Developing a test frame for distributed systems.

D. Dewolfs

29th August 2003

Abstract

The computational needs of complex numerical processing are rapidly growing beyond the capabilities of regular personal computers or workstations, and supercomputing solutions are becoming essential for tackling challenges in research areas like quantum physics or engineering. Often, distributed computing platforms offer a solution to these problems. A specific area of interest is resource scavening, which uses idle computing resources of non-dedicated machines for solving various scientific problems. This paper discusses an attempt to lay the foundations for a test frame to evaluate the qualities of available solutions in these areas.

1 Basics

To get a clear image of the performance issues involved with the practical application of a distributed platform to a given problem, it is necessary to design a system for gathering empiric data to quantify these performance characteristics. It is the goal of the research discussed in this paper to investigate different angles on how to implement such a performance profiling system.

Performance, in the context of this paper, is defined as relative deviation of the measured total wallclock time needed to complete a single experiment (the nature of an experiment is further discussed below in 3) from a theoreticaly estimated ideal. As research in this area advances, this definition will probably become more specific and/or grow in scope as metrics are added and refined. Conclusions gathered from an experiment should also discuss the other issues involved with deploying the system under test, like ease of use and ease of programming with the system, available tools, deployment time and needs etc. These must be treated separately from the pure performance figures but should not be forgotten when deciding in favor of a single system.

The test frame should be generic enough to evaluate the properties of a system under test with regard to a wide array of problems and the technical needs associated with them (node intercommunication, large data tranfers, high-frequency data transfers, highly heterogeneous computing loads etc.). In order to accomplish this goal, the basis of such an abstract test frame must be founded in a philosophy of simplicity and maximum abstraction.

2 General approach

A preliminary, proof of concept implementation for such a test frame was developed to provide a better practical insight into the different facets of the research area. The following paragraphs present an outline of this implementation.

To test and compare a set of distributed platforms, a cluster is set up with the necessary software for each selected system. A basic farmer-worker (FW) data distribution mechanism is then implemented for each tested system (FW was chosen primarily because of it's simplicity and the origins of this research in investigating a number of LINDA-spaces[3] based systems which are naturally suitable to this approach). The farmer then feeds the systems under test with a set

of data packets to be processed by the worker nodes and the total wallclock time needed for running the experiment is measured when the farmer node accumulates results for all given packets. Finally, a set of simple formulae is used to calculate overhead.

3 Generating the test sets

The stated requirements for the testframe bring up a number of questions : what kind of calculation will be used for the "benchmark" ? How are the test sets of data packets generated? What kind of data is sent in the packets?

To meet the requirement of abstraction, it was decided not to opt for known "popular benchmark algorithms" (e.g. a sorting operation[6] or solving a Fourier series), but to use a system based on "sleep-tasks" : tasks that consist of suspending operation of the running thread for a predetermined duration (as per the "sleep"-function in UNIX). This has the added benefit that the amount of time to be "slept" can be used as an absolute metric for the "execution weight" of a given data packet (the weight of the packet on the operation of the whole system, e.g. as a function of the amount of data to be analysed, it's complexity with regard to calculation, etc.) and for aproximating the expected total wallclock time for the experiment.

Using this approach, it is theoretically possible to create multiple profiles of data packet sets (e.g. a set of data packets of equal weight, a set of packets with "roughly" equal weight, a set of packets with weights that grow as the experiment progresses, an uneven mix of packets with greater and smaller weights etc.). Using this approach, it is possible to reach the goal of genericity by providing the ability to build generic "problem profiles" which can then be run through a given distributed system. Sleeps also allow for using the same abstract measurement for "execution weight" over a heterogeneous set of slave machines as they will roughly take similar times to complete over a range many different platforms.

The following input parameters are used for generating the test sets in the preliminary implementation : the number of data packets, a mean duration for each sleep operation associated with a data packet and a standard deviation, randomly creating a set of packets with sleep durations spread according to a normal distribution. The result is a set of sleep durations for each data packet in the test set, which is then written to an XML file for parsing by the software that is to feed the given RSS at the time of the experiment. The MPICH experiment

4 First experiments

Data test sets created through the test set generator were used to feed a basic, proof-of-concept MPICH-based C implementation of a FW system and similar systems based upon SUN's JavaSpaces and the Condor system. A series of experiments was performed for different sets of generated data, repeated tenfold, and results were gathered in XML files. A straightforward analysis of the results was made to reach a conclusion on (a) the performance efficiency of the systems, and (b) usability of the test frame as a tool for evaluating these qualities. Issues encountered during deployment of the experiment were also taken into account for the final conclusion.

The experiments were successfully performed and the MPICH experiment was able to present conclusions on the influence of the input parameters on 2 metrics : percentile deviation from the estimated total wallclock time for an experiment and deviation from the estimated total speedup as the number of slave processors rises.

5 Future work

It is now necessary to develop further from this basic framework and add more functionality. The first thing under investigation is the issue of test sets, and current research is focusing on creating realistic communication profiles for a number of practical problems. Another line of investigation to be followed

is the inclusion of communication models other than the FW paradigm. The theoretical models used for making estimates of the ideal wallclock times and speedups will also need to be adapted as the variety of test sets grows : a basic queueing model was sufficient for the MPICH-based experiment, as it used packets with roughly equal sleep durations - more complex situations with sleeps of widely differing length will require a more complex approach. A last element under investigation is the capability of adding network and CPU overhead to the cluster model, under the form of network and CPU parasites and realistic payloads for the data packets.

References

[1] W. Gropp, E. Lusk and A. Skjellum. A high-performance, portable implementation of the MPI Message-Passing Interface standard. Parallel Computing, 22(6):789-828. 1996

[2] M.J. Litzkow, M.K. Livny and M.W. Mutka. Condor - a hunter of idle workstations. Eight International Conference on Distributed Computing Systems, San Jose, California:104-111. 1988

[3] S.C. Associates. Linda User's Guide and Reference Manual. http://www.lindaspaces.com/downloads/lindamanual.pdf. 2000

[4] Sun Microsystems. Javaspaces service specification version 1.2. Technical Report. 2001

[5] The CoMP website : http://www.ruca.ua.ac.be/cmp

[6] R.E. Eggen, S. Gwizdak, M. Eggen. SOAP and XML as Parallel Distributed Environments : An Empirical Cost/Benefit Evaluation. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2002). 2002

[7] F. Hancke, G. Stuer, D. Dewolfs, J. Broeckhove, F. Arickx and T. Dhaene. Modelling overhead in JavaSpaces. Proceedings Euromedia 2003:77-81. 2003. http://www.ruca.ua.ac.be/hancke.