KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 341–345

A TSPACES BASED FRAMEWORK FOR PARALLEL-DISTRIBUTED APPLICATIONS

E. SCHEIBER

Abstract. A framework enabling the deployment and the execution of a parallel-distribute application is presented. The application is build on an imposed template based on the master-slave model. The application as well as the infrastructure of the framework use TSpaces servers. The coordination of the activities of the application is of data driven type.

1. Introduction

The purpose of this note is to present a framework enabling the deployment and the execution of a parallel-distribute application, whose development is based on the template presented in [4].

The development programming language is Java, used in many projects, frameworks and products for parallel-distribute computing and for high performance computing. The Java technology is improving continuously. Java was designed to meet the real world requirement of creating interactive, networked programs. Java supports multithreaded programming, too.

The model of the template is that of *master-worker* and the coordination is of *data driven* type based on a shared dataspace [3]. The idea of the developed approach is that of the Piranha project [3], and in order to carry out the proposed tasks we integrate as much as possible software components that are free of charge at least for a non commercial application.

The framework and the application template allows to solve problems which may be solved using MPI [5, 2, 1] in a network of workstations.

2. On the template of the parallel-distribute application

The instance of the *Dispatcher* class corresponds to the master, while an instance of the *Worker* class corresponds to a worker. The *Dispatcher* and the

2000 *Mathematics Subject Classification.* 68N99, 68N19.
Key words and phrases. parallel-distribute framework, Java, TSpaces.

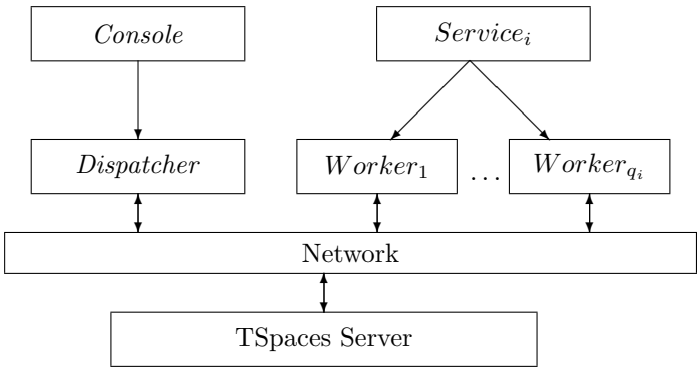
Worker classes are threads. The `run` methods of these Java threads contain the specific activities to solve the given problem.

For this version of the framwork we use *TSpaces 2.1.2,* from I.B.M., [6], as the shared data space for coordination and as a warehouse of the messages. *TSpaces* is a Java implementations of the Linda computational model.

Between the dispatcher thread and the worker threads there are asynchronous message exchanges. These messages are kept by a TSpaces server - the application associated tuple space. The data to be exchanged between the dispatcher and the workers are wrapped into objects. To distinguish between different kind of data, a tag field may be introduced. A message consumer (dispatcher or worker) waits until all the required messages are available. In this way the synchronization problems are solved as well as the coordination of the activities.

The dispatcher is launched to run by a *Console* program. On a workstation it is possible to start several worker threads. These threads are instantiated and started by a *Service* program. The planed number of the worker threads are equaly distributed to the involved workstations. This kind of organization of an application is similar to that used in the *JCluster* software [1].

The *Service* program is a servlet and we use the *apache-tomcat* as a servlet container Web server. The *Console* program which starts the dispatcher makes the requests to the *Service* servlets, too. The *Console* and *Service* classes are independent from an application.



The i index denotes a workstation on which q_i worker threads are run.

3. The framework

We suppose that several computers in a network will perform the required computation. On each workstation, the *apache-tomcat* Web server must be active.

To the network of workstations it is associated a second TSpaces server to keep the names of the involved computers-denoted as the tuple space of the network.

A scheme of the deployment of the framework with the required programs is given in Fig. 1.

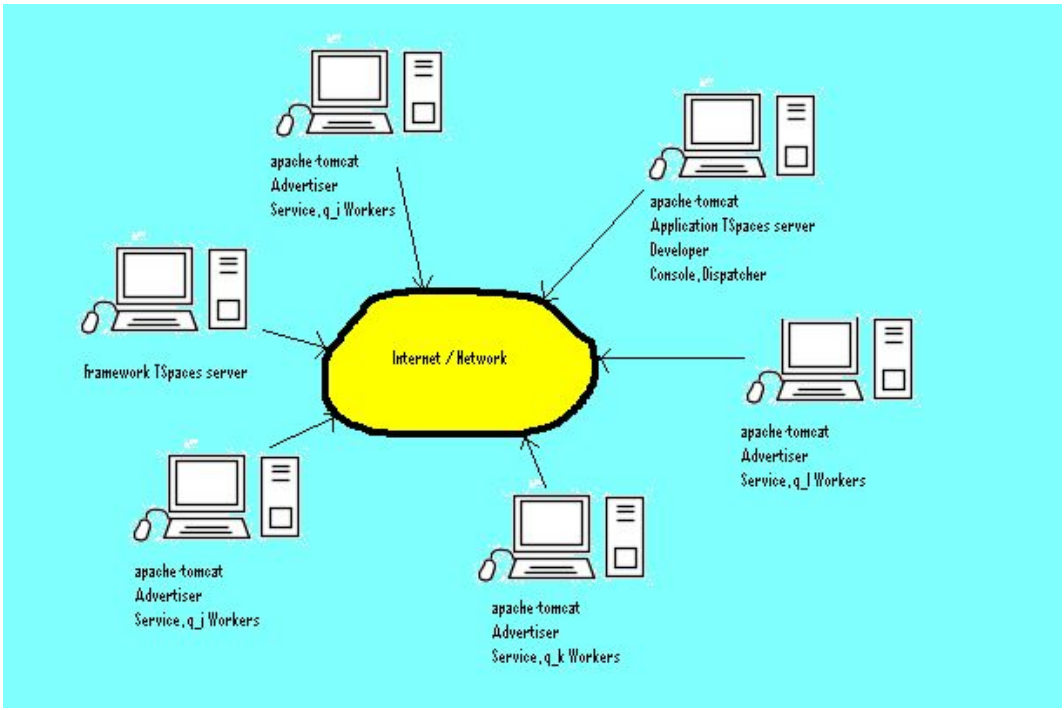


FIGURE 1. The running programs within the framework.

Two tools defines the framework:

- The *Advertiser* tool serves to state the network of workstations. Using this tool a computer is linked to the network of workstations to perform a parallel-distribute computing.

  The graphical interface of the *Advertiser* tool allows to
  - declare the computer as a member of the network of workstations - i.e. a tuple with the name of the computer is written into the tuple space of the network;
  - remove the computer from the network - i.e. remove the above define tuple from the tuple space of the network;
  - show the list of the computers in the network of workstations.

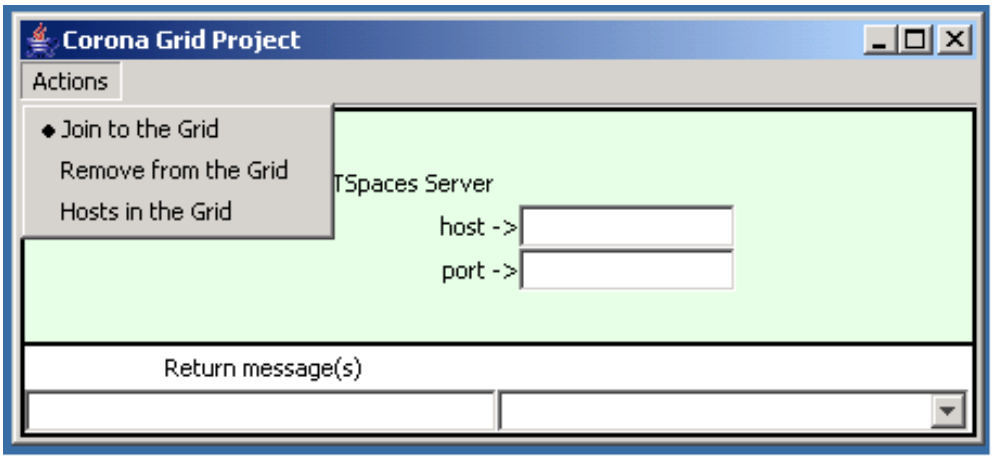


FIGURE 2. The window of the *Advertiser* tool.

- The *Developer* tool allows to
  - compile and arhive the service part of the application;
  - deploy the service part to the workstations of the network. The deployment is done with the *apache - tomcat - deployer;*
  - compile the console part of the application.
  - launch the console to run. In the *Console* class the request are programmed using the *commons - httpclient* software;
  - undeploy the service part of the application.

  These targets are executed throught *apache-ant.*

  A number of parameters are required: the name, the host and the port of the application tuple space, the number of the worker threads, the path to the *apache-ant* and the username and password for the `manager` application of *apache - tomcat.*

  The list of the computers name in the network is required, too. This list may be generated with the *Advertiser* tool, contained in the *Developer* tool, too.

The scenario to run a parallel-distribute application using the framework involves:

(1) To set up the network of workstations:
  (a) On launches the tuple space associated to the network;
  (b) On each workstation starts the *apache-tomcat* Web server and, with the Advertiser tool, declares the availability of that computer joining to the network of workstations.

(2) Using the Developer tool, on the host workstation install, deploy, launch the tuple space associated to the application and execute the parallel-distribute computation.

Conclusions. A framework to deploy and execute parallel-distribute applications is developed. The framework can be ported to any Java enabling platform. Before running an application, workstations can be added or removed. As a drawback of the framework, if a workstation dies then a running application never finishes. We intend to work on this problem. The security constraints are that of the *apache-tomcat* Web server.

The current version of the framework may be downloaded from the author's Web page *http://cs.unitbv.ro/site/pagpers/scheiber* The arhive contains several examples.

References

[1] Z. BAOYIN, "Jcluster A Java Parallel Environment," Distributed with the software, version 1.0.5, 2005.

[2] R. BISSELING, *Parallel Scientific Computation. A structured approach using BSP and MPI,* Oxford Univ. Press, 2004.

[3] G. A. PAPADOPOULOS, F. ARBAB, Coordination models and languages. CWI Report, SEN-R9834, 1998.

[4] E. SCHEIBER, Template for a parallel-distribute Application Based on a messaging Service. Proceedings of ICCCC 2006, Băile Felix, Oradea, Romania, 410-416.

[5] M. SNIR, D. OTTO, S. HUSS-LEDERMAN, D. WALKER, J. DONGARRA, *MPI: The Complete Reference.* MIT Press, Cambridge, MA, 1996.

[6] * * * , "TSpaces-User's Guide & Programmer's Guide," Distributed with the software, Version 2.1.2, 2000.

Transilvania University of Braşov, Romania
E-mail address: scheiber@unitbv.ro