Database and State Replication in Multiplayer Online Games

Paula Prata ¹,²
Etelvina Pinho ²
Eduardo Aires ²

¹Institute of Telecommunications
²Department of Computer Science
University of Beira Interior (UBI), Covilhã, Portugal
Outline

- The problem: availability of multiplayer online games
- State of the art
- Fault Tolerant – Dungeons & Dragons (FT-D&D)
- Simulations of several simultaneous games
- Conclusion
Online Games

- Highly demanding applications
- Millions of simultaneous players (MMOG)
- Highly profitable market

Example: “World of Warcraft”
  - millions of subscriptions
  - profit of 471 million dollars (in 2006)
Requirements

- **High scalability**
  - Achieved with replication and load balancing techniques

- **High availability**
  - Achieved with fault tolerant mechanisms

- Few hours of game downtime represent significant losses
- Game credibility
- In 2006 a fault in “World of Warcraft” cost about $26198 per hour
Objectives:

- Build highly available online games using:
  - Database replication
  - State replication

- Build a fault tolerant version of the Dungeons & Dragons game: FT-D&D

- Study the cost of replication with and without faults
Very competitive market

- Little information about fault tolerance (FT) on MMOG’s

There are some studies about data replication

There are some studies about state replication

Mainly concerning the scalability
Limitations of existent studies:

- The impact of a fault was not studied
- They did not include the replication of database and state simultaneously
- In relation to state replication, no quantitative studies are known on the impact of the performance of the games
Based on Dungeons & Dragons game

Medieval fantasy game

Multiplayer game

Browser based

Data persistence

Each game is played by 5 players representing the characters: Master, Cleric, Rogue, Fighter, Wizard
3. FT–D&D

3.1. The game

3.2. Data replic.

3.3. State replic.

3.4. Summary
Data replication implementation:

- Two MySQL database servers
- Following a “lazy primary copy” protocol
- Model Master – Master in active-passive mode
- Asynchronous protocol that allows role change between the master and the slave.
1. The problem
2. State of the art
3. FT–D&D
4. Simulations
5. Conclusion

3.1. The game
3.2. Data replic.
3.3. State replic.
3.4. Summary

First International Workshop on Networking and Games (N&G 2010)  Perth, Australia, 23 April 2010
If a fault occurs:

- In the master,
  - The requests are redirected to the slave
  - The slave becomes master

- In the slave,
  - Has no impact in the user

- In both cases a fault in the database server is transparent to the user
State replication implementation:

- **Cluster**
  - Two Tomcat web servers
  - Self discover of members

- One “load balancing” server
- DNS server
- Synchronization of clocks (NTP protocol)
1. The problem
2. State of the art
3. FT–D&D
4. Simulations
5. Conclusion

3.1. The game
3.2. Data replic.
3.3. State replic.
3.4. Summary

First International Workshop on Networking and Games (N&G 2010)  Perth, Australia, 23 April 2010
If a fault occurs:

- In the primary,
  - All requests are redirected to the secondary

- In the secondary,
  - The Tom Cat instance is removed from the cluster
  - Therefore stops receiving/sending messages

- In both cases a fault in the web server is transparent to the user
In FT-D&D, data and state replication allows transparency of:

- Fault in any database server
- Fault in any web server

The “load balancing”:
- Refers to the web server fault, it is used to redirect the clients to the secondary server
JMeter

- Load tests and measure performance
- Multithreading tool
  - Allows the simulation of several games simultaneously

- Simulation of 1, 15, 30 and 60 simultaneous games
- That is, 5, 75, 150 and 300 players
Test of 60 simultaneous games

- 20 low level games
  - 4 monsters of low level difficulty

- 20 medium level games
  - medium level monsters, one level change

- 20 high level games
  - high level monsters, more than two level changes
4. Simulations

4.1. JMeter

4.2. Games

4.3. Results
<table>
<thead>
<tr>
<th></th>
<th>% of throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 game</td>
</tr>
<tr>
<td>1. Without replication</td>
<td>100</td>
</tr>
<tr>
<td>2. State replication</td>
<td>90</td>
</tr>
<tr>
<td>3. Database replication</td>
<td>81</td>
</tr>
<tr>
<td>4. Database/state replication</td>
<td>76</td>
</tr>
</tbody>
</table>

Fault Simulation

<table>
<thead>
<tr>
<th></th>
<th>% of throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. Web server fault</td>
<td>61</td>
</tr>
<tr>
<td>6. Database fault</td>
<td>4</td>
</tr>
<tr>
<td>7. Web server/db fault</td>
<td>4</td>
</tr>
</tbody>
</table>
- Small impact of state replication
  - Decrease of 5% in the game throughput

- Considerable impact of data replication
  - Decrease of 33% in the game throughput

- Significant impact of a fail in database server
  - Decrease of 55% in the game throughput
When the number of players increases:

- The impact of replication becomes smaller
- There is a decrease in the distance between the throughput of all the cases
- The main factor to the decrease of performance is the increasing number of players
Future work

- Use semi-asynchronous data replication
- Evaluate the fault coverage by fault injection
Questions ...