Towards a Grid Applicable Parallel Architecture Machine

Karolj Skala and Zorislav Sojat

Ruder Boskovi, Institute, Center for Informatics and Computing
Bijenicka 54, HR-10000 Zagreb, Croatia.
{skala,sojat}@irb.hr

Abstract. An approach towards defining a flexible, commonly programmable, efficient multi-user Virtual Infrastructural Machine (VIM) is proposed. A VIM is a software infrastructure which enables programming a Grid infrastructure scalable at the lowest possible level. This shall be attained by a Virtual Machine with inherent parallel execution, executing compiled and interpreted computer languages on a very complex p-code level. Based on this interactive deformalized (i.e. natural language like) human-Grid interaction languages should be developed, enabling parallel programming to be done by inherently parallel model description of the given problem.

1 Introduction

The development of the e-science, which, predictably, will evolve towards the e-*society*, justifies the necessary efforts towards better understanding of computer implementation, through active Grids, of naturally distributed parallelism of the environment we live in.

It is a fact that new technologies, clusters and specifically the Grids, are in the execution of programmed tasks quite different from classical single- or multi-processor computing environments by using the parallel architecture of a heap of hugely independent computers, connected through different communication links.

This poses a great many problems in the development of a viable Grid system. The main issues lie in developing such a structure (i.e. software, including middleware issues), infrastructure (i.e. hardware, including low level software and firmware) and applications (including high level portals etc.) which can effectively utilize the extremely different execution environments[1], and actually enable a seamless integration of different kinds of Grid-enabled equipment, including inter alias scientific instruments, experimentation and production tools, mobile units of all kinds etc. into one Grid system.

To enable effective utilization of the multiplicity and variety of Grid resources, in this paper we propose a direction for further scientific and practical exploration towards a human usable viable Grid system, i.e. a specific set of, as we see it, necessary steps towards defining a Grid Applicable Parallel Architecture Machine (GAPAM).

[1] I.e. cross-Grid, cross-platform and cross-cluster execution on computers of different makes, operating systems, performances and other parameters.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 119–123, 2004.

2 The Grid Software Infrastructure

Actually the approach taken is an attempt to define the software infrastructure of a complex programmable system which has to be *flexible*, *easily programmable* in commonly known programming languages, and *efficient* in resource utilization and execution speed. A further requirement is for it to be multi-user, where all user privacy and group activities are supported. It shall be scalable, to encompass the whole range of Grid-enabled equipment, from simple user Grid Terminals, up to clusters of huge computing or data power, including fast and large data experiment, collection and visualisation equipment.

A major *flexibility* prerequisite is that the final system should not be an Operating System (OS), but a Virtual Infrastructural Machine (VIM), which should be applicable either in hardware, firmware, as a kernel, or as a layer, daemon or just an application. Specific programmable devices, like visualization equipment, measurement equipment, experimental equipment, Grid Terminals and other similar equipment, as well as different scientific, technical and production tools, have also to be taken into account.

We state that most of the problems mentioned may be much easier to solve, and some of the inherent heterogeneity problems would be solved in advance, by construction of a simple and consistent high performance flexible software infrastructure as a Virtual Infrastructural Machine (VIM). A VIM is actually a Virtual Machine (VM), defined in such a way that its basic states are distributed Grid states, i.e. its basic design is *infrastructural*.

This can be obtained by providing a lowest programmable common denominator, in the form of a Virtual Infrastructural Parallel Architecture Machine, so a viable Grid system could be written in programming languages translated into the interpretative, therefore easily reconfigurable, VIM p-code, executed in a closed interpretative software (or even hardware) driven machine. Such a Grid Applicable Parallel Architecture Machine (GAPAM) would tackle the particular sensitivity of all of the resources on a Grid to security issues in a strictly controlled way, as the execution of Grid software is done in a closed module fashion of a Virtual Machine.

Actually, we are advocating a goal of developing a system easily usable by not only computer experts and programmers, but also a wide circle of scientists and other common users. To attain such a goal, it is not only necessary to define, as mentioned above, a GAPAM, but also to develop, based on this VIM, a human interaction interface, in the form of interactive parallel model description language(s).

3 Approach

To attain this overall goal, we should primarily explore the inherent parallelism principles of most complex problems, through the prism of their immanent natural distributed parallelism, and to find effective and efficient methods of implementation of interactive programming systems which are able to automatically execute inherently parallel described models on parallel architectures, by a sequence of coordinated articulation and dearticulation of the model.

On one side there is the goal of producing such a VIM which allows proper articulation of a formal programming language into elements which enable loosely synchronized parallel execution. This will ease the strain of programming parallel applications.

On the other side, our goal should be producing targeted dearticulated and deformalized, in other words natural-language like, model definition languages, which will, through an interactive programming environment, facilitate the use of complex active Grid technologies, specifically enabling their widespread use by non-specially educated users.

The existing parallel programming languages (e.g. Occam, Orca etc.) are primarily based on explicit parallelism rules, i.e. the human user has to explicitly state the possibilities of parallel execution of certain sub-modules, i.e. sub-algorithms.

It is obvious from general knowledge on programming languages that the underlining principles of some of the "lateral" and less popular languages are not derivatives of the single instruction stream thought process. By this we primarily mean the object-oriented languages like Smalltalk, inherently interactive languages like Forth, Prolog, Lisp, and inherently parallel languages like APL, J and similar.

Fitting languages are primarily those that enable conceptual parallelism of natural process description, independent of their present day single-stream implementations. Those are then the languages that shall be used as the basis for their further dearticulation and deformalization.

The systematic and inter-compatible articulation[2] of the implementation principles of a particular programming language in parallel computing means finding those articulations of a specific programming principle which are elegantly and efficiently applicable onto programming of co-work of a set of identical or different computers (like in a cluster or a Grid).

The dearticulation of a programming language is actually the production of a "superstructure" above the programming system. This so called "superstructure" dearticulated from the programming language is effectively such a language which is specifically targeted for the description of particular scientific (or other user) models in a way as to be linguistically as close to the user as possible, and by this also more usable. By dearticulating the language, a level of interaction easier acceptable by a human is obtained.

A major principle leading to proper dearticulation, which is, per se, in the end very user-specific, is interactivity. To augment the measure of acceptability and possibility of such interactive programming, specifically by the scientists, which in their work, in modern science, always use formalized models of specific objects or areas of their scientific endeavours, special targeted dearticulation shall be used. By such dearticulation GAPAM languages would be adapted to specific targeted user communities.

The main means of attaining such dearticulation is the definition of a "superstructure" of the language by using denotational synonyms, and the exploration of the possibilities of deformalisation of such dearticulated languages. In this sense *deformalisation* is to be regarded as development of synonyms, easy and partially automatic definition of abstracts, i.e. abstraction layers, and a non-too-formal

[2] The articulation of a language is extracting of a lover level of symbols, articulating sentences into words, words into morphemes etc. Language articulation on any level may be syntactic (like phonetics), or semantic (like semiotics).

grammar, which enables interactive, step by step, and recursive formalisation of human input, by leading to the creation of a model description language in a form of a knowledge base for specific user or user groups and types.

The method to be used for the given purpose and goals is primarily consisting of proper *reformalization*, and consequently proper articulation of those model-definition languages, deformalized and dearticulated in the described way, into strictly formal forms of chosen programming languages. The mentioned model-definition languages, due to their deformalized and dearticulated nature may not strictly be called programming languages.

The articulation of those languages into formal forms shall be done in such a way that it is reasonably easy, and certainly feasible, to have a high quality implementation of such model-definition languages on clusters and Grids.

The implementation of the whole system shall, consequently, be done by the principle of imbedding, where a lower level executional virtual machine, i.e. the Virtual Infrastructural Machine (VIM), is used to implement a higher level interpretative Virtual Machine (VM), on a multilevel principle, where each successive higher level VM implements a specific formal language or deformalized dearticulation of such a language towards a specifically targeted human language.

As any dearticulated and deformalized language, the same as any natural (i.e. non-formal) language, allows (and necessitates) polysignificance and unobvious ness, the interactivity of the system must support guided significance definition of each specific model definition. In such a way the reformalization of the dearticulated deformalized language is attained in an interactive way.

The user programs written in such a way would be multiply understandable and executable. The multiplicity of the understanding and execution possibilities of such user programs is essential for the attainment of better human-machine interaction principles. They would be understandable on the scientist's dearticulated language, i.e. the community that uses this particular target language for their model descriptions will be able to understand the notation and notions used in such an executable model. As such the dearticulated and deformalized language may be quite close to common scientific jargon used in the specific field; such a model description can be easily readable and understandable for all humans knowledgeable in the particular field. On the other hand, the model description programmes will be, after their first articulation, and the interactive reformalization, understandable on the chosen underlying programming language. This has a consequence of transferability to other machines outside the described Virtual Infrastructural Machine system, as well as the understandability to other humans which do not know the specific science field dearticulated language, but do know the formal programming language.

Such model description programmes will be executable in a classical manner on a single sequential execution machine, as well as on fast parallel architectures, specifically on clusters and Grids. These parallel implementations are attained by the previously explored implementation articulation through the programming language imbedding system.

4 Conclusion and the Vision of Future

In this paper we tried to give a vision of an approach towards higher human interactivity with future Grid systems by defining several steps we see necessary regarding the conception of a system able to accept problem definitions in dearticulated and deformalized, natural-like, model description languages, enabling non-technical users to start using full power of the Grids.

Our vision of the future is a society where integrated Grids allow easy programming and interactive solving of user problems in real time by execution of particular models described by the model description near natural languages on an amount of machines chosen for the most effective execution, but possibly highly distributed even for basic VIM parallel complex instructions. A common user approaches the Grid as approaching a companion, a highly developed tool, which understands a great deal of the users near-natural language communication. It is such communication between the human and the machine that enables the Grid to start being, as W. Ross Ashby termed it, an *Intelligence Amplifier*.

And finally, regarding the dearticulation and deformalization, to cite A. N. Whitehead:

"By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race."

References

1. Raymond Greenlaw, Lawrence Snyder, "Achieving Speedups for APL on an SIMD Distributed Memory Machine", International Journal of Parallel Programming, 1990.
2. Charles Antony Richard Hoare, "Communicating Sequential Processes", Prentice Hall International, 1985.
3. Kenneth E. Iverson, "Concrete Math Companion", ISI, 1995.
4. Siniša Marin, Mihajlo Ristić, Zorislav Šojat, "An Implementation of a Novel Method for Concurrent Process Control in Robot Programming", ISRAM '90, Burnaby, BC, 1990.
5. Robin Milner, "Computing in Space", CARS, 2003.
6. Karolj Skala, "e-Science", Ruđer, Vol. 3, No. 7/8, Institute Ruđer Bošković, Zagreb, 2002, pp. 11-14.
7. Karolj Skala, Zorislav Šojat; Grid for Scientific and Economic devolopment of Croatia, 4th CARNet Users Conference - CUC 2002, September 25-27, 2002, Zagreb, Croatia
8. Zorislav Šojat, "An Approach to an Active Grammar of (Non-Human) Languages", 27. Linguistisches Kolloquium, Münster, 1992.
9. Zorislav Šojat, "An Operating System Based on Device Distributed Intelligence", 1st Orwellian Symposium, Baden Baden, 1984.
10. Zorislav Šojat, "Nanoračunarstvo i prirodno distribuirani paralelizam" ('Nanocomputing and naturally distributed parallelism'), Ruđer, Vol. 3, No. 7/8, Institute Ruđer Bošković, Zagreb, 2002, pp. 20-22.
11. W. M. Waite, "Implementing Software for Non-Numeric Applications", Prentice Hall, New York, 1973.
12. Krzysztof Zielinski, ed., "Grid System Components", AGridnet Consortium, draft, December 2002.