Parallelizing the ACGT OncoSimulator

Dominique LAVENIER and Julien JACQUES

***Abstract*— Nowadays, clinician can take advantage of *in-silico* technologies to get faster estimations on clinical treatments without performing time-consuming and expansive in-vivo experimentations. Among these techniques, tumor growth simulators can estimate tumor volume and the quantity of cells (stem, proliferating...) in function of the time and from a set of parameters. Within the framework of the ACGT European project [2], we are working on the development of the *ACGT Oncosimulator* which is based on the mathematical model from ICCS. In this paper, we relate the current computation techniques for increasing the *ACGT Oncosimulator* efficiency.**

I. MOTIVATIONS

To have a powerful simulator, the underlying program must be reliable, precise and fast. First, the reliability comes from the consistence of the mathematical model compared with in-vivo experiments. Second, the precision depends of the sampling area, that is the spatial 3D unit dedicated to the tumor discretization. Finally, speed is both related to the quality of the code and to the performance of the hardware resources. This paper deals with the last point.

Initially, the Oncosimulator has been developed for a stand-alone PC equipped with a single core processor. This approach obviously limits the program to sequential executions, even with today machines which are now integrating double or quad core processors. Furthermore, one of the goals of the ACGT project is to deploy a European computational grid able to support fast execution of the Oncosimulator. It was thus desirable to adapt the Oncosimulator code to benefit from the use of the ACGT grid power together with the latest technology improvements.

Basically, the ACGT end-users may exploit the Oncosimulator as follows:

- **intensively:** to find the better treatment, the clinician has to try many combinations of parameters provided by the Oncosimulator (size of the tumor, duration of the treatment, quantity of drugs, interval between two injections, etc.). In that case, many runs on different parameters have to be executed, and the execution times are directly correlated with the number of runs. It may be not unusual to have a few hundred of parameter combinations to test.
- **with high precision:** better the definition of the tumor, better the estimation of the tumor behavior. In our case, the definition of the tumor is directly linked to the 3D discretization. From a computational point of view, shrinking the discretization by 10 will increase the computational complexity by 1000 since the modelization operates in the 3D space.
- **interactively:** another way to focus to the right treatment is to run successive simulations guided by a human expert. Depending of the results of one simulation, the clinician will slightly tune a few parameters according to the tumor evolution. In that specific case, the response time of the simulator is critical. Ideally, an execution should not take more than ten to twenty seconds in order to have an efficient interacting tool.

In the rest of the paper, we explore how the various technologies available today can support these different kinds of requirements [1].

II. TECHNOLOGIES

A. Grid computing

Grid computing consists in a set of machines (called nodes) geographically spread and connected through Internet (Fig. 1). A grid generally provides a high computing power and a huge storage capacity. It is accessed through server managers as, for instance, the Globus-SGE [3] environment. These servers drive the node allocations according to the grid workload.

Grids are well adapted to simultaneous executions of many independent programs. The execution time of one program is the same as if it is executed into a single machine. However, since several programs are run in parallel, the global time T_g needed to achieve the whole computation corresponds to:

$$T_g = T_p \times \left\lceil \frac{N_p}{N_{\text{nodes}}} \right\rceil, \quad (1)$$

$$\text{where} \begin{cases} T_p : \text{execution time of single program} \\ N_p : \text{number of single programs} \\ N_{\text{nodes}} : \text{number of nodes} \end{cases}$$

Deploying a code on a grid is easy since no modification is required. Only the execution management is crucial in order to distribute program instances on the available resources.

The grid solution is well adapted to the first use of the Oncosimulator: a lot of executions can be run in parallel with different parameters, each run being assigned to a different machine. Results are automatically collected back to the user.

This work is supported by the European Community Framework Programme for Research, Technological Development and Demonstration (FP6)

D. Lavenier, Symbiose Project Team, CNRS, Irisa, Campus de Beaulieu, 35000 Rennes, France lavenier@irisa.fr

J. Jacques, Symbiose Project Team, INRIA, Irisa, Campus de Beaulieu, 35000 Rennes, France jjacques@irisa.fr

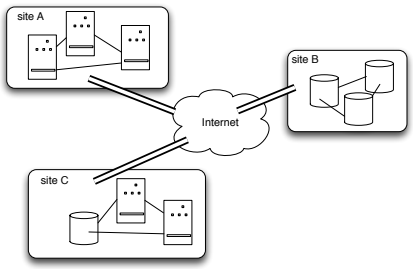

Fig. 1. *Example of a Grid infrastructure with repartition of the computation power and the storage capacity*

B. Cluster

A cluster is a set of identical machines connected through a high-speed network (Fig. 2). From an implementation point of view, there are basically two possibilities to execute a code on this support. The first one follows the grid idea: many instances of the same program are dispatched on the different nodes of the cluster. Difficulties and implementation are thus identical.

The second one required to modify the code for parallelizing the various parts of the program on several nodes. To get a fast program, computations must be shared between the processors in order to have a number of communications and synchronizations as small as possible.

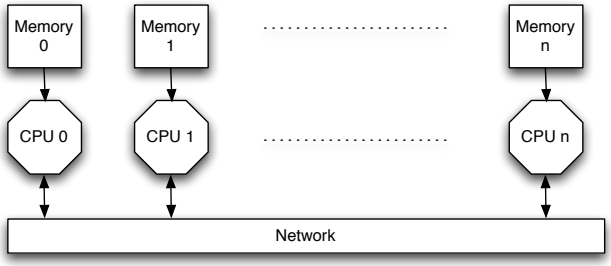

Fig. 2. *Single Program Multiple Data (SPMD) Architecture*

That makes the parallelization task quite difficult: Even by assuming well adapted distribution of data in the processor memories, it always remains, in many cases, a part of the program which cannot be parallelized and have to be done sequentially. We also need to keep in mind that communications add costs between processors, and these costs do not exist in the monoprocessor version. As a consequence, the execution time can be stated as follows:

$$T_{//} = \frac{T_{\text{parallel\_tasks}}}{N_p} + T_{seq} + C_{\text{comm}} \qquad (2)$$

In the case where $T_g < (N_p \times T_{//})$ then the parallelization is inadequate.

As for the grid technology, clusters are well adapted for running a large set of independent simulations with different parameters. If a smart parallelization can be done at the cluster level, this technology is thus able to satisfy all end-user requirements.

C. Multicore processors

A multicore processor is a component having, on the same die, several cores – or processing units – connected to the same memory (Fig. 3). In this architecture, the use of threads (or light processes) enables to explicitly express in the source code the parallel execution of various parts of the program.

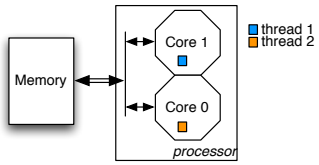

Fig. 3. *Two-core processor organization*

Using this technology, we need to extract parallel blocs that do not have data, time and spatial dependencies. In a sequential program, the process is composed of only one thread. Therefore, its execution on a multicore processor will be done only on one core. In a multithreaded program, the process is divided into many threads. As a result, the execution can be shared among all the cores.

Even if this solution covers all the end-user requirements for speeding up the simulation, it is much better suited for accelerating single instance of the Oncosimulator. With the next generation of processors, a larger number of cores will certainly be available, providing significant increase of the computing power. Thus, this is a necessity for the next version of programs like Oncosimulator programs which require high computational power to use this new opportunity.

III. Application to the ACGT Oncosimulator

As a first approach, we have made a cluster implementation on the INRIA Genouest bioinformatics platform where many instances of the Oncosimulator can be run simultaneously. Their submissions are done through a web browser at http://acgt.genouest.org.

The next step has been to integrate this work through the ACGT grid environment. The feedback from the ACGT end-users is encouraging since this implementation meets the need of simulating the tumor evolution according to a large set of parameters.

However, the execution time of each instance takes several minutes, leading to a non-interactive tool. Then, we first tried to develop a MPI version, but the code we developed was requiring too much synchronization points to get this version really efficient.

We are currently developing a multithreaded version targeted multicore processors with the objective to fall below an execution time of ten seconds.

The last point, which needs to be highlighted, is that all these technologies are not mutually exclusive. On the contrary! Grids are made of clusters which now include multicore processor nodes. Thus, parallelization improvement made on one of these technologies will have a direct impact on the whole ACGT project.

References

[1] M. Creel and W.L. Goffe, *Multi-core CPUs, Clusters and Grid Computing: a Tutorial*, Society for Computational Economics, Computing in Economics and Finance 2005, 438.
[2] *ACGT Project website*, http://eu-acgt.org
[3] *The Globus Alliance website*, http://www.globus.org