A Symmetrical Distributed Architecture for Multicast-Based Networks

Fabiano de O. Lucchese (candidate), Marco A. A. Henriques (advisor)

Department of Computer Engineering and Automation (DCA)
Faculty of Electrical and Computer Engineering (FEEC)
State University of Campinas (Unicamp)
Caixa Postal 6101, CEP 13083-970 – Campinas, SP, Brasil

{lucchese, marco}@dca.fee.unicamp.br

Abstract – Parallel processing, as defined in the early 70s, accounts for the use of multiple networked processing units working cooperatively to solve a single computational problem faster than a mono-processed computer. Since then, cluster and grid computing solutions have emerged, leveraged by the widespread use of network infrastructures such as the Ethernet LANs and the Internet. In this article the authors present a research proposal on distributed architectures that aim at defining the principles upon which symmetrical (serverless) systems should be built.

Keywords – parallel processing, distributed architectures, distributed memory, serverless, symmetrical

1. Introduction

A major scientific breakthrough that is often considered to be the very first step toward the modeling of modern computer systems was the formalization of the Turing Machine [1]. Depicted by the English mathematician Alan Turing in 1936, the Turing Machine consisted of the apparatus by which a comprehensive set of simple instructions could be sequentially executed in order to perform a complex task. Although nowadays computer systems are capable of performing computations rather complex operations at hundreds of thousands of times per second, the computing model upon which they are built is still based on the idea of sequentially executing an ordered set of simple instructions.

Despite its great success and large-scale adoption through the past decades, this well established model has its limitations [2]. The advances on miniaturization and acceleration of semiconductor-based digital circuits made possible the implementation of high-speed and cost-effective processing units, but led the computer industry to a dead end. If in one hand the semiconductor technology brought powerful computers to everybody's desk, the physical limitations imposed by the quantum properties of the semiconductors refrains the computer industry from keeping the same pace toward faster systems [34]. The speed at which digital systems can work may not significantly exceed the 3 GHz barrier.

These limitations have long been regarded by the industry and scientific community as a claim for alternative computing models and/or extensions to the Turing model. An extension that has been successfully explored by the processors industry and that is a well-known ally of the scientific research is the use of parallelism [3][4]. By using multiple Turing-Machine-like processing units in a coordinated fashion, computer systems can reach higher degrees of performance even though they are still based on the same technological platform. Brand-new consumer computers have not less than two processing units (also known as cores when they are packed on a single chip), and expensive high-performance systems can have up to hundreds or thousands of multi-core processors.

Although parallelism has proven to be much more than a promising paradigm, its large-scale adoption suffers from the lack of common knowledge about how to develop applications that can benefit from parallel architectures. Even today, when multi-core computers are part of any company infra-structure, applications developers aren't used to think parallel; the processing/communication relation is often ignored when building high-demand applications and, as a consequence, they have to be totally redesigned by specialized experts to be able to run efficiently on parallel systems.

A number of approaches have been taken by the software development industry to ease the process of creating parallel applications. As an example, the new Microsoft Visual Studio framework has a set of tools designed to automatically parallelize "for" loops and in/out operations, distributing parts of the same program to distinct cores [5]. Code automation and high-level languages are other examples of ways to encourage parallelism [6]. Cluster computing and grid computing platforms also offer a number of tools that provide the means to work with tightly or loosely coupled networked computer systems without having to care about group communications.

In [34][35], the authors identify a set of parallel applications, techniques and mechanisms that, according to them, can be regarded as fundamental parallel computation patterns. These patterns represent the dominant operations of the parallel computing world, embracing scientific computations, such as weather prediction calculations, car crash simulations, data-base applications, Monte Carlo simulations and others. They believe that defining such canonical set of “building blocks”, capable of representing all kinds of parallel codes at multiple abstraction levels, as well as a supportive design language (OPL), might serve as the initial step towards the development of specially-suited parallel hardware, novel programming tools/languages and software libraries that will ultimately help IT professionals deal with parallelism.

In this document we propose the research and development of parallel processing technology focused on one specific class of parallel architecture: distributed memory systems. As opposed to shared-memory architectures, distributed systems are built upon the aggregation of loosely coupled independent processing units interconnected by non-specialized networks, such as common PCs interconnected by Ethernet.

2. Background

Harnessing the power of general-purpose processing devices and network infra-structure and using it to solve large-scale problems is not a new idea. Academic projects to this end, such as Linda [7] and PVM [8], exist since the early 80s and several academic and commercial projects can be found nowadays.

SETI@home [9] is perhaps the most widely known distributed system aimed at solving large-scale problems. With over 760,000 users and over 1,700,000 hosts actively participating in July/2009 -- this project showed for the first time the enormous aggregated computing power obtainable from the computers connected to Internet. SETI@home is oriented to execute a single application that searches for signs of extraterrestrial intelligence by processing signals generated by radio telescopes. SETI@home inspired a general purpose project named BOINC [10] that has been used to execute other applications such as Einsten@home, a project that searches for spinning neutron stars, and Rosetta@home, a project aimed at determining three-dimensional shapes of proteins.

Java-based systems, such as ProActive [11] and HPJava [12], leverage the platform independence and standard API of Java to provide simple, unique platforms that can be executed on most of the existing architectures. These solutions differ in the application models they execute and in the attention they pay to key issues such as scalability and fault tolerance.

This kind of distributed parallel processing has many advantages over using supercomputing hardware to solve the same problems. One obvious advantage is cost, since bundling together a set of general-purpose processors is orders of magnitude cheaper than buying a supercomputer. Furthermore, the processing power of supercomputers is not easily scalable, making it a habit for their owners to periodically change them for newer ones when the size of the problem outgrows the supercomputer's capacity. On the other side, a parallel distributed system is able to increase its power by merely adding new processing devices.

Of course, parallel distributed systems have disadvantages as well [13]. Supercomputers have specially designed hardware and custom libraries that make process intercommunication much faster than what can be achieved with a regular network. High communication speed and low latency are vital to efficiently solve some problems and for these cases supercomputers still have an edge.

However, it has been noted that large-scale commercial applications, such as those found in financial services and energy industry, tend to present a type of parallelism known as data parallelism [14]. In these cases the application executes the same algorithm over a large amount of data, with the processing of each data item being relatively independent of other data items. This class of applications is well-suited for a distributed parallel approach, since it can take advantage of a large number of relatively independent processing devices. Commercial solutions provided by companies such as DataSynapse [15], Platform Computing [16] and United Devices [17] are aimed at this class of applications.

Existing solutions for parallel distributed systems tend to exhibit asymmetrical architectures, with elements like "brokers" or "directors" coordinating the effort of the "workers", who do the actual processing. This approach has several drawbacks, such as single points of failure, which demands the use of replication and fault-tolerance techniques with varying complexity. Asymmetry itself increases the complexity of implementation and deployment, besides wasting processing power as many times the "coordinating" computers cannot participate in the actual processing [18].

Thus, it would be desirable to have a distributed parallel system oriented to data-parallel applications with a symmetric architecture, where all the elements are equal and there are no single points of failure. Such a system would be easier to deploy and maintain and be better prepared to scale to a large number of nodes.

3. Proposal

In this document, we propose as the main challenge a research on infra-structure technologies that can lead us to the modeling of a totally distributed and symmetric (serverless) architecture with no single points of failure, made up by the aggregation of network-connected heterogeneous computers. We divide this goal into 3 sub-goals that are described in section 3.3. Additionally, we consider that, prior to dealing with these sub-goals, a number of considerations should be made regarding the communication and state representation layers, described in sections 3.1 and 3.2.

3.1. Communication Issues

This proposal is strongly based on some form of group communications. In our particular case, we are aiming at distributed systems built upon non-specialized and general-purpose interconnects. Although this research may produce theoretical results that are agnostic to the interconnection medium, we have chosen to adopt the Ethernet as the target network technology. This is due to the fact that Ethernet has long been adopted as the de facto standard for interconnecting networks of workstations and, within the past 5 years, is becoming the standard for specialized high-performance computers according to the TOP 500 annual supercomputers list [19].

Symmetric and heterogeneous distributed systems based on Ethernet are commonly referred to as peer-to-peer computer systems [20]. A number of group communication frameworks designed to support peer-to-peer models have been introduced [21][22], but the adoption of such frameworks in our proposal should be based on clearly established criteria. Having this in mind, the first goal of this thesis is answering to the following questions:

- Why use group communication?
- Assuming that group communication is really necessary, which are the requirements for parallel applications execution support?
- Which are the influences of the interconnection technology over the group communication layer? Is Ethernet a suitable medium?
- How can a group communication benefit from the Ethernet structure?

Preliminary evaluations made with Java-based communication frameworks (ProActive [11] and JGroups [23]) have shown that the multicast nature of Ethernet networks is rarely taken into consideration into their design. We strongly believe that multicast-based operations, such as message exchange and file transfers, could greatly enhance such frameworks, making them more suitable for Ethernet-based parallel application executions.

3.2. State Representation Issues

Another challenge would be defining the distributed state representation of this system across the participating computers [24]. In this phase, the questions that we expect to be answered are:

- What is the best approach to represent the state of our distributed system? Is it an object model?
- Which types of entities this model should provide? Which properties should they present?
- Which are the qualities of service that should be guaranteed by this model? Which levels of performance and fault tolerance are required?

A number of experiments have already been conducted with different object models and a set of entities was designed as a first approach. However, the resulting model should be validated and, possibly, extended under the light of a more formal analysis. We expect the definition of a few entities with distinguished communication and persistence properties.

3.3. Higher Level Services

Communication and state representation are the basic tools upon which higher level distributed services are built. To support the execution of parallel applications, we need the following services:

- **Security**: a comprehensive set of tools for user authentication and authorization across the network, as well as communication and storage encryption.
- **File Staging**: making application files available where they need to be accessed is a basic requirement of a distributed system.
- **Application Execution Engine**: the execution of a parallel application involves scheduling tasks across the participating computers, detecting and circumventing failures and monitoring the system as a whole.

These are the sub-goals mentioned in the introduction of section 3. We believe that the modeling of a distributed system with the desired properties can be based on these three services.

4. Chronology

This thesis proposal represents an opportunity to analyze and extend a series of important results obtained by several research groups in the past few years. We want to submit all important past decisions to the scrutiny of formal analysis and draw conclusions concerning their validity and scope. We also expect to extend these results and propose future developments that might lead to innovative distributed services based on the same key functionality.

Having in mind that a great implementation effort has already been done, which may speed-up all field experiments, we envisage the following chronology for this proposal:

- **Communication Issues**: from 6 to 9 months (month 1 to month 6-9), including biographic revision, theoretical analysis and practical tests.
- **State Representation Issues**: from 6 to 9 months (month 7-10 to month 12-18), including biographic revision, theoretical analysis and practical tests.
- **Higher Level Services**: from 12 to 18 months (month 13-19 to month 24-36), including biographic revision, theoretical analysis and practical tests.

We expect to produce one scientific article for each of the initial subjects (communication and state representation), and one for each higher level service.

References

[1] Turing, Alan M (1936), "On Computable Numbers, With an Application to the Entscheidungsproblem," Proc. London Math. Soc. Ser. 2 42, pp. 230-265

[2] AGARWAL, V. et al, Clock rate versus IPC: the end of the road for conventional microarchitectures, Special Issue: Proceedings of the 27th annual international symposium on Computer architecture (ISCA '00), Volume 28, Issue 2 (May 2000), Pages: 248 - 259

[3] TANENBAUM, A. S. et al, Distributed Operating Systems, Prentice Hall; US ed edition (September 4, 1994)

[4] KUMAR, R. et al, Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads and Scaling, Proceedings of the 32nd annual international symposium on Computer Architecture, 2005, Pages: 408 - 419

[5] POWERS, L., Microsoft Visual Studio 2008 Unleashed, Sams; 1 edition (June 9, 2008)

[6] VAN GERMUND, A. J. C., Performance prediction of parallel processing systems: the PAMELA methodology, Proceedings of the 7th international conference on Supercomputing, 1993, Pages: 318 - 327

[7] LELER, W., Linda meets Unix, IEEE Computer Society, Feb 1990, Volume: 23, Issue: 2, Pages 43 - 54

[8] GEIST, A. et al, PVM: Parallel Virtual Machine, The MIT Press, 1994, ISBN-10:
0-262-57108-0

[9] ANDERSON, D. P. et al, SETI@home: an experiment in public-resource computing, Communications of the ACM, Volume 45, Issue 11 (November 2002), Pages: 56 - 61

[10] ANDERSON, D. P. et al, BOINC: A System for Public-Resource Computing and Storage, Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Pages: 4 - 10

[11] ProActive - Parallel, Distributed, Multi-Core Solutions with Java, http://proactive.inria.fr/

[12] CARPENTER, B. et al, HPJava: Data Parallel Extensions to Java, Concurrency Practice and Experience, 1998

[13] MULLENDER, S., Distributed Systems, Addison Wesley Publishing Company; 2 Sub edition (July 1993)

[14] DANIEL HILLIS, W., Data parallel algorithms, Communications of the ACM,

Volume 29 , Issue 12 (December 1986), Pages: 1170 - 1183

[15] DataSynapse | Leader in Dynamic Application Service Management, http://www.datasynapse.com/

[16] Clusters, Grids, Clouds - Platform Computing, http://www.platform.com/

[17] United Devices, http://www.ud.com/

[18] BUSCHMANN, F., Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing (v. 4), Wiley (May 8, 2007)

[19] TOP500 Supercomputing Site, http://www.top500.org/

[20] ORAM, A., Peer-to-Peer : Harnessing the Power of Disruptive Technologies, O'Reilly Media, Inc.; 1st edition (March 15, 2001)

[21] RIPEANU, M. Peer-to-Peer Architecture Case Study: Gnutella Network, pp.0099, First International Conference on Peer-to-Peer Computing (P2P'01), 2001

[22] JUNGINGER, M., LEE, Y, The Multi-Ring Topology — High-Performance Group Communication in Peer-to-Peer Networks, pp.49, Second International Conference on Peer-to-Peer Computing (P2P'02), 2002

[23] JGroups - The JGroups Project, http://www.jgroups.org/

[24] LIU, J. et al, Distributed state representation for tracking problems in sensor networks, Proceedings of the 3rd international symposium on Information processing in sensor networks, 2004, Pages: 234 - 242