7th International Conference on Hydroinformatics
HIC 2006, Nice, FRANCE

EXECUTING HYDRODYNAMIC SIMULATION ON DESKTOP GRID WITH OBJECTWEB PROACTIVE

DENIS CAROMEL, VINCENT CAVÉ, and ALEXANDRE DI COSTANZO
INRIA Sophia – I3S – CNRS – Université de Nice Sophia Antipolis
France

CÉLINE BRIGNOLLES, BRUNO GRAWITZ, and YANN VIALA
Société du Canal de Provence et d'aménagement de la région provençale
Le Tholonet CS 70064 – 13182 Aix en Provence Cedex 5 –, France

In this paper, we show how with the grid middleware ObjectWeb ProActive, we have improved and adapted the deployment of a numerical Hydrodynamic Simulation application, the TELEMAC parallel System, for desktop grid.

INTRODUCTION

Two-dimensional mathematical models are increasingly used for the simulation of flood propagation in river plains. This type of model are now preferred to the former 1D network model with the conceptualization of the flood plain by a series of ponds, which necessitate adjusting a number of parameters in order to describe the conceptualized hydraulic transfer between ponds.

However, the problem with 2D models is that they require large amount of processing time especially when large areas are to be studied. SCP_{id}, is using on a regular basis the finite element software TELEMAC-2D. Recently it has been used for modelling the Durance River from Cadarache to Mallemort, which represent more than 50 km of river. Such a model represents 20,000 calculation nodes, and the calculation time is more than 38 hours, which prevents the engineers from taking advantage of all the benefit that the model can provide.

The availability of powerful computers and high-speed network technologies as low-cost commodity components is changing the way we use computers today. These technological opportunities have led to the possibility of using distributed computing platforms as a single, unified resource, leading to what is popularly known as Grid computing.

Grids enable the sharing, selection and aggregation of a wide variety of resources including supercomputers, storage systems, and specialized devices that are geographically distributed for solving large-scale computational and data intensive problems in science, engineering, and commerce.

In this paper, we present the results of a collaborative effort towards the distributed, numerical Hydrodynamic Simulation. First, we describe our river modelling using finite element method. Second, we present an Open Source middleware for the Grid,

ObjectWeb ProActive, featuring distributed objects and deployment. Then, using ProActive, we demonstrate how to design and program a distributed version of TELEMAC, wrapping legacy code and deploying it on desktop grids.

RIVER MODELING USING FINITE ELEMENT METHOD

Introduction

In the past decade, hydroinformatics have seen the apparition of more powerful tools in order to improve the knowing of natural phenomenon. Some of them are two dimensional free surface flow models which are especially useful for floodplain modeling. The Société du Canal de Provence (SCP_{id}) uses a software called TELEMAC 2D to build such hydraulic models. This is a program for the solution of the two dimensional Saint-Venant equations in their depth-velocity form using triangular finite elements. It has been developed by EDF-DRD.

Application description

In order to review the flood risk management plan all along the river Durance, the French government asked SCP_{id} to build a numerical model of part of the river to study floods expansions and propagations [1]. The concerned area represents more than 50 km of river between the Cadarache dam and the Mallemort dam. Also are presents many levees or dikes that are important in flood propagation which must be included in the model.

Finite elements description is particularly well adapted to such topography because of the possibility to refine the mesh where it is important to have a good description of the field. Finally, the obtained mesh of this area account more than 20,000 calculation nodes. The computation of one flood algorithm can then takes up to 38 hours on a dedicated workstation.

The Saint-Venant equations

The model uses the depth-velocity form of the 2D Saint Venant equations (continuity and momentum equations):

$$\frac{\partial h}{\partial t} + \mathbf{U} \cdot \nabla h + h \nabla \cdot \mathbf{U} = S \tag{1}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{U} = -g \nabla Z + h\mathbf{F} + \nabla \cdot (h \nu_e \nabla \mathbf{U}) + \frac{S}{h}(\mathbf{U}_S - \mathbf{U}) \tag{2}$$

Where h is the water depth, Z is the free surface elevation, $\mathbf{U}$ is the velocity vector, $\mathbf{F}$ is the friction force vector, S is the bottom source term, $\mathbf{U_S}$ is the source term velocity vector, ν_e is the effective viscosity, which includes the dispersion and the turbulence contributions.

The friction force is given by the Strickler formulae:

$$\mathbf{F} = -\frac{1}{\cos\alpha}\frac{g}{h^{4/3}K^2}\mathbf{U}U \tag{3}$$

Where α is the steepest slope at the point and K is the Strickler coefficient.

Boundary conditions

Physically we distinguish between two types of boundary conditions: the solid boundaries and the liquid boundaries.

In solid boundaries there exists an impermeability condition: no discharge can take place across a solid boundary. In order to take account of friction the following relation is imposed:

$$\frac{\partial \mathbf{U}}{\partial n} = a\mathbf{U} \tag{4}$$

Where a is the boundary friction coefficient and n is the normal vector.

Liquid boundary condition assumes the existence of fluid domain that does not form part of the calculation domain. Four types of liquid boundaries are distinguished:

- Torrential inflow: velocity and depth prescribed
- Fluvial inflow: velocity prescribed and free depth
- Torrential outflow: free velocity and depth
- Fluvial outflow: free velocity and prescribed depth

Initial conditions

In order to obtain realistic initial conditions, we start from a fictive reasonable state satisfying all the boundary conditions. Keeping the upstream discharge equal to the initial value of the flood hydrograph, the calculation is performed, and a final stationary state is obtained, with water only in the main channel. It is this stationary state which is used as initial state for the flood study.

The algorithm

The Saint-Venant equations are solved in two steps. The first step allows computing the convection term with the characteristic method. In the second step, the program solves the propagation, diffusion, and source term of the equations with a finite elements scheme. Variationnal formulation associated with time and space discretization changes continuous equations in discreet linear system where unknown quantities are depth and velocity at each node. Iterative method is then used to solve this system [2].

Results

The results of such a model are depth and velocity field at each node and at each time step. The analysis is done through time or space variations or maximum values of quantities. Figures 1 is an analysis of the flood dynamic. It represents the flow on small part of the modelled area. As the software compute the complete velocity field, it is possible to distinguish main flow from secondary flow like overtopping or weir flow above dikes. It is also possible to evaluate the modification in first bottom flow when breaches occur. Figure 2 represents the depth field. Places where water depth is more than one meter are identified in black while places where water depth is more than half a meter are identified in grey.

Running time of the software

A calculation on such a model requires a running time of the same order than the propagation time of the flood on a dedicated workstation with two 2.4 GHz processors and 1 GB RAM. Therefore, it is difficult to optimize the model as time between one modification and its result is very long.

The Durance model is the largest one built in SCP_{id}. We have also built a model of the coastal plain of the Siagne river [3], which accounts up to 10,000 calculation nodes, for running time of about 10 hours. This model has been used for the first tests of grid computing.

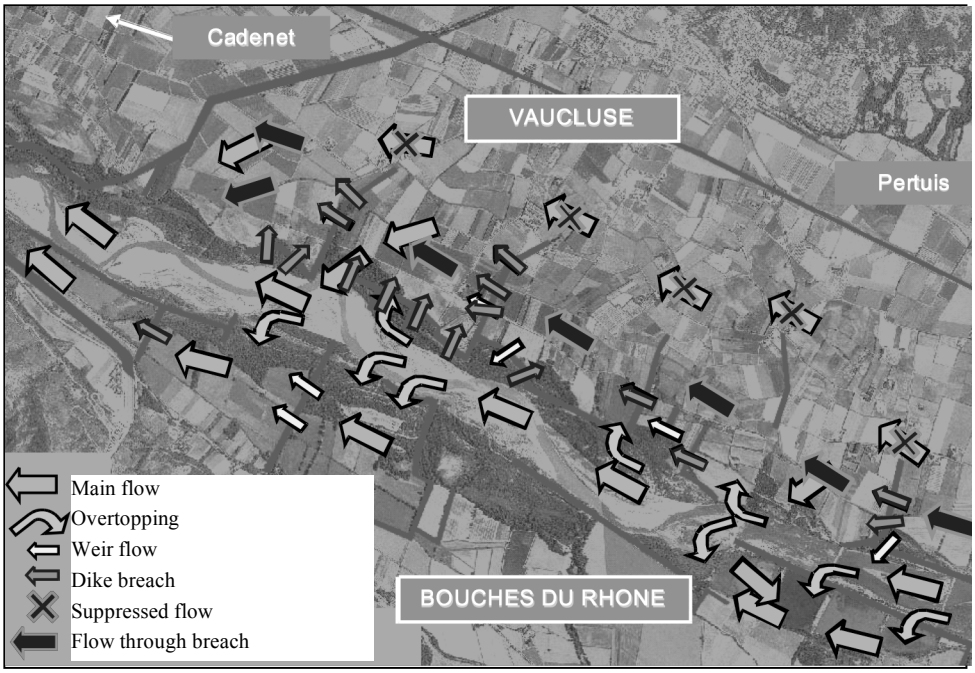

Figure 1: Dynamic analysis of flood.

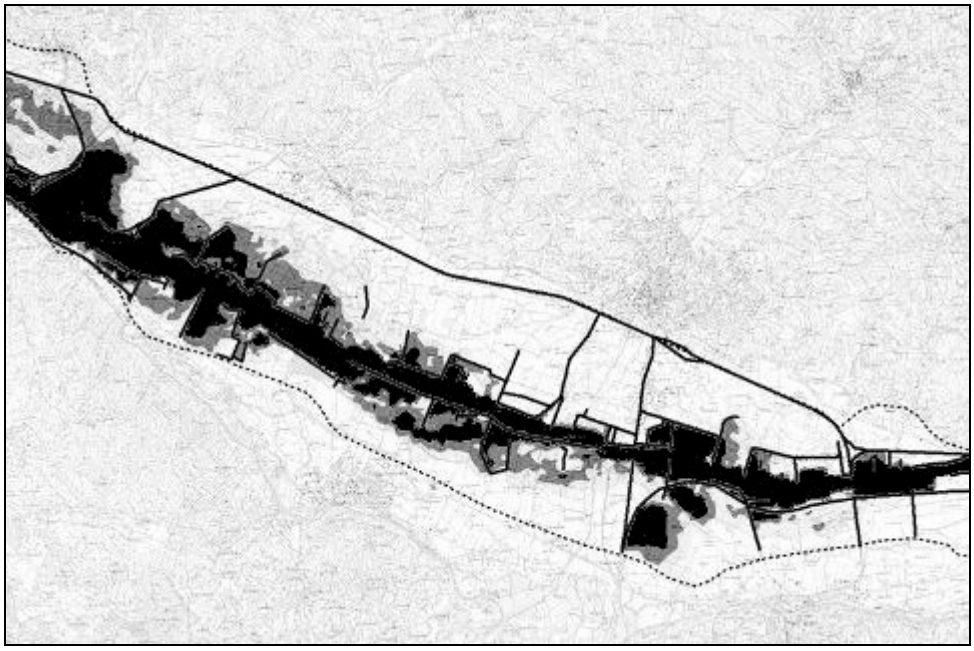

Figure 2: Depth field.

OBJECTWEB PROACTIVE MIDDLEWARE

The ProActive middleware is a 100% Java library, which aims to achieve seamless programming for concurrent, parallel, distributed, and mobile computing. As it is built on top of standard Java API, it does not require any modification of the standard Java execution environment, nor does it make use of a special compiler, pre-processor or modified virtual machine.

The base model is a uniform *active object* programming model. Each active object has its own thread of control and is granted the ability to decide in which order to serve the incoming method calls, automatically stored in a queue of pending requests. Active objects are remotely accessible via method invocation. Method calls with active objects are asynchronous with automatic synchronization. This is provided by automatic *future objects* as a result of remote methods calls, and synchronization is handled by a mechanism known as *wait-by-necessity* [4].

ProActive currently uses the RMI Java standard library as a portable communication layer, even if the transport layer may be changed by relying on an adapter object that it is in charge of protocol interface with ProActive. The communication protocols now supported by ProActive are *RMI*, *HTTP*, *Jini*, *RMI/SSH*, and *Ibis*.

In the context of the ProActive middleware, nodes designate physical resources from a physical infrastructure. They can be created or acquired. The deployment framework is

responsible for providing the nodes mapped to the virtual nodes used by the application [5]. Nodes may be created using remote connection and creation protocols. Nodes may also be acquired through lookup protocols, which notably enable access to the ProActive Peer-to-Peer infrastructure as explained below.

The main goal of the Peer-to-Peer (P2P) infrastructure is to allow applications to transparently and easily obtains computational resources from grids composed of both clusters and desktop machines. The P2P infrastructure has three main characteristics. First, the infrastructure is decentralized and completely *self-organized*. Second, it is flexible, thanks to parameters for adapting the infrastructure to the location where it is deployed. Last, the infrastructure is portable since it is built on top of JVMs, which run on cluster nodes and on desktop machines.

THE TELEMAC SYSTEM PARALLEL VERSION WITH PROACTIVE

The parallel version of the TELEMAC System is based on the Single Program Multiple Data (SPMD) programming model, which focuses on distributing the same program on different computers, each program computes some parts of data. The TELEMAC System uses the Message Passing Interface (MPI) [6] to distribute the computation among several computers.

In order to distribute computations, an MPI process needs, before to start computing, to have on the targeted host an MPI Daemon (MPD) previously running, i.e. a runtime for MPI processes. The deployment of the MPI program is handled by the MPI job launcher (MPIrun), which deploys MPI processes on specified MPDs.

TELEMAC System users have to follow steps for deploying theirs simulations, these steps are:

1) Editing the TELEMAC System simulation configuration file to specify which parallel mode to use and how many CPUs are needed.
2) Starting *MPDs* on each host.
3) Writing *MPI* configuration files with addresses of MPDs for each MPI process.
4) Starting the TELEMAC System computation: prepare simulation data and run the *MPI* job launcher.

All these steps imply for users to modify two configuration files and to start process (MPDs) on all involved hosts. Furthermore, users have to run all these steps each time they have to perform a simulation.

Users with good computer background can easily write scripts for automating these burden steps. Nevertheless, regular TELEMAC System users are not friendly with command-line tools, scripts, etc. Therefore, in this work we propose to automat all these painful steps.

Our solution turns this static deployment into a dynamic and automated one, which does not require special skills from users. The main points of the deployment mechanism which we will focus are:

- automating configuration file edition,
- providing a dynamic and self-organized infrastructure for deploying simulations.

Therefore, we propose a seamless system, which is composed of a graphical client interface for managing simulations and a self-organized desktop grid infrastructure for running simulations. In this work, we focus on this desktop grid maintained by the ProActive middleware.

In order to achieve TELEMAC System simulation deployment seamlessly, we have built a desktop grid, materialized by a Peer-to-Peer infrastructure. Concretely it is composed of ProActive daemons running peers on heterogeneous desktop workstations of INRIA, connected by a classical 100 Megabits Local Area Network (LAN). Therefore, this infrastructure allows applications to discover available resources, book some of them, and then deploys some active objects that implement user business logic.

Using this kind of architecture, we have developed a graphical job management application for deploying TELEMAC System simulations. Users only have to follow some basic steps:

1) Starting the job management application.
2) Selecting a TELEMAC System simulation file.
3) Selecting the desired resources from the desktop grid.
4) Starting computation.

Thus the application entirely hides connections to the desktop grid, acquisitions of resources and remote MPD deployment using active objects. It also automates all the MPI configuration file creation and modification edition process. Thereby the application eliminates all the burden steps required with the classical TELEMAC System.

EXPERIMENTATIONS

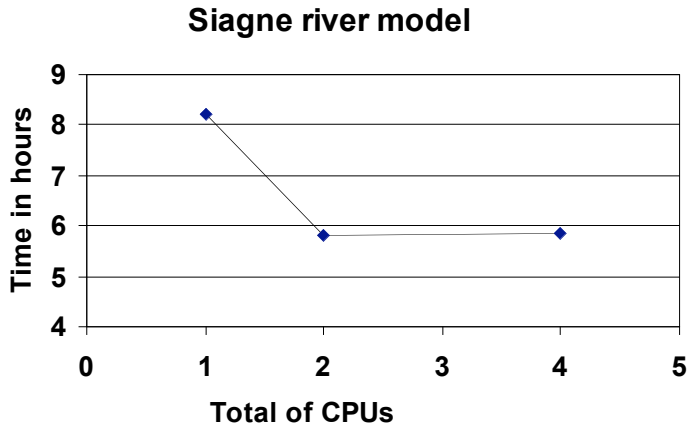

Figure 3: Benchmarks of the Parallel version of TELEMAC deployed on a desktop grid.

Experimentation results for the Siagne river model using the parallel version of the TELEMAC System are shown by Figure 3. These experiments have been run out using a desktop grid deployed at INRIA Sophia Antipolis lab; this grid is composed of Windows desktop workstations running Pentium 4 at 3.6 GHz. We notice that the execution times are very close for both two CPUs and four CPUs. We believe that this phenomenon is due to the parallelization technique that uses sub-domain decomposition. Therefore when sub-domains are too small, workstations are communicating more than they compute, thus increasing the overall computation time.

CONCLUSION AND FUTURE WORK

In this work we show that an MPI application, the TELEMAC System, with a static-based deployment requires significant efforts to be deployed on a desktop grid. Thus it involves end-users in burden tasks such as managing available resources and editing configuration files, both of them manually. Therefore we have developed a lightweight TELEMAC System job management application. This job manager addresses previous problems without requiring any source code modification of the original MPI application. In addition we believe that this tool is not only limited to TELEMAC System and can be commonly used to deploy and manages other MPI SPMD-like applications.

Experimentation shows that even with a small desktop grid there is a real execution time gain while running the simulation.

We are now working on optimizing the execution time of the MPI application by enabling the job manager to automatically and dynamically reconsider deployment regarding resources availability.

REFERENCES

[1] Brignolles C., Royer L., Calvet F., Viala Y., de Saint Seine J., "Modélisation en deux dimensions de la dynamique des crues de la Durance", *Proc. Systèmes d'information et risques naturels*, Montpellier, France, (2005).

[2] Hervouet J-M., "*Hydrodynamique des écoulements à surface libre"*, Presse de l'école nationale des Ponts et chaussées, (2003).

[3] Viala Y., Leroy H., Brignolles C., Sau J., "Modelling of the floods of the river Siagne", *Proc. 19th international congress of ICID – Use of water and land for food security and environmental sustainability*, Beijing – China (2005).

[4] Caromel, D., *"Towards a Method of Object-Oriented Concurrent Programming"*, Communications of the ACM 36, (1993), pp 90–102.

[5] Baude F., Caromel D., Huet F., Mestre L., Vayssiere J. , "*Interactive and Descriptor-based Deployment of Object-Oriented Grid*", HPDC-11, Scotland, pp. 93-102, (2002).

[6] Gropp W., Lusk E., Doss N., Skjellum A., "*A High-Performance Portable Implementation of the MPI Message Passing Interface Standard*" (1995).