A Virtual World Distributed Server developed in Erlang as a Tool for analysing Needs of Massively Multiplayer Online Game Servers

Michał Ślaski
Erlang Training and Consulting Ltd.
London, United Kingdom
michal@erlang-consulting.com

Marcin Gazda
AGH University of Science and Technology
Al. Mickiewicza 30
30-059 Krakow, Poland

ABSTRACT
At present massively multiplayer online games allow several thousands of players to stay in a single, persistent virtual world. Because of the fast growing interest in this type of servers, we started researching their efficiency and scalability. Our target was an analysis of the MMOG server, which could service up to 1000 players in a single virtual world. We made an assumption that the server will be distributed and running on a dedicated cluster. As the implementation platform we chose Erlang/OTP its main advantages being integration with a distributed database, soft real-time and supporting distributed applications. In this paper we discuss the realisation of the project, and practical aspects of the measurement of server parameters.

Keywords
Massively Multiplayer Online Game, Erlang, Load Testing, Distributed Server.

1. INTRODUCTION
Massively multiplayer online games (MMOG) are a dynamically developing segment of the computer games industry. Even though there are many difficulties that you need to overcome while building this type of systems, there has been a significant increase in the interest in MMOG throughout the last couple of years. In the year 2004 the total income from selling online games was over 1.5 billion US dollars. In the year 2006 the total income is predicted to be twice as big.

In classic multiplayer games, like Quake, the number of players usually is under 20. At present one of the most popular MMOG in Europe and USA is World of Warcraft released by Blizzard Entertainment, in which several thousands users play on each server. MMOGs are characterised by big demands for the the system and for the network infrastructure. Thousands of players staying concurrently in the same virtual world, interacting with each other and changing the state of the game environment, generate heavy network traffic and a heavy server load.

Not many titles were as successful as the World of Warcraft. Most of them had technical faults and lacked in attractive gameplay. To provide a good level of gameplay, every user needs to have an up-to-date information about the state of the virtual world. This state changes every time when the user takes an action (like moving his character or collecting an object) leading to the conflict between capacity and coherence. It is impossible to guarantee that a dynamically shared game state will change frequently and that every user will have a permanent access to the same and most current state.

2. PROTOTYPING
The main reason of failures in creating MMO games is the fact that you can only fully test it at the end of the developing process. All of the leaks and mistakes in the design can be noticed during beta tests when thousands of users start playing it. It is important to prototype every solution in order to simulate players and check if the solution is appropriate. During the phase of prototyping you can experiment with the functional and technical aspects of the system and you should determine the architecture of the system.

2.1 Using Erlang/OTP for the prototype
Usually prototypes are developed on hardware not as strong as the target one, so it is important that the platform used for prototyping is compatible with different operation systems and it is not dedicated to a specific one. Another requirement for the platform is the possibility of a quick and easy development process. There is a need for mechanisms that can support distributed architectures, because these are the most promising directions of research. Supporting high availability and efficient internal communication are also significant. This is why Erlang Open Telecom Platform was chosen. Important features from our point of view are: open source, its own virtual machine, light processes and the distributed database Mnesia.

Handling a big amount of users requires concurrent oriented programming. This can be accomplished with Erlang processes – one process for each user. There is a need of scalability of the system, what can be realised by distributing the server on a cluster of machines. Erlang allows processes to communicate with each other by knowing only their PID and without knowing if they are run on the same node, so you can treat the cluster as one coherent system. Such persistence allows you to build prototypes effectively.

3. IMPLEMENTATION

For the sake of research a game client in Java and with Java3D library was developed. The application provides functionality similar to role playing games. Users can move their characters, collect objects, chat with each other and do some magic. Messages with the information about user actions are sent to the server and then dispatched to other connected users.

The server was implemented in Erlang. All of the game data is stored in Mnesia tables. The Mnesia's scheme was configured to keep the replicas of all tables on all nodes. Most frequently used data like socket binding or player's position are stored in ETS hash tables.

Erlang processes are organised in a supervision tree. The root process on every node is responsible for starting services assigned to this node. Services were implemented with the *gen_server* behaviour. There are two kinds of services: processing network traffic and processing user's actions. When 'network processing' service is run, the node becomes an access point and it starts to listen on a TCP port for new connections. When 'action processing' service is run, then the node takes part in distributing the computations.

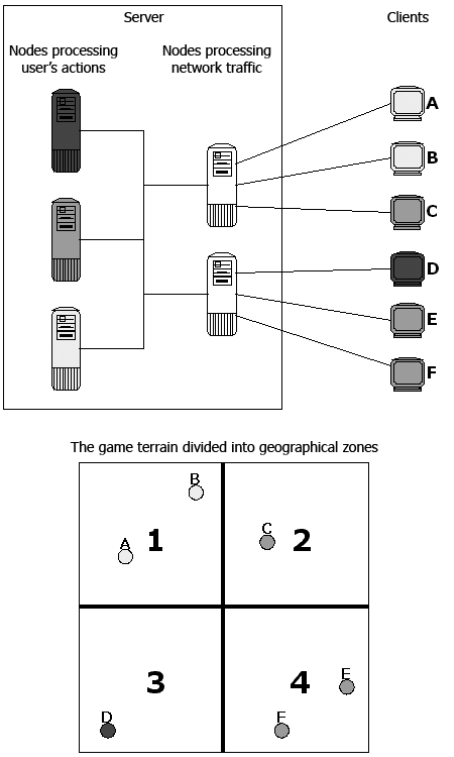

Figure 1. System's architecture

The game terrain is divided into geographic zones assigned to the nodes processing the player's actions. Every player is assigned to the zone in which he is standing. Processes controlling the players placed within the same zone are run on the same node. When the player moves to another zone, its controlling process is moved to the appropriate node. This operation needs a lot of calculations, so the algorithm was implemented to prevent often zone switches. The player is switched into another zone not when he crosses a zone border, but when he gets out of the zone range.

4. SYSTEM PERFORMANCE ANALYSIS

When the implementation of the system was finished, a game session with real players was organised. Seven people were playing for several hours so we could log all of their actions and carry out analysis on how often statistically a user takes an action. Then we generated scenarios for IDX-Tsunami which is a distributed load testing tool that can simulate TCP clients. A plug-in for IDX-Tsunami was developed to support our protocol and to log the server performance.

The research was done in two phases. During the first one two types of architectures were tested. The architecture of the first type is built by single-function nodes, which means that every node is processing either network traffic or player's actions. The architecture of the second type is built by double-function nodes, which means that every node provides both services.

For every architecture the maximum number of players that it can handle was determined. We determined the number by analysing several different indicators, such as the time in which the server replies for messages, outgoing network traffic, number of outgoing packets and the utilisation of the CPU on the server side.

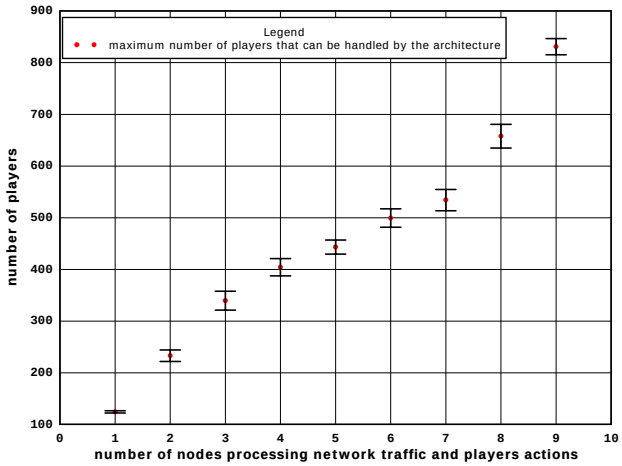

Figure 2. Architectures with double-function nodes

According to research, zone based solutions are highly dependant on the distribution of players in the world. We made an assumption that players are spread around the world equally and this kind of situation could be possible if the game world is big enough. But the fact is that when many players are standing in the same region, like during battles, the server can be overloaded.

During the second phase we worked on synchronizing issues. Let imagine that two players are looking at the same mushroom lying on the ground. One of them moves near the mushroom and picks it up. Then the message is sent to the server. Before the server passes this message to the second player, he can still see the mushroom, so he can try to pick it up. If he dose so, obviously he will fail, because the mushroom is already collected. We measured the probability that a player would or would not fail in two kinds of situations: when the server is not overloaded and when it is overloaded. In the second case players more often cannot take successful actions, because the state of the world which they can observe is desynchronized from the state on the server.

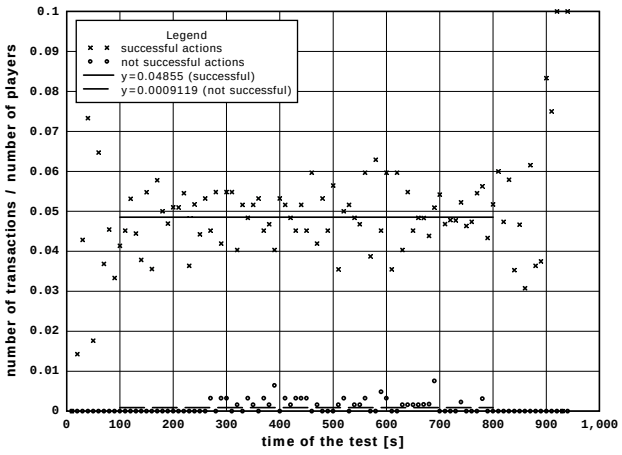

Figure 3. Server not overloaded, more successful actions.

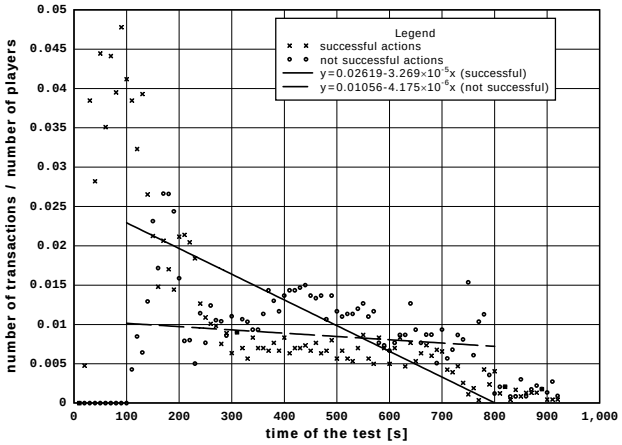

Figure 4. Server overloaded, more actions fail.

5. CONCLUSION

To summarize we found the Erlang Open Telecom Platform very appropriate for developing prototypes of the distributed MMOG system. Because of the fact that you can quickly build a solution and then experiment with it, you can examine different algorithms in a relatively short period of time.

6. ACKNOWLEDGMENTS

University supervisor: Stanisław Ciszewski, AGH University of Science and Technology, Kraków, Poland.

Nicolas Niclausse, author of the IDX-Tsunami tool, which was used for the players' simulation.

7. REFERENCES

[1] J. Armstrong, R. Virding, C.Wikstrom, M. Williams, Concurrent Programming in Erlang. *Prentice-Hall*, 1996.

[2] A.G. Bosser, Massively Multiplayer Online Games: matching Game Design with Technical Design, *IMAGINA*, June, 2004.

[3] IGDA Online Games SIG, 2004 Persistent Worlds Whitepaper, *http://www.igda.org/online/*, December, 2004.

[4] B. Knutsson, H. Lu, W. Xu, B. Hopkins, Peer-to-Peer Support for Massively Multiplayer Games, *INFOCOM 2004*, March, 2004.

[5] D. Saha, S. Sahu, A. Shaikh, A Service Platform for On-Line Game, *Proceedings of NetGames 2003 Workshop*, May, 2003.

[6] MMOGCHART.COM, *http://www.mmogchart.com*

[7] K. Milligan, Massively successful - MMORPGs come of age, *http://keathmilligan.net/view.php?id=448*, December, 2004.

[8] IDX-Tsunami distributed load testing tool, *http://tsunami.idealx.org/*