The Design of the IPACS Distributed Software Architecture

Heinz Kredel
University of Mannheim
IT Center, L 15,16
68131 Mannheim, Germany
kredel@rz.uni-mannheim.de

Matthias Merz
University of Mannheim
IT Center, L 15,16
68131 Mannheim, Germany
merz@rz.uni-mannheim.de

Abstract

The IPACS-project (Integrated Performance Analysis of Computer Systems) was founded by the Federal Department of Education, Science, Research and Technology (BMBF) in the program High-Performance-Computing to define a new standard for measuring system performance. One part of this research project is the design of a distributed architecture for execution of benchmarks on High Performance Computers. Its objective is to guide benchmarkers along a mostly automated benchmarking process cycle in compiling and executing benchmarks, and finally gathering and presenting the measured results on a central website. In this paper we present a distributed software architecture which allows immediate analysis of the results, is robust and flexible to support this benchmarking process for the benchmark community.

1. Introduction

The analysis of the performance of computer systems and applications has lead to a vast number of benchmark programs and suites. Among these the TOP500 ([17]) list, based on the Linpack benchmark is the most public visible benchmark in the world. Its success comes first from the scalability of the Linpack benchmark over all computer architectures over the last 25 years, second from the open availability of the source and supporting code together with the community validation of the results and third from the interest of computer manufacturers to publish the best Linpack numbers for their systems as a competitive comparison. For customers the Linpack numbers are a prime corrective to the peek advertised performance (PAP) of computer systems, since a real benchmark program must be executed on a existing computer in order to obtain the performance numbers.

Many other benchmark initiatives fail short on some aspects: the benchmarks are only meaningful for certain architectures, hardware features or certain system sizes (e.g. NAS PB [11]), to obtain and publish the benchmark one must be a member of an organization and follow certain procedures (e.g. TPC [18], SPEC [16]) or there is only an academic interest in the results of a benchmark (see the Netlib link collection [3]).

However, its success is due to a limitation of the Linpack benchmark, is assesses the suitability of a computer system only by computing a solution to a dense and arbitrarily big system of linear equations. But many of today's applications incorporate new algorithms with different system stress patterns or algorithms based on new mathematical theories. On the practical side it is relatively easy to run the Linpack with reasonable results by a benchmark professional but for most new or young benchmarkers it is very hard to tune and optimize the Linpack software configuration to achieve good results.

The IPACS-project ([6]) wants to improve this situation by augmenting Linpack with a set of low level and application benchmarks and in easing the execution of these benchmarks. The evaluation and selection or the development of augmenting benchmarks is part of other IPACS publications, in this paper we focus on the development of an distributed and easier to use benchmark environment.

1.1 Related Work

Other benchmarking activities do not aim at such an highly automated process cycle. There is one project 'Repository in a Box' (RIB, [15]) which is a software package for creating web metadata repositories which can contain metadata for benchmark suites for various application domains. This tool helps finding benchmarks or other software in a specific application domain that does not contain the benchmark code or benchmark results.

The Performance Database Server ([12]) is a web-server which contains results of various benchmarks from Dhrystone to Linpack. The results from Linpack are mostly up to date, but other tables contain merely historical data. Data input seems to be sent via email to the maintainers. The goal of the Performance Evaluation Research Center ([13]) is a scientific understanding and improvement of the performance of HPC systems. Although they develop benchmarks and performance models for predictions (just as IPACS) it seems not to be intended to facilitate the benchmarking and publishing process. The HPC Challenge ([4]) with PaMaC project ([14]) also aim at a suitable benchmark suite which can complement the Linpack/TOP500 benchmark. The proposed benchmarks are primarily based on Linpack and its software infrastructure. The web-site contains an archive of benchmark results and provides a web-form to be filled out and submitted together with the benchmark result file. User validation is via email response with an activating URL.

So the IPACS concept of integrating a benchmark code repository, a benchmark result repository and automated process cycle contributes new ideas and experiences in benchmarking.

1.2 Outline of the paper

The remainder of this paper is organized as follows: Section 2 recalls the traditional process of benchmarking and highlights the improvements of the automated IPACS benchmark cycle. Section 3 introduces the distributed software architecture with its main components: benchmark client, repository server and web-presentation. Section 4 concludes this paper.

2. Process of Benchmarking

The process of benchmarking High Performance Computers is quite complex and assumes fundamental benchmarking experience, in particular by configuring and compiling benchmark sources with approved compiler flags and proper operating system settings. The IPACS benchmark environment should guide the (novel) benchmarkers in the selection and deployment of the benchmarks, guide the execution and tuning of a benchmark and automatically collect the results, publish them and presenting them for comparison with other computer systems and benchmark runs.

Let us first recall the well-known Linpack process cycle (fig. 1): a person selects the Linpack source code from Netlib web-site, the Linpack is transferred to the target computer system, compiled, tuned and run, finally the results are pasted into a web-form at TOP500 or send by email to Jack Dongarra (the developer and maintainer of the benchmark). If the results are meaningful or bad can be seen by feedback from the TOP500 team or the bi-yearly published and moderated result lists on www.top500.org.

IPACS aims at an automated process cycle (fig. 2): a person connects with the IPACS client to the IPACS repository server, the client guides the person in the determination of basic facts about the High Performance Computer, based on this information a suitable benchmark is downloaded. The IPACS client guides the person in the compilation, tuning and running of the selected benchmark, once results from a benchmark are available, they are transmitted as XML forms setup by the IPACS client and stored in the repository database. The client then

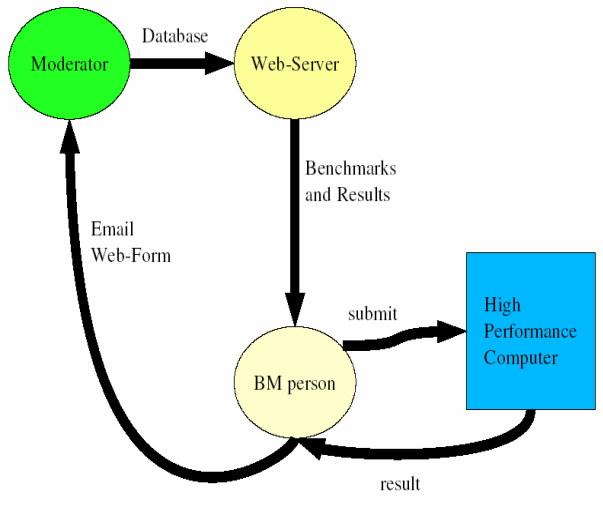

Figure 1: Linpack Benchmarking Process

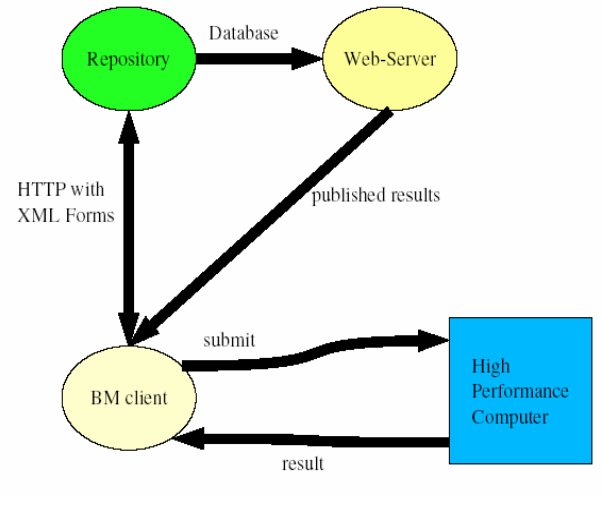

Figure 2: IPACS Process Cycle

guides the browsing of the repository contents with the new results.

3. Software Architecture

As discussed in the previous sections the IPACS Distributed Software Architecture includes three primary elements as illustrated in fig. 3: benchmark client, repository server and web-presentation. The objectives to be met by each of these components are outlined below:

3.1 Benchmark Client

The benchmark client mediates as graphical user interface between benchmarker and repository server. Starting the client for the first time on a High Performance Computer, the client is registering itself at the repository server. Afterwards the client tries to detect the current hard- and software environment partly by asking the benchmarker for further information. To avoid redundant data the client does not store any information locally, with the exception of a simple identification number. Thus, every information about the hard- and software environment, the measured results and further information entered by a benchmarker are directly stored at the repository servers database, which leads to another advantage: the repository server - and with it also the web-presentation - has always the most up-to-date information.

In cooperation with the repository server and independent of the current hard- and software environment, the client suggests the execution of useful benchmarks according to the IPACS benchmark methodology. By starting the execution of a suggested benchmark the client automatically downloads either a pre-compiled benchmark for the current operating system if available or an existing benchmark source code. After the execution of the benchmark the client imports the measured results and transmits them to the repository server.

To cope with different hardware and operating systems, the benchmark client was completely developed in the object-oriented portable programming language Java. The client uses the standard HTTP-protocol with XML messages to communicate with the repository server. Other message oriented middleware such as CORBA (Common Object Request Broker Architecture) or RMI (Remote Method Invocation) have been considered in the design phase. Although these are much higher-level standards and provide easier programming APIs they require a sophisticated infrastructure and are therefore more difficult to deploy in an unknown environment. So CORBA requires a setup of an ORB (object request broker) and accompanying services together with the installation and configuration of the respective classes which

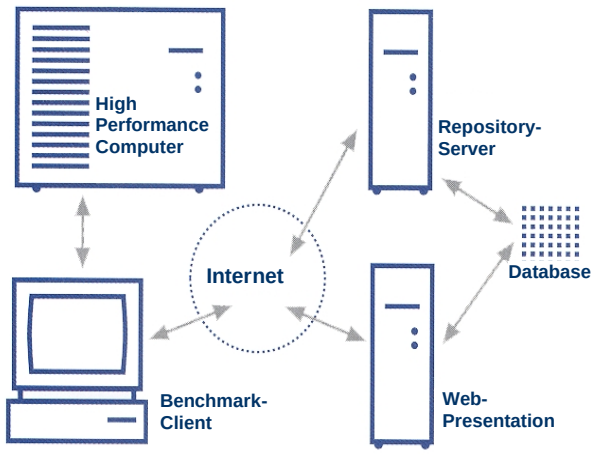

Figure 3: IPACS Architecture

makes the deployment of the client much more complicated. Basically RMI has the same disadvantages as also services on certain ports must be provided. Despite of these problems and since also the CORBA Firewall Traversal Specification was not settled during our design phase, we decided to stick to the more robust and ubiquitous HTTP-protocol on port 80. Thus our concept avoids problems with the infrastructure, especially with firewalls of HPC centers.

Up to now all IPACS benchmark suites produce only ASCII text files as output. The relevant information in these files is identified and parsed to be structured for further processing. Together with the benchmark developers we designed a global XML-structure (DTD) to describe a common XML-output for all involved benchmarks. As a result the benchmark client has only to transmit these XML-files via HTTP to the repository server without additional data-interpretation and on the other side the repository server is always able to deal with the received information. Compared to other solutions, e.g. serialized objects instead of XML-messages or well-known web-services our approach leads to an further advantage: In the rather unlikely case of firewall-problems with port 80, the XML-files could also be transmitted with other services e.g. ftp or email.

3.2 Repository Server

While the benchmark client was designed as a lightweight approach without extensive intelligence, the main work is done on the repository server. Once a new client is registered, the repository server requests information about the current hard- and software environment. Based on the received facts and probably already stored information the repository server determines useful benchmarks for execution on the High Performance Computer. It has to deal with information about available benchmarks, current versions and their location at the IPACS web-server. After benchmark execution it has to collect the measurement results and to prepare the data for the web-presentation. One fact to keep in mind is that measurement results are only meaningful even they are presented together with the current hard- and software configuration.

To realize our distributed client-server environment, several different scenarios are possible. Our first solution was based one the widespread Enterprise Java-Beans (EJB) component model [10] as part of the J2EE platform [9]. But due to financial limitations and the flaws of the available open source J2EE implementations (e.g. memory leak in JBoss 2.x) when our project started, we decided to realize our implementation based on Apache's open-source servlet-container Tomcat [2]. Afterwards we discussed possible persistence solutions for our backend. Our first model was based on an object-oriented database, because object-oriented databases appear to be the conceptually most appealing kind of data store. But due to the need of the web-presentation we decided to use a relational database even though this lead to the so-called impedance mismatch between the object-oriented and component-based business logic in the application layer. At first glance the new Java Data Objects (JDO) standard [8] might well be a considerable solution as the persistence layer of choice. But whereas in none-managed environments are at least a few open source implementations available (with all kind

of restrictions and limited functionality), currently only expensive commercial JDO-implementations are existing with the use in managed environments.

Mapping benchmark results and computer information from the communicated XML files to fixed relational tables with hardcoded SQL/JDBC (Structured Query Language/) leads to a disadvantage regarding flexibility. Even the use of available O/R mapping tools often locks the developer into a particular vendor and leads to a restriction in application portability. Therefore we developed a suitable persistence layer for the special requirements in benchmarking. Based on a XML-mapping file our approach allows to change the information structure in our database tables and relations without code-modifications to the repository server. This ensures an easy adaptability of all components to future requirements.

3.3 Web-Presentation

Once the benchmarks are executed and the output files are available one would like to compare the performance results to other results in the repository. The comparison will reveal deficits in the software tuning of the benchmark with respect to similar architectures and problems of the hardware of new computers if no improvements can be seen. So facilitating easy comparisons is a mayor goal of IPACS. There are at least three software designs to meet these goals: first we could augment the client with sufficient capabilities to present the comparison data, second we could augment the server with the public Web-publishing functionality to present the data and third we could add some private Web-publishing functionality on the server.

The first design would give the benchmarkers private and immediate comparison data, however it would mean that the client software must be considerably more complex and would need longer development and testing time. So we decided to implement all comparison functionalities only on the Web-server, where it must be implemented anyway. With private comparison the benchmarker could compare their results with the data contained in the repository before making their results public available. The private comparison could be interesting in cases a benchmark is not yet tuned sufficiently or a new hardware is benchmarked which is not yet disclosed or finally optimized for performance. On the other hand the immediate public availability of the results can be inspiring and motivating for other benchmarkers. Immediate publishing was also used with the online Java Linpack benchmark [7] which was widely accepted. The implementation of a private comparison capability would require some session tracking and per session views on the repository database. So for the software architecture we decided to keep the design simple in the first place and not to implement the private comparison functionality. From the maintainers point of view the private comparison seems also unattractive, as we could end up with a repository containing thousands of results but only a small number that is publicly visible. This would considerably lower the value of the IPACS services to the community.

For the software to realize the Web-service we stick to the proven robust combination of the Apache HTTP-Server with PHP scripts to access the MySQL database of the repository. The presentation of the benchmark results is straight forward. Every benchmark has three levels of details: a single number which presents the most condensed result (as the R_{max} of Linpack), a set of (three) numbers which give more insight into the performance characteristics (e.g. N_{max}, $N_{1/2}$ in Linpack) and finally the textual output of the benchmark run. The results are first presented in the most condensed form together with the main hardware and software characteristics and a visitor can then click on appropriate buttons or links to uncover more and more details. This approach is also used on other benchmarking activities, e.g. by the web-presentation of the SPEC benchmarks.

A final point in the presentation of the results to consider is the order in which the computer systems are listed. For Linpack/TOP500 the computers are easily ordered with respect to their R_{max} value. But for IPACS with more than six different benchmarks it is not possible to find a meaningful way to order the systems. The IDC HPC [5] tries to define a rank based on a (equal) weighting scheme of the different benchmarks to construct a single number for ordering. However there has been disagreement about this scheme in the scientific benchmark community [1] as a weight between different performance numbers depends on particular application characteristics. Each visitor has its own applications which imply different importance (or weights) between the numbers of different benchmarks. Therefore IPACS will not imply any ordering on the overview results but will provide visitors with the option of selection of own ordering schemes for assessment.

4. Conclusion

With the IPACS software environment we presented a new step in the support of an easy and efficient benchmarking cycle. The proposed software architecture is robust and simple to be deployed in a wide variety of heterogeneous client environments. The database design is elaborate enough to start with the benchmarking but is also flexible in design to be easily adaptable to future requirements. The presentation of the benchmark results is designed to be most useful for the benchmark community. All results are immediately transferred to the web site via the repository server to make them available for comparison and analysis with other results.

We thankfully acknowledge the fruitful discussions with our colleagues from the IPACS project and others. In particular thanks to F.J. Pfreund for the project initiative, to M. Meuer for the IPACS web presentation, to H. Bockhorst, A. Geiger, D. Merten, E. Strohmaier, C. Simmendinger, D. Waschk for clarifying the requirements of the benchmark process. We also thank our colleagues from the IT Center and the department Information Systems of the University of Mannheim for their continued support of the project.

References

[1] Aad van der Steen (2002), How informative is the IDC Balanced Rating HPC Benchmark? http://www.hoise.com/primeur/02/articles/weekly/AE-PR-03-02-60.html

[2] Apache Software Foundation: The Jakarta Project – Tomcat, http://jakarta.apache.org/tomcat/index.html

[3] High Performance Linpack (HPL) at Netlib: URL: http://www.netlib.org/benchmark/hpl/

[4] HPC Challenge: URL: http://icl.cs.utk.edu/hpcc/

[5] IDC market research: URL: http://63.143.40.111/benchmark/

[6] IPACS Project: URL: http://www.ipacs-benchmark.org/

[7] Java Linpack: URL: http://www.netlib.org/benchmark/linpackjava/

[8] Java Community Process (2003): JSR-000012 Java Data Objects Specification 1.0.1

[9] Sun Microsystems (2003): Java 2 Platform Enterprise Edition 1.4., http://java.sun.com/j2ee/

[10] Sun Microsystems (2002): Enterprise JavaBeans Specification 2.1, http://java.sun.com/products/ejb/

[11] NAS Parallel Benchmark: URL: http://www.nas.nasa.gov/

[12] Performance Database Server: URL: http://performance.netlib.org/performance/html/PDStop.html

[13] Performance Evaluation Research Center (PERC): URL: http://perc.nersc.gov/main.htm

[14] Performance Modeling and Characterization (PaMaC): URL : http://www.sdsc.edu/PMaC/Benchmark/

[15] Repository in a Box (RIB): URL: http://icl.cs.utk.edu/rib/

[16] Standard Performance Evaluation Corporation (SPEC): URL: http://www.specbench.org/

[17] TOP500 Supercomputer Sites: URL: http://www.top500.org/

[18] Transaction Processing Council (TPC): URL: http://www.tpc.org/