Re-engineering Jake2 to work on a grid

Gonçalo Amador and Ricardo Alexandre and Abel Gomes
Universidade da Beira Interior, Instituto de Telecomunicações, Rua Marquês D'Ávila e Bolama, 6201-001 Covilhã, Portugal
Phone: +351-275319891, Fax: +351-275319899, e-mail: a14722@ubi.pt

***Abstract*—Re-engineering Jake2 (Quake2 port to Java) to work on a grid is not a trivial task. For that purpose, we had to migrate a networking client-server model to a grid-based model. Networking games is an area where considerable computational gains might be obtained using this migration, in particular management and optimization of resources (i.e. processor and memory usage).**

I. INTRODUCTION

Grids are collections of heterogeneous computation and storage resources scattered along distinct network domains. Grids provide tools that allow to users to find, allocate and use available resources [1,2].

In order to control and manage the various resources that grids can offer, various grid middlewares have been developed, namely: Optimal Grid [8]; Ice [6]; GridGain [7]. We have chosen GridGain because of its modern design (it also supports cloud computing) and because it is based on Java programming language, as adequate for networking systems and applications.

This paper deals with re-engineering (i.e. re-design and re-implementation) a multiplayer computer game, called Jake2 (Fig. 1), which is a Quake2 port to Java, with the objective of running it on a grid.

Fig. 1. Jake2 running shot.

II. JAKE2 ARCHITECTURE

Jake2 [10, 11] is a brute force port of the original Quake2 source code to Java. Like Quake2 it is a client-server networking game. The code is divided in eight modules (taken from [11]):

1) "client – key handling, visual effects, screen drawing and message parsing.

2) game – all game logic: triggers, monsters, weapons, elevators, explosions, keys and secrets.

3) qcommon – common support functionality (file system, message buffer, config variables, checksums and splash screen).

4) renderer – 3D graphics displays (jogl, fastjogl, lwjgl).

5) server – world handling, movement, physics, game control and communication.

6) sound – sound implementations: joal, lwjgl, and a proprietary native java sound driver.

7) sys – methods dealing with the operating system (network, keyboard handling).

8) util – 3d maths, some libc methods, etc."

Jake2 has two playing modes: single player and multiplayer. When a game starts off, a client and a server are created locally (Fig. 2). In single player mode, the client and the server run on the same machine. When a single player game is started, the player connects to its local server. In multiplayer mode, the client can either join to or start a network session; joining a network session means that a client connects to a remote server, and its local server becomes idle (it exists but does not serve any client). Starting a new network session implies that client's local server becomes globally visible to other clients wishing to join the session. It is also possible to create a dedicated server (only does server work and is globally visible), setting the local client idle.

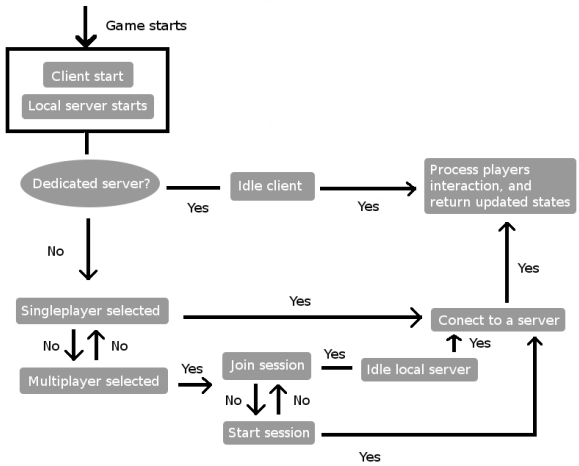

Fig. 2. Jake2 working logic.

Each client passes its state information to the server (local or remote depending of the previous referred choices), the server processes the received data (world handling, movement, physics, game control) and returns state updates to the connected clients, so the clients can redraw their world area**.** The game multiplayer mode has no bots (artificial intelligence controlled entities or agents) support.

III. GRID ARCHITECTURE

Grid computing is a kind of parallel computation. A grid can be described as a network of interconnected computers, where parallel work is done. In grid computing, issues like communication time is a more pertinent issue than in supercomputers, where parallel computation is done locally. On the other hand, a grid is more scalable than a supercomputer, because we only need to add new, but not necessarily homogeneous, network nodes to expand its computational capacity.

By definition, a grid work is a set of tasks performed by the grid work nodes. This means that a work is divided among the number of existing nodes, where a node is a sub-grid that may be composed by one or more computers. There are two main types of nodes, control and work nodes. A grid network has several control nodes that ensure work division and gathering of results. Every control node is aware of the other control nodes, while a work node only sees the work nodes directly connected to its parent control node (see Fig. 3).

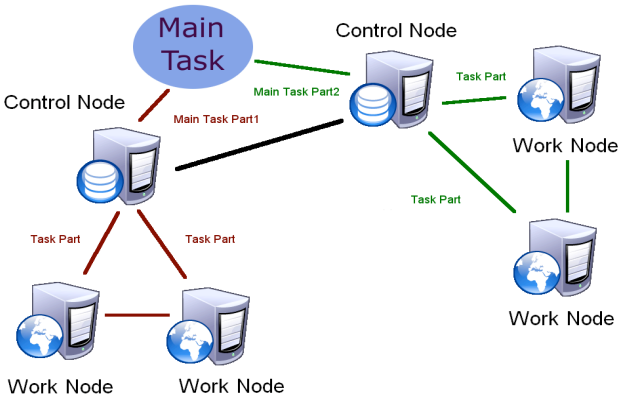

Fig. 3. Grid networks doing a task division.

As usual in parallel computing, there are two ways of dividing tasks by work nodes: *split/reduce* and *map/reduce*. The split mechanism equally divides the main task by work nodes regardless of which ones are going to accomplish the subtasks (Fig. 4). On the contrary, the map mechanism assigns the subtasks to specific work nodes. This means that, unlike the map logic, the split logic assumes that each node has the same work capacity.

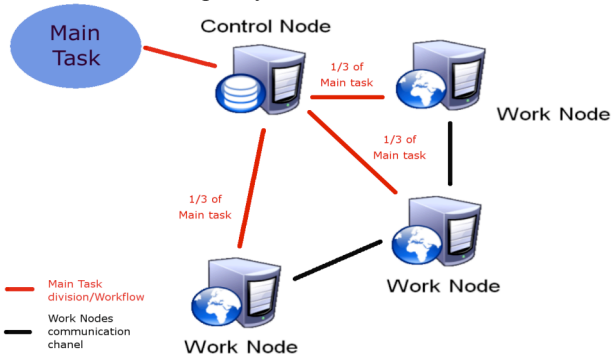

Fig. 4. Work division by non-specific nodes using split.

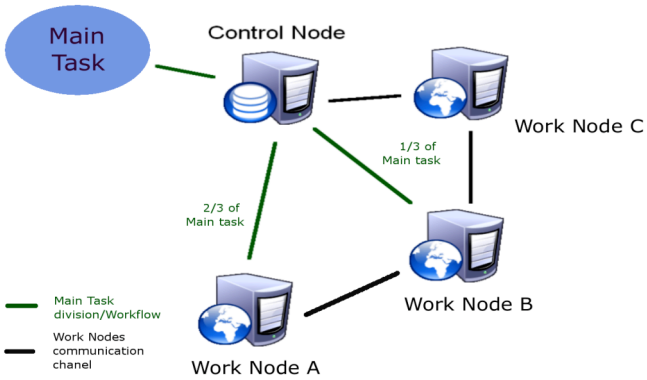

Fig. 5. Work division by specific nodes using map.

IV. GRID-BASED JAKE2

Jake2 is a client-server FPS (First-Person Shooter) computer game that works in real time up to a maximum of 32x8=256 clients or players (for 8 servers with the maximum of 32 players per server). Therefore, Jake2 is not a MMOG (Massively Multiplayer Online Game). Similar to other FPSs, most of the processing work of Jake2 is carried out in the server side (player-player and player-world interaction), after which the server communicates the updated data back to the players. The Jake2 world consists of 32 zones, where each zone corresponds to a game level. That is, the server takes care of 32 clients maximum playing in one of the 32 zones.

In theory, we do not need to have a grid to transform Jake2 into a MMOG. It would be enough to have 32 server computers, with each server doing the computations required by each level, resulting in a maximum possible number of 32x32=1,024 online players. Note that a game is a MMOG if it supports more than 1,000 online players. However, this scaling strategy does not overcome the problem of having crowded zones while others are empty. Using a grid middleware, which is inherently capable of dealing with unbalanced processing load over different game levels, can solve this problem.

In our work, we have used the GridGain grid middleware. Jake2 was re-designed to run on GridGain. In particular, we changed the client and server modules, and incorporated a grid-specific communication module.

In respect to the client, the changes were:

- instead of connecting to a LAN (Local Area Network) server or to configure a connection to a global server that processes one level of the game world, we connect to a cluster server running one out of 32 levels;
- the number of servers a client can connect to is now 32, i.e. the client sees all the servers of the cluster when he/she has to select a level to play.

The changes carried out in the client had nothing to do with the grid. On the contrary, the server changes were related to the grid infrastructure, namely:

- each cluster server is now a grid control node;
- a control node may also work as a work node;
- each control node uses the map/reduce work division mechanism.

We do not use the split/reduce policy because it is less efficient in the distribution of MMOG tasks. This inefficiency arises when there are more nodes than tasks to be performed. In these circumstances, some tasks are repeated by different nodes because of the distribution policy based on workload equality of the split/reduce mechanism.

To take advantage of the grid infrastructure, we had first to identify which server methods could be parallelized. Without it there is no gain in parallelizing a sequential method on a grid because there is no gain in processing time, with the drawback of having an extra communication time.

The parallelized methods were those related to the physics engine, including collisions. The world handling methods (e.g. world partitioning methods) are also parallelizable as described in [3,4], but they were not re-designed and re-implemented yet. Processing a parallelized method involves the distribution of its tasks among other grid work nodes from a control node (Fig. 5). After finishing their tasks, the work nodes return the results to the control node, which passes such results (i.e. updated states) to clients (Fig. 6).

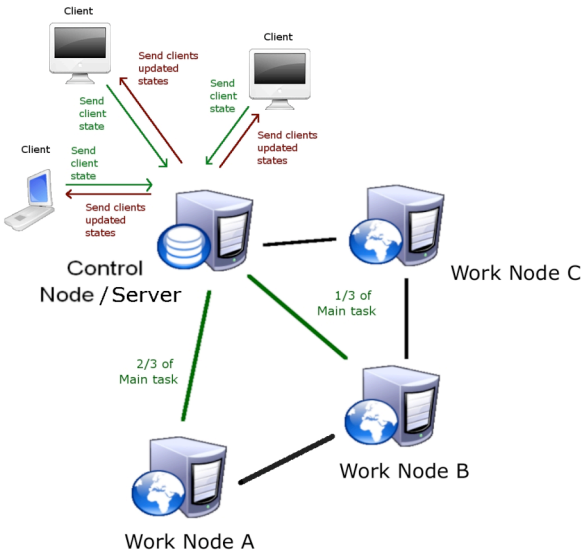

Fig. 6. Jake2 grid architecture.

For example, when a player moves around the game world, it is necessary to check whether any other player lies within his/her area of interest and to proceed accordingly to establish possible interactions. Assuming that we have N+1 players, the control node that is running the `move` method delegates the N interaction tests between players into available work nodes.

We could use another grid architecture for Jake2. For example, we could transform clients into work nodes, each one of which only takes care of its area of interest, also called interaction area. They communicate their updated state to the server, which keeps track the state of interacting players and sends such updated state to each player that requires it. This way, each server ensures that each player gets all the needed data to handle its area of interaction.

V. GRID MANAGER

During GridGain usage learning process, we needed to create several work nodes over our network. Since no graphical interface application was available to the GridGain community, it was built a small graphical application to create remote or local work node sets (a given number of work nodes per machine). This application is still a prototype that can be extended by incorporating other interesting features, namely an info collector that gathers data from the nodes and redirect such data to the graphical application. Another interesting feature could be to allow for node shutdown.

Currently, the grid manager uses OpenSSH [9] to communicate remote node activation commands. OpenSSH is a free implementation of the SSH1 and SSH2 protocols that allows remote execution of commands between two machines, after some sort of authentification. To allow a key pair authentication, for the password login requirement to be bypassed, some configuration scripts were made for for Linux (OpenSuse and Ubuntu) and Windows (XP and Vista).

VI. CONCLUSIONS AND FUTURE WORK

Grid computing is a suited technology to solve problems that require intensive processing. But in the beginning, it was not clear that it could cope with real time applications as needed in networking computer games. The real bottleneck in using grid computing for games is whether the communication time between nodes increases to a point of degrading the overall game performance. As seen above, this problem can be surrounded with a correct selection of the parallelizable methods.

Thus, applying grid computing to networking games such as massively multiplayer online first person shooters (MMOFPSs) or massively multiplayer online role playing games (MMORPGs) is not a trivial task. This requires re-implementing the original code from scratch or, as in our case, redesigning much of the code.

We have modified Jake2 to run on the GridGain infrastructure, as illustrated in Fig. 6. This has led us to re-design the server code; in particular, nine server methods have been parallelized. The result is a more scalable Jake2, i.e. more adaptable game to an increasing number of players.

REFERENCES

[1] Stephen Jarvis, Nigel Thomas, and Aad van Moorsel, “Open issues in grid performability”, I.J. Of Simulation, vol 5, pp. 3–12, 2004.

[2] Nigel Thomas, “Challenges and opportunities in grid performability”, Technical Report 842, School Of Computing Science, University Of Newcastle Upon Tyne, School Of Computing Science, Claremont Tower.

[3] Sergio Caltagirone, Matthew Keys, Bryan Schlief and Mary Jane Willshire, “Architecture for a massively multiplayer online role playing game engine”, Journal of

Computing Sciences vol 18 , pp. 105–116, December 2002.
[4] G. Deen, M. Hammer, J. Bethencourt, I. Eiron, J. Thomas, J. H. Kaufman, “Running Quake II on a Grid”, IBM Systems Journal, vol 45, 2006, pp. 21-44.
[5] James Kaufman, Tobin Lehman, Glenn Deen, John Thomas, “OptimalGrid -- autonomic computing on the Grid”, developerWorks.
[6] Ice: http://www.zeroc.com/, last access 22-01-09.
[7] GridGain: http://www.gridgain.com/, last access 22-01-09.
[8] Optimal Grid: http://www.alphaworks.ibm.com/tech/optimalgrid, last access 22-01-09.
[9] OpenSSH : http://www.openssh.com/, last access 22-01-09.
[10] Jake2: http://bytonic.de/html/jake2.html, last access 22-01-09.
[11] *Unofficial Jake2 Resource* Jake2: https://wiki.informatik.tu-chemnitz.de/bin/view/RST/Jake2, last access 22-01-09.