2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies

Dynamic Creation of BSP/CGM Clusters on Cloud Computing Platforms

Alessandro Kraemer*, Junior Cesar de Oliveira†, Fabio André Garaluz dos Santos†,
Ana Claudia Maciel†, Alfredo Goldman‡ and Daniel Cordeiro‡
*Department of Computer Science, Technological University of Paraná, UTFPR
Campo Mourão, Brazil
Email: kraemer@utfpr.edu.br
†Department of Internet System Technology, Technological University of Paraná, UTFPR
Campo Mourão, Brazil
Email: {juniorcesar.utfpr,fgaraluz,anacm.maciel}@gmail.com
‡Department of Computer Science, University of São Paulo
São Paulo, Brazil
Email: {goldman,danielc}@ime.usp.br

***Abstract*—Cloud computing platforms have the potential to benefit scientific projects on all fields of knowledge. Its virtualized resources and large storage capacity enable any scientist to have access to high performance computing platforms at low costs. In this paper, we present the MyCloud project, a cloud computing infrastructure for the Technological University of Paraná campuses, in southern Brazil. We show how to build and instantiate virtual machines templates for BSP/CGM applications that can be used on private cloud computing platforms.**

***Keywords*—Bulk Synchronous Parallel; Coarse Grained Multicomputer; Virtual Machines; Cloud Computing.**

I. Introduction

The demand for computational power is increasing. Scientists from different fields of knowledge (like Computer Science, Biology, Engineering, etc.) now relies on high performance computing to conduct their experiments.

Any scientist can now have access to computational power equivalent to that of supercomputers at low costs, using public cloud computing platforms like Amazon EC2 [1]. Despite that, there is no easy way for the developers to set up a cloud computing platform for their work. It is up to the developer to choose the best parallel or distributed programming model and how to deploy their programs on cloud computing platforms.

The MyCloud project, being developed at the Technological University of Paraná (UTFPR), aims to provide a cloud computing platform and the its respective support tools to allow scientists from all areas to use cloud computing technology. This basic support is built using the Bulk Synchronous Parallel (BSP [2]), the Coarse Grained Multicomputer (CGM [3]) programming models, and the underlying virtualization technologies that guarantees an optimized use of the available computational resources.

On cloud computing platforms, customized virtual machine instances are created and executed on clusters controlled by some kind of virtual machine manager module. It is up to the developers to set up the images, to choose how many instances of them will be executed and how they interact. On a university environment, with developers from different areas of knowledge, a question arises:

- How to simplify the use of cloud computing platforms by non-experts?

We believe that our users do not need to have technical knowledge about how to build a cluster on cloud platforms. In order to simplify this process, we are developing a new framework (based on open-sourced software), that is based on the idea of having templates of virtual machines that are optimized for a standard distributed programming model.

The remaining of this paper is organized as follows. Section II presents the UTFPR scientific context on which this project is being developed. Existing open source frameworks for build private cloud platforms are discussed on Section III and the chose programming model (BSP/CGM) is presented on Section IV. On Section V we present our solution for the dynamic creation of test beds, based on the use of virtual machines templates. We also show some preliminary tests to validate the solution and to verify the scalability of this solution when using BSP algorithms. Finally, we present how this solution fits the UTFPR needs and a discussion of the results on Section VI.

II. Scientific Context

The UTFPR has 12 campuses and 223 academic research groups addressing scientific projects in CFD (Computational Fluid Dynamics), Bioinformatics (pattern analysis), Medical Informatics (DNA analysis and data mining), 3D simulation Numerical analysis, Electronic Structure and Chemistry. Several algorithms are used as proof-of-concept in these fields of knowledge. The MyCloud project goal is to share the computational infrastructure available on each campus, in order to create a big cloud computing platform, and to include supporting tools to execute scientific applications. The first step of the MyCloud project is the deployment of the software and hardware infrastructure in one campus (Campo Mourão).

The authors would like to thank Fundação Araucária (#345/2012) and FAPESP (#2012/03778-0) for their financial support.

978-0-7695-5044-2/13 $26.00 © 2013 IEEE
DOI 10.1109/EIDWT.2013.140

Table I. CLOUD COMPUTING FRAMEWORK FEATURES

Framework	Open Source	User interface	Hypervisors
Amazon Web Services [1]	no	AWS Management Console, AWS APIs and Command Line Interface (CLI)	XEN
Apache CloudStack [7]	yes	CloudStack API, AWS APIs, CLI, Web Interface	XEN, KVM, QEMU, VMware and Hyper-V
Eucalyptus [8]	yes	AWS APIs, CLI, Web Interface	XEN, KVM, QEMU, VMware and Hyper-V
OpenNebula [9]	yes	AWS APIs, Open Cloud Computing Interface (OCCI), CLI, Sunstone Web Interface	XEN, KVM, QEMU, VMware and Hyper-V
OpenStack [10]	yes	AWS APIs, CLI, OpenStack Dashboard Web Interface	XEN, KVM, QEMU, PowerVM, Hyper-V, LXC and VMware
VMWare vCloud [11]	no	vCloud API, vCloud Director and vCloud Connector	VMWare

Then each campus will be incrementally integrated to the cloud platform. The distributed infrastructure will improve the service availability and will expand the computational resources available to the scientists.

The first parallel and distributed programming language for cloud supported by the project is the BSP/CGM model, using the Messaging Passing Interface (MPI) as the interprocess communication model. Evidently any other programming model can be used on the platform, notably the well-known MapReduce model [4], that is commonly used on "big data" problems. Starting with support to BSP/CGM is a sensible choice, not only because these models are widely know among our Computer Science teachers and students, but also because our preliminary studies [5] on the suitability of the BSP/CGM model for cloud computing suggests that this model scales well on such platforms. The BSP/CGM model is explained in details on Section IV.

III. CLOUD COMPUTING FRAMEWORKS

A cloud computing framework is made of several components [6], each of which deals with different aspects of the virtual machine execution on the available physical machines. For instance, cloud computing frameworks usually have components to control the user interfacing, the deployment of virtual machine images into cloud nodes, load balancing, the virtual machine monitoring (managed by the *hypervisor*), etc.

We have evaluated some of the available cloud computing frameworks regarding features like the availability of the source code, the user interface exposed to the developers and the support for different hypervisors. The results are summarized on Table I.

The cloud platform can be exposed to the end users using different models. The most basic service model is the Infrastructure as a Service (IaaS) model, where the virtual machines instances can be acquired and released on demand by the users (the request can be made using different user interfaces such as an API, a command line interface, etc.). Each time a user requests a new virtual machine, the platform chooses a physical node (that can host multiple virtual machines) and then the hypervisor instantiates and execute the virtual machine image. It is up to the developer to configure the multiple virtual machines being used (from setting up the SO to installing and configuring the program to be executed). This setting can be assisted by software managers, such as Chef [12] and Puppet [13], but the process is not trivial and more error prone.

Another model to expose the cloud computing platform is known as Platform as a Service (PaaS), where the developer writes its program to a specific framework provided by the cloud computing provider. Amazon, for example, exposes the Apache Hadoop framework [14] through the Elastic MapReduce (EMR) API. The developer just configure the cluster configuration (e.g., the maximum number of nodes allowed) and uploads the MapReduce application. The entire execution is controlled and managed automatically by the EMR framework, from the configuration of the virtual images, to the monitoring and execution of the MapReduce program itself.

Our solution lies somewhere in between these two models. We provide the flexibility of the IaaS model — enabling the developer to customize the operating systems and libraries to its specific needs — and some of the facilities provided by the PaaS model, by supplying specialized virtual machine templates, optimized for some distributed programming model. This way, a developer can simply choose a template optimized for its programming model of choice (e.g. the BSP/CGM programming model with MPI communications) and the framework dynamically instantiates and starts the virtual machines for the user, setting up all the environment configurations needed by the software stack (e.g. configuring the IP addresses in the MPI stack, the BSPlib [15] and the CGMlib [16]). In our implementation, we provide a dashboard web interface, where the users can configure the size of the required cluster and the SSH access the acquired machines.

IV. THE BSP AND CGM MODELS

The Bulk Synchronous Parallel (BSP) model was first introduced by Valiant [2]. The computation is organized as a sequence of *supersteps*. Each superstep is composed of multiple parallel computational phases where each processor (on all nodes) only performs local computations, followed by a communication phase, where the communications of data involve all the processors in a personalized all-to-all communication. Between two supersteps there is a synchronization barrier, which guarantees the simultaneous start of all processors for the next step.

Having global synchronizations at the end of each superstep may be seen as cumbersome, but there are significant advantages of doing so, especially for large scale distributed systems. First of all, to find a good way to overlap communication and computation, a precise knowledge of the hardware is essential. This means that a fine tuning of the application must be redone whenever possible in the hardware. Secondly, global synchronization points can be used straightforwardly to provide checkpoints to store consistent partial states. Thirdly,

there are several works exploring how the h-relation can be effectively done, providing efficient implementations on a large diversity of scenarios [17]. An h-relation represents the sending and the receiving of h amounts of data in each processor [18] at each communication step.

Dehne *et al.* [3] introduced a slightly different model for parallel programs that, in fact, can be considered as a special case of the BSP model. The Coarse Grained Multicomputer (CGM) model defines how a problem of size n will be executed by p processors, each with a local memory limited to $O(n/p)$. Like in BSP, a CGM algorithm consists of a sequence of rounds with distinct phases for local computation and global communication. Usually, each processor will apply the best sequential algorithm for the data available locally. The goal when designing CGM algorithms is to minimize the number of h-relations executed at the communication [19] phase.

Several implementations of the BSP model have been developed since the initial proposal by Valiant. They provide to users full control over communication and synchronization in their applications. A comparison of several libraries was conducted by Krusche [20]. On the CGM side, we can note the implementation provided in the CGMlib [21], among others.

Works using BSP as a programming model for cloud computing first appeared on the context of large graph processing applications. These applications naturally appear on problems from different areas of knowledge (Computer Science, Biology, Engineering, etc.) and use graphs to represent relations among different entities, such as links on web pages, user relations on social networks, flow of goods, order of DNA fragments, etc. Goldman *et al.* [5] have analyzed the suitability and scalability of BSP algorithms on cloud computing platforms.

Many algorithms have been developed using BSP or BSP/CGM. These include: Prefix Sum, Partition and Parallel Sorting [21]; Next Element Search [22]; Longest Repeated Suffix Ending at Each Point in a Word [23]; Two-Dimensional Parallel Pattern Matching [24]; Graph Planarity Testing [25]; Maximum Weight Matching in Trees [26]; Dynamic Programming for Solving the String Editing Problem [27]; All-Substrings Longest Common Subsequence [28]; Maximal Independent Set Problem and P-Quantiles [29]; Knapsack [30]; Maximum Matching in Convex Bipartite Graphs [31]; Matrix Chain Product [32]; Transitive Closure [33]; Biological Sequence Comparison [34]; Approximation Hitting set Algorithm for Gene Expression Analysis [35]; Integer Sorting [36]; List Ranking, Euler Tour, Connected Components, Spanning Tree, Lowest Common Ancestor, Open Ear Decomposition and Biconnected Components, and Chordal Graph Recognition [37].

V. How to Simplify Development on Clouds Using Virtual Machine Templates?

The main goal of this work is to simplify the use of cloud computing platforms for scientists of all areas. These scientists have increasingly computational needs — so they must have the option to customize the computational platform in order to optimize it for their needs — while, at the same time, should not have to deal with all the technicalities related to building and configuring clusters of nodes on cloud platforms.

Our proposal is based on the usage of virtual machines templates optimized for a particular distributed programming

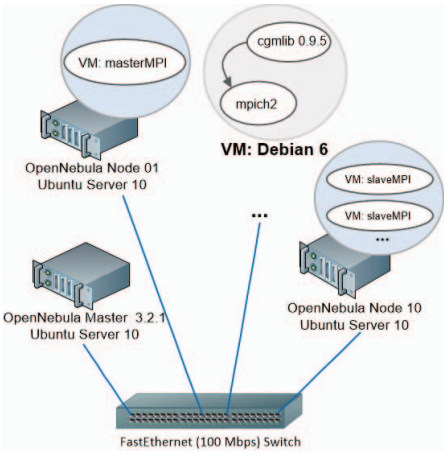


Figure 1. Virtual machine instances inside the OpenNebula framework.

model. Although optimized for a specific model, the virtual machine is completely exposed to the users and can be tuned and customized for particular needs. This enables the creation of high performance computing platforms with little effort.

We have implemented a prototype framework, based on open-sourced software, that enables the dynamic creation and configuration of a cluster of nodes inside a cloud computing platform that is well suited for the efficient execution of BSP/CGM applications using the MPI protocol.

MPI applications modeled using BSP/CGM have often a master/worker architecture, where a master defines which part of the computation each worker will execute. They also generally use some kind of distributed file system to distribute the data that must be processed by the application and to collect the results. We leverage this common architecture in our prototype implementation to automatically assign a master server to be responsible for controlling the distributed file system and the distributions of controlling of the IP addresses for the workers. Our implementation relies on the NFS [38] distributed file system, a DNS [39] and a DHCP [40] server on the master and DHCP clients on the workers.

The first and only manual step is to upload the application to the master node using SSH. Then, in order to execute a MPI application, each worker must be able to gather the following information: (i) the IP address of the NFS server; (ii) the application to be executed; (iii) the list of all IP addresses of all nodes belonging to the cluster being deployed (MPI requires this to execute all-to-all communications). Using our proposed architecture, (i) can be easily discovered by the system using the DHCP protocol, that not only configures the IP of each worker, but also inform which node is the DNS server and, therefore, the NFS server. When the worker node mounts the NFS file system, it automatically have access to the application to be executed (ii) and also informs the master of its IP address. This list of all IP address can be straightforwardly obtained from the master using the NFS command `showmount`, which solves the last problem. Once the list of IPs is obtained, the workers can configure the MPI framework "hostfile" and start the execution of the application.

We have implemented our prototype using the Oracle

VM VirtualBox[1] virtualization software package. Each virtual image have installed a Debian 6.0 GNU/Linux distribution. In order to execute MPI applications, we have also the portable and high-performance MPICH2[2] implementation of the MPI standard. The BSP/CGM applications have at their disposal the BSPlib [41] and the Cgmlib [21] version 0.9.5.

The virtual images are controlled by the OpenNebula [9], an IaaS (Infrastructure as a Service) framework. The final user can control its virtual image instances using the web dashboard based on the OpenNebula Sunstone interface.

A. Experimental evaluation

In this section we present some experiments using an OpenNebula test bed and BSP/CGM virtual machine templates. All nodes uses Qemu KVM hypervisor and are managed by the OpenNebula framework, as shown on Figure 1. The figure also depicts the physical infrastructure utilized in our experiments. Each computer node used in the experiment has 1 Core Duo E6750 2.66 GHz, 2 GB of RAM, a 230 GB HD SATA II and NIC FastEthernet (100 Mbps).

Scalability is an important requirement for scientific algorithms. The use of virtualization by the cloud platforms can result on bigger network latencies and some processing overhead. The goal of our experimental evaluation is to validate our proposed architecture and templating schema and also to study the scalability of BSP/CGM algorithms on the MyCloud cloud platform.

We have executed our experiments using two BSP/CGM applications described by Chan *et al.* [21]: *Partition* and *Parallel Sorting*. These algorithms are relevant because they are part of the solution of several scientific applications. Figures 2a and 2b shows our results with the Parallel Sorting and Partition algorithms, respectively. Each figure presents the average execution and communication times obtained over the increase on the number of virtual machines deployed. Each result is presented with a confidence interval of 95%.

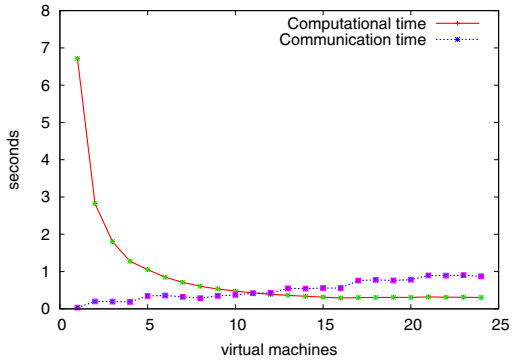
The MPI framework treats each virtual machine as an available processor. Thus, our experiments triggers the execution of up to 24 workers distributed among eight physical machines (three virtual machines instantiated on each cloud node). The time spent by the programs were measured using the `CGMTimers` class, part of CGMlib library [16].

The results suggest that the computational and communication time of the algorithms evaluated scale well with the increase on the number of virtual machines. They also show that for a higher number of virtual machines, the time spent by communication have a high impact on the algorithms performance. This increase in the communication times can be partially explained by the mechanism used to virtualize the network (the virtual network).

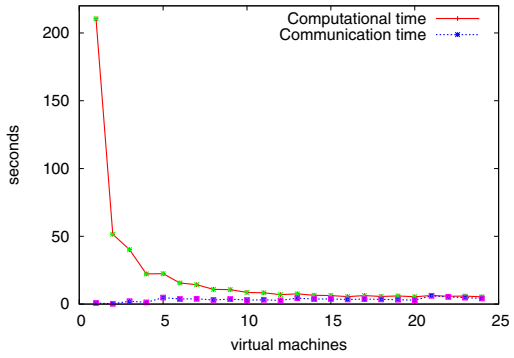
Figure 3 depicts the virtual networks are build over the real physical NICs. The network communication are managed by the virtual network (that can be used in bridge or switch mode). The virtual network optimizes communication between different virtual machines by bypassing the real NIC when only

[1]https://www.virtualbox.org/

[2]http://www.mpich.org/



(a) Parallel Sorting algorithm (n=400,000)



(b) Partition algorithm (n=9,999,999)

Figure 2. Evaluation of Partition and ParallelSorting algorithms [21].

intra-node communication is performed. In our experiments, the virtual NIC was managed by the Virtio [42] driver, a device accelerator used by the Qemu KVM hypervisor.

Our results corroborate the results obtained by other works, like the one from Jamal *et al.* [43], that also analyzed the scalability of virtual machines on multi-core systems. They have also found that the scalability on virtualized platforms suffers from the overhead caused by communication. Other approaches to ensure efficient execution on cloud computing platforms include QoS guarantees for the virtual machines [44], [45] and clever scheduling heuristics designed to assign virtual machines to physical machines based on characteristics like the multi-core cache structure [46] or the analysis of the I/O and CPU bounds of the applications [47].

VI. Conclusions

In this paper we presented the MyCloud project and our approach to address different needs from scientists using an easy-to-use virtual machine templating scheme. We presented our prototype implementation, based on open-sourced software, that relies on OpenNebula and on Oracle VM VirtualBox to instantiate and manage the virtual images on a cluster of computing nodes. We presented and evaluated or templating schema using virtual machines templates optimized for applications using the BSP/CGM programming model. Our preliminary results showed that BSP/CGM algorithms can scale well on the MyCloud platform, and that the performance of these applications are bounded by the overhead caused

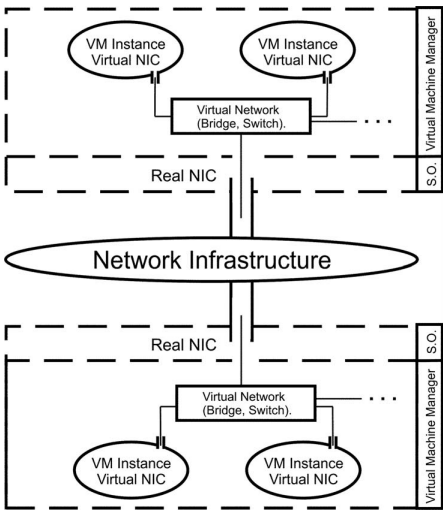


Figure 3. The virtual network infrastructure.

by inter- and intra-node communication when the number of virtual machines increases.

A. Future Work

As future work, we intend to investigate more implementation details that could improve the performance of the virtual machines on our platform (like the use of the Virtio network device accelerator, that was essential to guarantee the performance of our experiments). We would like also to investigate how to improve the communication performance of BSP/CGM algorithms, and use this particular programming model to create a fault-tolerant system.

For the final users (scientists at UTFPR), we intend first to increase the number of available templates optimized for other programming models (like, for instance, the MapReduce model). We want also to polish the dashboard web interface, that are now dependent on the OpenNebula Sunstone interface, in order to assist non-expert users to configure the number of required computational resources and ease the upload of the applications to the cloud platform. We also plan to implant a parallel programming education project to graduate students to help scientists to make a better use of the platform. When all UTFPR campuses finishes the integration of their computational resources to the platform, we expect that the platform will be open to scientists and students from other universities as well.

REFERENCES

[1] Amazon.com, Inc. (2013). Amazon web services, [Online]. Available: http://aws.amazon.com/.

[2] L. G. Valiant, "A bridging model for parallel computation", *Communications of the ACM*, vol. 33, no. 8, pp. 103–111, Aug. 1990, ISSN: 0001-0782.

[3] F. Dehne, A. Fabri, and A. Rau-Chaplin, "Scalable parallel geometric algorithms for coarse grained multicomputers", in *Proceedings of the ninth annual symposium on Computational geometry*, ACM, 1993, pp. 298–307.

[4] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters", *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[5] A. Goldman, D. Cordeiro, and A. Kraemer, "The suitability of BSP/CGM model for clouds", in *Proceedings of the International Advanced Research Workshop on High Performance Computing, Grids and Clouds*, ser. HPC, Abstract, Cetraro, Italy, Jun. 2012.

[6] L. Youseff, M. Butrico, and D. da Silva, "Toward a unified ontology of cloud computing", in *Grid Computing Environments Workshop*, Austin, TX, USA, Nov. 2008. DOI: 10.1109/GCE.2008.4738443.

[7] A. Huang *et al.* (2013). Apache cloudstack, [Online]. Available: http://cloudstack.apache.org/.

[8] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system", in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09, Washington, DC, USA: IEEE Computer Society, 2009, pp. 124–131, ISBN: 978-0-7695-3622-4.

[9] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds", *IEEE Internet Computing*, vol. 13, pp. 14–22, 2009, ISSN: 1089-7801. DOI: 10.1109/MIC.2009.119.

[10] OpenStack, LLC. (2011). The openstack project, [Online]. Available: http://openstack.org/.

[11] O. Krieger, P. McGachey, and A. Kanevsky, "Enabling a marketplace of clouds: vmware's vcloud director", *ACM SIGOPS Operating Systems Review*, vol. 44, no. 4, pp. 103–114, 2010.

[12] J. Robbins *et al.* (2013). Opscode Chef, [Online]. Available: http://www.opscode.com/chef/.

[13] L. Kanies, "Puppet: next-generation configuration management", *The USENIX Magazine*, vol. 31, pp. 19–25, 1 2006.

[14] D. Cutting *et al.* (2013). Apache hadoop project, [Online]. Available: http://hadoop.apache.org/.

[15] J. Hill, B. McColl, D. C. Stefanescu, M. W. Goudreau, K. Lang, S. B. Rao, T. Suel, T. Tsantilas, and R. H. Bisseling, "BSPlib: the BSP programming library", *Parallel Computing*, vol. 24, no. 14, pp. 1947–1980, 1998.

[16] A. Chan and F. Dehne, "CGMgraph/CGMlib: implementing and testing CGM graph algorithms on PC clusters", in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Springer, 2003, pp. 117–125.

[17] A. Goldman, J. G. Peters, and D. Trystram, "Exchanging messages of different sizes", *Journal of Parallel and Distributed Computing*, vol. 66, no. 1, pp. 1–18, 2006.

[18] W. F. McColl, "Scalable computing", in *Computer Science Today*, Springer, 1995, pp. 46–61.

[19] C. E. R. Alves, E. N. Cáceres, F. Dehne, and S. W. Song, "A CGM/BSP parallel similarity algorithm", in *Proceedings of the I Brazilian Workshop on Bioinformatics*, 2002, pp. 1–8.

[20] P. Krusche, "Experimental evaluation of BSP programming libraries", *Parallel Processing Letters*, vol. 18, no. 01, pp. 7–21, 2008.

[21] A. Chan, F. Dehne, and R. Taylor, "CGMgraph/CGMlib: implementing and testing cgm graph algorithms on pc

clusters and shared memory machines", *International Journal of High Performance Computing Applications*, vol. 19, no. 1, pp. 81–97, 2005.

[22] A. Chan, F. Dehne, and A. Rau-Chaplin, "Coarse grained parallel next element search", in *Parallel Processing Symposium, 1997. Proceedings., 11th International*, IEEE, 1997, pp. 320–325.

[23] T. Garcia and D. Seme, "A coarse-grained multicomputer algorithm for the longest repeated suffix ending at each point in a word", in *11-th Euromicro Conference on Parallel Distributed and Network based Processing (PDP'03*, 2003, pp. 349–356.

[24] H. Mongelli and S. W. Song, "Efficient two-dimensional parallel pattern matching with scaling", in *13th IASTED International Conference on Parallel and Distributed Computing and Systems*, 2001, pp. 360–364.

[25] E. N. Cáceres, A. Chan, F. Dehne, and S. W. Song, "Coarse grained parallel graph planarity testing", in *PDPTA*, 2000.

[26] A. Chan and F. Dehne, "A coarse grained parallel algorithm for maximum weight matching in trees", *12th International Conference on Parallel and Distributed Computing and Systems (PDCS)*, pp. 134–138, 2000.

[27] C. E. R. Alves, E. N. Cáceres, and F. Dehne, "Parallel dynamic programming for solving the string editing problem on a CGM/BSP", in *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, ACM, 2002, pp. 275–281.

[28] C. E. R. Alves, E. N. Cáceres, and F. Dehne, "Sequential and parallel algorithms for the all-substrings longest common subsequence problem", Institute of Mathematics and Statistics, University of São Paulo, Tech. Rep. RT-MAC-2003-03, Apr. 2003.

[29] A. Ferreira and N. Schabanel, "A randomized BSP/CGM algorithm for the maximal independent set problem", *Parallel Processing Letters*, vol. 9, pp. 411–422, 1999.

[30] E. N. Cáceres and C. Nishibe, "0-1 knapsack problem: BSP/CGM algorithm and implementation", in *17th IASTED International Conference on Parallel and Distributed Computing and Systems*, 2005, pp. 14–16.

[31] J. Soares and M. Stefanes, "BSP/CGM algorithm for maximum matching in convex bipartite graphs", in *Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing*, IEEE, 2003, pp. 167–174.

[32] E. N. Cáceres, H. Mongelli, L. Loureiro, C. Nishibe, and S. W. Song, "A parallel chain matrix product algorithm on the InteGrade grid", in *International Conference on High Performance Computing, Grid and e-Science in Asia Pacific Region*, 2009, pp. 304–311.

[33] E. N. Cáceres and C. C. A. Vieira, "Revisiting a BSP/CGM transitive closure algorithm", in *16th Symposium on Computer Architecture and High Performance Computing*, IEEE, 2004, pp. 174–179.

[34] C. E. R. Alves, E. N. Cáceres, F. Dehne, and S. W. Song, "A parallel wavefront algorithm for efficient biological sequence comparison", in *Proceedings of the 2003 international conference on Computational science and its applications: PartII*, SPRINGER, 2004, pp. 249–248.

[35] C. E. R. Alves, E. N. Cáceres, and F. Dehne, "A parallel approximation hitting set algorithm for gene expression analysis", in *14th Symposium on Computer Architecture and High Performance Computing*, IEEE, 2002, pp. 75–81.

[36] A. Chan and F. Dehne, "A note on coarse grained parallel integer sorting", *Parallel Processing Letters*, vol. 9, pp. 533–538, 1999.

[37] E. Cáceres, F. Dehne, A. Ferreira, P. Flocchini, I. Rieping, A. Roncato, N. Santoro, and S. W. Song, "Efficient parallel graph algorithms for coarse grained multicomputers and BSP", in *Automata, Languages and Programming*, Springer, 1997, pp. 390–400.

[38] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, and D. Noveck, "Network file system (NFS) version 4 protocol", RFC Editor, RFC 3530, Apr. 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3530.txt.

[39] P. V. Mockapetris, "Domain names – implementation and specification", RFC Editor, RFC 1035, Nov. 1987. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1035.txt.

[40] R. Droms, "Dynamic host configuration protocol", RFC Editor, RFC 2131, Mar. 1997. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2131.txt.

[41] R. H. Bisseling, "Parallel scientific computation: a structured approach using bsp on mpi", in. Oxford University Press, 2004, p. 324.

[42] R. Russell, "Virtio: towards a de-facto standard for virtual I/O devices", *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 95–103, 2008.

[43] M. H. Jamal, A. Qadeer, W. Mahmood, A. Waheed, and J Ding, "Virtual machine scalability on multi-core processors based servers for cloud computing workloads", in *Networking, Architecture, and Storage, 2009. NAS 2009. IEEE International Conference on*, IEEE, 2009, pp. 90–97.

[44] M. M. Rahman, R. Thulasiram, and P. Graham, "Differential time-shared virtual machine multiplexing for handling qos variation in clouds", in *Proceedings of the 1st ACM multimedia international workshop on Cloud-based multimedia applications and services for e-health*, ACM, 2012, pp. 3–8.

[45] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and QoS in cloud computing environments", in *International Conference on Parallel Processing*, IEEE, 2011, pp. 295–304.

[46] W. Emeneker and A. Apon, "Cache effects of virtual machine placement on multi-core processors", in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, IEEE, 2010, pp. 2261–2266.

[47] C. Tingwei and Z. Shanjie, "Classify virtualization strategy in cloud computing", in *Computer Science & Education (ICCSE), 2012 7th International Conference on*, IEEE, 2012, pp. 192–196.