Grid Anywhere: An Architecture for Grid Computing Able to Explore the Computational Resources of the Set-Top Boxes

Fabiano Costa Teixeira, Marcos José Santana, Regina Helena Carlucci Santana, and Julio Cezar Estrella

Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, Brazil
(teixeira,mjs,rcs,jcezar}@icmc.usp.br

Abstract. This paper shows a grid computing architecture called Grid Anywhere which aims at allowing the sharing of the computational resources from set-top boxes (receiver) of an Interactive Digital Television System. This sharing is made in a such a way that the TV broadcaster can use the set-top boxes tuned to it to get a high computational power and the user can also use remote resource of others devices (set-top boxes or personal computers) to increase the performance of the application executed in his receiver.

Keywords: Interactive Digital Television, Grid Computing, Peer-to-Peer networks.

1 Introduction

Brazil is a country which has a large number of TV devices, almost 54 millions [1]. In February 2, 2007 started in Brazil the transmissions of digital television signals. The reception of this signal and the audio/video reproduction are made by a device called set-top box.

The set-top box, in addition of the function cited in the previous paragraph, is able to execute applications which can be sent by the television signal broadcaster together with the audio and video streams. So, this device has computational resources, including cpu, memory and hard disk.

In the next years Brazil will have approximately 80 millions of digital receivers (set-top box) [11], then can be interesting the possibility of sharing computational resources hosted by this devices. This share can be made to offer a high performance execution environment. In addition, the user will be able to use remote resources hosted in other receivers or conventional personal computers to increase its computational power to execute applications. In this context this paper presents a proposal of a grid computing architecture that has as participants the conventional personal computers, interactive digital television signal broadcasters and set-top boxes. This document is organized as follows: the sections 2 and 3 show, respectively, a short introduction about Interactive Digital Television and grid computing paradigm. The section 4 shows the proposal of the architecture. The section 5 presents some object migration issues and section 6 shows a performance evaluation of object transfers using Java Sockets. Finally, the section 7 concludes the paper and presents some future works.

T. Doulamis et al.: (Eds.): GridNets 2009, LNICST 25, pp. 79–88, 2010.

2 Interactive Digital Television

In a digital television system many of the components are different from those found in an analogical one. Changes are found at the program production, transmission and reception of digital signal [2]. In figure 1, the production is made using a digital camera and the images are sent to a MPEG encoder which has the function of making the compression of data to be sent to the viewers (users). The data compression allows that a unique channel that today (in an analogical system) is able to transmit only one TV program can be used to transmit more than one simultaneously, depending of the video quality adopted.

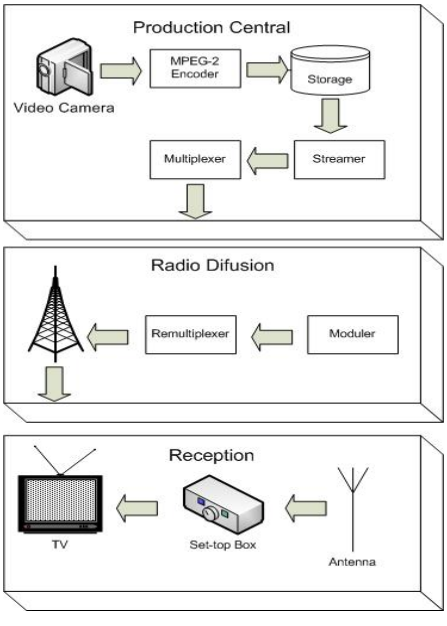

Fig. 1. Digital Television System Architecture. Based on [2].

The MPEG-2 Systems [3], technology used in the transport layer of the digital television system, allows that the audio, video and data streams can be multiplexed in a unique transport stream that is sent using the radio diffusion. An antenna has the function of receiving the signal propagated via broadcast and forward it to the set-top box.

The set-top box, device with computational resources (cpu, memory, operation system), in addition of its main functions (reception and decoding of the signal, reproduction of audio and video) is able to execute applications. Since a data stream can be transmitted, it can be used to transport an application (java application, for example) sent by the television broadcaster. So, this application can be received by the viewer's set-top box and executed.

Since the sent of the application is made via broadcast, when the set-top box is tuned in a channel the data stream transmission could be in curse. For audio and video

streams it is not a problem. However, when an application has been transmitted, the receiver needs to get the whole file. The Data Carousel address this question: the data are divided in modules that are transmitted cyclically. So, if the initial modules have already been transmitted, the receiver waits for the retransmission of them [15].

The set-top boxes can be manufactured by various companies that, potentially can adopt different hardware and software solutions. So, in order to the same application sent by a television broadcaster could be executed in different types of receivers, as showed in figure 2, there is a software layer called middleware. The middleware has the function of providing transparency of hardware and software to the application. This transparency is achieved offering a generic API (application program interface).

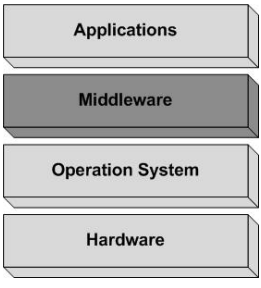

Fig. 2. Set-top Box's Layer Model. Based on [2].

In order to an application executed in the viewer's set-top box be able to send data to the television broadcaster or another entity, it is necessary a communication channel that has the function of allowing set-top box to access an IP network (Internet). This communication channel can be implemented using a lot of technologies like WiMax, ADSL, Cable [12].

3 Grid Computing

In 90's middle a new computing paradigm has been proposed to allow that computational resources geographically distributed and under independent administrations could be shared [4][5].

The resources to be shared can be since processors and disks to software licenses.

The possibility of sharing resources among users geographically distributed allows the building of a very powerful computational system, which can contribute in a significant way in the solution of problems, without using new hardware devices.

Institutions and individuals can be organized with the intention of sharing their resources. This group that determines a set of sharing rules and polices are called Virtual Organization (VO) [6].

Some projects of philanthropic computation have been using resources offered by voluntary users to contribute with the processing of larges information sets. Boinc Project is a framework that allows the building of this kind of application [7]. It hosts projects like SETI@home and Rosseta@home that work with processing of radio

signal to look for intelligent extraterrestrial life and analyze of amino acid chains in the proteins formation, respectively.

Building a computational grid involves a lot of requirements. In order to abstract these requirements, the utilization of a middleware is very interesting. In this context, Globus Toolkit [8], an open source tool, was presented in 1998. Is gives a set of programs, services and libraries that aim the construction of a grid computing providing mechanisms for security, resource management, application submission and data transfer.

A Brazilian product called OURGRID [9] also gives facilities to build a grid computing. However, the system developed uses peer-to-peer architecture and the tasks are submitted using a BoT (Bag of Tasks) approach. So, the applications can't communicate with each other.

A middleware called DGET (Data Grid Environment and Tools) [10], using Java objects migration, allows the construction of a grid computing.

The work proposed in [11] describes a grid computing architecture called TVGrid, which uses the idle computational resources found in a set-top box. This approach is based on the transmission of the application via broadcast to be executed in the viewer receiver. These applications are executed and the results are sent back to the broadcaster.

4 Grid Anywhere Architecture

Nowadays, Brazil has a large number of TV devices, approximately 54 millions [1] and potentially in 2017 the country will have almost 80 millions of set-top boxes [11].

Since each set-top box is composed by computational resources, in a near future a big computational park will be found in Brazil and lots of these resources could be idle in some periods of time.

The architecture proposed in this paper aims the construction of a grid computing using conventional personal computers and set-top boxes. However, the proposal has a bidirectional approach where a set-top box could act in two roles: resource provider and resource consumer.

When a receiver acts as a resource provider, a complex application could be divided in short parts to be sent, by the television broadcaster, in order to be executed by the set-top boxes. When the processing finish, the results are sent back via a communication channel. Acting as a resource consumer, a limited set-top box can use idle computational resources found in others receivers or conventional personal computers to increase its computational power to execute applications.

To build this grid computing, peer-to-peer architecture has been used, supposing that the receivers used in Brazil will have a communication channel that alows a direct communication between the set-top boxes. So, as presented in figure 3, each set-top box (TV Peer), personal computer (PC Peer) and TV broadcaster (Broadcaster Peer) represents a node of this network.

The resource sharing is made using Java object migration between the participants. Therefore, an object can be exported to another peer, its methods can be remotely invoked and the result or the object come back to the original peer.

The PC Peer is a participant that can either contact or be contacted by any other participant under a peer-to-peer connection to migrate the Java objects to be executed.

The Broadcaster Peer represents a high computational power entity, since it can receive a Java object and send it via broadcast to all set-top boxes tuned to it.

Finally, the TV Peer is the participant that has all PC Peer's characteristics. However, it is able to receive Java Objects sent via broadcast by Broadcaster Peer.

Since the viewers need to share their set-top box, this proposal hopes that will be some business model that incentives them to enable the set-top boxes to be used to execute applications sent by the broadcaster.

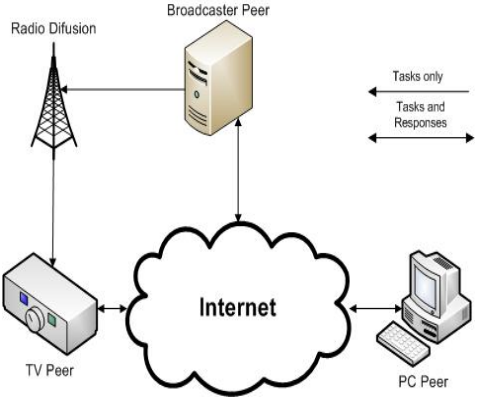

Fig. 3. Grid computing architecture

4.1 Middleware

This proposal includes a middleware architecture that is instaled in the peer, which can be either a PC, a TV Broadcaster or a set-top box.

The Grid Anywhere Middleware aims at offering to the application programmer an environment to build grid applications in a transparent way. Since the grid is based on Java objects migration and remote method invocation, a programming model where the programmer defines the classes that can be accessed remotely has been adopted. The middleware, using code instrumentation [13], automatically manages the migration and remote method invocation. So, the user doesn't needs to worry about the details of the process.

As showed in figure 4, the middleware has a core API that can be invoked in two ways in order to get advanced information about the grid and to take actions: explicitly by expert programmers or by instrumented java code.

The *scheduler,* when required, analyzes the grid environment to find the best peer to send a serialized object to be hosted there and executed. The programmer needs to define if the object requests either a single host (so, it can be sent to a PC Peer or a TV Peer in a unicast mode) or a high performance host. The second case is indicated when there is a large data set that can be processed by a same algorithm. So, the object which implements that algorithm can be sent to a Broadcaster Peer which sends

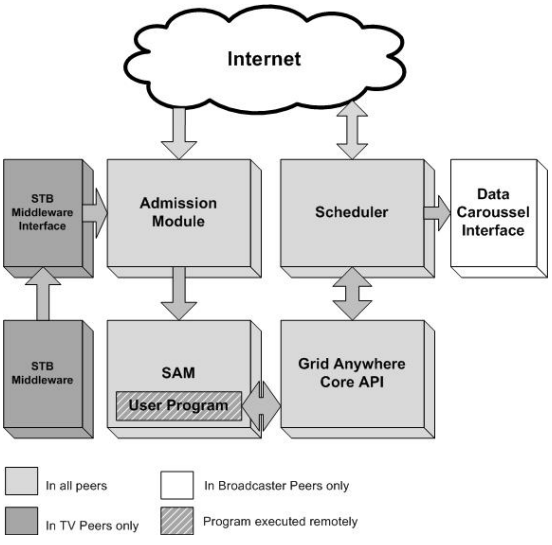

Fig. 4. Grid Anywhere middleware architecture

this object to every set-top box tuned to it. The set-top box receives the object and executes it. The Java program executed can use conventional communications interfaces (like sockets) or the middleware API to get the data to be processed. When the processing has been done, the results are sent back to the original peer.

The *admission control* module is used when the peer is in the role of resource provider. It negotiates the execution with the client *scheduler* and receives a Java Object to be hosted in the local peer. Since the object has been received, in order to give security guarantees to the resource provider, the Java Object is sent to be executed in the SAM module (a sand box responsible to execute the applications in a secure way).

When the resource provider is a TV Peer, the reception of Java Objects sent by the Broadcaster is made by the set-top box middleware. So, there is a *set-top box middleware interface* module that is responsible to get the received object and forward it to the admission module. In this first work, an interface to Ginga (the Brazilian Middleware) will be implemented. In order to integrate the Grid Anywhere to another Interactive Digital Television System, it is possible to implement the set-top box middleware interface, what makes the middleware flexible.

In situations where there is no set-top box middleware, it is necessary to implement programs to manage the incoming data carousel. In future work, this module will be implemented to give "independency" to the Grid Anywhere middleware.

The Broadcaster Peer uses the Data Carousel Interface to forward the Java Object to the Broadcaster's application responsible for multiplexing data with audio and video to be sent to the viewer receivers (set-top boxes).

4.2 Security

The security is a very important issue in the Grid Environment implemented by the proposal presented in this paper. The peers execute programs in most times

implemented by unknown programmers. So, the program execution in an insecurity mode can be very dangerous to the local system.

The SAM is a sand-box responsible to execute Java program in a secure way. To do it the user program is executed over a *Java Security Manager* and an *Environment Monitor*, and both over a *Police System*, as show in figure 5.

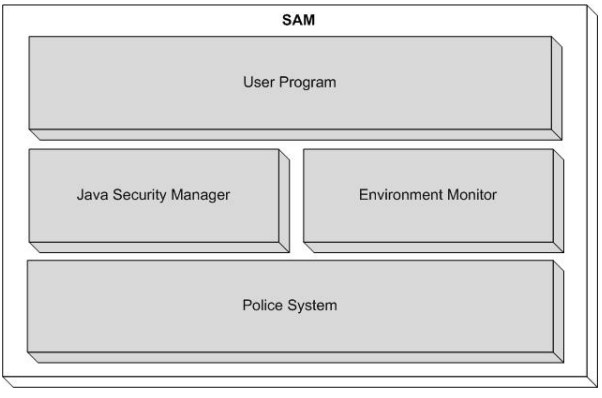

Fig. 5. SAM layered model architecture

The *Police System* is a set of rules defined by the user, which defines what a program can do(open files, sockets) and the amount of resources that can be used (memory, cpu, bandwidth). The Java Security Manager and the Environment Monitor are responsible to make these controls respectively.

The Java Security Manager used in this architecture is an override of the original one distributed with JRE. Instead of using the Access Controller [14] to define the access rules, the SAM implementation uses an own Police System to enable that more flexible rules can be defined.

Middleware users can define the rules based in the concept of user groups, subgroups and users. In a Grid Computing environment, the group entity can be mapped to a Virtual Organization (VO). So, in a trusted VO the access rights can be more than in an untrusted one. In addition, other rules types can be established, giving a high flexibility to define the security polices.

Since the middleware aims at giving an easier tool which could be used by users without experience in computing, the interfaces used to configure it need to be very simple. In a Interactive Digital Television System context, the user need to be able to configure the polices using the TV's remote control. So, the SAM has a friendly and intuitive graphical tool to define the security rules.

5 Object Migration

Since our approach aims at increasing the processing throughput and decrease the response time (when an interactive application is used), the Grid Anywhere middleware is based on Java objects migration. Every time that a grid peer needs to get more computational power to execute its application, some objects are serialized and sent to

another peer (which rebuild the object). A XML document is used to describe the method invocation (method name and parameters values). When the processing is done three actions are possible: a) the value returned from the method is sent to the caller; b) if a *void* method is called, the object still at the destination host with a possible state change; c) the object is serialized again and returned to the caller. Figure 6 shows a sequence diagram that presents the message pattern exchange (MEP) of Grid Anywhere.

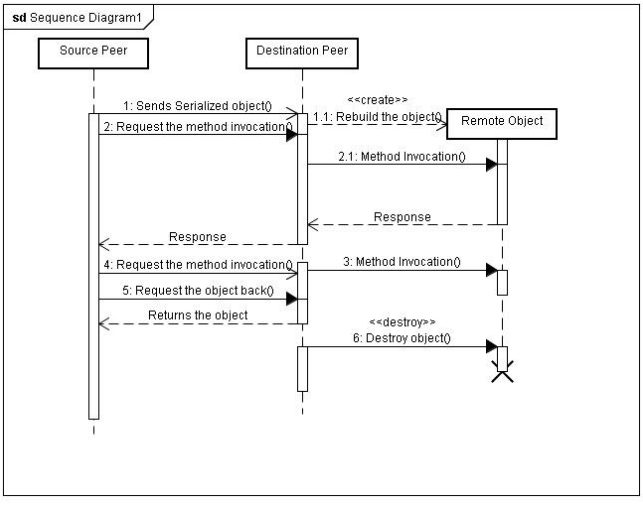

Fig. 6. Message pattern exchange for Grid Anywhere

The sequence 1 explains a synchronous invocation, whereas sequence 4 shows an asynchronous invocation (without response). Finally, the sequence 5 presents a process to request the object back.

To do the object transfer our middleware uses Java Sockets. It is important to consider the object's size before migration because of the transfer overhead. The scheduler could make a time prediction that will be spent to migrate an object and take the decision of either move the object (and to decide the destination peer) or keep it locally. In order to verify if the time prediction is possible, a performance evaluation has done and presented in the next section.

6 Test Environment

In order to verify if scheduler could make a transfer's overhead prediction, an initial experiment has been done. Using two computers geographically separated and two cable modems, data sets with different sizes (from 1KB to 20KB) was transmitted between the machines. The same data set was transmitted ten times using the API Java Sockets and the time spent was stored in order to calculate the time average, standard deviation and confidence interval for each data size.

Figure 7 shows the transfer behaviors when a cable modem has been used.

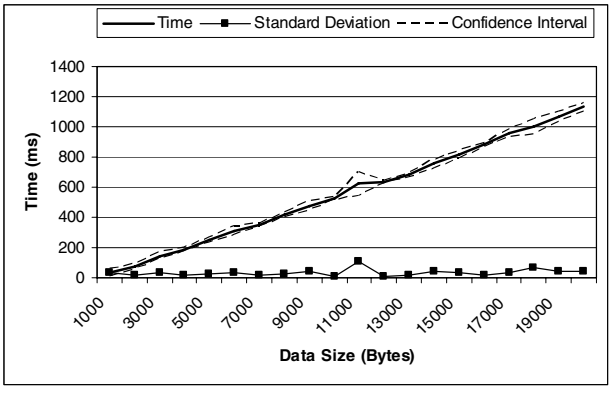

Fig. 7. Behavior of transfers using cable

When the cable technology was used the transfer's time has increased in a linear way (in function of object's size). With a discrete standard deviation and a short confidential interval, a mathematical function could be implemented in order to compute the transfer's time for a given Java object. Therefore, time predictions could be done, in some cases, with a minimal insurance. However, other link technologies could present different behaviors.

A mechanism to determine, in execution time, if a transfer's overhead prediction could be done in function of the data link used by the peer is been created.

7 Conclusion

In Brazil there is a large number of TV devices and the popularization of the internet could create a big computational park composed by resources of the set-top boxes.

Grid Anywhere, the architecture proposed in this paper, enables the construction of a grid computing able to use the idle resources of the set-top boxes in the processing of complex applications. This can be very useful to the general engineering and science since a high computational power could be found in a architecture with a high number of set-top boxes using the middleware.

Since the Grid Anywhere also allows that the set-top box uses the remote idle resources found in other receivers or personal computers to increase its computational power, the architecture can contribute to the digital inclusion in Brazil enabling that viewers with limited financial resources could execute more complex application.

Actually, a prototype of the security and object migration modules has been implemented. In future work the interfaces with the set-top box middleware and data carousel generator will be constructed in order to allow a performance evaluation of the object migration in a Interactive Digital Television System.

References

1. Zuffo, M.K.: TV Digital Aberta no Brasil – Políticas Estruturais para um Modelo Nacional. Escola Politécnica, USP, São Paulo (2003)
2. Fernandes, J., Lemos, G., Silveira, G.: Introdução à Televisão Digital Interativa: Arquitetura, Protocolos, Padrões e Práticas. Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação (2004)
3. Sarginson, P.A.: MPEG-2: A Tutorial Introduction to the System Layer. In: IEE Colloquium on MPEG-2: What it is and what it isn't (1995)
4. Foster, I.: The Grid: A New Infrastructure for 21st Century Science. Physics Today 55, 42–47 (2002)
5. Teixeira, F.C., Toledo, M.B.F.: Arquitetura de Grade Computacional Baseada em Modelos Econômicos. I2TS, Cuiabá, Brasil (2006)
6. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications 15, 200–222 (2001)
7. Boinc Home Page. Disponível em, `http://boinc.berkeley.edu`
8. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. In: Jin, H., Reed, D., Jiang, W. (eds.) NPC 2005. LNCS, vol. 3779, pp. 2–13. Springer, Heidelberg (2005)
9. Andrade, N., Costa, L., Germóglio, G., Cirne, W.: Peer-to-Peer grid computing with the OurGrid Community. SBRC, Fortaleza-CE (2005)
10. Hudzia, B., Ellahi, T.N., McDermott, L., Kechadi, T.: A Java Based Architecture of P2P-Grid Middleware, Arxiv preprint cs.DC/0608113 (2006)
11. Batista, C.E.C.F., Araujo, T.M.U., Omaia, D., Anjos, T.C., Castro, G.M.L., Brasileiro, F.V., Souza Filho, G.L.: TVGrid: A Grid Architecture to use the idle resources on a Digital TV network. In: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, pp. 823–828 (2007)
12. Farias, M.C.Q., Carvalho, M.M., Alencar, M.S.: Digital Television Broadcasting in Brazil. IEEE Multimedia 15(2), 64–70 (2008)
13. Chiba, S., Nishizawa, M.: An Easy-to-Use Toolkit for Efficient Java Bytecode Translators. In: Pfenning, F., Smaragdakis, Y. (eds.) GPCE 2003. LNCS, vol. 2830, pp. 364–376. Springer, Heidelberg (2003)
14. Java Security Architecture, `http://java.sun.com/j2se/1.4.2/docs/guide/security/spec/security-specTOC.fm.html`
15. Herpel, C.: Elementary Stream Management in MPEG-4. IEEE Transactions on Circuits and Systems for Video Technology 9, 315–324 (2002)