GRID COMPUTING AND STOCHASTIC DISTRIBUTED SIMULATION

David R.C. Hill
ISIMA / LIMOS UMR CNRS 6158
Blaise Pascal University
BP 10125 – 63177 Aubiere Cedex France
www.isima.fr/hill

ABSTRACT

Grid computing, can be considered as distributed computing taken to the next evolutionary stage. This paper presents an historical perspective of distributed computing followed by the concepts of grid computing and a presentation of some grid systems. This paper also deals with the distribution of stochastic simulations and experiments (in particular Monte Carlo simulations). The different techniques in use to distribute random numbers are also presented as well as future research directions.

INTRODUCTION

Before getting to the current status of grid computing, lets start with an historical perspective of distributed computing. If we had to trace what could be the beginning of distributed systems, we have to recall to our mind that at the beginning of the sixties, Leonard Kleinrock a scientist working at MIT, published a paper on packet switching entitled "Information flow in large communication Nets" (Kleinrock 1961). Still at MIT, J.C.R. Licklider envisioned a global interconnected set of computers where everyone could access programs and information from any sites. The first book on the packet switching theory was published in 1964, again by Kleinrock (Kleinrock 1964). In 1965, the first wide area network was built with a low speed telephone line connecting Massachusetts and California time-shared computers that were able to work together. Also in 1965, the US Department of Defence Advanced Research Project Association (DARPA), started to work on ARPANET (the Advanced Research Program Agency Network). In 1966, the concept of "computer network" was introduced at DARPA and published in 1967. In 1969, the File Transfer Protocol (FTP) helped in distributing the RFCs (Request For Comments) which were fast and easy ways to share ideas amongst network researchers. It was also at the end of the sixties that Ken Thompson, Dennis Ritchie and others started to work on what will become Unix at AT&T Bell Laboratories. At the same time in 1969, ARPANET was connecting 4 universities & research institutes (Stanford Research Institute, UCLA, UC Santa Barbara and the University of Utah). In 1972, the InterNetworking Working Group (INWG) was built to control the growing network and the first e-mails were introduced to public. The next year saw the introduction of the Ethernet Protocol at Xerox PARC (Palo Alto Research Center). Year 1974 saw both the arrival of Telnet, a commercial version of ARPANET and the rewriting of Unix V.4 with the C language, thus enabling a better portability of this operating system among different computers. In 1979, the USENET newsgroup was introduced by students from North Carolina and Duke Universities. The beginning of the eighties saw the spreading of IBM personal computers and its compatibles. The term Internet was introduced in 1982, and in 1983, the old NCP (Network Control Protocol) introduced in 1970 was replaced by the current TCP/IP protocol. Five years later, in 1988, the spreading of the Internet Worm showed the vulnerabilities of the Internet, thus leading to the formation of the Computer Emergency Response Team (CERT) to address security issues. At the Palo Alto DEC Research Center, the distribution of computing tasks sent via e-mails to workstations inside and outside their laboratory was successfully achieved in 1988 (results were also collected via e-mails).

A few years later, in 1991, the concept of World Wide Web is born and in the same year Linus Torvalds distributed the source code of the Linux kernel as freeware. In 1993, the first Web browser, named Mosaic is released applying the Web concepts developed at CERN ; in the same year, the first RSA Security challenge is won by a team of 600 volunteers providing their distributed computing resources. In 1994, version 1.0 of Linux, produced by Linus Torvalds and a world wide team of hackers, is available for download. The annual growth rate of the Web between 1993 and 1994 is over 300%. The following year saw the introduction of an alpha version of Java, an object-oriented language with a portable byte-code able to be transmitted and executed on the Internet. The GIMPS distributed computing project, searching for larger and larger prime numbers is launched in 1996. On August the 26th of 1997, the PI challenge is launched by Fabrice Bellard to find the 1000th billion binary digit of PI, this goal is reached on September 22nd. In 1999, the concept of Internet computing was providing at least 3 times more computing power (33 Teraops) than any other supercomputers with the Seti@home project of Berkeley University, which analyzed data from the famous Arecibo radio-telescope. Clients software were available for Windows, Macintosh-OS and Unix. More useful projects appeared with the new millennium. In year 2000, the Casino-21 project goal was to simulate the dynamics of the earth's climate for the next century. In year 2001, Barabasi from Notre Dame

University introduced in (Barabasi et al. 2001) the concept of parasitic computing, explaining and showing how Internet protocols could be used to compute complex linear programming problems. At the end of year 2001, the French Decrypton project was setup to compute and compare 50000 protein sequences with a distributed client software developed by Genomining and IBM. More than 75000 Internet users participated to reach the first results in 2 months, 1170 years would have been necessary to achieve the same result on a single computer. The collected data is now publicly available for the international community of scientists working in bioinformatics. A taxonomy of distributed technologies that have emerged following all these events can be found in a paper written with Fabrice Bernadi (Bernardi et al. 2004). The former includes classical client/server systems, metacomputing, massive distributed architectures (P2P, Internet computing,…) agent-based systems and the communication models able to be implemented in distributed systems. In addition, many presentations and definitions of distributed systems can be found in the literature from the Flynn classification (Flynn 1972) to various recent books from Andrew Tannebaum. In this paper the main focus will be given to grid computing concepts and systems.

CONCEPTS AND SYSTEMS OF GRID COMPUTING

Grid computing is defined as “coordinated resource sharing and problem solving in large, multi-institutional virtual organization" by (Baker et al. 2002). The term Grid suggests a computing paradigm similar to an electric power grid. This was a very new concept and is currently under a heavy phase of research and development. A quite complete review can be found in (Allan and Ashworth 2001). (Buyya 2002) provides an economical perspective of this kind of systems, while (Paindaveine 2003) provides a review of Grids in the health sector. Grid applications couple resources that cannot be replicated at a single site or may be globally located for other reasons. Thus, a Grid is an infrastructure that can unify globally and diverse resources. The main issues for computational grids are heterogeneity since a grid involves a multiplicity of heterogeneous resources; scalability since a grid might grow from few resources to millions; dynamicity or adaptability since the probability of some resource failing is high with so many resources. There are four components necessary to form a grid :

- a Grid Fabric comprising all the resources (computers, clusters,...) geographically distributed and accessible from anywhere on the Internet;
- a Grid Middleware offering core services such as remote process management, co-allocation of resources, storage access,...;
- a Grid Development Environment offering high level services that allows programmers to develop applications and brokers that act across global resources;
- some Grid Applications and Portals developed using Grid-enabling languages.

There are many grid projects worldwide. Among the Grid Computing Systems we try to select the most recognized one, which are however still in their research and development phases. Some of them, like Globus, have been employed in very strong and precise researches (see references). More grid projects can be found at (http://www.globus.org/about/related.html), at (http://www.gridcomputing.com) or at (http://dsonline.computer.org/gc/gcprojects-european.htm), and many of the systems presented here are described more precisely in the excellent review provided by (Kacsuk and Vajda 1999). Here is a non exhaustive list, excerpt of grid computing systems we selected in (Bernardi et al 2004) :

- Globus: Globus is the most known and used of grid-enabling systems. It is released as a toolkit proposing a set of components that can be used independently or together. We can cite as components a resource allocation manager, security infrastructure, a monitoring and discovery service or a failure detector as described in (Foster and Kesselman 2001) or (Allcock et al. 2001). Globus 3.0 is the first full major implementation of an open standard for grid computing OGSA (Open Grid Services Architecture) well described at http://www-106.ibm.com/developerworks/grid/library/gr-visual/index.htm%l) and presented in (Alpdemir et al. 2003). Main URL: http://www.globus.org
- ALiCE: ALice (Adaptive and scalable Internet-based Computing Engine) is a platform-independent, Java based grid middleware. It comprises three types of entities: a resource broker, service producers and service consumers. Main URL: http://www.comp.nus.edu.sg/~teoym/alice/alice-tech.htm
- Condor: Condor is defined as a specialized workload management system for computer-intensive jobs. The goal of the project is to develop mechanisms and policies that support high-throughput computing on large collections of computing resources. (Thain et al. 2003), (Basney et al. 1997) and (Tannenbaum et al. 2002) present this project. Main URL: http://www.cs.wisc.edu/condor/
- Grid Engine: Sun ONE Grid Engine is available in two versions: SGE and SGE Enterprise Edition. This middleware requires no alterations to applications to be distributed. More information can be found in (Coomer and Chaubal 2002). Main URL: http://wwws.sun.com/software/gridware/
- GriPhyN: GriPhyN is a collaboration of computer science researchers and experimental physicists to enable a grid system following two approaches: apply a methodical and organized data management using the virtual data concept and developing automated request scheduling mechanisms in order to make large grids easy to use as single computers. Main URL: http://www.griphyn.org/index.php
- Legion: Legion enables the connection of networks of computers using an independent element approach. There is no central element and the proposed framework permits scheduling and distributing processes as in a single, virtual machine. More

information can be found in (Natrajan et al. 2001). Main URL: http://www.cs.virginia.edu/~legion/

- Metacomputer OnLine: Metacomputer OnLine (known as MOL) is designed as an open, extensible software system comprising a variety of software modules. In contrast to other grid environments, MOL is not based on specific models or tools. Main URL: http://www.uni-paderborn.de/pc2/projects/
- NetSolve: NetSolve is a grid framework focusing on three main points: ease of-use for the end-user, efficient use of the resources and the ability to integrate any arbitrary software component as a resource into the Net-Solve system. More information, as well as a grid concept description can be found in (Arnold et al. 2001). Main URL: http://icl.cs.utk.edu/netsolve/

STOCHASTIC PARALELL SIMULATION

Parallel and distributed simulation has been extensively studied for deterministic models. The major references in this field (Chandy and Misra 1979) for conservative approaches (Jefferson and Sowizral 1985) for Time Warp an optimistic approaches ; (Fujimoto 2000) presents the current state of the art in this research direction. However, research on stochastic parallel simulation is much tougher and this domain is less studied. Few are aware of the potential dangers associated to the distribution of sequential stochastic simulations. A well known paper in this domain warned scientists at the end of the last century to be very careful with parallel Monte Carlo (Hellekalek 1998). Indeed, if sequential stochastic simulations always require a statistically sound RNG (Random Number Generator), in the case of parallel and distributed stochastic simulations, this requirement is crucial. Major scientists in this research field such as Paul Coddington state that: *"Random number generators, particularly for parallel computers, should not be trusted. It is strongly recommended that all simulations be done with two or more different generators, and the results compared to check whether the random number generator is introducing a bias"* (Coddington 1996). The main advice that specialists recommend is that random numbers should be generated in parallel, i.e. each Logical Processor (LP) should autonomously get its own sub-sequence of a global random sequence. Designers of parallel stochastic simulations always have to answer this fundamental question: how can we make a safe RNG repartition in order to keep, on the one hand, efficiency, and on the other hand a sound-statistical quality of the simulation in order to obtain credible results ? Indeed the validation of such parallel simulations is a critical issue. Many research studies have been undertaken to design good sequential RNGs. Many others concern Parallel Random Number Generators (PRNGs). It appears that the key focus, which is how to assess the quality of random streams, remains a hard problem, and many widely-used techniques have been shown to be inadequate for some specific applications.

An obvious way to get parallel random numbers streams is to partition a sequence of a given generator into suitable independent sub-sequences. This can be done in three major ways (James 1990) (Traore and Hill 2001) :

1. The Leap Frog (LF) method allocates in turn the random numbers of a sequence to a partition of streams like a deck of cards dealt to card players. Thus, for a partitioning of a global sequence $\{x_i, i = 0,1,2,\ldots\}$ into N parallel streams, the jth stream is $\{x_{kN+j-1}, k = 0,1,2,\ldots..\}$. Given the period p of the global sequence, the period of each stream can reach p/N.
2. The Sequence Splitting (SS) method splits a sequence into non overlapping contiguous blocks. Thus, for a partitioning of a sequence $\{x_i, i= 0, 1, 2, \ldots\}$ into N streams, the jth stream is $\{x_{k+(j-1)m}, k = 0, \ldots, m-1\}$, where m is the length of each stream. One major difficulty is the determination of a good value for 'm', which must be chosen so that each stream is long enough to achieve the stochastic simulation performed by the corresponding processes.
3. The Independent Sequences (IS) method builds a partition of N streams by initializing the same generator with N different seeds. Such a technique can lead to overlapping streams, since a random number generated in one stream can match the seed used for another stream. Some generators exhibit several sub-cycles that can be encapsulated into other streams. Such generators seem to be well adapted to the IS method, since the seeds to use for parallel streams can be the first terms of the sub-cycles.

Each way has its own specific flaw and whatever the solution retained, we have to check the correlations between subsequences. Indeed, the correlation phenomena observed in a sequential RNG are often induced in the parallel streams that are obtained from this generator. For instance, an initial correlation between distant numbers of a sequence can lead to auto-correlation inside parallel streams produced with the LF method, or cross-correlation between parallel streams produced with the SS method. Variants have been developed for three methods presented above such as the shuffling leap frog or the parameterization method which is a variant of the IS technique (See the work of Prof. Mascagni Team http://www.ncsa.uiuc.edu/Apps/SPRNG/ www/paper/). Its principle is to parameterize both the seed and recursive function. One of the main contributions of this variant is that it results in a scalable period, i.e. the number of different random numbers that can be used increases with the number of parallel streams. An interesting parameterization method is used in (Percus and Kalos 1989) to get parallel streams from an LCG, by choosing for the jth stream a multiplier a(j) and an additive constant c(j). It is shown in this case that good results can be obtained if c(j) is the jth prime number $\leq \sqrt{(m/2)}$ where m is the modulus. However testing pseudo-random numbers in their parallel context is essential. The usual approach in testing a PRNG is to apply standard tests which are available for sequential RNG to each random number parallel streams, and then to all streams combined (Mascagni et al. 1995). New techniques appear which are specific to PRNGs http://archive.ncsa.uiuc.edu/Science/CMP/RNG/www/pape r/index.html). A parallel version of the well known spectral test has been developed in (Percus and Kalos 1989) for parallel LCG. So far, little work has been done in this area. Since not many rigorous results are known about the properties of PRNG, more stringent and varied tests are necessary. The next section will present an application were we have chosen the Sequence Splitting parallelization method.

SAMPLE APPLICATION AND DISCUSSION

The use of Monte Carlo Simulations for clinical treatment is now showing its efficiency to obtain better precision in cancer treatments with radiotherapy than what is currently obtained with analytical methods. However, the computing time using Monte Carlo calculations must stay comparable to what is currently obtained with analytical calculations. Since particle emission can be simulated independently, the parallelization of Monte Carlo simulations by splitting pseudorandom numbers generator sequences into a multitude of streams is a simple, rigorous and generic method (i.e. it can be applied for each Monte Carlo codes used in physics with a pseudorandom number generator). In (Maigne et al 2004) we have recently presented the comparison between a local and a parallel computation of some physical results of interest showing that the output data stayed unchanged and therefore validated the parallelization method used. Additional tests should be conduced with different pseudorandom numbers.

Despite many constraints due to Network latency and Grid load, the gain in computing time obtained by splitting the simulations were very encouraging. To speed up the calculations, we can think about enabling a new submission for the jobs that are waiting for a too long time in the Grid batch queue in order to be computed. In the future, it is envisaged to setup a waiting time in the batch queue equivalent to 10 % of the total estimated computing time of the jobs, after this delay, jobs still in the waiting status will be stopped and submitted again. The new submission will be achieved on another grid site where either the number of free CPUs is consequent or the running status is reached quickly for the submited jobs. Additional optimization can also be done for Monte Carlo simulations using unrolling techniques (Hill 2003). Local files of pre-computed random number subsequences can be accessed with memory mapping techniques on local worknodes without efficiency losses compared to a classical memory access. Indeed the use of random numbers in a Monte Carlo simulation is always predictable and cacheable in processor caches if pre-computed and stored in arrays.

CONCLUSION

Grid systems are very promising for Monte Carlo Simulations. Parallel Monte Carlo simulation can be complex to setup with sound statistical bases, however parallelizing techniques for pseudo-random numbers are now carefully studied. Recent work precisely analyzed Large-Scale Grid-Based Monte Carlo Applications (Li and Mascagni 2004). Independent replications should now always be done in parallel, this will always imply a performance gain (Li and Mascagni 2003), boostraping can also benefit of this new computing power. The current perspectives should now focus again on basic operation research and performance evaluation studies that should be done to optimize job submission with tools like QNAP2 (Queuing Network Package 2) or with dedicated software like GridSim. The GridSim toolkit is an Australia based grid research project that enables users to conduct design and performance evaluations of scheduling algorithms. The primary objective of such tools is to is to create simulated grid environments that reflect real-world scenarios involving a varying number of resources and users with different (policy) requirements (Buyya 2002). Additional simulations, corresponding to metamodels have to be used to test resource allocation techniques and the effectiveness of resource brokers and scheduling algorithms. Current grid systems are in development phase and there are limits to how many machines can run grid jobs simultaneously. When a user is lucky, he may get 20 machines, but the design of large parallel applications where thousands of machines with many different load conditions, many different kinds of user scenarios should be considered for the evaluation of Grid scheduling systems. From this paper, we hope it is understood that Grid computing is not a silver bullet enabling any application to run faster and faster. There are many applications that cannot be parallelized, many others will need a large amount of work to allow better performances and in addition the grid configuration, reliability and infrastructure will also play an important role in the general performance.

REFERENCES

(Allan and Ashworth 2001) R.J. Allan and M. Ashworth. A Survey of Distributed Computing, Computational Grid, MetaComputing and Network Information Tools. Technical report, UKHEC, 2001.

(Allcock et al. 2001) W. Allcock, A. Chervenak, I. Foster, L. Pearlman, V. Welch, and M. Wilde. Globus Toolkit Support for Distributed Data-Intensive Science. In Proceedings of Computing in High Energy Physics (CHEP '01), 2001.

(Alpdemir et al. 2003) M.N. Alpdemir, A. Mukherjee, N.W. Paton, P.Watson, A.A. Fernandes, and A. Gounaris. OGSA-DQP - A Service-based Distributed Query Processor for the Grid. In EPSRC, editor, Proceedings of UK e-Science All Hands Meeting Nottingham, 2003.

(Arnold et al. 2001) D.C. Arnold, S.S. Vahdiyar, and J.J. Dongarra. On the Convergence of Computational and Data Grids. Parallel Processing Letters, Vol. 11, N°2&3, pp. 187-202, 2001.

(Baker et al. 2002) M. Baker, R. Buyya, and D. Laforenza. Grids and Grid Technologies for Wide-Area Distributed Computing. Software - Pratice and Experience, Vol. 488, 2002.

(Barabasi et al. 2001) A.L. Barabasi, V.W. Freeh, H. Jeong, and J.B. Brockman. Parasitic Computing. Nature, Vol. 412, N°30 August 2001.

(Basney et al. 1997) J. Basney, M. Livny, and T. Tannenbaum. High Throughput Computing with Condor. HPCU News, Vol. 1, N°2, 1997.

(Bernardi 2004) F. Bernardi, J-F. Santucci and D. Hill. A Survey of Distributed Systems in Bioinformatics, LIMOS Research Report 04-02. 2004.

(Buyya 2002) R. Buyya. Economic-based Distributed Resource Management and Scheduling for Grid Computing. PhD thesis, Monash University, 2002. Melbourne, Australia.

(Chandy and Misra 1979) K.M.Chandy and J. Misra, Distributed Simulation. IEEE Transactions on Software Engineering, Vol. 5, N°55, pp. 440-452. September 1979.

(Coddington 1996) P. Coddington, "Random Number Generators For Parallel Computers", *NHSE Review*, Second Issue, 1996.

(Coomer and Chaubal 2002) J. Coomer and C. Chaubal. Introduction to the Cluster Grid - Part 1 and Part 2. Technical report, Sun BluePrints, 2002.

(Flynn 1972) M.J. Flynn. Some Computer Organizations and their Effectiveness. IEEE Transactions on Computers, C-21, 1972.

(Foster and Kesselman 2001) I. Foster and C. Kesselman. Globus - A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications, Vol. 11, N°2, 2001.

(Fujimoto 2000) Fujimoto R.M. 2000. Parallel and Distributed Simulation Systems. Wiley Interscience January. 2000.

(Hellekalek 1998) P. Hellekalek, "Don't trust parallel Monte Carlo!", *Proceedings of the 12th Workshop on PADS*, Banff, Alberta, Canada, pp. 82-89, 1998.

(Hill 2003) Hill D. 2003. URNG: A portable optimisation technique for software application requiring pseudorandom numbers. *Simulation Modelling Practice and Theory*. Vol. 11, pp. 643-654.

(James 1990) James, A Review of Pseudorandom Number Generators, Computer Phys. Comm. Vol. 60, pp. 329-344, 1990.

(Jefferson and Sowizral 1985) Jefferson D. Sowizral H., "Fast concurrent simulation using the time warp mechanism". Distributed Simulation SCS, Vol 15 n° 2. January. San Diego. pp. 63-69. 1985.

(Kacsuk and Vajda 1999) P. Kacsuk and F. Vajda. Network-Based Distributed Computing – Meta-computing. Technical report, ERCIM, 1999.

(Kleinrock 1961) L. Kleinrock. Information Flow in Large Communication Networks. RLE. Quarterly Progress Report, July 1961.

(Kleinrock 1964) L. Kleinrock. Communication Nets - Stochastic Message Flow and Delay. McGraw-Hill, 1964. later re-issued by Dover Books.

(Li and Mascagni 2003) Y. Li and M. Mascagni, Improving Performance via Computational Replication on a Large-Scale Computational Grid, Parallel Computing, Vol 29, pp. 69-94. 2003.

(Li and Mascagni 2004) Y. Li and M. Mascagni, Analysis of Large-Scale Grid-Based Monte Carlo Applications International Journal of High Performance Computing Applications (IJHPCA) Vol. 17 pp. 369-382. 2004.

(Maigne et al. 1994) L. Maigne, D. Hill, P. Calvat, V. Breton, R. Reuillon, Y. Legr and D. Donnarieix, Parallelization of Monte Carlo Simulations and Submission to a Grid Environment, Parallel Processing Letters, Vol. 14, N° 2, pp. 2004.

(Mascagni et al. 1995) M. Mascagni, S.A. Cuccaro and D.V. Pryor, Techniques for Testing the Quality of Parallel Pseudorandom Number Generators. *Proceedings of the 7th SIAM Conference on Parallel Processing for Scientific Computing*. pp. 279-284, 1995.

(Natrajan et al. 2001) A. Natrajan, M. Humphrey, and A. Grimshaw. Capacity and Capability Computing in Legion. In Proceedings of the 2001 International Conference on Computational Science, 2001.

(Paindaveine 2003) Y.P. Paindaveine. HealthGRID - Old Wine in New Bottle. In Healthgrid, Proceedings of the First European HealthGrid Conference, Lyon, France pp. 20-27, 2003.

(Percus and Kalos 1989) Percus O.E. and Kalos M.H., Random number generators for MIMD parallel processors. *Journal of Parallel and Distributed Computing*. Vol. 6. pp. 477-497, 1989.

(Tannenbaum et al. 2002) T. Tannenbaum, D. Wright, K. Miller, and M. Livny. Condor - A Distributed Job Scheduler, chapter 15. The MIT Press, 2002. Beowulf Cluster Computing with Linux.

(Thain et al 2003) D. Thain, T. Tannenbaum, and M. Livny. Condor and the Grid, chapter 11. John Wiley, Grid Computing. Making The Global Infrastructure a Reality edition, 2003.

(Traore and Hill 2001) Traore M., Hill D., "The use of random number generation for stochastic distributed simulation: application to ecological modeling". 13th European Simulation Symposium, Marseille, October 18th-20th, pp. 555-559, 2001.

ACKNOWLEDGEMENT

We give a special thank to Dr. Mamadou Kaba Traore, Dr. Fabrice Bernardi and to Lydia Maigne (Ph.D. student) for their help in the research presented in this paper.