CORBA Benchmarking*

František Plášil[1,2], Petr Tůma[1], Adam Buble[1,2]

[1] *Charles University*
Faculty of Mathematics and Physics,
Department of Software Engineering
Malostranské náměstí 25
118 00 Prague 1
Czech Republic
phone: +420-2-2191 4266
fax: +420-2-532 742
e-mail:{plasil, tuma, buble}@nenya.ms.mff.cuni.cz

[2] *Institute of Computer Science*
Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2
182 07 Prague 8
Czech Republic
Phone: +420-2-6605 3291
Fax: +420-2-858 57 89
e-mail: plasil@uivt.cas.cz

Abstract. This TR was created in preparation of the Charles University response to the ORBOS Benchmark RFI (OMG document bench/98-05-02); this pre-specified the outline of the text. As the main contribution, the authors suggest to reduce ORB benchmarking complexity by factoring ORB request-handling functionality into *prime actions* which can be the subject of a separate benchmarking. To measure the prime actions, ORB has to be equipped with a light-weight intercepting mechanism. It is pointed out that the standard interceptors defined in CORBA are not very useful for this purpose. In addition, a way to overcome the difficulty that some prime actions can be performed in parallel is proposed.

1 Introduction

With our industrial partner, MLC Systeme Ratingen, Germany, we were involved in the CORBA Comparison Project [1]. The goal of the project was to select a CORBA implementation that could support a backbone of a distributed information system. The criteria the evaluation was based on included, e.g., ORB performance, CORBAservices support and interoperability. Naturally, benchmarking was a portion of the project (for evaluating ORBs' performance). Published on the Internet (http://nenya.ms.mff.cuni.cz), the project report drew a lot of attention. This has encouraged us to reflect our benchmarking experience gained during the project as this article (response to the ORBOS Benchmark RFI, bench/98-05-02).

2 Responses to RFI objectives

2.1 What are the goals of CORBA benchmarking in terms of domains, types of applications, etc.?

In our experience related to [1], we identified particularly the following domains where benchmarking gets employed:

- potential ORB users often want to know whether a particular ORB implementation is suitable for their specific application, or which implementation to choose (*user orientation domain*),
- ORB vendors naturally want to know the criteria their customers employ when choosing an ORB, and also want to test their product to locate possible sources of problems. For this domain (*ORB vendors domain*), the goals incline towards getting more low-level details about specific fragments of ORB functionality.

---

* In a slightly modified form, this TR was submitted as the Charles University Response to the OMG Benchmark PSIG RFI (OMG document bench/98-05-02), and is available as the OMG document bench/98-10-04.

Other domains can be found as well, such as using a benchmarking support to diagnose running systems. Overall, however, we see the user orientation domain as the primary target of our RFI response, because this is the domain that puts most emphasis on portability of benchmarks and comparability of results, which we see as the proper areas to be addressed by the OMG standardization activity (as far as the ORB vendors domain goes, most vendors have their test suites already anyway).

2.2 What kinds of benchmark measurements are appropriate in a CORBA environment (e.g. throughput, scalability, reliability, memory footprint, etc.)?

The quality of an ORB is influenced by many factors; some of them can be complex and depend on other factors, e.g. throughput and scalability. It is inherently difficult to evaluate any benchmark-based measurement of a complex factor, as the influence of other (sub)factors involved in the benchmark is usually hard to identify and analyze.

The important factors that make up the overall quality of the ORB are:

a) the effectivity of the request/response handling mechanism (complex factors such as throughput, scalability, and latency can be evaluated here),

b) the reliability and robustness of the ORB implementation,

c) the support for CORBAservices (especially for those services that require close cooperation with the ORB, the Security and Transaction services in particular),

d) the existence and functionality of extensions such as the multithreading support,

e) the interoperability.

In our project, benchmarking was used to test a) and partially b); in fact, there is another benchmark to test parts of d) and e) as well. Briefly, to test the effectivity of the request/response handling mechanism, benchmark suites with the following goals were designed:

- measuring differences in invocation times when using different load patterns and different invocation strategies,

- measuring differences in invocation times related to different aspects of the structural complexity of IDL interfaces,

- measuring the overall throughput of the ORB and the overhead associated with passing specific data types,

- measuring invocation times and resource consumption as functions of the number of objects, number of proxies, number of connections, etc.

In our experience, problems with reliability of an ORB demonstrate themselves in unusual settings, unusual load patterns, etc. Contrary to that, benchmarking is usually done in typical settings, with typical load patterns. Hence, we believe that it is not practical to try devising a benchmarking suite to test ORB reliability (although some test suites might be set up to partially verify, e.g., the correctness of the IIOP implementation, but the OMG Branding Program already does this).

In a simplified view, robustness is determined by upper limits of entities handled, e.g. the maximum size of requests, the maximum number of objects on the server and client sides, etc. Naturally, tests to measure these limits can be defined.

Interoperability is addressed by the OMG Branding Program (http://www.omg.org/members/branding.htm), hence we do not discuss it in any depth in this article.

2.3 Are there meaningful, and possibly interoperable, ways of measuring these?

In our experience, benchmark results (in terms of absolute numbers) vary depending on the benchmarking platform (extremely in many cases); hence it is difficult to conclude on any meaningful results unless one has actually run all the important benchmarks on all the ORBs in question on the same platform (possibly the application target platform in the user orientation domain).

The variance of results also makes it difficult to verify any published data, because the exact settings of the benchmark might be too hard or too expensive to duplicate (not to mention the fact that the correctness of the settings might also be questioned).

A way to address at least some of these difficulties (and the difficulties addressed in Section 2.2) might be to measure partial factors that influence the quality of an ORB separately. As a subject of our current research, we try to base our approach to the ORB benchmarking on expressing the ORB functionality in terms of well-defined *prime actions* (e.g. marshalling, dispatching), and on benchmarking these actions. Using a concept analogous to CORBA interceptors, benchmarking of the prime actions can be based on intercepting *prime points* (points separating prime actions) in the request handling path.

2.4 What kinds of CORBA benchmarking technologies are already available, either as (part of) vendor or publicly available components?

We have already mentioned our CORBA Comparison project [1]. Douglas Schmidt, University of Washington, St. Louis, developed a number of benchmarks (http://siesta.cs.wustl.edu/~schmidt/ACE_wrappers/); another benchmark suite is MCITT, developed in NIST (http://www.mel.nist.gov/msidstaff/flater/mcitt/). There is also a small benchmarking program enclosed with IONA's Orbix, and some of the other ORBs include very simple benchmarks as well.

As an aside, we keep a list of benchmarking results and other related resources at our web site (http://nenya.ms.mff.cuni.cz/thegroup/).

2.5 Is it possible (and practical) to create a library of benchmarking reference algorithms, procedures, and/or applications?

It is our hypothesis that it is possible to create a suite of benchmarks with the goal to measure (and evaluate separately) the prime actions. This is a subject of our current research.

It would be desirable to create such a benchmark suite library for the following reasons:

- the library could be used to execute the benchmarks in a specific setting, e.g., the target application settings in the user orientation domain, to obtain results relevant to that particular setting,
- vendors could provide benchmark library implementations certified to be correctly implemented for a particular ORB,
- publishing the library instead of the results would resolve the problems related to verifying the results mentioned in Section 2.3.

As an aside, in compliance with TPC [4], we believe that there is a difference between creating good benchmarks and creating a good process of reviewing and monitoring those benchmarks. We believe OMG is in a good position to become involved in the latter.

2.6 What sort of ORB instrumentation might be needed or helpful to assist in the gathering of comparative benchmark data?

In section 2.3, we have outlined the idea of reducing benchmarking complexity by measuring prime actions. Such a benchmark needs to be able to find out what prime actions are being carried out by the ORB during the request processing; this can be done, e.g., by intercepting the prime points in the request processing path. To do this, the ORB needs to be instrumented with an adequate intercepting mechanism. We believe that the standard interceptors as defined in the CORBA specification are not very useful for benchmarking purposes, because their very existence can slow down the overall performance of the ORB. Therefore, we suggest to equip ORBs with a set of lightweight measurement upcalls that would signal when a particular request is passing a prime point (based on the publisher-subscriber pattern, a benchmark could register with the ORB to get notified, e.g., about start/stop of marshalling, start/stop of network transport).

The measurement of prime actions is complicated by the fact that the ORB can actually perform several prime actions in parallel, or perform a single prime action in several phases. To overcome this difficulty, the lightweight upcall mechanism would simply be obliged to call the corresponding "start", resp. "stop", function whenever a thread enters, resp. leaves, an activity related to the monitored action (thus potentially re-opening the same action several times during a single request, or opening several actions concurrently).

As presented here, the solution does not resolve issues such as collecting precise statistics of thread activity when the ORB uses multithreading to perform several prime actions in parallel. So far, we have only managed to do this using system-dependent profiling tools, but a definition of a portable benchmark can hardly rely on these.

2.7 What are the potential problems associated with benchmarking (e.g., not measuring what you thought you were, vendor optimization to artificial benchmarks to the detriment of general applicability, copyright, intellectual property, liability issues related to benchmark algorithms, procedures, applications, and results)?

In this section, we address only the benchmark-oriented optimization (benchmark special implementations in [4]) whose purpose is performance optimization of benchmark results without any corresponding applicability to real-word applications and environments - we call this anti-benchmarking.

In principle, we see the following key ways to tackle anti-benchmarking:

- an administrative restriction, e.g. clause 0.2 (anti-benchmark prohibition) in TPCs [4],
- anti-benchmark avoidance, e.g. by a sophisticated design of the benchmark set.

When employing the measurement points approach (sections 2.3 and 2.6), anti-benchmarking related to request handling path measurements should be detectable by creating cross-benchmarks - benchmarks designed to cross-check the partial results by comparing their sums with overall measurements.

At the same time, we believe that an anti-benchmark optimization (e.g., faster delivery based on shortcuts in the dispatcher possible when using only a small numbers of objects) should be detectable by other benchmarks as a hard-to-explain anomaly.

It is also important to note that if an optimization has a real-world motivation (realistic special cases of the number of objects, number of connections, number of threads, etc. taken into account), then all involved parties benefit. The benchmarking suite should have a wide-enough scope to distinguish these optimizations from anti-benchmarking, and, possibly, to characterize the situations where the optimized results apply.

3 References

[1] *"CORBA Comparison Project, Final Report"*, Distributed Systems Research Group, Charles University, Prague, June 1998, http://nenya.ms.mff.cuni.cz/thegroup/

[2] Michi Henning, *"Binding, Migration and Scalability in CORBA"*, DSTC, St. Lucia, Oct. 1998, http://www.dstc.edu.au/BDU/staff/michi-henning.html

[3] Aniruddha S. Gokhale, Douglas C. Schmidt, *"Measuring and Optimizing CORBA Latency and Scalability Over High-speed Networks"*, Washington University, St. Louis, Oct. 1998, http://www.cs.wustl.edu/~schmidt/corba-research-performance.html

[4] Kim Shanley, *"History and Overview of the TPC"*, Transaction Processing Performance Council, Feb. 1998, http://www.tpc.org/articles/tpc.overview.history.1.html