Testing for Feasible Performance of a Distributed Manufacturing System: a Brief Experience Report

David Flater
National Institute of Standards and Technology
100 Bureau Drive, Stop 8260
Gaithersburg, MD 20899-8260
U.S.A.

ABSTRACT

NIST researchers conducted a series of informal performance tests to establish confidence that an existing manufacturing process control system could successfully be replaced by a Common Object Request Broker Architecture (CORBA)-based system. Relative to the application, it was found that performance was seriously impacted by platform configuration and operational issues, while the CORBA overhead as such was negligible.

Keywords: Benchmarking, CORBA, Distributed, Performance, Testing.

1. INTRODUCTION

In 1997, the author of this report performed *ad hoc* distributed object benchmarking in response to requests from industrial partners in the Advanced Process Control Framework Initiative [1], hereinafter referred to as "APC." While there was no attempt to define a general method for distributed computing performance testing or to repeat the tests in other contexts, the informal tests have continued to be of interest. We therefore humbly submit the following brief summary of the testing that was conducted in hopes that it may prove valuable to those working in this field.

2. SUMMARY OF TESTING

The type of software of interest to APC in 1997 was manufacturing process control software. The goal of the testing was to establish confidence that an existing control system could successfully be replaced by a Common Object Request Broker Architecture [2] (CORBA)-based system.

APC was first and foremost concerned with measuring those limits of CORBA that could be show-stoppers for migrating the APC system to CORBA. Our first test, therefore, was to find the limit on the number of connections that could be sustained by a CORBA server, irrespective of the amount of traffic on those connections. We subsequently tested the server by confronting it with concurrent requests from 100 clients and measuring the response times for those requests. With the exception of a solitary "robustness test," the remainder of the tests concentrated on response time for various simple data operations. We used data operations that were similar, but not identical to, the APC workload to establish that the performance of the system would meet APC's requirements and to learn how to structure data to optimize that performance.

Following is a summary of the tests that we performed. The source code to most of these tests is available under the demos/APC subdirectory of the Manufacturer's CORBA Interface Testing Toolkit [3] distribution found in ftp://ftp.cme.nist.gov/pub/mcitt/.

1. Find limit on number of connections, one client one server.

   The test client binds to the test server repeatedly, without releasing previous connections, until an exception is thrown.

2. Determine time to acknowledge 19K text string, collocated Object Request Broker (ORB), client, and server.

   The test server provides an operation to receive an argument declared "in string" and returns a short integer as acknowledgement. The test client binds to the test server and invokes the operation 200 times in a timed loop with a 19456 byte (19K) string as the argument. The client, server, and ORB all run on the same machine with the connection being made through the loopback IP address (127.0.0.1, a.k.a. "localhost").

3. Determine time to acknowledge 262K text string, collocated ORB, client, and server.

   This test is identical to test 2 except that the test data item is larger.

4. 100 clients retrieving 200 19K strings, collocated ORB, clients, and server.

   The test client starts 100 copies of itself and executes against a single server. Execution involves binding to the server, waiting for a particular time (to achieve approximate synchronization with the other clients), and then retrieving a 19K text string 200 times in a timed loop. Clients, server, and ORB are all running on the same machine with the connection being made through the loopback IP address. The 19K string retrieved by each client on each request is immediately freed.

5. Robustness test: normal server, pointer error in client.

   The test server implements an operation, checkWidget, that accepts a widget as an in-argument. checkWidget invokes an operation on the widget passed to it, then returns a result.

   The test client exercises checkWidget first with a valid widget reference, then with a reference to a non-widget – specifically, a pointer to the test server itself. The purpose is to determine whether the CORBA implementation isolates failures in a single component, or whether failures in one program can spread through CORBA to other programs.

6. Determine time to acknowledge 19K text string via local area network.

   This test is identical to test 2 except that the client and server are on different hosts instead of collocated. The purpose is to determine whether network latency is more or less significant than CORBA overhead for data of size 19K.

7. Retrieval times for different granularities, collocated ORB, client, and server.

   The test client finds the times required for the following operations with a collocated ORB, client, and server:

   - Retrieve and free a single 19K string.
   - Retrieve and free a 1K string 19 times.
   - Retrieve and delete a structure containing 19 1K strings.
   - Retrieve and free an empty string.

   The first three operations are different possible ways to package the same 19K of data – as one big string attribute, as 19 separate string attributes of smaller size, or as a structure containing 19 separate strings. The last operation gives some idea of the overhead involved in a retrieval operation.

8. Retrieval times for different granularities, non-collocated.

   This test is identical to test 7 except that the client and server are not collocated.

9. Access times for 10,000 1K first-class CORBA objects, collocated.

   The test server creates 10,000 CORBA objects, each of which allocates 1K for attribute storage (not counting overhead). The test client binds to the server and executes 10,000 iterations of a timed loop in which (1) an object is returned by a request on the server, (2) a simple request (returning boolean) is made of the returned object, and (3) the object is released. Objects are chosen using a uniform random distribution.

10. Determine time to retrieve APCRunData::RunDataSequence, collocated.

    The test client retrieves and deletes a very complex data structure from a collocated server 200 times in a timed loop. The data structure is based on proprietary data supplied by APC, so the test is not publicly available. The objective is see what effect the complexity and size of the real-world data structure have on retrieval time.

11. Determine time to retrieve APCRunData::RunDataSequence, non-collocated.

    This test is identical to test 10 except that the testing client is run on a remote machine. The network setup is identical to previous non-collocated tests.

12. 100 clients retrieving 200 19K strings, non-collocated.

    This test is the same as test 4 except that the clients are run on a different machine than the server.

The previously mentioned Manufacturer's CORBA Interface Testing Toolkit (MCITT) enabled us to run the same test on multiple platforms through the use of code generation techniques. The tester would select a "binding" and the toolkit would pull code templates from that binding to generate test programs (clients and servers) that would run on that particular platform.

One of the tests suggested by APC was "maximum number of objects." At the time, it did not seem likely that we would find any such limit except by running out of memory, but we later learned that there exist CORBA products that break down after 32K CORBA objects are created, presumably from exhausting the supply of 2-byte object identifiers. With the benefit of perfect hindsight, we obviously should have performed that test.

3. OBSERVATIONS

The performance of successful CORBA operations proved not to be an issue threatening the feasibility of the manufacturing system. However, platform choice and configuration proved to be of critical importance, as minor changes often determined the success or failure of remote operations under stressful conditions.

The results that we obtained from running the tests on different platforms varied drastically, as did the results that we obtained running the tests on the same platform with different configurations. In some cases, the limits that our tests encountered were "soft" limits that could be increased if one knew how. This behavior is very platform-specific; an experienced tester is needed to judge whether the results are "real" or whether something external to the ORB needs to be reconfigured. In other cases, the tests chanced upon quirks of the particular collection of third-party software that we had installed, which had a drastic impact on performance. A re-execution of the tests with a slightly different configuration, perhaps merely a newer version of the same software, would have obtained extremely different results.

Following are examples of the kinds of operational difficulties that we encountered.

- Find limit on number of connections

  Some CORBA implementations consume one file descriptor for each CORBA connection. In order to attempt to determine a true limit, it was necessary to increase the soft limit on the number of file descriptors. Nevertheless, having increased the supply of file descriptors, we ran afoul of other system limitations and mysterious failures that differed by platform.

- 100 clients

  The inrush of clients attempting to contact the ORB simultaneously caused failures and fatal performance problems that differed by platform and configuration. Modes of failure included:

  - The ORB process exhausting the supply of file descriptors.
  - Delayed responses apparently caused by undiagnosed software interactions and/or resource contention.
  - Clients failing to detect the ORB at all.
  - Clients being starved of CPU time.

- Robustness

  Minor differences in product version or platform determined whether the result was a helpful error message or a complete denial of service.

- APCRunData::RunDataSequence

  The complexity of the generated test code triggered latent bugs in one of the compilers used to compile the test. To obtain a usable executable, it was necessary to change the compilation options.

As a government research laboratory, it is not our role to promulgate results that are specific to particular products. But it is valuable nonetheless to observe that if quantitative benchmark results for CORBA systems are to be meaningful, they must be accompanied by full documentation on the configuration and tweaks that are necessary to achieve the claimed result. Besides being necessary for the sake of rigor, this would be a useful resource for users who run afoul of "default configuration syndrome" as we did.

4. CONCLUSION

APC was hoping for an *a priori guarantee* that their system would port to CORBA. We were not able to supply that guarantee. All benchmarks face the problem of uncertainty ("Your Mileage May Vary") when users try to rely on them for make-or-break decisions, and we can never have 100% confidence that unforeseen operational issues will not appear. However, as object-sharing technology and operating systems mature, the operational issues will become less significant, and scientific approaches to simulation and testing of these systems will become more useful predictors of success.

5. REFERENCES

[1] Project Brief: Advanced Process Control Framework Initiative, http://jazz.nist.gov/atpcf/prjbriefs/prjbrief.cfm?ProjectNumber=95-12-0027, National Institute of Standards and Technology, 1996.

[2] CORBA/IIOP 2.3.1 Specification, http://www.omg.org/cgi-bin/doc?formal/99-10-07, 1999.

[3] MCITT home page, http://www.mel.nist.gov/msidstaff/flater/mcitt/, 1999.