

# Web-Based Parallel Simulation of AGVs Using Java and JINI

Rong Ye, Wen-Jing Hsu, and Ze-Hua Liu

Centre for Advanced Information Systems,  
School of Computer Engineering, Nanyang Technological University,  
639798, Singapore  
yerong@bigfoot.com, Hsu@ntu.edu.sg

**Abstract.** The vision of Computational Grids promises an exciting future for the distributed simulation community. In this project we make a small but practical step toward the grand vision of distributed simulation by using certain prevailing Internet technologies to enable access of simulation services anytime and anywhere. Specifically, this project focuses on accessing distributed simulation of AGVs (Automated Guided Vehicle) in container port operations through the World Wide Web. The objectives are to explore and address relevant issues, evaluate various approaches, demonstrate a workable version. We initially construct the AGV simulation system in an indirect communication model and identify its merits and demerits. Then, we explore the use of JINI technology for an efficient and robust direct communication architecture.

## 1 Introduction

The vision of computational grid has been well expanded in the book by Ian Foster and Carl Kesselman *The Grid: Blueprint for a new Computing Infrastructure*[6], which simply put, is an infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities. This will result in increased delivered computation by five orders of magnitude within a decade brought about by increased demand-driven access to computational power, increased utilization of idle capacity, greater sharing of computational results, and new problem solving techniques and tools.

Meanwhile, in the past decade, the “Internet revolution” has been the most significant technological development. The technological development has crossed all the frontiers of time and space and truly reduced the world into a global village. Presently the development of the Internet appears to be driven by the momentum created by the “network-centric model”[3], the ultimate goal of which is to turn the network into the computer and turn the client into a “thin” client, i.e. shifting computing burden from the client to server. It brings about the concept of balancing the computational burdens throughout the network amongst clients and servers with minimum resources expended on each specific client. Although arising from a different context, mainly driven by the rise of PC and Internet, this network-centric computing concept actually coincides with that of the Computational Grids which has been championed by people from super-computing

arena. In this project we make a small step toward the vision by using some of the prevailing Internet technologies. Specifically, the project focuses on Web-based distributed simulation of AGVs by using JINI and Java. The objectives are to explore and address relevant issues, evaluate various approaches, demonstrate a workable JINI-enabled version for a distributed AGV simulation system.

The rest of paper is organized as follows: Section 2 briefly introduces a traditional AGV simulation system and its Web-enabled counterpart; Section 3 presents an *ad-hoc* 3-tier approach; Section 4 elaborates on an efficient and robust JINI-enabled framework; Section 5 concludes the paper and discusses future work.

## 2 Container Port Simulation

To manage the complexities of the processes at the port, container operations include scheduling of the port operations, allocation of resources and various traffic control schemes. An AGV simulation system concentrates on all or part of the above aspects. The statistics collected in the simulation would give useful information to both the route layout designers and the routing algorithm designers for the AGVs.

We have developed a prototype AGV simulation system[8]. It could run on SGI, SUN SPARC Workstation or other UNIX systems.

The user could specify the number of AGVs involved and various other control parameters at the beginning of a simulation run. Afterwards, the user can observe the visualization output of simulation execution. In the end, results are analyzed and reports are presented to the user. Usually, the user will invoke the simulation system with different parameters and repeat them many a time before coming to a conclusion.

We now port the original system to the Web and Internet by utilizing latest Internet technologies such as JAVA and JINI in the following sections.

We employ the “server fat approach” which means the critical computing is done on the server, usually a high performance machine, so as to tackle the complicated computing involved in a simulation in time. A Java “wrapper” program is developed for our legacy non-Java AGV simulation service. The GUI part inclusive of both input and output handling is ported onto the Web-Browser by means of Java applets. In our framework, the client is the Web-Browser, or more specifically the applet embedded in an HTML page; the server is a Java-coded or a JINI-enabled simulation service. Figure 1 is a snapshot of visualization outputs of a running AGV simulation on a Web-Browser.

## 3 An *Ad-Hoc* Approach

To support Web-based applications, it has become a rather standard practice to adopt a 3-tier architecture(see Fig. 2)[5].

We developed an initial *ad-hoc* version based on a generic approach, where the Middle-tier servers mediate between sophisticated back-end services and the Web front-ends. This approach applies an indirect communication model. By name,

the clients will talk with the back-end services through a so-called "bridge". This model has both merits and demerits. Clearly, the biggest problem is that the "bridge" may become a bottleneck in the system with the number of links increased.

The user downloads Java applets from the Web server. The applets will then connect the client machine to a Lookup Service and send requests to the server. In response to a service request, the Lookup Service locates and forwards requests to the relevant AGV simulation service provider.

On the high performance computer side, an Application Server Daemon is responsible for registering itself in a Lookup Service and waiting for incoming requests. Upon receiving a request, the Daemon will invoke the high performance computing application with the given parameters. Finally, the results will be transferred back to the clients via the Lookup Service server.

This version was developed by using JDK(Java Development Kit) with TCP/IP as its communication protocol. Besides the shortcomings tagged with the indirect communication model, there are other drawbacks. The system has to deal with joining and leaving process of any particular AGV service provider. The most difficult part is to handle all kinds of faults or exceptions, such as a sudden crash of a service provider, failures of a subnetwork.

## 4 A Structured Approach: JINI-Enabled Framework

JINI is a framework for building scalable, robust and truly distributed systems using Java [2,4]. Using JINI is a new approach to demonstrate the concept of Web-enabled AGV simulation system. The JINI approach provides a number of benefits including instant availability of services, impromptu community software and high flexibility and fault tolerance.

By employing Java and JINI technology, we have designed, analyzed, and implemented a JINI-enabled framework for Web-based distributed AGV simulation system.

JINI-enabled version adopts a direct communication model, where the clients will communicate with back-end services directly through a proxy provided by the service. A proxy is an arbitrary serializable Java object in the service item. It contains the information of how to interact with the service.

The advantage of the direct scheme is obvious: firstly, there is no more "bridge"-like bottleneck; secondly, the communication delay will be reduced greatly [1]. A possible disadvantage of the direct communication model is related to the security issue. Malicious attacks to the back-end server are possible because the server's network address is published in its proxy. We here assume that the back-end machine itself has been well protected from intrusion.

### 4.1 System Architecture: Services Federation

Figure 3 shows a schematics of a so-called AGV simulation service federation which consists of one or more JINI Lookup Services.

For AGV Service Providers (see Fig. 3):

- *register*: The AGV service providers will join the federation by registering themselves in one or more JINI Lookup Services in the federation. This step could occur at any time when a high performance machine starts up and is willing to provide a service. The service programs may be one that has been installed on the machine or one downloaded from an AGV services repository. If the services programs are downloaded from a services repository, firstly, they should be configured to be able run on the local machine. That may involve compiling, linking and other necessary processing.

For Service Agents (see Fig. 3):

1. *Step (1)*: Firstly, the Service Agent should register itself in one or more Lookup Services so as to expose itself to clients and to be ready for serving. Meanwhile, it registered its interest in AGV Service in the federation.
2. *Step (2)*: Then, the agent will collect matched AGV Service Providers' proxies in the federation. When a new Service Provider comes to join the federation or a Service Provider leaves the federation, the JINI Lookup Services will notify the agent of the change. In this way, the agent could always keep a collection of all live Service Providers' proxies without polling individuals in the federation.

## 4.2 Distributed AGV Simulation Serving Scenario

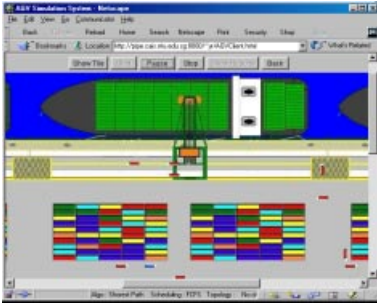
Figure 4 shows the scenario of how the Web-enabled distributed AGV simulation system serves a user.

1. A user accesses the portal Web page of the AGV service, downloads Java applets and logs on the federation.
2. The applet will then connect the user machine to a JINI Lookup Service and search the Service Agent. When a Service Agent is located, the applet will forward the user's requirements to the Agent.
3. After that, the Agent will utilize all available AGV simulation computing resources to execute AGV simulation tasks according to the user's requirements. Multiple independent modules or multiple batch jobs will be performed concurrently.
4. The clients could monitor the execution of an individual task and observe its visualization output through the back-end server's proxy.

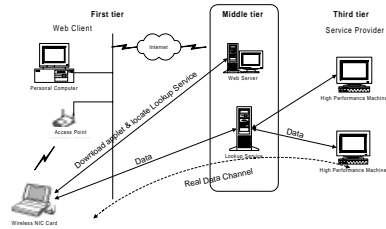
On the high performance computer side, upon receiving a request, a Service Provider Daemon will invoke the high performance computing application with the given parameters from the agents and finally the results will be transferred back to the requester.

Obviously, the "batch work" feature of simulation executions makes distributed multiprocessing possible. With the enforced system, we now are able to apply the agent approach to the following two categories of AGV simulation scenarios.

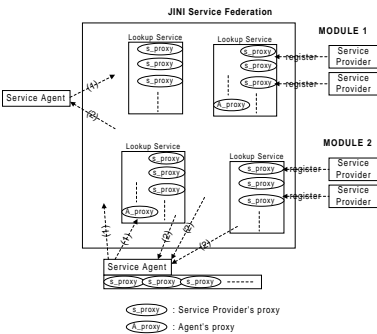
- *Iterative simulation and batch jobs* As we mentioned earlier, usually, a simulation run will be repeated for many a time with same parameters or different options before reaching a more objective result. In both cases, the agents system could dynamically distribute "batch tasks" among the service providers in the federation thus provide clients with a timely service.



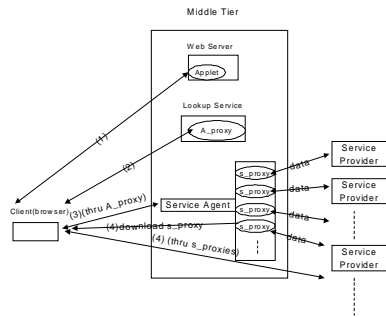
**Fig. 1.** A Sample Session in Automated Guided Vehicle Simulation



**Fig. 2.** 3-tier View of Web-Based Simulation System



**Fig. 3.** An AGV Simulation Services Federation



**Fig. 4.** Service Access Protocol in the Agent-Based Framework

- *Time-parallel simulation* This is an alternative way to Space-Parallel decomposition of simulation. The key idea is to decompose the simulation along the time dimension for multiprocessing [7]. Usually, the computing involved in intervals is independent. From another point of view, we have a batch of jobs again.

**4.3 Scheduling and Self-Healing System**

Clearly, scheduling is an important issue in our system. First of all, the agent will estimate a weight value for each service provider based on an estimation function:

$$V_{provider} = f(NumOfProc, PeakPerf, Workload, Latency, Credit, \dots)$$

*NumofProc* represents the number of the service provider’s physical processors; *PeakPerf* denotes its peak performance; *Workload* indicates its current computing workload; and *Latency* means the communication delay. These parameters are dynamic information.

*Credit* is an additional information. Each Service Provider is tagged with a *Credit* value which is evaluated by the Agent depending on the number of failures of the Service Provider in earlier scenarios. Important and nontrivial tasks are always dispatched to a Service Provider with a high *Credit* value.

With JINI technology, a provider joining or leaving the federation could be observed by the agent. The agent will recalculate the weight values of service providers and refresh the ordered list regularly. If any task is currently assigned to a failed service provider, it will be reassigned to another provider. Meanwhile, any newly joined resource will be noticed and be harnessed immediately. In other words, it is an adaptive and self-healing system.

## 5 Conclusions and Future Work

This project arises from the need to bridge the gap between the supercomputers and the general users by utilizing some of the prevailing Internet technologies. Through a JINI-enabled distributed AGV simulation services community, the remote client could access the AGV simulation resources on a network even on the Internet with a common Web-Browser. The following are recaps of our main contributions:

- We have evaluated two approaches in terms of their suitability to the Web-based AGV simulation application.
- A Web-based and JINI-enabled distributed AGV simulation system has been implemented with Java and JINI technologies resulting in intelligence, self-heal, improved efficiency, high scalability, high reliability, fault tolerance, and with security measure.

## References

1. Hyok Kim, Hongki Sung, and Hoonbock Lee: Performance Analysis of the TCP/IP Protocol Under UNIX Operating Systems for High Performance Computing and Communications. HPC ASIA'97 IEEE. (1997) 499 -504
2. Ken Arnold, Bryan Osullivan, Robert W Scheifler and Jim Waldo: The Jini Specification. Addison-Wesley (1999)
3. Bernard Conrad Cole: The Emergence of Net-centric Computing: Network computers, Internet Appliances, and Connected PCs. Prentice Hall (1999)
4. W. Keith Edwards: Core Jini. Prentice Hall (1999)
5. Jeri Edwards: 3-Tier Client/Server At Work (revised edition). Robert Ipsen (1999)
6. Ian Foster: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
7. David M. Nicol and Richard M. Fujimoto: Parallel simulation today. *Annals of Operations Research*, 53:249-285, 1994
8. Voon-Yee Vee, Rong Ye, Shah Sneha Niranjana and Wen-Jing Hsu: Meeting Challenges of Container Port Operations for the Next Millennium. Report for Supercomputer Programming Contest(CrayQuest'99), Singapore. (1999), Gold Award Winner
9. Voon-Yee Vee, Rong Ye, Liu Zehua, Shah Sneha Niranjana, Wen-Jing Hsu and Frank Reichert: Web-Based Parallel Computations: Challenges and Opportunities of HPC in the Network-Centric Era. Report for Supercomputer Programming Contest (CrayQuest'2000), Singapore. (2000) Grand Champion (Industry Category)