on-line monitoring scheme for remote parallel applications. An extension of the MPI-library supporting this scheme, examples of the use and tests are also shown. The last part of this article is our conclusions and future researches.

2 W4AP Basic Features

2.1 Architectural Overview

The W4AP is an integrated WEB-based system consisting of the server and a set of user's workstations (clients). Server and clients are connected via Internet. The W4AP-user works with standard WEB-browsers. We use also standard graphical interfaces working in the same style on different browsers. HTTP-protocols, PHP scripts and Java-applets are used for the information exchange.

One can distinguish the following basic components in the W4AP system:

- **The User's Workstation (Client)** implements a client part of the W4AP.
- **Cluster Node (C-node)** can be a supercomputer with its own Local Resource Management System (LRMS).
- **Working Nodes (W-nodes)** are a part of the C-node architecture controlled by LRMS.
- **W4AP-server** receives requests from clients and executes required software modules on their behalf. It stores also sources, results of computations, user's project data, etc.
Each user operates only with his/her private projects including sources and data files as well as data and tools for automatic generation of executable code and control of the program execution. The set of project types is open and extendable. It depends on specifics of solving problems as well as supercomputers used.

2.2 User's Project Templates

Some preliminary parameters needed for preparing and executing of applications are stored in special structures called templates. Each template is highly customizable. The system administrator can maintain parameters of computational system as well as programming environment. The user can create his/her project by choosing suitable template. All non-system-specific project options are accessible for the user. The user's template inherits all system specifics from a project template. The W4AP system includes the three template types: **Sequential project**, **Parallel project** and **Parallel GRID project**.

**Sequential project** is to design and implement traditional sequential applications. It also may be used to test computational algorithms and learn a programming language. The application created in the framework of this project is limited in resources and restricted to run on server's host computer.

**Parallel project** is to design and implement parallel applications (Fig. 1) based on well known and popular MPI-platform. The administrator must specify a particular C-node and default system-specific parameters for the project template. These settings are the C-node remote hostname, the set of compiler and executor commands and necessary command-line parameters. The user can specify application parameters such as: name and options of the compiler used, link libraries, launch options, etc. The system provides a mechanism for automatic data replication from/to C-node. The application is compiled and executed inside C-node environment with all files transferred from data storage server. After finishing successful compilation or execution, all data is automatically replicated back to the data storage. Connections between data storage and the host computer of C-node are based on standard secure (SSH, SCP) and non-secure (RSH, RCP) protocols.

![Figure 1: Parallel Project Architecture](image)

**GRID project** is to create cross-platform parallel applications for various C-node types by choosing target C-node automatically according to the specified options such as: amount of memory, number W-nodes, etc. The user can specify all necessary limits for resources to run his program. There is a special database of computational resources maintained. So, the W4AP GRID project module can choose suitable C-node according to specified limits and collected statistics.

This flexible organization allows the efficient execution of application programs as well as on-line representation of intermediate and final results.

3 On-line Monitoring Scheme

Usually, there are two approaches to monitoring a program execution: post mortem and runtime debugging. In post mortem debugging, the compiled program keeps track of everything that happens during execution and logs every event in special files called trace files. After the parallel program terminates, these trace files are then parsed using special tools to help the user guess what went wrong (and when) during execution of a buggy program. Runtime tools, on the other hand, have access to the program's memory on each machine. Using operating system calls, runtime debuggers can stop or resume program execution in any execution node. These tools can change variables during the
execution and can break the program flow under certain conditions.

In our approach, we use the on-line monitoring scheme based on active cooperation between computational nodes and W4AP server. This means that a program submits information about its current situation to the server, which immediately makes these data visible to the user. It is also possible to organize trace files.

4.1 Resource Allocation and Remote Applications Control

Let us explain the proposed scheme in more detail. We consider that any computational resource has the four important parameters: total and free amount of computational nodes, as well as a number of free nodes, the number of executing tasks, total and free amounts of memory.

The Resource Management Module was designed to track executing jobs and allocate free resources for new ones (Fig. 2). This module allows add/remove jobs to/from queue, set limits for maximum number of nodes for each task and maximum number of user’s simultaneous jobs. It also provides remote data replication, garbage collection and resource reallocation when the job is terminated or finished successfully. The resource manager will terminate unfinished job on timeout.

In order to watch a state of a program, the Monitor Module allows controlling selected variables. Every time when the control spot is reached, all variables from all nodes are collected and a new HTML report is generated in the main node. This report is a table containing information about control spot name, node number and node variables state. This module is a regular ANSI C library compatible with the most MPI distributions and can be used separately. To do this, it is only necessary to have MPI engine installed.

The presented scheme cannot be considered as a full runtime debugger because the monitor cannot control the program from outside. It just allows a programmer to watch the computational process via Internet in real time. It is extremely useful for large-scale jobs, which may use about several hours, days or more of processors time.

3.2 MPI-monitor Library

As was pointed above, the Monitor Module is realized as the MPI-library including a set of routines activating and controlling on-line monitoring. These tools provide run-time errors control and HTML online-report. Below we show description of the main functions of the MPI-monitor Library.

Monitor initialization: void WT_Init(int *argc, char **argv[]). This function is to initialize internal parameters of MPI environment (topology size, type, etc) and allocate memory for them. It must be called after MPI_Init () in every node.

Mark as observable: void WT_AddWatch(MPI_Datatype, void *data, char *comment). The pointer to variable, data type and comment must be specified. It is better to mark variables at the program beginning. Each node can have its own list of marked variables, if necessary.

Block control spot: void WT_Collect(char *). The list of variable names and current values is maintained automatically having reached this function call in program. It should be called in all computational nodes. If some node is not responding, it may have affect on whole program that may cause a deadlock. A single parameter is used as a comment for control spot.

Stop blocking control spot: void WT_Icollect(char *, double). If some node is not responding, the information from previous call about this node is used. Therefore, only operational nodes have the recent values in report. It is implemented with MPI_Isend and MPI_Irecv routines, so the deadlock probably will not occur.

Figure 2: Monitor operational scheme
There are two parameters: comment for control spot and timeout in seconds to wait for nodes response.

**Report dumping:** void WT_Dump(). This routine is to store the current report to the trace file.

**Finalize Session:** void WT_Finalize(). This function frees all allocated memory and flushes all buffers. It must be called before MPI_Finalize().

The typical monitoring scheme consists of the following stages:

1. Initialize the W4AP-monitor,
2. Mark necessary variables as observable,
3. Collect variable values in the main node,
4. Store data into the trace file,
5. Finalize session after finish of monitoring.

Each call of the WT_Dump renews the trace file content. Importantly, monitoring information is accessible just after the first recording into a trace file. There are several types of variable data representation like tables, graphics, visual objects, etc. It is possible to display not only scalars but also vector and matrix data using a special technique [11].

### 4 Examples of the Monitor Usage

As example, we show an MPI-program fragment implementing implicit iterative calculations (Seidel Iteration). In this example, parallel computations are organized using the pipeline principle. When a processor 0 calculates iteration \(i\), the processor 1 computes an iteration \(i-1\), etc. Data exchange between processors is implemented for both directions. All processors execute the same program.

```c
#include "webtrace.h"
int rank, size, M;
int main(int argc, char *argv[])
{...
    int iter=0; double max;
    MPI_Init(&argc, &argv);//Initialize MPI
    WT_Init(&argc, &argv);//Initialize Monitor
    ...
    // iter and max -->watchable
    WT_AddWatch(MPI_LONG, &iter, "Iteration: ");
    WT_AddWatch(MPI_DOUBLE, &max, "Residual: ");
    ...
    do { /* Calculating residual between old and new iterations. Collecting it in max for all computational nodes */
        ...
        WT_Icollect("Iteration end");
        // Send data to the main node. If calling node is the main, get data from all.
        WT_Dump(); // Put data in the trace file
        iter++; } while (max<EPS);
    WT_Icollect("Computation end");
    WT_Dump();
    WT_Finalize(); // End Monitoring
    MPI_Finalize(); // End MPI}
```

This example shows that the program collects periodically the content of observable variables \(max\) and \(iter\). This information is transferred to the main node, and can be visualized. Example of this on-line table representation is shown in Fig. 3. We show also an example of visualization of a triangular matrix distributed in eight nodes (Fig.4).

![Figure 3: Example of On-line Monitoring](image-url)
6 Conclusion

In this paper, we have presented on-line W4AP monitor, a tool that utilizes standard WWW tools and Java for interactive monitoring of parallel remote applications. It allows controlling parallel program execution by assigning suitable variables and arrays to be observable. Monitor data can be showed in both text and graphical forms. The prototype provides a graphical user interface at minimal overhead. It can be run from anywhere on the Internet. Most important, it can run with no modifications on virtually any console, requiring only the presence of a Java-enabled WWW browser on the client side. The rich possibilities of WWW-browsers and integration data strategies make possible to control a big amount of remote parallel processes.

The tool in its present implementation is currently being tested by Internet users at various locations with the help of a WWW-based hosting environment that provides the full capabilities of the W4AP to remote Internet users without compromising security on the local server. We are currently working on extending this prototype to an integrated MPI-aware environment for program development, debugging, and performance monitoring.

References: