2003

A Performance Analysis of Distributed Algorithms in JavaSpaces, CORBA Services and Web Services

Suresh Sunku
University of North Florida

A PERFORMANCE ANALYSIS OF DISTRIBUTED ALGORITHMS IN JAVASPACES, CORBA SERVICES AND WEB SERVICES

By

Suresh Sunku

A Professional Option Project submitted to
The Department of Computer and Information Sciences
In partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

April, 2003

The professional option project "A Performance Analysis of Distributed algorithms in JavaSpaces, CORBA services and Web services" submitted by Suresh Sunku in partial fulfillment of the requirements for the degree of Master of Science in Computer Information and Sciences has been

Approved By | Date

Signature Deleted

5/2/2003

Dr. Roger Eggen
Project Director

Signature Deleted

5/2/2003

Dr. Charles N. Winton
Graduate Director

Signature Deleted

5/5/03

Dr. Judith L. Solano
Department Chairperson

ACKNOWLEDGEMENT

To my lovely wife Sulekha and my dearest children Nikhil and Nisha, without their continuous support and encouragement it would not have been possible to achieve graduation. I also wish to express my gratitude to my co-workers Laura Scholl-Smith, Dorothy Dean and Franco Venturi who have put up with me for more than 2 ½ years.

Table Of Contents

Table Of Figures

ABSTRACT

Implementation of distributed parallel algorithms on networked computers has always been very difficult until the introduction of service-oriented architectures (SOA) like JavaSpaces service, CORBA services and Web Services. Algorithms of the type Master/Worker pattern are implemented with relative ease using the SOAs. This project analyzes the performance of such algorithms on three contemporary SOAs namely JavaSpaces service, CORBA services and Web Services. These architectures make the implementations of distributed algorithms reasonably fault tolerant and highly and dynamically scalable. Also, the systems built on these architectures are generally loosely coupled and operate asynchronously.

In this project we measure and analyze the latency, speed-up and efficiency metrics of an insertion sort of O (n^2) complexity on all the three SOAs. We then draw conclusions of overall performance and scalability on all the three architectures.

Chapter 1

INTRODUCTION

1.1 Early beginnings

Today, with the advent of the Internet and the E-Commerce, many organizations have deployed some type of distributed computing solutions either to scale up or to integrate many disparate systems within and outside the organizations. With the increased adoption of distributed computing paradigm, came the increased number of protocols and technologies thus increasing the complexity of distributed computing.

The early solutions to the distributed computing involved low-level communication such as sockets programming where the messages between client and server were usually encoded in application-level protocols [RMISpec1.4] . So, the next genre of protocols such as RPC (Remote Procedure Call), MPI (Message Passing Interface) and PVM (Parallel Virtual Machine) have encapsulated the low-level communication, but the applications tended to be tightly coupled as with socket programming. In other words, the applications invoked procedures on each other to exchange data or messages and to do so they needed to know a lot about each other. The distributed system developed on these protocols tended to be very unstable and increased the number of points of failure. In other words, when one system or host in the distributed network application failed, then the whole application failed.

As the object-oriented languages provided the developers an ability to abstract away procedures into objects in which both the data and the functions that work on them are

bound together, they became very popular. This led to the adoption of object-oriented protocols such as DCOM, CORBA and RMI. Overall, these protocols have done an excellent job of providing distributed object communication but they are no more than RPC calls with the added ability to marshal the objects when the objects are used as parameters in the method calls. Hence, These protocols did not improve fault-tolerance of applications built using them nor they reduced the tight coupling needed between the components. However, as they were adopted with increasing popularity, there was also a need for object persistence across several invocations [JSSpec1.1].

These drawbacks have motivated the search for a better model and lead to the creation of service-oriented architectures such as JavaSpaces service, CORBA services and Web Services. These new service architectures introduced a new paradigm of developing distributed algorithms that are naturally scalable and reasonably fault tolerant. Also the applications built using these architectures tend to be loosely coupled and dynamically scalable [MKRM].

1.2 Master-Worker Pattern

We have chosen a parallel algorithm based on the Master-Worker pattern, a popular parallel algorithm, to measure the performance on all the three architectures. In a Master-Worker pattern, the master divides a task in to several sub-tasks and then allocates each of the tasks to workers who are ready to execute and return the results. This pattern provides a natural scalability because each worker receives the number of sub-tasks based

its capacity to execute. The faster workers will get larger number of sub-tasks while the slower workers process less number of sub-tasks.

Chapter 2

SERVICE ORIENTED ARCHITECTURES

2.1 JavaSpaces

Sometime around 1998, Sun Micro Systems introduced new JAVA paradigm based on Linda Systems that is known as JavaSpaces. The JavaSpaces technology consists of a tuple space that provides mechanism for storing a group of related objects and retrieving them based on value-matching lookup for specified fields' [JSSpec1.1]. This ability to store and retrieve objects within JavaSpaces not only provides distributed object persistence but also aids in the design of distributed algorithms.

The figure 1 illustrates a distributed algorithm utilizing the space for parallel computing.

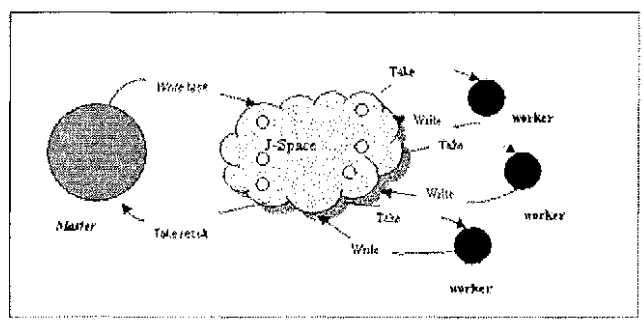

Figure 1

The distributed algorithms implemented using JavaSpace service are naturally scalable since one can start a new worker if there are more tasks written to the space than the number of tasks taken from the space for processing. Also, the applications are not only loosely coupled but they are anonymous too because the workers and the master/client are never aware of each other.

The JavaSpace service is a JINI service and hence can be combined with JINI transaction service to provide and maintain integrity of the space. A worker can start a transaction before taking a task from the space and end the transaction after the successful write of the result-task back to the space. This provides a degree of fault tolerance since the transaction manager will restore the original task back into space when a worker crashes or fails to complete and this allows another worker to take the task and process to completion.

The JavaSpaces API consists of 4 main method types:

- Write() - writes an entry to a space.
- Read() - reads an entry from a space.
- Take() - reads an entry and deletes it from a space.
- Notify() - registers interest in entries arriving at a space.

All objects must be serializable since Write() writes the serialized bytes to the space and the Read()/Take() reconstructs the object from the serialized bytes.

2.2 CORBA Services

Around 1997 the OMG group published various service specifications that run on top of CORBA specification. Although there is no single CORBA service that is comparable to JavaSpaces service, one can combine multiple services such as the Trader service and the Transaction service to build and implement distributed parallel algorithms. The CORBA's Trading Service is similar to the yellow pages; it allows objects that can

provide a service to register their abilities and it then allows clients to locate those services by describing the functionality they require. From then on, the client deals directly with the service provider object.

Since the Trader service is defined to be language neutral and designed to run on any architecture, it is not designed to store objects itself but to store CORBA remote references to the objects. Hence, the algorithms using the Trader service must interact with the service provider object to interchange the actual sub-tasks and the result. Typical steps involved using Trader service is shown in figure 2.

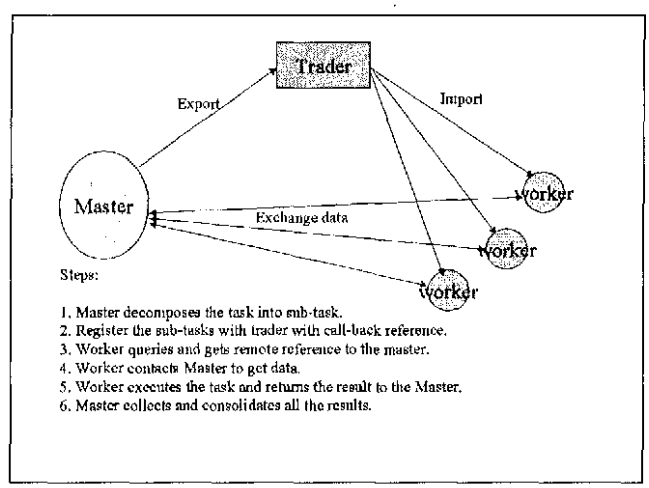

Figure 2

Like JavaSpaces, here too, the number of tasks performed by a worker depends on their capacity and the load on their respective computers. One can easily start a new worker to improve performance and when combined with CORBA transaction service, one can build very robust and fault-tolerant solutions. However, unlike in the JavaSpaces, the

master and the worker are not anonymous, as they need to know of each other to exchange the data.

2.3 Web Services

The Web Services (WS) architecture is modeled more like CORBA trader service in that it also employs a trader like service registry that can be queried to discover services and it is also designed to be language neutral. However, the WS APIs are designed to be operable across World Wide Web and firewalls. Typical steps involved in using Web Services are shown in figure 3.

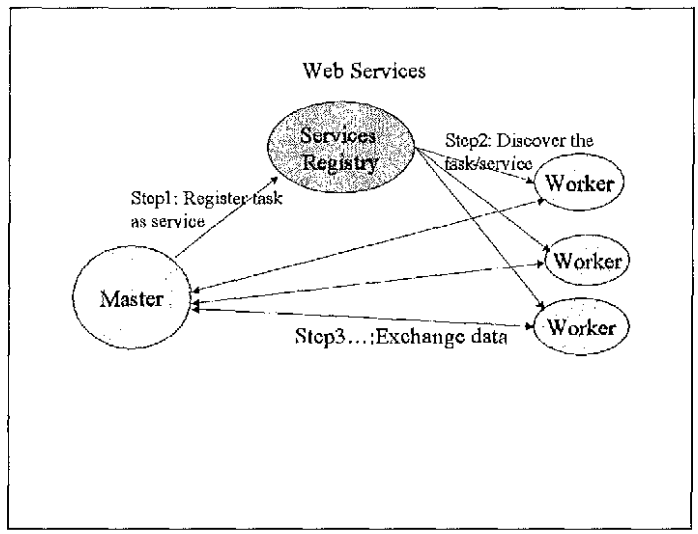

Figure 3

Unlike the above 2 architectures, here the Master can not create and register sub-tasks because there is no way for a Worker to remove a service from the registry once it is consumed. Hence, the Master registers the whole task as one service but divvy up a portion of array to each Worker as the Workers contact for sub-tasks.

The Web Services specifications consists of 3 sets of API namely UDDI (Universal description, discovery and integration), WSDL (Web Services Description Language) and SOAP (Simple Object Activation Protocol). The service providers register the service offerings using UDDI and/or WSDL. And the service consumers query the service registries to obtain protocol bindings to the end points, in other words the service providers. After the discovery the service provider and the consumer exchange data directly.

The web services specification is still a work-in-progress and many features such as objects as payloads, security and choreography (transactions across multiple messages) are still being worked on.

Chapter 3

THE RESEARCH

3.1 Overview

First, we developed a distributed algorithm to perform an insertion sort using JavaSpaces. The insertion sort was chosen because it has both average and worst case performance in $O(n^2)$ and can put significant execution demand on servers. We then developed a similar algorithm using CORBA services and Web Services while making every effort to keep the algorithms as similar as the architectures allowed us to be.

The goal of the study is to evaluate the performance of distributed algorithms using JavaSpaces, CORBA Services and Web Services by measuring latency, speed-up and efficiency. We then draw our conclusions on scalability of each of the three architectures. However, In this study, we do not attempt to make a straight comparison between JavaSpaces, CORBA Services and Web Services because there are many issues that can affect the performance other than the architectures itself. For one thing, our implementation in their respective architectures may not be optimal and may be contributing to the overhead. Also, the products that we used may not be the best of the breed in the market and hence adding to the overhead. However, we made every effort to implement our algorithm consistently across the architectures and followed the best practices of the respective architectures. One example is that in our implementation using CORBA services, the remote references are cleaned up as soon their need is over, causing an additional communication loop to the remote reference end-point. While in

JavaSpaces the objects are serialized and de-serialized at the site where they are used and hence there is no additional trip to destroy the objects. And in Web Services the objects are never exchanged and so the clean up never arises.

3.2 Hardware

The hardware for this study consists of a cluster of homogeneous workstations all running RedHat Linux v7.2. The machines are all Intel based PCs consisting of single 500 MHz processors connected by 100 megabit fast Ethernet.

3. 3 Software

The software is Java (TM) 2 Runtime Environment, Standard Edition (build 1.3.1) [Orbacus] available free from Sun. In order to keep the variables in performance evaluation to low, the Java language environment is used. We have used GigaSpaces [Orbacus] an implementation of JavaSpaces, ORBACUS 4.0.5 [Giga] an implementation of CORBA services and Axis an implementation of SOAP to develop and evaluate the performance of distributed parallel algorithms. All of the software mentioned above is available for free from the GigaSpaces corp., Iona technology inc., and Apache foundation respectively for academic use.

Chapter 4

THE RESULTS

4.1 Testing

Although the initial goal was to measure the performance of SOAs only, but we have decided to include the performance metrics of a MPI (Message Passing Interface) implementation of Master/Worker pattern. Since MPI is more popular for parallel architectures, we thought it would be more meaningful to contrast the observations of SOAs against the MPI's numbers.

We ran a series of executions for each version of the architectures using 8K, 16K, 32K & 64K integers and using 1, 2, 4, and 8 workers/servers. The data are distributed so that each server has the same amount of data. The servers do all the work while the client only distributes and receives data. All tests were executed under similar conditions for each of the SOAs during a time when the load on the dedicated network and servers was at a minimum.

Observed data: The table below summarizes the observed data.

CORBA Services					JAVA Spaces Service			
Input Size	Ts (P=1)	Tp (P=2)	Tp (P=4)	Tp (P=8)	Ts (P=1)	Tp (P=2)	Tp (P=4)	Tp (P=8)
8K	6060	4573	4368	4385	3892	2674	2534	2595
16K	11160	6087	5050	4545	9058	4093	2753	2961
32K	29598	11210	6454	5498	29962	9350	4735	3526
64K	110607	33292	15317	8720	111780	30272	11236	6713
Web Services					MPI			
Input Size	Ts (P=1)	Tp (P=2)	Tp (P=4)	Tp (P=8)	Ts (P=1)	Tp (P=2)	Tp (P=4)	Tp (P=8)
8K	197624	130936	89821	70893	2314	2107	1900	1593
16K	330012	259660	177830	136700	8128	2789	2234	2429
32K	689287	518070	352723	272821	31558	9496	3316	2646
64K	1488396	1056166	709469	538059	106362	29044	10120	4466

Figure 4

Note:

Ts: Time taken for a sequential process, in other words time taken with 1 worker.

Tp: Time taken for a parallel process with P processors.

4.2 Latency:

We measured latency as time taken for an integer to make a round trip. The figure 3 shows the latency for each implementation. We have observed higher latency in using CORBA services and Web Services over other architectures.

	Latency (micro sec)
JavaSpaces	1842
CORBA Services	4200
Web Services	4203
MPI	912

Figure 5

The following summarizes our assumptions for the increased latency:

A) CORBA: It takes 2 additional hops to complete the loop Master-Worker-Master than the other three architectures due to the CORBA's use of remote references that requires additional steps to clean up the remote objects. While in JavaSpaces it takes only four hops because JavaSpaces architecture serializes the objects at source and de-serializes the objects at destination. Since the objects are instantiated locally at destination, there is no need for communication over the wire to destroy an object reference. Also in JavaSpaces the un-referenced objects are deleted asynchronously, the time to destroy an object is never included in the measured time. The following shows the six steps needed to complete a loop:

- The Master writes remote reference to a sub-task to the trader.
- The Worker extracts the reference.
- The Worker gets data using the remote reference.
- The Worker invokes remote delete on the Sub-task reference.
- The Worker then instantiates an Iterator for result set and pass the reference to Master

- Master uses the remote iterator to get data and then invokes a remote delete of iterator.

B) Web Services: We think that the implementation of AXIS as a servelet running in TOMCAT servlet container is one of the major reasons for high latency in our Web Services architecture. Also our set-up and configuration (or lack of it) of TOMCAT could have contributed to latency. We think that our service was instantiated on the first call to service, which in turn caused for a new JVM (Java Virtual Machine) to come up and thus increasing the latency.

4.3 The Speed-up

Speed-up is defined as ratio of time taken to process sequentially to time taken to process in parallel. We measured sequential time as the time taken with one worker.

The table below shows the speed-up of observed data:

CORBA Services					JAVA Spaces Service			
Input Size	Ts (P=1)	Tp (P=2)	Tp (P=4)	Tp (P=8)	Ts (P=1)	Tp (P=2)	Tp (P=4)	Tp (P=8)
8K	1	1.3252	1.3874	1.382	1	1.4555	1.5359	1.4998
16K	1	1.8334	2.2099	2.4554	1	2.213	3.2902	3.0591
32K	1	2.6403	4.586	5.3834	1	3.2045	6.3278	8.4974
64K	1	3.3223	7.2212	12.684	1	3.6925	9.9484	16.651
Web Services					MPI			
Input Size	Ts (P=1)	Tp (P=2)	Tp (P=4)	Tp (P=8)	Ts (P=1)	Tp (P=2)	Tp (P=4)	Tp (P=8)
8K	1	1.6423	1.7558	1.7344	1	1.0982	1.2179	1.4526
16K	1	1.9421	3.0936	3.0153	1	2.9143	3.6383	3.3462
32K	1	3.1004	6.1728	8.332	1	3.3233	9.5169	11.927
64K	1	3.7909	10.487	17.408	1	3.6621	10.51	23.816

Figure 6

Figure 7, 8, 9 and 10 display the speed-up achieved by all the three architectures. It is clear from the charts that all the systems had increasing speed-up as the processors and input size are increased. The SOAs have achieved a speed-up comparable to the speed-up observed with MPI .

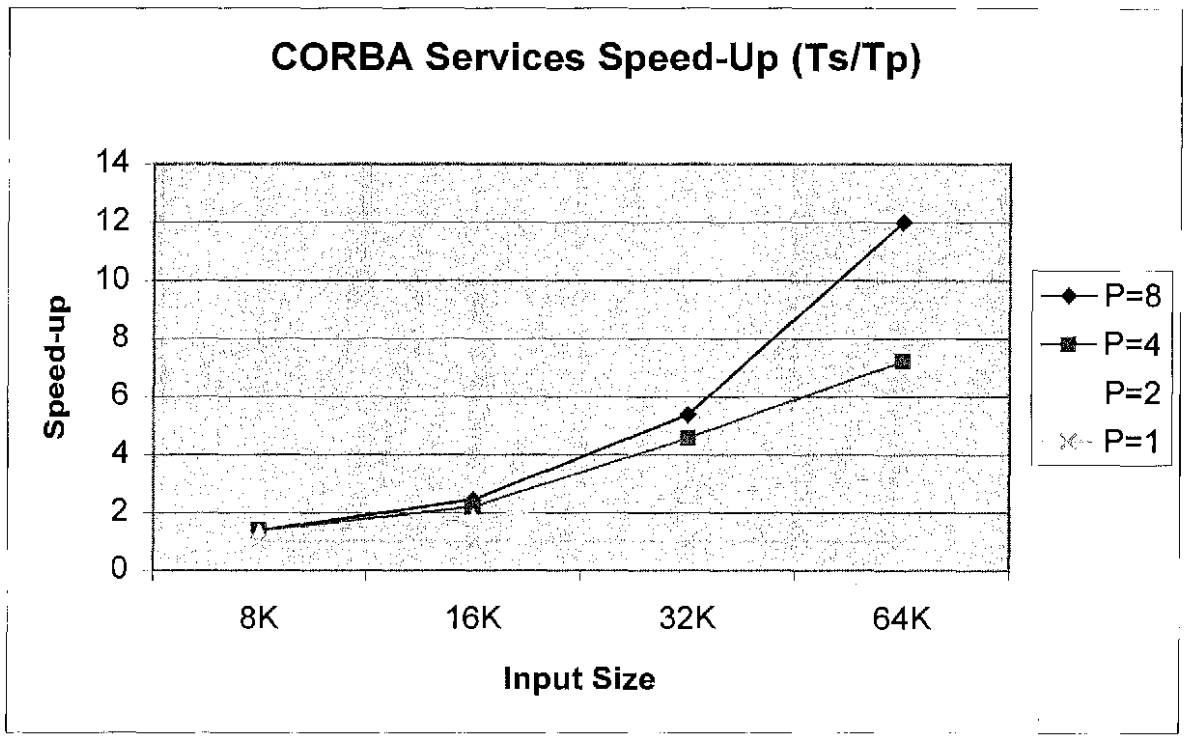

Figure 7

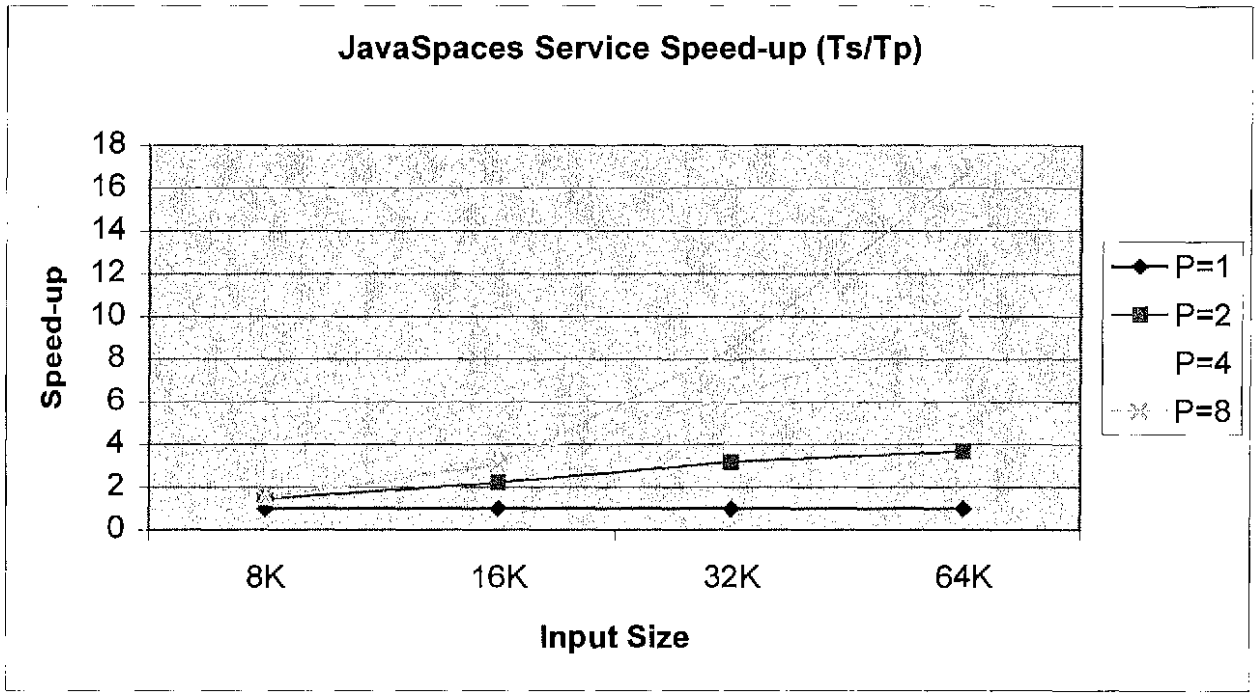

Figure 8

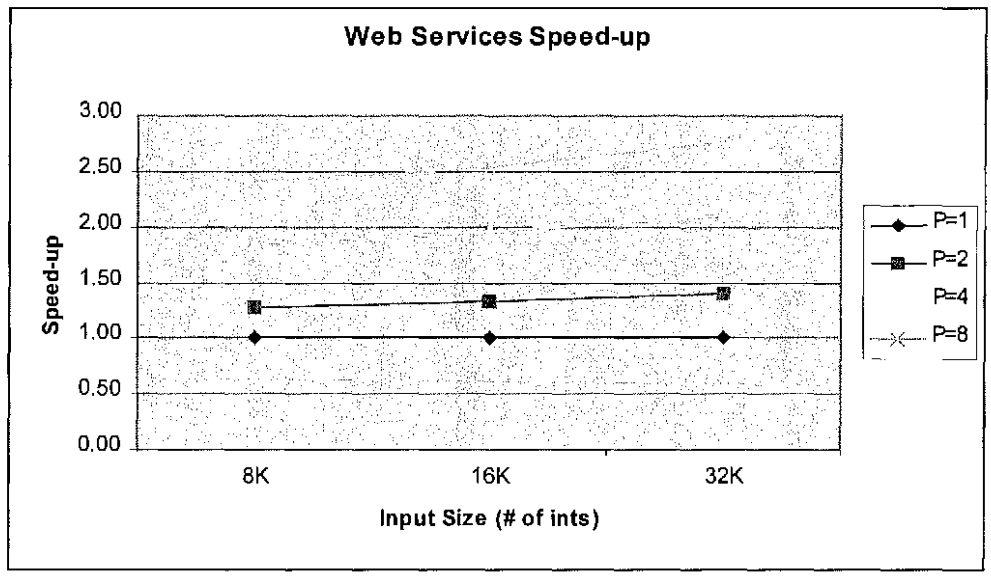

Figure 9

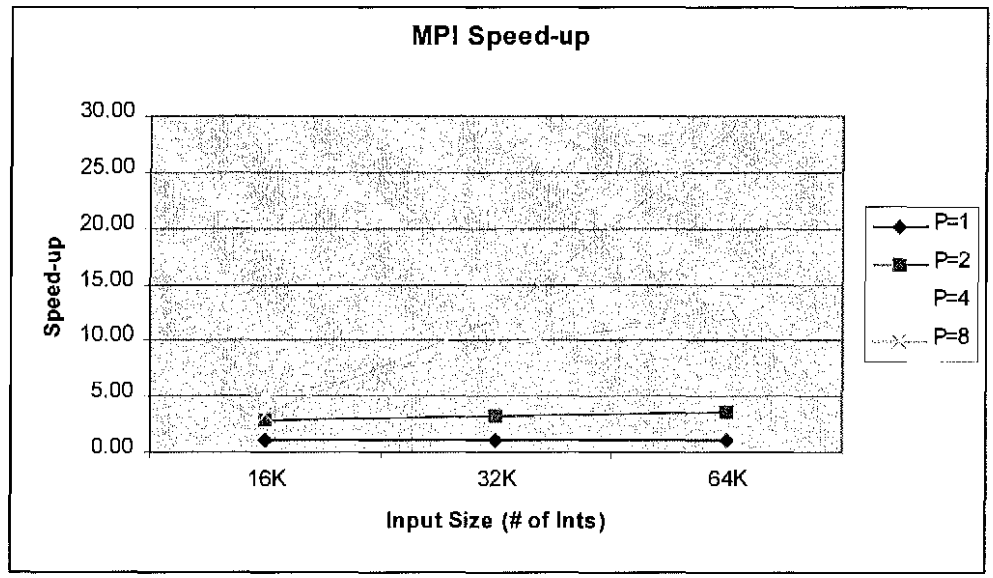

Figure 10

4.4 Efficiency

The efficiency is defined as Ts/P*Tp. The figure below charts the efficiency of all the four architectures. It uses the diagonal values from the table of observed data, in other words it uses the data in the cells where the input and the processors are doubled.

Efficiency: (Ts/PTp)	1	2	4	8
CORBA Services	1	0.9167	1.1465	1.5855
Java Spaces	1	1.1065	1.5819	2.0814
Web Service	1	0.6355	0.4886	0.3456
MPI	1	1.4572	2.3792	2.977

Figure 11

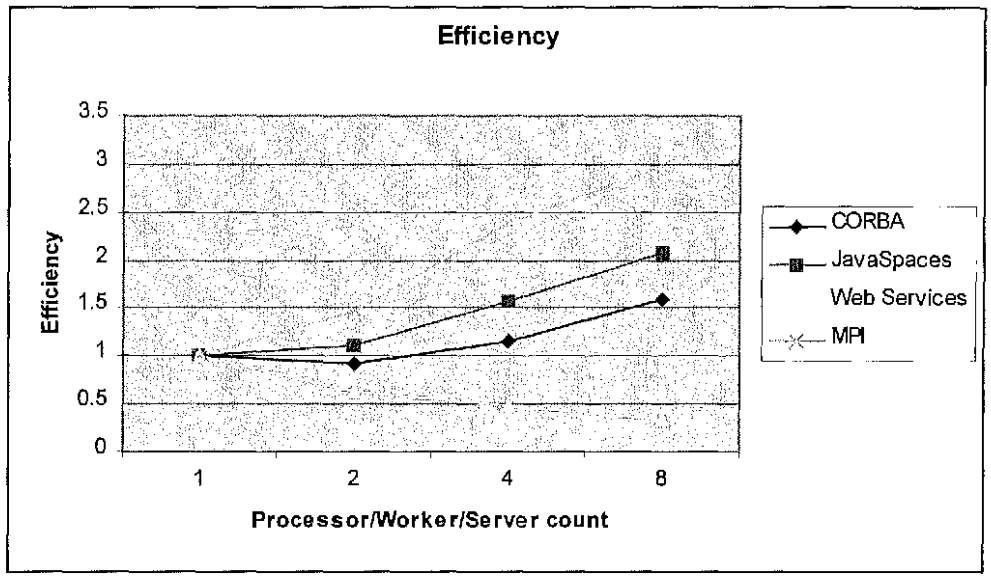
Efficiency
3.5
3
2.5
2
1.5
1
0.5
0
Efficiency
1
2
4
8
Processor/Worker/Server count
CORBA
JavaSpaces
Web Services
MPI

Figure 12

Chapter 5

SUMMARY AND CONCLUSIONS

Before we make any conclusion from the data, one must keep in mind that the performance metrics from MPI is for reference only. We can not expect any of the SOAs, as they are implemented by us, to out perform MPI implementation because the MPI code was written in "C", which allowed us to run the executable. While all of the other implementations were done in JAVA, which uses byte code and runs in interpreter mode. However, the CORBA and the Web Service specifications are language neutral and that means that we could have implemented in "C++" for better performance but we wanted to be consistent across all the implementations of the SOAs.

Overall the From the observed data we can conclude that the distributed parallel algorithms of type Master-Worker pattern have performed very well on all of the SOAs except in Web Services architecture. The observed speed-up and efficiency for CORBA Service and JavaSpace service is comparable to that of MPI solution and indicates that these architectures really scale well. However, we can not say the same thing for implementation of the Web Services. Also all the three SOA implementations have higher latency but it seems to be constant irrespective of input size and processor count and that suggests that for large tasks the latency is negligible.

As it is shown above that these architectures provide tremendous benefits in terms of fault-tolerance and dynamic scalability that the performance hit we take is negligible over traditional parallel implementations such as MPI. Since the SOAs support object-oriented

languages, the benefits of Object Oriented programming can be and probably should be realized when doing parallel distributed processing. No study is complete with out mentioning the ease of use and implementation of the three SOAs. The development and implementation of Java Spaces was the easiest of all since the specification only has five commands and the implementation is a lot cleaner. However, it is limited in that the JavaSpaces can only be implemented on Java platforms supporting JINI architecture. Relatively the use and implementation of Web Services is harder than JavaSpaces, it is not difficult to master. However, we can not say the same for CORBA specification but CORBA's specification is language and platform independent and also provides very good scalability. Of the three SOAs, we find that debugging is lot easier in Web Services because the wire protocol is Unicode, which allowed us to read message payload with out any need of special tools.

Today there is a lot of work is being done in the field of Grid computing also known as On-Demand computing where the concept of cluster computing is expanded to embrace network of computers that are spread over wide area some times over different continents. The grid-computing platform appears to be a natural platform to implement distributed and parallel algorithms. A performance study should be conducted to see how well the grid platform scale up for distributed parallel algorithms.

Very recently the Grid-computing community has adopted service-oriented architecture to provide the users access to computing resources on-demand which only validates the benefits of using SOAs.

REFERENCES

[RMISpec1.4]
RMI Specification, http://java.sun.com/j2se/1.4/docs/guide/rmi/spec/rmi-intro2.html

[JSSpec1.1]
JavaSpaces Service Specification
http://wwws.sun.com/software/jini/specs/js1_1.pdf

[JDK1.3]
The source for Java Technology http://java.sun.com

[Giga]
GigaSpaces http://www.gigaspaces.com

[Orbacus]
Iona Technologies Orbacus home page
http://www.iona.com/products/orbacus_home.htm

[Trader]
Trading Object Service 1.0
http://www.omg.org/technology/documents/formal/trading_object_service.htm

[MKRM]
E. Freeman, S. Hupfer; "Make Room For JavaSpaces, Part 1 Ease the Development Of Distributed Apps with JavaSpaces"
http://www.artima.com/jini/jiniology/js1.html

[UDDI 1.0]
Universal Description, Discovery and Integration standard organization:
http://www.uddi.org

APPENDIX A

JavaSpaces Code Listings

```
*****************************************************
*            JavaSpaces Client listing              *
*                JspaceClient.java                  *
*****************************************************
package client;
import java.util.*;
import java.io.*;
import net.jini.space.*;
import net.jini.core.transaction.*;
import net.jini.core.transaction.server.*;
import net.jini.core.lease.Lease;
import com.j_spaces.core.IJSpace;
import java.rmi.RMISecurityManager;
import com.j_spaces.core.client.SpaceFinder;
import com.j_spaces.core.client.LookupFinder;
import utils.*;

/**
 * This class is responsible for splitting the task of sorting ints
 * into sub tasks, gathering the partial results and return
 * the final result.
 */

public class JSpaceClient
{
    private static Properties clientprop;
    private static utils.arrayHolder thearray = null;

    public static void
    main(String args[]) throws Exception
    {
    // check usage
    if (args.length != 2)
        usage();

      long starttime=System.currentTimeMillis();

    // extract params
    String lookupHost = args[0];
    int taskcount = Integer.parseInt(args[1]);

    // set RMI Security Manager
     if ( System.getSecurityManager() == null )
     {
       System.setSecurityManager( new RMISecurityManager() );
      }

    // get references to space and transaction services
    JavaSpace space = (JavaSpace)SpaceFinder.find( lookupHost );

    System.out.println("Looking for Transaction Manager...");
```

```
    TransactionManager trManager =
(TransactionManager)LookupFinder.find(
                              null,       // service name
                              new Class[] {
net.jini.core.transaction.server.TransactionManager.class }, // service
class name
                              null,       // service attributes
                              "localhost", // unicast lookup host
                            null,       // lookup groups
                                  10*1000     // timeout 10 seconds
                            );

    if ( trManager == null )
    {
     System.out.println("Transaction Manager can not be found...");
     System.exit(1);
    }
    else
     System.out.println("Found: Transaction Manager." );


     clientprop = new Properties();
     try
     {
      FileInputStream pFIS = new FileInputStream("client.properties");
                  clientprop.load(pFIS);
                  pFIS.close();
     }
     catch(Exception e)
     {
      e.printStackTrace();
            System.exit(9);
     }

      System.out.println("Loading ints.....");
     try
     {
            thearray = new utils.arrayHolder(
                  clientprop.getProperty("input","testfile"),

Integer.parseInt(clientprop.getProperty("maxsize","0")));
            System.out.println("Done loading ints: " +
thearray.getLength());
     }
     catch(Exception e) {e.printStackTrace();System.exit(9);}
     // sort the ints.
     sortInts(space, trManager, taskcount);
     // finish
     long endtime=System.currentTimeMillis();

        System.err.println("==========JS
Sort======================");
        System.err.println(" Problem Size: " + thearray.getLength());
        System.err.println(" Processor count: " +  taskcount);
        System.err.println(" Elapsed time: " + (endtime - starttime));

System.err.println("=========================================");
```

```
      System.exit(0);
}

  public static void sortInts(JavaSpace space, TransactionManager
trManager, int taskcount)
    throws Exception
  {
    // create a new transaction
    Transaction.Created tCreated = TransactionFactory.create(
            trManager, 3600 * 1000);
    Transaction tr = tCreated.transaction;

    // break task into several tasks and write them to space in one
transaction
    //int numOfTasks = (int) Math.sqrt((double) thearray.getLength());
    int numOfTasks = taskcount;
    int numbersPerTask = (thearray.getLength()) / numOfTasks;
    System.out.println("Creating " + numOfTasks + " tasks, " +
      numbersPerTask + " numbers per task.");
    for (int i=0; i<numOfTasks; i++)
    {
      int start = i * numbersPerTask;
      if (i == numOfTasks - 1)
       numbersPerTask = thearray.getLength() - start;
      SortTask task = new SortTask(thearray.getArray(start,
numbersPerTask));
      space.write(task, tr, Lease.FOREVER);
    }

    // commit transaction
    tr.commit();

    // wait for any result.
    // create a new transaction
    System.out.println("Waiting for results");
    tCreated = TransactionFactory.create(
      trManager, 3600 * 1000);
    tr = tCreated.transaction;
    int numOfResults = 0;
    int[][] resultsets = new int[numOfTasks][];
    while (numOfResults < numOfTasks)
    {
      ResultTask template = new ResultTask();
      template.sortedarray = null;
      ResultTask taskResult = (ResultTask)
        space.take(template, tr, Long.MAX_VALUE);

      resultsets[numOfResults] = taskResult.getResult();
      numOfResults++;
    }

    tr.commit();
    PrintUtils.printInts(resultsets, numOfResults);
    return ;
  }
    static void
```

```
    usage()
    {
        System.err.println();
        System.err.println("Usage:  java client.JSpaceClient [-h]
SpaceName TaskCount");
            System.err.println("also refer to client.properties file to
set " +
                        "properties.");
        System.err.println();
        System.err.println("Options:");
        System.err.println();
        System.err.println("  -h    Show this help message");
        System.exit(1);
    }
}
*******************************************************
*            JavaSpaces Worker listing                *
*                 JSpaceWorker.java                   *
*******************************************************
package worker;
import java.util.*;
import java.io.*;
import net.jini.space.*;
import net.jini.core.transaction.*;
import net.jini.core.transaction.server.*;
import net.jini.core.lease.Lease;
import com.j_spaces.core.IJSpace;
import java.rmi.RMISecurityManager;
import com.j_spaces.core.client.SpaceFinder;
import com.j_spaces.core.client.LookupFinder;
import utils.*;

/**
 * This class is responsible for extracting from the space and
executing
 * the task of sorting ints. Once the sort is completed, the result is
 * written back to the space.
 */

public class JSpaceWorker
{
    public static void
    main(String args[]) throws Exception
    {
    // check usage
    if (args.length != 1)
        usage();

    // extract params
    String lookupHost = args[0];

    // set RMI Security Manager
     if ( System.getSecurityManager() == null )
     {
       System.setSecurityManager( new RMISecurityManager() );
      }
```

```
    // get references to space and transaction services
    JavaSpace space = (JavaSpace)SpaceFinder.find( lookupHost );

    System.out.println("Looking for Transaction Manager...");
    TransactionManager trManager =
(TransactionManager)LookupFinder.find(
                              null,       // service name
                              new Class[] {
net.jini.core.transaction.server.TransactionManager.class }, // service
class name
                              null,       // service attributes
                              "localhost", // unicast lookup host
                            null,       // lookup groups
                                  10*1000     // timeout 10 seconds
                            );

    if ( trManager == null )
    {
     System.out.println("Transaction Manager can not be found...");
     System.exit(1);
    }
    else
     System.out.println("Found: Transaction Manager." );


     // executing the task.
     executeSort(space, trManager);
     // finish
     System.exit(0);
}

  public static void executeSort(JavaSpace space, TransactionManager
trManager)
    throws Exception
  {

    // create a new transaction

    // get task if any
      SortTask template = new SortTask();
      template.sortarray=null;
    while(true)
    {
       Transaction.Created tCreated = TransactionFactory.create(
            trManager, 3600 * 1000);
      Transaction tr = tCreated.transaction;
System.out.println("Searching space for SortTask...");

      SortTask task = (SortTask) space.take(template, tr,
Long.MAX_VALUE);
      if (task == null)
      {
        tr.commit();
          break;
      }
System.out.println("Got the handle for SortTask!");
      ResultTask resulttask = new ResultTask(task.run());
```

```
        space.write(resulttask, tr, Lease.FOREVER);
        // commit transaction
        tr.commit();
      }

      return ;
    }
      static void
      usage()
      {
          System.err.println();
          System.err.println("Usage:  java client.JSpaceWorker [-h]
SpaceName ");
          System.err.println();
          System.err.println("Options:");
          System.err.println();
          System.err.println("  -h    Show this help message");
          System.exit(1);
      }
}
*****************************************************
*            JavaSpaces Utilities listings          *
*****************************************************
package utils;
import java.io.*;
import java.util.*;
public class arrayHolder
{
      public int[] thearray= null;
      public arrayHolder(String infile, int maxsize)
            throws Exception
      {
            int arraysize=0;
            if (maxsize == 0)
                  maxsize = 160000;
            int[] intarray = new int[maxsize];

            BufferedReader br = new BufferedReader(new
FileReader(infile));

            String line = null;
            StringTokenizer stk = null;
            while((line=br.readLine()) != null)
            {
                  stk = new StringTokenizer(line);
                  while(stk.hasMoreTokens())
                  intarray[arraysize++] =
Integer.parseInt(stk.nextToken());
            }
            thearray = new int[arraysize];
            for(; arraysize > 0 ; arraysize--)
            thearray[arraysize-1] = intarray[arraysize-1];
      }
      public int getLength() { return thearray.length;}
      public int[] getArray(int start, int size)
      {
            int[] retarray = new int[size];
```

```
            for (int i = 0; i < size; i++)
            {
                  retarray[i] = thearray[i+start];
            }
            return retarray;
      }
}
package utils;

import net.jini.core.entry.Entry;

/**
 * This object encapsulates a task entry in space. The performer of
this task
 * simply executes the run method.
 */
public class ResultTask
  implements Entry
{
  public int[] sortedarray;
  public ResultTask()
  {
  }

  public ResultTask(int[] inarray )
  {
    sortedarray = inarray;
  }

  public int[] getResult()
      {
            return sortedarray;
      }
}
package utils;

import net.jini.core.entry.Entry;

/**
 * This object encapsulates a task entry in space. The performer of
this task
 * simply executes the run method.
 */
public class SortTask
  implements Entry
{
  public int[] sortarray;
  public SortTask()
  {
  }

  public SortTask(int[] inarray )
  {
    sortarray = inarray;
  }

  public int[] run()
```

```
      {
System.out.println("Length of the array to be processed is: " +
sortarray.length);
             for (int i = sortarray.length; --i>=0;)
            {
                  boolean flipped = false;
                  for (int j = 0; j<i; j++)
                  {
                        if (sortarray[j] >  sortarray[j+1])
                        {
                              int T = sortarray[j];
                              sortarray[j] = sortarray[j+1];
                              sortarray[j+1] = T;
                              flipped = true;
                        }
                  }
                  if (!flipped) { return sortarray; }
            }
System.out.println("completed sort of: " + sortarray.length);
            return sortarray;
      }
}
```

APPENDIX B

CORBA Trader Service Code Listings

```
*****************************************************
*            CORBA Trader IDL (SortInt.idl)         *
*****************************************************
module worker
{
      typedef sequence<long>IntSet;
      exception EndOfSetReached{};

//iterator to pass ints between clients
      interface IntIterator {
            boolean getNextSet(in long setsize, out IntSet subset)
            raises(EndOfSetReached);
            oneway void destroy();
      };
//clients callback object for the server to return sorted ints
      interface ClientCallBack {
            oneway void getSortedInts(in long intcount, in IntIterator
srvrItr);
      };
//sort int server
      interface Sorter {
            oneway void sortInts(in long intcount, in IntIterator
clientItr, in ClientCallBack ccb);
      };
};


*****************************************************
*             CORBA Trader Client listing           *
*****************************************************
package client;
import java.util.*;
import java.io.*;

public class TraderClient
{
      private static Properties clientprop;
      private static int psize;
      private static int pcount;
    private static int
    run(org.omg.CORBA.ORB orb, String args[])
    {
        //
        // Use argc to position us past the last option
        //
        int argc;

        //
        // Check options
        //
        for(argc = 0 ; argc < args.length ; argc++)
```

```
            if(args[argc].equals("-h"))
                usage();

        String constraint, preference ,policyFile;
            constraint = clientprop.getProperty("constraint");
            preference = clientprop.getProperty("preference");
            policyFile = clientprop.getProperty("policyfile");

        //
        // Connect to Trading Service
        //
        org.omg.CosTrading.Lookup lookup =
utils.SortUtils.connect(orb);

        if(lookup == null)
            usage();

        try
        {
            String type = utils.SortUtils.getServiceType();

            org.omg.CosTrading.Policy[] policies;
            if(policyFile != null)
                policies = getPolicies(orb, policyFile);
            else
                policies = new org.omg.CosTrading.Policy[0];

            org.omg.CosTrading.LookupPackage.SpecifiedProps
desiredProps =
                new org.omg.CosTrading.LookupPackage.SpecifiedProps();
            desiredProps.__default(
                org.omg.CosTrading.LookupPackage.HowManyProps.all);

            org.omg.CosTrading.OfferSeqHolder offers =
                new org.omg.CosTrading.OfferSeqHolder();
            org.omg.CosTrading.OfferIteratorHolder iter =
                new org.omg.CosTrading.OfferIteratorHolder();
            org.omg.CosTrading.PolicyNameSeqHolder limits =
                new org.omg.CosTrading.PolicyNameSeqHolder();

            //
            // Perform the query
            //
            System.out.println("TraderClient querying Trader...");
            lookup.query(type, constraint, preference, policies,
desiredProps,
                         20, offers, iter, limits);

            if(offers.value.length == 0 && iter.value == null)
            {
                System.out.println("No offers found.");
                return 0;
            }
            else
                System.out.println(offers.value.length +
            " offers found!!!!!!!!!!");
```

```
            //Execute the offers
            //

        if (offers.value.length > 0)
                    executeOffers(offers.value, orb);
        else
            //
            // If we received an iterator, then add all of its
            // offers to the vector
            //
            if(iter.value != null)
            {
                org.omg.CosTrading.OfferSeqHolder seq =
                    new org.omg.CosTrading.OfferSeqHolder();
                iter.value.next_n(20, seq);
                executeOffers(seq.value, orb);
                iter.value.destroy();
            }
          }
          catch(org.omg.CosTrading.IllegalServiceType e)
          {
              System.err.println("Illegal service type '" + e.type +
"'");
          }
          catch(org.omg.CosTrading.UnknownServiceType e)
          {
              System.err.println("Unknown service type '" + e.type +
"'");
          }
          catch(org.omg.CosTrading.IllegalConstraint e)
          {
              System.err.println("Illegal constraint");
          }
          catch(org.omg.CosTrading.LookupPackage.IllegalPreference e)
          {
              System.err.println("Illegal preference");
          }
          catch(org.omg.CosTrading.LookupPackage.IllegalPolicyName e)
          {
              System.err.println("Illegal policy '" + e.name + "'");
          }
          catch(org.omg.CosTrading.LookupPackage.PolicyTypeMismatch e)
          {
              System.err.println("Policy type mismatch for '" +
                                 e.the_policy.name + "'");
          }
          catch(org.omg.CosTrading.LookupPackage.InvalidPolicyValue e)
          {
              System.err.println("Invalid policy value for '" +
                                 e.the_policy.name + "'");
          }
          catch(org.omg.CosTrading.IllegalPropertyName e)
          {
```

```
            System.err.println("Illegal property name '" + e.name +
"'");
        }
        catch(org.omg.CosTrading.DuplicatePropertyName e)
        {
            System.err.println("Duplicate property name '" + e.name +
"'");
        }
        catch(org.omg.CosTrading.DuplicatePolicyName e)
        {
            System.err.println("Duplicate policy name '" + e.name +
"'");
        }
        catch(org.omg.CORBA.SystemException e)
        {
            e.printStackTrace();
        }

        return 0;
    }

    public static void
    main(String args[])
    {
      long starttime=System.currentTimeMillis();

            clientprop = new Properties();
            try
            {
                  FileInputStream pFIS = new
FileInputStream("client.properties");
                  clientprop.load(pFIS);
                  pFIS.close();
            }
            catch(Exception e)
            {
                  e.printStackTrace();
            System.exit(9);
            }

        java.util.Properties systemprops = System.getProperties();
        systemprops.put("org.omg.CORBA.ORBClass", "com.ooc.CORBA.ORB");
        systemprops.put("org.omg.CORBA.ORBSingletonClass",
                  "com.ooc.CORBA.ORBSingleton");

        int status = 0;
        org.omg.CORBA.ORB orb = null;

        try
        {
            args = com.ooc.OBCORBA.ORB_impl.ParseArgs(args,
systemprops, null);
            orb = org.omg.CORBA.ORB.init(args, systemprops);
            status = run(orb, args);
        }
        catch(RuntimeException ex)
        {
```

```
            ex.printStackTrace();
            status = 1;
        }

        if(orb != null)
        {
            try
            {
                ((com.ooc.CORBA.ORB)orb).destroy();
            }
            catch(Exception ex)
            {
                ex.printStackTrace();
                status = 1;
            }
        }

      long endtime = System.currentTimeMillis();
      System.err.println("============CORBA
Sort====================");
      System.err.println(" Problem Size: " + psize);
      System.err.println(" Processor count: " +  pcount);
      System.err.println(" Elapsed time: " + (endtime - starttime));
      System.err.println("========================================
=");
        System.exit(status);
    }

    static void
    executeOffers(org.omg.CosTrading.Offer[] offers, org.omg.CORBA.ORB
orb)
    {


        String servername=null;
        pcount = offers.length;
      utils.arrayHolder thearray = null;
      System.out.println("Loading ints.....");
        try
        {
            thearray = new utils.arrayHolder(
                  clientprop.getProperty("input","testfile"),

Integer.parseInt(clientprop.getProperty("maxsize","0")));
      psize = thearray.getLength();
            System.out.println("Done loading ints: " +
thearray.getLength());
        }
        catch(Exception e) {e.printStackTrace();System.exit(9);}



      //
      //create clientcallback object for sorter to return result set
    //
      try
      {
```

```
            org.omg.PortableServer.POA root =
                  org.omg.PortableServer.POAHelper.narrow(
            orb.resolve_initial_references("RootPOA"));
            root.the_POAManager().activate();
            worker.ClientCallBack_impl ccbImpl =
                        new worker.ClientCallBack_impl(root,
offers.length);
            worker.ClientCallBack ccb = ccbImpl._this(orb);
      System.out.println("Created ClientCallBack Object...") ;

            //
            // determine the length of each int sub-set
            //
            int subsetsize = thearray.getLength()/offers.length;


        for(int i = 0 , startIndex = 0; i < offers.length ; i++)
        {
            try
            {

                worker.Sorter sorter = null;
                try
                {
                    sorter =
worker.SorterHelper.narrow(offers[i].reference);
                }
                catch(org.omg.CORBA.BAD_PARAM ex)
                {
                    System.out.println("  Object is not a Sorter -
skipping");
                    continue;
                }

                //
                // Extract the properties of this offer
                //

                org.omg.CosTrading.Property prop =
offers[i].properties[0];

                if(prop.name.equals("servername"))
                   servername = prop.value.extract_string();

                System.out.println();
                System.out.println("Executing Offer from:" +
servername);
            //
            //get sub-set of ints
            //
            if ((offers.length - i) == 1)
                  subsetsize = thearray.getLength() - startIndex;
            worker.IntIterator clientItr =
                  (new worker.IntIterator_impl(root,
                  thearray.getArray(startIndex,
subsetsize))).this(orb);
            startIndex +=subsetsize;
```

```
            sorter.sortInts(subsetsize, clientItr, ccb);
            }
            catch(org.omg.CORBA.SystemException ex)
            {
                System.out.println("  Unable to contact " +
servername);
                        ex.printStackTrace();
                        System.exit(2);

            }
        }
            //
            //wait until the call back recieves sorted arrays from all
servers.
            //
            while(!ccbImpl.isDone()){}
       }
       catch(Exception e) {e.printStackTrace();System.exit(9);}
    }

    static org.omg.CosTrading.Policy[]
    getPolicies(org.omg.CORBA.ORB orb, String file)
    {
        org.omg.CosTrading.Policy[] result = null;

        java.io.File f = new java.io.File(file);

        if(f.exists())
        {
            System.out.println("Reading policies from " + file +
"...");

            try
            {
                //
                // Add all policies to vector
                //
                java.util.Vector vec = new java.util.Vector();

                java.io.FileReader fr = new java.io.FileReader(f);
                java.io.BufferedReader reader = new
java.io.BufferedReader(fr);

                String line;
                while((line = reader.readLine()) != null)
                {
                    String trimline = line.trim();
                    if(trimline.length() == 0 ||
trimline.startsWith("#"))
                        continue;

                    int pos = trimline.indexOf('=');
                    if(pos < 0 || trimline.length() == pos + 1)
                    {
                        System.err.println("Error: Invalid line '" +
line +
                                           "' in file " + f);
```

```
                    System.exit(1);
                }

                String name = trimline.substring(0, pos);
                String value = trimline.substring(pos + 1);

                org.omg.CosTrading.Policy policy = null;

                if(name.equals("exact_type_match"))
                    policy = getBooleanPolicy(orb, name, value);
                else if(name.equals("use_modifiable_properties"))
                    policy = getBooleanPolicy(orb, name, value);
                else if(name.equals("use_dynamic_properties"))
                    policy = getBooleanPolicy(orb, name, value);
                else if(name.equals("use_proxy_offers"))
                    policy = getBooleanPolicy(orb, name, value);
                else if(name.equals("search_card"))
                    policy = getULongPolicy(orb, name, value);
                else if(name.equals("match_card"))
                    policy = getULongPolicy(orb, name, value);
                else if(name.equals("return_card"))
                    policy = getULongPolicy(orb, name, value);
                else if(name.equals("link_follow_rule"))
                    policy = getFollowOptionPolicy(orb, name,
value);
                else if(name.equals("hop_count"))
                    policy = getULongPolicy(orb, name, value);
                else
                {
                    System.err.println("Error: Unknown policy '" +
name +
                                       "' in file " + f);
                    System.exit(1);
                }

                vec.addElement(policy);
            }

            fr.close();

            result = new org.omg.CosTrading.Policy[vec.size()];
            vec.copyInto(result);
        }
        catch(java.io.FileNotFoundException ex)
        {
            System.err.println("Error: Unable to open file " + f);
            System.exit(1);
        }
        catch(java.io.IOException ex)
        {
            System.err.println("Error: Unable to read file " + f);
            System.exit(1);
        }
    }
    else
        result = new org.omg.CosTrading.Policy[0];
```

```
        return result;
    }

    static org.omg.CosTrading.Policy
    getBooleanPolicy(org.omg.CORBA.ORB orb, String name, String value)
    {
        org.omg.CosTrading.Policy result = null;

        if(value.equalsIgnoreCase("true"))
        {
            org.omg.CORBA.Any any = orb.create_any();
            any.insert_boolean(true);
            result = new org.omg.CosTrading.Policy(name, any);
        }
        else if(value.equalsIgnoreCase("false"))
        {
            org.omg.CORBA.Any any = orb.create_any();
            any.insert_boolean(false);
            result = new org.omg.CosTrading.Policy(name, any);
        }
        else
        {
            System.err.println("Error: Invalid boolean value for policy
'" +
                               name + "'");
            System.exit(1);
        }

        return result;
    }

    static org.omg.CosTrading.Policy
    getULongPolicy(org.omg.CORBA.ORB orb, String name, String value)
    {
        org.omg.CosTrading.Policy result = null;

        try
        {
            int n = Integer.valueOf(value).intValue();
            org.omg.CORBA.Any any = orb.create_any();
            any.insert_ulong(n);
            result = new org.omg.CosTrading.Policy(name, any);
        }
        catch(NumberFormatException ex)
        {
            System.err.println("Error: Invalid numeric value for policy
'" +
                               name + "'");
            System.exit(1);
        }

        return result;
    }

    static org.omg.CosTrading.Policy
    getFollowOptionPolicy(org.omg.CORBA.ORB orb, String name, String
value)
```

```
    {
        org.omg.CosTrading.Policy result = null;

        if(value.equals("local_only"))
        {
            org.omg.CORBA.Any any = orb.create_any();
            org.omg.CosTrading.FollowOptionHelper.insert(any,
                org.omg.CosTrading.FollowOption.local_only);
            result = new org.omg.CosTrading.Policy(name, any);
        }
        else if(value.equals("if_no_local"))
        {
            org.omg.CORBA.Any any = orb.create_any();
            org.omg.CosTrading.FollowOptionHelper.insert(any,
                org.omg.CosTrading.FollowOption.if_no_local);
            result = new org.omg.CosTrading.Policy(name, any);
        }
        else if(value.equals("always"))
        {
            org.omg.CORBA.Any any = orb.create_any();
            org.omg.CosTrading.FollowOptionHelper.insert(any,
                org.omg.CosTrading.FollowOption.always);
            result = new org.omg.CosTrading.Policy(name, any);
        }
        else
        {
            System.err.println("Error: Invalid FollowOption value for "
+
                               "policy '" + name + "'");
            System.exit(1);
        }

        return result;
    }

    static void
    usage()
    {
        System.err.println();
        System.err.println("Usage:  java client.TraderClient [-h] ");
            System.err.println("refer to client.properties file to set
" +
                           "properties.");
        System.err.println();
        System.err.println("Options:");
        System.err.println();
        System.err.println("  -h    Show this help message");
        System.exit(1);
    }
}
*****************************************************
*             CORBA Trader Worker listings          *
*****************************************************
package worker;

public class SortServer
{
```

```
    private static final String offerIdFile_ = "offers.dat";

    // ------------------------------------------------------------------
--
    // main() and supporting methods
    // ------------------------------------------------------------------
--

    private static int
    run(org.omg.CORBA.ORB orb, String args[])
        throws org.omg.CORBA.UserException
    {
        org.omg.PortableServer.POA root =
            org.omg.PortableServer.POAHelper.narrow(
                orb.resolve_initial_references("RootPOA"));

        boolean showHelp = false, withdrawOffers = false;
            String servername = null;

        //
        // Check arguments
        //
        for(int i = 0 ; i < args.length ; i++)
        {
            if(args[i].equals("-w"))
                withdrawOffers = true;
            else if(args[i].equals("-h"))
                showHelp = true;
                  else if(servername == null) // assume it is the
server name
                servername = args[i];

            else // unknown argument
                usage();
        }

            if (servername == null)
                  servername = "testserver";
            if (showHelp)
                  usage();
        //
        // Start server event loop - this is necessary in case
        // the trader calls us back (i.e., _get_interface)
        //
        root.the_POAManager().activate();

        //
        // Connect to Trading Service
        //
        org.omg.CosTrading.Lookup trader =
utils.SortUtils.connect(orb);

        if(trader == null)
            usage();

        //
```

```
        // Quit now if -w option was specified
        //
        if(withdrawOffers)
      {
            utils.SortUtils.cleanOffers(trader, offerIdFile_);
            return 0;
      }

        //
        // Instantiate our sorter implementation
        //
        Sorter_impl SorterImpl = new Sorter_impl(root, servername);
        worker.Sorter sorter = SorterImpl._this(orb);

        //
        // Install the 'sortints' service type and export our service
        // offers
        //
        utils.SortUtils.installServiceType(orb, trader);
        exportOffers(orb, trader, servername, sorter);

        System.out.println("Server for '" + servername + "' is
ready...");
        orb.run();

        return 0;
    }

    public static void
    main(String args[])
    {
        java.util.Properties props = System.getProperties();
        props.put("org.omg.CORBA.ORBClass", "com.ooc.CORBA.ORB");
        props.put("org.omg.CORBA.ORBSingletonClass",
                  "com.ooc.CORBA.ORBSingleton");

        int status = 0;
        org.omg.CORBA.ORB orb = null;

        try
        {
          args = com.ooc.OBCORBA.ORB_impl.ParseArgs(args, props, null);
            orb = org.omg.CORBA.ORB.init(args, props);
            status = run(orb, args);
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
            status = 1;
        }

        if(orb != null)
        {
            try
            {
                ((com.ooc.CORBA.ORB)orb).destroy();
            }
```

```
            catch(Exception ex)
            {
                ex.printStackTrace();
                status = 1;
            }
        }

        System.exit(status);
    }


    static void
    exportOffers(org.omg.CORBA.ORB orb,
                 org.omg.CosTrading.Lookup trader,
                 String servername,
                 worker.Sorter sorter)
    {
        String result = null;

        System.out.println("Exporting service offers...");

        try
        {
            //
            // Write the ID of each offer we export to a file
            //
            java.io.FileWriter fw = new
java.io.FileWriter(offerIdFile_);
            java.io.PrintWriter writer = new java.io.PrintWriter(fw);

            org.omg.CosTrading.Register reg = trader.register_if();

            if(reg == null)
            {
                System.err.println("Error: Trader does not support " +
                                   "the Register interface");
                System.exit(1);
            }

                org.omg.CosTrading.Property[] props =
                    new org.omg.CosTrading.Property[1];

                props[0] = new org.omg.CosTrading.Property();
                props[0].name = "servername";
                props[0].value = orb.create_any();
                props[0].value.insert_string(servername);

                String id = reg.export(sorter,

utils.SortUtils.getServiceType(),
                                       props);
                writer.println(id);

            writer.flush();
            fw.close();
        }
        catch(java.io.IOException ex)
```

```
        {
            System.err.println("Error: Unable to write file " +
offerIdFile_);
            System.exit(1);
        }
        catch(org.omg.CosTrading.RegisterPackage.InvalidObjectRef e)
        {
            System.err.println("Error: Invalid object reference");
            System.exit(1);
        }
        catch(org.omg.CosTrading.IllegalServiceType e)
        {
            System.err.println("Error: Illegal service type '" + e.type
+
                               "'");
            System.exit(1);
        }
        catch(org.omg.CosTrading.UnknownServiceType e)
        {
            System.err.println("Error: Unknown service type '" + e.type
+
                               "'");
            System.exit(1);
        }
        catch(org.omg.CosTrading.RegisterPackage.InterfaceTypeMismatch
e)
        {
            System.err.println("Error: Interface type mismatch for '" +
                               e.type + "'");
            System.exit(1);
        }
        catch(org.omg.CosTrading.IllegalPropertyName e)
        {
            System.err.println("Error: Illegal property name '" +
                               e.name + "'");
            System.exit(1);
        }
        catch(org.omg.CosTrading.PropertyTypeMismatch e)
        {
            System.err.println("Error: Property type mismatch for '" +
                               e.prop.name + "'");
            System.exit(1);
        }
        catch(org.omg.CosTrading.ReadonlyDynamicProperty e)
        {
            System.err.println("Error: Readonly dynamic property '" +
                               e.name + "'");
            System.exit(1);
        }
        catch(org.omg.CosTrading.MissingMandatoryProperty e)
        {
            System.err.println("Error: Missing mandatory property '" +
                               e.name + "'");
            System.exit(1);
        }
        catch(org.omg.CosTrading.DuplicatePropertyName e)
        {
```

```
            System.err.println("Error: Duplicate property '" + e.name +
"'");
            System.exit(1);
        }
        catch(org.omg.CORBA.SystemException e)
        {
            e.printStackTrace();
            System.exit(1);
        }
    }

    static void
    usage()
    {
        System.err.println();
        System.err.println("Usage:  java worker.SortServer [-h] [-w] "
+
                           "server-name");
        System.err.println();
        System.err.println("Options:");
        System.err.println();
        System.err.println("  -h   Show this help message");
        System.err.println("  -w   Withdraws existing offers and
exits");
        System.exit(1);
    }
}
//
**********************************************************************
//
// Generated by the ORBacus IDL to Java Translator
//
// Copyright (c) 2002
// IONA Technologies, Inc.
// Waltham, MA, USA
//
// All Rights Reserved
//
//
**********************************************************************

// Version: 4.1.2

package worker;

//
// IDL:worker/IntIterator:1.0
//
/***/

public class IntIterator_impl extends IntIteratorPOA
{
    private org.omg.PortableServer.POA poa_;
      private int[] thearray;
      private int nextindex;

    public
```

```
IntIterator_impl(org.omg.PortableServer.POA poa, int[] fullset)
{
    poa_ = poa;
        thearray = fullset;
        nextindex = 0;
}

public org.omg.PortableServer.POA
_default_POA()
{
    if(poa_ != null)
        return poa_;
    else
        return super._default_POA();
}

//
// IDL:worker/IntIterator/getNextSet:1.0
//
public boolean
getNextSet(int setsize,
           IntSetHolder subset)
    throws EndOfSetReached
{
        if (nextindex >= thearray.length)
              throw new EndOfSetReached("Message from Iterator");
    boolean _r = true;
        int newarraysize = setsize;

        if ((thearray.length - nextindex) < setsize)
        {
              _r = false;
              newarraysize = thearray.length - nextindex;
        }

    subset.value = new int[newarraysize];
        int i = 0;
        for (; i < newarraysize; i++)
        {
              subset.value[i] = thearray[nextindex + i];
        }
        nextindex += i;
    return _r;
}

//
// IDL:worker/IntIterator/destroy:1.0
//
public void
destroy()
{
        try
        {
              _poa().deactivate_object(_object_id());
        }
        catch(Exception e)
        { e.printStackTrace();}
```

```
    }
}
//
**********************************************************************
//
// Generated by the ORBacus IDL to Java Translator
//
// Copyright (c) 2002
// IONA Technologies, Inc.
// Waltham, MA, USA
//
// All Rights Reserved
//
//
**********************************************************************

// Version: 4.1.2

package worker;
import utils.*;

//
// IDL:worker/ClientCallBack:1.0
//
/***/

public class ClientCallBack_impl extends ClientCallBackPOA
{
    private org.omg.PortableServer.POA poa_;
      private int[][] resultsets;
      private int to_arrive_setcount;
      private int setcount;
      private boolean done = false;

    public
    ClientCallBack_impl(org.omg.PortableServer.POA poa, int cbcount)
    {
        poa_ = poa;
            resultsets=new int[cbcount][];
            to_arrive_setcount=cbcount;
            setcount = cbcount;
    }

    public org.omg.PortableServer.POA
    _default_POA()
    {
        if(poa_ != null)
            return poa_;
        else
            return super._default_POA();
    }

    //
    // IDL:worker/ClientCallBack/getSortedInts:1.0
    //
    public void
    getSortedInts(int intcount,
```

```
                    IntIterator srvrItr)
    {
            resultsets[to_arrive_setcount -1] =
                    SortUtils.buildArray(intcount, srvrItr);
            to_arrive_setcount--;
            if (to_arrive_setcount == 0)
            {
                    SortUtils.printInts(resultsets, setcount);
                    done = true;
            }

    }
      public boolean isDone()
      {
                    return done;
      }
}
//
**********************************************************************
//
// Generated by the ORBacus IDL to Java Translator
//
// Copyright (c) 2002
// IONA Technologies, Inc.
// Waltham, MA, USA
//
// All Rights Reserved
//
//
**********************************************************************

// Version: 4.1.2

package worker;
import utils.*;

//
// IDL:worker/Sorter:1.0
//
/***/

public class Sorter_impl extends SorterPOA
{
    private org.omg.PortableServer.POA poa_;
      private String servername;

    public
    Sorter_impl(org.omg.PortableServer.POA poa, String name)
    {
        poa_ = poa;
            servername = name;
    }

    public org.omg.PortableServer.POA
    _default_POA()
    {
        if(poa_ != null)
```

```
            return poa_;
        else
            return super._default_POA();
    }

    //
    // IDL:worker/Sorter/sortInts:1.0
    //
    public void
    sortInts(int intcount,
             IntIterator clientItr,
             ClientCallBack ccb)
    {
            System.out.println(servername+ " :begin to sort the array
....");
            int[] result =
            SortUtils.sort(SortUtils.buildArray(intcount, clientItr));
            System.out.println(servername+
                  " :Finished sorting the array of size " +
                  result.length);

//SortUtils.printArray(result);
            try
            {
                  IntIterator_impl serverItrImpl = new
IntIterator_impl(poa_, result);
//the following 2 lines are standard way of creating a incarnated corba
object.
                  org.omg.CORBA.Object obj =
poa_.servant_to_reference(serverItrImpl);
                  IntIterator srvrItr = IntIteratorHelper.narrow(obj);
// the following is ORBACUS way of creating an incarnated object
//                IntIterator srvrItr = serverItrImpl._this();
                  ccb.getSortedInts(result.length, srvrItr);
            }
            catch(Exception e) { e.printStackTrace();}

    }
}
*****************************************************
*            CORBA Trader Utilities listings        *
*****************************************************
package utils;

import java.io.*;
import java.util.*;
public class arrayHolder
{
      public int[] thearray= null;

      public arrayHolder(String infile, int maxsize)
            throws Exception
      {

            int arraysize=0;
            if (maxsize == 0)
                  maxsize = 160000;
```

```
            int[] intarray = new int[maxsize];

            BufferedReader br = new BufferedReader(new
FileReader(infile));

            String line = null;
            StringTokenizer stk = null;
            while((line=br.readLine()) != null)
            {
                  stk = new StringTokenizer(line);
                  while(stk.hasMoreTokens())
                  intarray[arraysize++] =
Integer.parseInt(stk.nextToken());
            }

            thearray = new int[arraysize];
            for(; arraysize > 0 ; arraysize--)
            thearray[arraysize-1] = intarray[arraysize-1];

      }

      public int getLength() { return thearray.length;}

      public int[] getArray(int start, int size)
      {
            int[] retarray = new int[size];
            for (int i = 0; i < size; i++)
            {
                  retarray[i] = thearray[i+start];
            }
            return retarray;

      }



}
package utils;
import worker.*;
// Things are just too ugly without this
import org.omg.CosTradingRepos.ServiceTypeRepositoryPackage.*;

public abstract class SortUtils
{
      private static final String serviceType_ = "SortInts";

      public static int[] sort(int a[])
      {
            for (int i = a.length; --i>=0;)
            {
                  boolean flipped = false;
                  for (int j = 0; j<i; j++)
                  {
                        if (a[j] >  a[j+1])
                        {
                              int T = a[j];
                              a[j] = a[j+1];
```

```
                        a[j+1] = T;
                        flipped = true;
                    }
                }
                if (!flipped) { return a; }
            }
            return a;
        }
        public static int[] buildArray(int maxsize, IntIterator itr)
        {
            int[] result = new int[maxsize];
            int nextindex = 0;
            IntSetHolder intsetholder = new IntSetHolder();
            int fetchcount = 65538;
            try
            {
                boolean morerows=itr.getNextSet(fetchcount,
intsetholder);
                do
                {
                    int i = 0;
                    for (;i < intsetholder.value.length;i++)
                    {
                        result[nextindex+i] =
intsetholder.value[i];
//System.out.println("From buildarray: " + result[nextindex+i]);
                    }
                    nextindex += i;
                    morerows=itr.getNextSet(fetchcount,
intsetholder);
                }
                while(morerows);
                itr.destroy();
            }
            catch(EndOfSetReached eosr)
            { itr.destroy();
            }
            catch(Exception e)
            {e.printStackTrace();}
            finally{ return result;}

        }

        public static String
        getServiceType()
        {
            return serviceType_;
        }

        //
        // Connect to Trading Service by resolving the 'TradingService'
        // initial reference.
        //
        public static org.omg.CosTrading.Lookup
        connect(org.omg.CORBA.ORB orb)
        {
            org.omg.CosTrading.Lookup result = null;
```

```
            //
            // Resolve the 'TradingService' initial reference
            //

            try
            {
                org.omg.CORBA.Object obj =
                    orb.resolve_initial_references("TradingService");
                result = org.omg.CosTrading.LookupHelper.narrow(obj);
            }
            catch(org.omg.CORBA.ORBPackage.InvalidName ex)
            {
                System.err.println("Error: Unable to resolve initial " +
                                   "reference 'TradingService'");
            }

            return result;
        }

        public static void
        installServiceType(org.omg.CORBA.ORB orb,
org.omg.CosTrading.Lookup trader)
        {
            boolean typeExists = false;
            org.omg.CosTradingRepos.ServiceTypeRepository repos = null;

            //
            // Verify that the type "SortInts" already exists in the
            // service type repository; if not, we need to install it
            //
            try
            {
                org.omg.CORBA.Object obj = trader.type_repos();
                repos =
org.omg.CosTradingRepos.ServiceTypeRepositoryHelper.
                        narrow(obj);
                repos.describe_type(serviceType_);
                typeExists = true;
            }
            catch(org.omg.CosTrading.IllegalServiceType ex)
            {
                // ignore
            }
            catch(org.omg.CosTrading.UnknownServiceType ex)
            {
                // ignore
            }
            catch(org.omg.CORBA.SystemException e)
            {
                System.err.println("System error occurred");
                e.printStackTrace();
                System.exit(1);
            }

            if(!typeExists)
            {
```

```
                System.out.println("Installing service type '" +
                                   serviceType_ + "'...");

                try
                {
                    String[] superTypes = new String[0];

                    //
                    // Define the properties for this service type
                    //

                          PropStruct[] props = new PropStruct[1];

                    props[0] = new PropStruct();
                    props[0].name = "servername";
                    props[0].value_type = orb.create_string_tc(0);
                    props[0].mode = PropertyMode.PROP_NORMAL;


                    repos.add_type(serviceType_,
                                   worker.SorterHelper.id(),
                                   props, superTypes);
                      }
                catch(org.omg.CosTrading.IllegalServiceType ex)
                {
                    System.err.println("Illegal service type: " +
ex.type);
                    System.exit(1);
                }
                catch(org.omg.CosTrading.UnknownServiceType ex)
                {
                    System.err.println("Unknown service type: " +
ex.type);
                    System.exit(1);
                }

catch(org.omg.CosTradingRepos.ServiceTypeRepositoryPackage.
                    ServiceTypeExists ex)
                {
                    System.err.println("Service type '" + ex.name +
                                       "' already exists");
                    System.exit(1);
                }

catch(org.omg.CosTradingRepos.ServiceTypeRepositoryPackage.
                    InterfaceTypeMismatch ex)
                {
                    System.err.println("Interface of service type '" +
                                       ex.derived_service +
                                       "' does not match super type '" +
                                       ex.base_service + "'");
                    System.exit(1);
                }
                catch(org.omg.CosTrading.IllegalPropertyName ex)
                {
                    System.err.println("Illegal property name '" +
ex.name + "'");
```

```
                System.exit(1);
            }
            catch(org.omg.CosTrading.DuplicatePropertyName ex)
            {
                System.err.println("Duplicate property name '" +
                                   ex.name + "'");
                System.exit(1);
            }

catch(org.omg.CosTradingRepos.ServiceTypeRepositoryPackage.
                ValueTypeRedefinition ex)
            {
                System.err.println("Property type for '" +
                                   ex.definition_2.name +
                                   "' is not compatible with the
definition " +
                                        "in service type '" +
ex.type_1 + "'");
            }

catch(org.omg.CosTradingRepos.ServiceTypeRepositoryPackage.
                DuplicateServiceTypeName ex)
            {
                System.err.println("Duplicate super type: " +
ex.name);
                System.exit(1);
            }
            catch(org.omg.CORBA.SystemException e)
            {
                System.err.println("System error occurred");
                e.printStackTrace();
                System.exit(1);
            }
        }
    }

    public static void
    cleanOffers(org.omg.CosTrading.Lookup trader, String fileName)
    {
        //
        // If the offer ID file exists, then read the offer IDs from
the
        // file and withdraw them
        //

        java.io.File f = new java.io.File(fileName);

        if(f.exists())
        {
            String[] offers = readOffers(f);

            System.out.println("Removing old offers...");

            try
            {
                org.omg.CosTrading.Register reg =
trader.register_if();
```

```
                    if(reg == null)
                    {
                        System.err.println("Error:  Trader does not
support " +
                                           "the Register interface");
                        System.exit(1);
                    }

                    for(int i = 0 ; i < offers.length ; i++)
                    {
                        try
                        {
                            reg.withdraw(offers[i]);
                        }
                        catch(org.omg.CORBA.UserException ex)
                        {
                            // ignore
                        }
                    }
                }
                catch(org.omg.CORBA.SystemException e)
                {
                    System.err.println("System error occurred");
                    e.printStackTrace();
                    System.exit(1);
                }
            }
        }

        private static String[]
        readOffers(java.io.File f)
        {
            String[] result = null;

            java.util.Vector vec = new java.util.Vector();

            try
            {
                java.io.FileReader fr = new java.io.FileReader(f);
                java.io.BufferedReader reader = new
java.io.BufferedReader(fr);

                //
                // Read all the offer IDs in the file, one ID per line
                //

                String id;
                while((id = reader.readLine()) != null)
                    vec.addElement(id);

                fr.close();

                result = new String[vec.size()];
                vec.copyInto(result);

                //
```

```
            // Get rid of the file
            //
            f.delete();
        }
        catch(java.io.FileNotFoundException ex)
        {
            System.err.println("Error:  Unable to open file " + f);
            System.exit(1);
        }
        catch(java.io.IOException ex)
        {
            System.err.println("Error:  Unable to read file " + f);
            System.exit(1);
        }
        catch(org.omg.CORBA.SystemException e)
        {
            System.err.println("System error occurred");
            e.printStackTrace();
            System.exit(1);
        }

        return result;
    }
    public static void printInts(int[][] sortedarray, int arraycount)
    {
          int arrayptr = Integer.MIN_VALUE;
          int[] ptr = new int[arraycount];
          boolean numfound = true;
          while(numfound)
          {
                int lownum = Integer.MAX_VALUE, lowarray =
Integer.MIN_VALUE;
                numfound = false;
                for (int i = 0; i < arraycount; i++)
                {
                      if (sortedarray[i] != null)
                      {     arrayptr = ptr[i];
                            if (sortedarray[i][arrayptr] <= lownum)
                            {
                                  lownum = sortedarray[i][arrayptr];
                                  lowarray = i;
                                  numfound = true;
                            }
                      }
                }
                if (!numfound) continue;
                ptr[lowarray]++;
                if (ptr[lowarray] >= sortedarray[lowarray].length)
                {
                      sortedarray[lowarray] = null;
                }
                System.out.println(lownum);
          }
    }
    public static void printArray(int[] inarray)
    {
          for (int i = 0; i < inarray.length;i++)
```

```
            {
                System.out.println(i+":"+inarray[i]);
            }
        }

}
```

APPENDIX C

WebServices Code Listings

```
*****************************************************
*            SOAP Client listings                   *
*****************************************************
import java.util.*;
import java.io.*;
public class multipleTester
{
      static long starttime=0, endtime=0;
    public static void main(String [] args) throws Exception
    {
            Properties prop = new Properties();
            FileInputStream pFIS = new
FileInputStream("client.properties");
            prop.load(pFIS);
            pFIS.close();

            System.out.println("Loading ints.....");
            arrayHolder thearray = new arrayHolder(
                        prop.getProperty("input","testfile"),

      Integer.parseInt(prop.getProperty("maxsize","0")));
            System.out.println("Done loading ints: " +
thearray.getLength());

            int servercount =
Integer.parseInt(prop.getProperty("webServiceCount","1"));
            int startPtr = 0;
            int incr = thearray.getLength()/servercount;

            threadedTester[] testers = new threadedTester[servercount];
            int start = 0;
            for (int i=0,j=1; i < servercount; i++,j++)
            {
                  if ((servercount - i) == 1)
                        incr = thearray.getLength() - start;

                  testers[i] = new threadedTester(
                              thearray.getArray(start, incr),
                              prop.getProperty("url"+j));
                  start +=incr;
            }
            starttime = System.currentTimeMillis();
            Thread[] t = new Thread[servercount];
            for (int i = 0;i < servercount; i++)
            {
                  System.out.println("Starting thread " + i);
                  t[i] = new Thread(testers[i]);
                  t[i].start();
            }
            for (int i = 0; i < servercount; i++)
            {
```

```
                    t[i].join();
                    System.out.println("Therad " + i + " ended.");
              }

              int[][] sortedarray = new int[servercount][];
              for (int i = 0; i < servercount; i++)
              {
                    sortedarray[i] = testers[i].getSortedArray();
              }
        //    printInts(sortedarray, servercount);
              endtime = System.currentTimeMillis();
              System.err.println("Time to sort and print " +
thearray.getLength() + " using " +
                    servercount + " servers: " + (endtime - starttime));
        }
        public static void printInts(int[][] sortedarray, int arraycount)
        {
              int arrayptr = Integer.MIN_VALUE;
              int[] ptr = new int[arraycount];
              boolean numfound = true;
              while(numfound)
              {
                    int lownum = Integer.MAX_VALUE, lowarray =
Integer.MIN_VALUE;
                    numfound = false;
                    for (int i = 0; i < arraycount; i++)
                    {
                          if (sortedarray[i] != null)
                          {     arrayptr = ptr[i];
                                if (sortedarray[i][arrayptr] <= lownum)
                                {
                                      lownum = sortedarray[i][arrayptr];
                                      lowarray = i;
                                      numfound = true;
                                }
                          }
                    }
                    if (!numfound) continue;
                    ptr[lowarray]++;
                    if (ptr[lowarray] >= sortedarray[lowarray].length)
                    {
                          sortedarray[lowarray] = null;
                    }
                    System.out.println(lownum);
              }
        }

}

/**
 * SortIntSrvc.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis Wsdl2java emitter.
 */

public interface SortIntSrvc extends java.rmi.Remote {
```

```
    public int[] sortInts(int[] a) throws java.rmi.RemoteException;
}
/**
 * SortIntSrvcService.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis Wsdl2java emitter.
 */

public interface SortIntSrvcService extends javax.xml.rpc.Service {
    public String getSortIntSrvcAddress();

    public SortIntSrvc getSortIntSrvc() throws
javax.xml.rpc.ServiceException;

    public SortIntSrvc getSortIntSrvc(String address) throws
javax.xml.rpc.ServiceException;

    public SortIntSrvc getSortIntSrvc(java.net.URL portAddress) throws
javax.xml.rpc.ServiceException;
}
/**
 * SortIntSrvcServiceLocator.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis Wsdl2java emitter.
 */

public class SortIntSrvcServiceLocator extends
org.apache.axis.client.Service implements SortIntSrvcService {

    // Use to get a proxy class for SortIntSrvc
    private final java.lang.String SortIntSrvc_address =
"http://localhost:8080/axis/services/SortIntSrvc";

    public String getSortIntSrvcAddress() {
        return SortIntSrvc_address;
    }

    public SortIntSrvc getSortIntSrvc() throws
javax.xml.rpc.ServiceException {
       java.net.URL endpoint;
        try {
            endpoint = new java.net.URL(SortIntSrvc_address);
        }
        catch (java.net.MalformedURLException e) {
            return null; // unlikely as URL was validated in WSDL2Java
        }
        return getSortIntSrvc(endpoint);
    }

    public SortIntSrvc getSortIntSrvc(String address) throws
javax.xml.rpc.ServiceException {
       java.net.URL endpoint;
        try {
            endpoint = new java.net.URL(address);
        }
```

```
        catch (java.net.MalformedURLException e) {
            return null; // unlikely as URL was validated in WSDL2Java
        }
        return getSortIntSrvc(endpoint);
    }

    public SortIntSrvc getSortIntSrvc(java.net.URL portAddress) throws
javax.xml.rpc.ServiceException {
        try {
            return new SortIntSrvcSoapBindingStub(portAddress, this);
        }
        catch (org.apache.axis.AxisFault e) {
            return null; // ???
        }
    }

    /**
     * For the given interface, get the stub implementation.
     * If this service has no port for the given interface,
     * then ServiceException is thrown.
     */
    public java.rmi.Remote getPort(Class serviceEndpointInterface)
throws javax.xml.rpc.ServiceException {
        try {
            if
(SortIntSrvc.class.isAssignableFrom(serviceEndpointInterface)) {
                return new SortIntSrvcSoapBindingStub(new
java.net.URL(SortIntSrvc_address), this);
            }
        }
        catch (Throwable t) {
            throw new javax.xml.rpc.ServiceException(t);
        }
        throw new javax.xml.rpc.ServiceException("There is no stub
implementation for the interface:  " + (serviceEndpointInterface ==
null ? "null" : serviceEndpointInterface.getName()));
    }

}
/**
 * SortIntSrvcSoapBindingStub.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis WSDL2Java emitter.
 */

public class SortIntSrvcSoapBindingStub extends
org.apache.axis.client.Stub implements SortIntSrvc {
    private java.util.Vector cachedSerClasses = new java.util.Vector();
    private java.util.Vector cachedSerQNames = new java.util.Vector();
    private java.util.Vector cachedSerFactories = new
java.util.Vector();
    private java.util.Vector cachedDeserFactories = new
java.util.Vector();

    static org.apache.axis.description.OperationDesc [] _operations;
```

```
    static {
        _operations = new org.apache.axis.description.OperationDesc[1];
        org.apache.axis.description.OperationDesc oper;
        oper = new org.apache.axis.description.OperationDesc();
        oper.setName("sortInts");
        oper.addParameter(new javax.xml.namespace.QName("", "in0"), new
javax.xml.namespace.QName("http://DefaultNamespace",
"ArrayOf_xsd_int"), int[].class,
org.apache.axis.description.ParameterDesc.IN, false, false);
        oper.setReturnType(new
javax.xml.namespace.QName("http://DefaultNamespace",
"ArrayOf_xsd_int"));
        oper.setReturnClass(int[].class);
        oper.setReturnQName(new javax.xml.namespace.QName("",
"sortIntsReturn"));
        oper.setStyle(org.apache.axis.enum.Style.RPC);
        oper.setUse(org.apache.axis.enum.Use.ENCODED);
        _operations[0] = oper;

    }

    public SortIntSrvcSoapBindingStub() throws
org.apache.axis.AxisFault {
         this(null);
    }

    public SortIntSrvcSoapBindingStub(java.net.URL endpointURL,
javax.xml.rpc.Service service) throws org.apache.axis.AxisFault {
         this(service);
         super.cachedEndpoint = endpointURL;
    }

    public SortIntSrvcSoapBindingStub(javax.xml.rpc.Service service)
throws org.apache.axis.AxisFault {
        if (service == null) {
            super.service = new org.apache.axis.client.Service();
        } else {
            super.service = service;
        }
            java.lang.Class cls;
            javax.xml.namespace.QName qName;
            java.lang.Class beansf =
org.apache.axis.encoding.ser.BeanSerializerFactory.class;
            java.lang.Class beandf =
org.apache.axis.encoding.ser.BeanDeserializerFactory.class;
            java.lang.Class enumsf =
org.apache.axis.encoding.ser.EnumSerializerFactory.class;
            java.lang.Class enumdf =
org.apache.axis.encoding.ser.EnumDeserializerFactory.class;
            java.lang.Class arraysf =
org.apache.axis.encoding.ser.ArraySerializerFactory.class;
            java.lang.Class arraydf =
org.apache.axis.encoding.ser.ArrayDeserializerFactory.class;
            java.lang.Class simplesf =
org.apache.axis.encoding.ser.SimpleSerializerFactory.class;
            java.lang.Class simpledf =
org.apache.axis.encoding.ser.SimpleDeserializerFactory.class;
```

```
            qName = new
javax.xml.namespace.QName("http://DefaultNamespace",
"ArrayOf_xsd_int");
            cachedSerQNames.add(qName);
            cls = int[].class;
            cachedSerClasses.add(cls);
            cachedSerFactories.add(arraysf);
            cachedDeserFactories.add(arraydf);

    }

    private org.apache.axis.client.Call createCall() throws
java.rmi.RemoteException {
        try {
            org.apache.axis.client.Call _call =
                    (org.apache.axis.client.Call)
super.service.createCall();
            if (super.maintainSessionSet) {
                _call.setMaintainSession(super.maintainSession);
            }
            if (super.cachedUsername != null) {
                _call.setUsername(super.cachedUsername);
            }
            if (super.cachedPassword != null) {
                _call.setPassword(super.cachedPassword);
            }
            if (super.cachedEndpoint != null) {
                _call.setTargetEndpointAddress(super.cachedEndpoint);
            }
            if (super.cachedTimeout != null) {
                _call.setTimeout(super.cachedTimeout);
            }
            if (super.cachedPortName != null) {
                _call.setPortName(super.cachedPortName);
            }
            java.util.Enumeration keys = super.cachedProperties.keys();
            while (keys.hasMoreElements()) {
                java.lang.String key = (java.lang.String)
keys.nextElement();
                _call.setProperty(key,
super.cachedProperties.get(key));
            }
            // All the type mapping information is registered
            // when the first call is made.
            // The type mapping information is actually registered in
            // the TypeMappingRegistry of the service, which
            // is the reason why registration is only needed for the
first call.
            synchronized (this) {
                if (firstCall()) {
                    // must set encoding style before registering
serializers

_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANT
S);

_call.setEncodingStyle(org.apache.axis.Constants.URI_SOAP11_ENC);
```

```
                    for (int i = 0; i < cachedSerFactories.size(); ++i)
{
                        java.lang.Class cls = (java.lang.Class)
cachedSerClasses.get(i);
                        javax.xml.namespace.QName qName =
                                (javax.xml.namespace.QName)
cachedSerQNames.get(i);
                        java.lang.Class sf = (java.lang.Class)
                                cachedSerFactories.get(i);
                        java.lang.Class df = (java.lang.Class)
                                cachedDeserFactories.get(i);
                        _call.registerTypeMapping(cls, qName, sf, df,
false);
                    }
                }
            }
            return _call;
        }
        catch (java.lang.Throwable t) {
            throw new org.apache.axis.AxisFault("Failure trying to get
the Call object", t);
        }
    }

    public int[] sortInts(int[] in0) throws java.rmi.RemoteException {
        if (super.cachedEndpoint == null) {
            throw new org.apache.axis.NoEndPointException();
        }
        org.apache.axis.client.Call _call = createCall();
        _call.setOperation(_operations[0]);
        _call.setUseSOAPAction(true);
        _call.setSOAPActionURI("");

_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANT
S);
        _call.setOperationName(new
javax.xml.namespace.QName("http://DefaultNamespace", "sortInts"));

        setRequestHeaders(_call);
        setAttachments(_call);
        java.lang.Object _resp = _call.invoke(new java.lang.Object[]
{in0});

        if (_resp instanceof java.rmi.RemoteException) {
            throw (java.rmi.RemoteException)_resp;
        }
        else {
            getResponseHeaders(_call);
            extractAttachments(_call);
            try {
                return (int[]) _resp;
            } catch (java.lang.Exception _exception) {
                return (int[])
org.apache.axis.utils.JavaUtils.convert(_resp, int[].class);
            }
        }
    }
```

```
}
import java.io.*;
import java.util.*;
public class arrayHolder
{
      public int[] thearray= null;

      public arrayHolder(String infile, int maxsize)
            throws Exception
      {

            int arraysize=0;
            if (maxsize == 0)
                  maxsize = 160000;
            int[] intarray = new int[maxsize];

            BufferedReader br = new BufferedReader(new
FileReader(infile));

            String line = null;
            StringTokenizer stk = null;
            while((line=br.readLine()) != null)
            {
                  stk = new StringTokenizer(line);
                  while(stk.hasMoreTokens())
                  intarray[arraysize++] =
Integer.parseInt(stk.nextToken());
            }

            thearray = new int[arraysize];
            for(; arraysize > 0 ; arraysize--)
            thearray[arraysize-1] = intarray[arraysize-1];

      }

      public int getLength() { return thearray.length;}

      public int[] getArray(int start, int size)
      {
            int[] retarray = new int[size];
            for (int i = 0; i < size; i++)
            {
                  retarray[i] = thearray[i+start];
            }
            return retarray;

      }



}
*****************************************************
*            SOAP Service listings                  *
*****************************************************
public class SortIntSrvc
{
```

```
	public int[] sortInts(int a[])
	{
		for (int i = a.length; --i>=0;)
		{
			boolean flipped = false;
			for (int j = 0; j<i; j++)
			{
				if (a[j] >  a[j+1])
				{
					int T = a[j];
					a[j] = a[j+1];
					a[j+1] = T;
					flipped = true;
				}
			}
			if (!flipped) { return a; }
		}
		return a;
	}
}
*****************************************************
*            Service deploy and undeploy files      *
*****************************************************
<!-- Use this file to deploy some handlers/chains and services      -->
<!-- Two ways to do this:                                           -->
<!--   java org.apache.axis.client.AdminClient deploy.wsdd          -->
<!--      after the axis server is running                          -->
<!-- or                                                             -->
<!--   java org.apache.axis.utils.Admin client|server deploy.wsdd   -->
<!--      from the same directory that the Axis engine runs         -->

<deployment
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <!-- Services from SortIntSrvcService WSDL service -->

  <service name="SortIntSrvc" provider="java:RPC">
      <parameter name="className" value="SortIntSrvc"/>
      <operation name="sortInts" returnQName="return" >
        <parameter name="a" type="tns:ArrayOf_xsd_int"
xmlns:tns="http://DefaultNamespace"/>
      </operation>
      <parameter name="allowedMethods" value="sortInts"/>
      <parameter name="scope" value="Request"/>

      <typeMapping
        xmlns:ns="http://DefaultNamespace"
        qname="ns:ArrayOf_xsd_int"
        type="java:int[]"

serializer="org.apache.axis.encoding.ser.ArraySerializerFactory"

deserializer="org.apache.axis.encoding.ser.ArrayDeserializerFactory"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      />
  </service>
```

```
</deployment>
<undeployment xmlns="http://xml.apache.org/axis/wsdd/">
 <service name="SortIntSrvc"/>
</undeployment>
```

VITA

Suresh Sunku has a Bachelor of Engineering from Mysore University India, in Mechanical Engineering, 1983 and expects to receive a Master Of Science in Computer and Information Sciences from the University Of North Florida, April 2003. Dr. Roger Eggen of the University Of North Florida is serving as Suresh's project director. Suresh is currently employed as a Database Administrator at Merrill Lynch for the past 5 years and has been employed at the company for the past 12 years. Prior to that, Suresh had worked as applications development manager at Merrill Lynch. Overall, Suresh has over 18 years of experience in the field of Information Sciences.

Suresh has on-going interest in the field of Grid/On-Demand computing and is working with University Of North Florida professors to develop services for the grid community. Suresh has extensive experience in C++, and Java programming languages and also worked with commercial database products such as IBM's DB2 and Microsoft's SQL Server. Suresh is married and has 2 children of age 10 and 9 years.