SWC: A Small Framework for WebComputing

Kevin Ying, David Arnow, and Gerald Weiss
Brooklyn College, CUNY
Brooklyn, NY 11210, USA
{kevin,arnow,weiss}@sci.brooklyn.cuny.edu

PAPER TYPE: Extensive

SECTION: (3) System Software & Hardware Architectures

The advent of the Java programming language, with its support for web-deliverable applets, has created a new, promising though peculiar parallel computing platform that some call WebComputing. The essential idea is that a master server, or collection thereof, in league with a collection of web servers coordinates the execution of tasks by applets running in parallel on an ever-changing set of unreliable, heterogeneous client machines. The promise of WebComputing is the potential for achieving an unheard-of degree of parallelism — in principle, a computation could harness every computer that is connected to the Internet. There have been a number of ambitious projects, including Charlotte [BKKW,96], Javelin [CCINSW,97], and Bayanihan [Sarmenta,98], that have explored this platform.

SWC is a small framework, originally developed as a tool for students, that simplifies developing and deploying WebComputing applications.

The framework supports a form of "worker" parallelism, suitable for WebComputing. A master process defines an initial set of tasks and integrates the results of those tasks, possibly defining new tasks along the way. The tasks comprising the computation are defined by lightweight data objects, carried out by workers (applets), and generated dynamically by both the master and the workers themselves. The master can also broadcast control information to all workers. Workers do not communicate directly with each other - they simply return results of tasks and request new ones to carry out.

To use the framework, an application programmer must extend a framework-defined set of abstract classes. These classes define the task computation, the master's actions, the data needed to define a particular task or result, a mechanism for recognizing task equivalence, and a mechanism for recognizing task completion. Using these classes, the framework itself oversees the entire computation and handles all communication.

Two implementations of SWC are provided. One is thread-based and runs on an SMP platform; it serves as development environment when a web-based one is not available or is inconvenient. The other implementation is our main purpose. It actually uses WebComputing. It consists of a multithreaded, servlet-enhanced HTTP server that provides an application control page, creates the master process, downloads the applets and handles all communication. It also provides the classes that define, for both the master process and the applets, their computational structure and their communication tools.

An SWC computation is initiated using a form in a web page provided by the server. A servlet responds by creating the necessary objects within the HTTP server along with an external master process. The master process and the server communicate using TCP and need not reside on the same machine. The master process, using programmer-provided classes creates an initial set of task definitions, which are sent off to the server. The server maintains a collection of task definitions, and uses eager scheduling, as is commonly done in WebComputing, to assign them to applets that have been downloaded into volunteering web clients. Because of the unreliable nature of the applets themselves, the fact that the most obvious candidates for WebComputing will not require large quantities of data to define tasks, and the desire to eliminate system-imposed limits on the number of connections on the server as a potential bottleneck, server-applet communication uses UDP. Because the server does not consider a task complete until the results are actually received, lost packets do not compromise the integrity of the application. To avoid failure to utilize an applet as a result of communication failure, applets use a timeout mechanism to repeatedly send the server their most recent response until the server provides additional work or instructs them to terminate.

As the server receives responses from applets it determines whether these are additional task definitions, in which case they are added to its collection of task definitions, or results from completed tasks in which case they are passed to the master. The latter may, in response, generate new task definitions or control information for the server to distribute to the applets. Control information is immediately broadcast to applets and made available to all future ones.

The SWC system provides a small framework for WebComputing. It has a simple programming interface that WebComputing application programmers may simply implement a number of predefined, application specific modules of the framework and run their application.

[AISS,97] A.D. Alexandrov, M. Ibel, K. E. Schauser, and C. J. Scheiman, *SuperWeb: Research Issues in Java-Based Global Computing*, Concurrency: Practice and Experience, June 1997.

[BKKW,96] Baratloo, M. Karaul, Z. Kedem and P. Wyckoff. *Charlotte: Metacomputing on the Web*, in Proceeding Of the 9th International Conference on Parallel and Distributed Computing System, 1996.

[CCINSW,97] P. Cappello, B. Christiansen, M.F. Ionescu, M.O. Neary, K.E. Schauser and D. Wu, *Javelin: Internet-based parallel computing using Java*, the Sixth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 1997.

[Sarmenta,98] Luis F. G. Sarmenta. Bayanihan: Web-Based Volunteer Computing Using Java. *2nd International Conference on World-Wide Computing and its Applications* (WWCA'98), Tsukuba, Japan, March 3-4, 1998.