Problem Independent Metacomputing classification

P. Hellinckx, F. Arickx, J. Broeckhove and T. Dhaene
Computational Modelling And Programming
Antwerp University
Middelheimlaan, 1 – 2020, Antwerp,
Belgium
E-mail: Pehe@ruca.ua.ac.be

ABSTRACT

This paper describes a new metacomputing characterisation approach. The *problem dependent* benchmarking is replaced by a dynamic *problem independent* characterisation approach by introducing a new level of abstraction. This allows for problem independent metacomputing classification.

INTRODUCTION

While in recent times the computational power of modern computing systems has grown spectacularly, the need for increasing computing capacity has grown even faster. A possible solution to this problem is the utilization of the idle CPU-cycles of workstations around the world. Ideally, all unused resources should be made available to whoever needs them. This concept is known as *resource scavenging*. DCS's (Distributed Computational Systems) try to approximate this ideal situation as closely as possible.

Although many DCS's, (e.g. the Linda Tuple Space implementations, Condor, …) were introduced during the last decade(s), there is no actual information on the performance characteristics of those systems. While, all of them are more or less reliable, the key question: "*Which one performs best in this particular situation?*" remains unanswered. Running currently available benchmarks results in static problem-dependent results which do not necessarily correspond to the problem at hand. One only gets exact performance results for the specific problems contained in the benchmark. For all other cases, an error prone interpolation has to be performed.

Ideally, this *characterisation* problem is solved by *a multidimensional performance lookup space constructor* which provides for low effort *DCS performance lookup table* construction. By comparing the problem specific performance data of the different lookuptables a problem specific DCS *classification* is realised.

LOOKUPSPACE ENTITIES

In order to construct and eventually use a lookup space an adequate entity description is required. As each entity, a distributed solution of a specific problem, is in fact a distribution of tasks on a certain infrastructure, the description consists of some tasklist descriptors and some infrastructure descriptors. In the current stage of this research the tasklist is only described by the number of tasks and the duration of each task. The infrastructure is only described by the number of available workers. Future research will include the expansion of the tasklist- and infrastructure descriptors by introducing for example the physical size of a task, the network infrastructure, the computational power and a lot more.

LOOKUPSPACE CONSTRUCTION

Constructing a lookupspace is in fact mapping the performance data on the corresponding distributed problems. The infinite amount of distributed problems makes an exact mapping impossible. To resolve this problem we tried a new approach. In stead of mapping the performance on each problem we analyzed the influence of the different descriptor parameters on the performance of the DCS.

If one could simulate each possible distributed problem by supplying its parameter tuple, it is possible to do an empirical detection of the impact of these, mutually dependent, parameters on the overall performance of the DCS. This can be represented as a multi-dimensional *lookup space* which maps the distributed problems to their corresponding performance on a given DCS. This mapping can be performed in two ways, (1) by a brute force approach in which all possible combinations are evaluated, and (2) by a dynamic adaptive statistical approach such as DOE (Montgomery 2001) in which only relevant tuples are selected. As the second method is still ongoing research of a companion project (Hancke et al. 2003) (De Neve 2003) currently the brute force approach is still in use.

DISTRIBUTED PROBLEM SIMULATION

The fundamental properties of the needed simulation technique are *total problem range coverage, simulation correctness* and *easy definability*. This means that for every existing distributable problem it should be possible to generate a simulation which is indistinguishable of the original problem and can be completely defined by a couple of parameters. By directly manipulating the fundamental building blocks of distribution problems, the tasks, the properties mentioned above can be obtained. Figure 1 illustrates a typical distribution scheme.

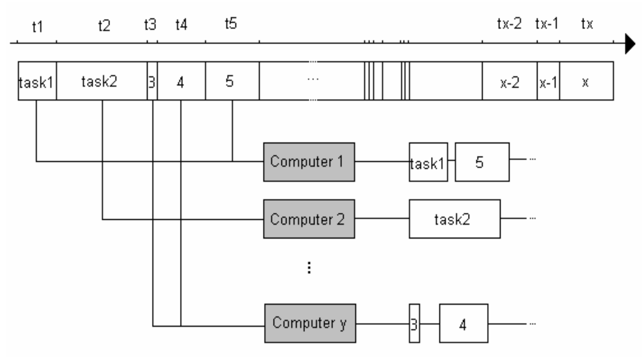

Figure 1: Typical distribution scheme: Each task 'a' takes time 'ta' to compute on one machine. All 'x' tasks are distributed on the 'y' available computers.

Such a scheme consists of 'x' different tasks which can all be computed independently of one another. There is no relation between the tasks and/or the computers that execute them.

Different distributive problems are distinguished by their task lists. Fulfilling the property of having *total problem range coverage* comes down to dealing with this difference in task lists. An easy way to generate every possible task list would solve this problem. This is done by replacing the actual tasks by a time consuming dummy function. See figure. 2.

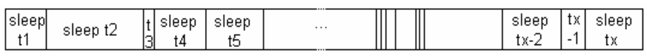

Figure 2: Introducing dummy tasks: each task 'a' is replaced by a sleep of length 'ta'

This function occupies the processor for a given time. In the current version of our testing technique the tasks are replaced by a sleep function which sleeps the actual running time of the original task (task1 will be replaced by a sleep of time t1). Applying this technique, one can build every task list one wants by generating the corresponding list of sleeps.

Due to the number of tasks, each with specific properties, in a task list it is impossible to define it manually. To resolve this problem a test-generator is build. Given the appropriate input, an XML-file (McLaughlin 2001) representing the intended task list is constructed. See figure 3.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE tasklist>
    <tasklist>
        <descriptor>
            <nrtasks>10</nrtasks>
            <duration>
                <NormalDistribution std="1.0" avg="10.0" />
            </duration>
        </descriptor>
        <tasks>
            <task id="1">
                <duration>9</duration>
            </task>
            <task id="2">
                <duration>10</duration>
            </task>

                ...

            <task id="10">
                <duration>9</duration>
            </task>
        </tasks>
     </tasklist>
```

Figure 3: Example of an XML file of a tasklist with 10

The use of an XML file has two main advantages:

1. It provides for platform independent storage medium for task lists.
2. Identical experiments can be repeated using the same task list.

Currently the task lists are build using two inputs:

1. The number of tasks.
2. The PDF (probability distribution function) of the task duration.

Currently only two PDF's are supported: fixed and Gaussian. In future versions more will be added. Ultimately, a completely user definable PDF will be supported.

CONCLUSION

In this paper a *problem-independent meta-computing characterization technique* was introduced. It provides a new approach to model a distributed computational systems. The proposed technique enables one to obtain, without performing any experiment, the performance of a problem solution on a given DCS by simply specifying its parameter tuple.

REFERENCES

Xml. URL: http://www.xml.com.

Chow R. and T. Johnson 1999. “Distributed Operating Systems & Algorithms.” 1999. *Addison-Wesley Pub Co* (March 1997)

Couvares, P. and T. Tannenbaum 2001,”Condor Tutorial” *First EuroGlobus Workshop* June 2001

Gropp, W.; L. Ewing and Skjellum A. 1999. “Using MPI”. *The MIT Press, second edition*, 1999.

Hancke, F.; G. Stuer; D. Dewolfs; J. Broeckhove; F. Arickx and T. Dhaene 2003. “ Modelling Overhead in JavaSpaces”. *Proceedings Euromedia* 2003, p77-81 Available at http://www.ruca.ua.ac.be/hancke/

McLaughlin, B..2001. “Java & XML”. *O'Reilly & Associates*; 2 edition (September 2001)

Montgomery, D.C. 2001.“Design and Analysis of Experiments”.the fifth edition. John Wiley and Sons Inc., New York, 2001.

Nester, C.; M.Philippsen and B. Haumacher . “A More Efficient RMI for Java”. In *Proc. of ACM 1999 Java Grande Conference*, pages 152–157, San Francisco, Calif., June 1999.

Neve, H.D. 2003. “Performantieanalyse van gedistribueerde systemen m.b.v Design Of Experiment”s. 2003. *Thesis UA Academiejaar* 2002 – 2003