ISSN: 2321-7782 (Online)
Impact Factor: 6.047

Volume 4, Issue 9, September 2016

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

A Study on Databases in Video Games

Kartik Nagpal[1]
B. Tech Scholar
Department of Information Technology and Engineering
JMIT, Radaur, Yamunanagar – India

Nitin Nanda[2]
B. Tech Scholar
Department of Computer Science and Engineering
JMIT, Radaur, Yamunanagar – India

Abstract: In this paper, we outline the uses and requirements of the databases in Video Games. We outline the architecture of different types of video games and how the database technology plays a big role in their design and output. We will study the different types of databases and designs which can be used, and their advantages and disadvantages over each other. The requirements of different video games are different, so the database used will also be different. The Massive Multiplayer Online games require very active database which changes the entries instantly and from all over the network. From this, we identify several new research directions to further improve the utilization of this technology in computer games.

Keywords: Video games, Database, Massive Multiplayer Online Game, Distributive Databases.

I. Introduction

Video Games have played a major role in our lives. They are a big portion of the entertainment of an average person and most of us have played video games at least one in our lives. Gaming Industry has developed so much, in the last decade, that it has covered almost every platform. On each platform, there is a dedicated industry and big corporations has been working to develop new and better games. According to the Entertainment Software Association estimates, the video games sales in 2006 were $7.6 billion dollars. The biggest game of recent times, Grand Theft Auto 5, alone has collected more than $2 billion dollars in sales and microtransactions. Clearly, people has been spending more of their money and time on video games. The video game industry has been constantly growing and expanding its business to new platforms. The mobile gaming industry is also growing at a remarkable rate and constantly growing. Games can also be used in areas like training and education or modeling and simulation.

Databases have a really important role to play in video games. With the time, the size and complexity of games have also been increasing. The games needed to be handled properly and certain elements were added later. During the starting years, Video Games had a single level gameplay where the user needs to play as further as he can. But with the time, the concept of the Video Games has changed. Now, the games provide the unique experience to each player according to his style of playing and decisions he makes during playing. To handle the complexity of modern video games, databases are used. These databases store the player progress, player's data and other things. Different types of databases are used which varies according to the type of the video game. Offline games only need a centralized database, but MMO's cannot work with centralized databases. They need a database which can collect data from all over the network and instantly. The margin of error is very less in MMO's because it will put down the whole gaming network of that game which will result in a huge loss.

In this paper, we will discuss the types of Video Games and types of databases which are required.

II. Types of Databases used in Game

There are multiple options to choose from when we are talking about the type of database to be used in our gaming application. It depends primarily on the type of game and its gameplay i.e. offline game or MMO (Massively Multi-Player

Online) gaming environment. Considering the fact that games, whether online or offline, produces huge amounts of data which needs to be updated periodically (not instantaneously, sometimes) in order to maintain consistency throughout the database.

1. Offline games

Undoubtedly, centralized databases will be used for this purpose as everything has to be stored on the local machine. RDBMS does this job best because of their simple implementation and almost every work can be performed with the help of basic commands like Select, Insert, Delete, Update. The data is stored in database objects called tables. It further consists of fields, rows, and columns. Despite the ease of usage, it is really hard to scale to such huge levels, so as to manage data of up to 100,000 users playing simultaneously in online.

2. Massive Multiplayer Online games

More user interactivity and real-world graphics have attracted huge numbers of gamers to play online and share their data and achievements. So, proper data storage and retrieval techniques have to be considered. For this, Distributed database has proved to be the best option of all. There is a local server and it's data warehouse is present in each geographical location, say a country. All the local hosts, that is the gamers are connected to the local server and therefore, the burden is shared among all the servers present globally. These individual servers have to be connected with each other in order to maintain consistency and update their local copy of data and also to tackle the scenario when one of the servers fails. In such cases, the traffic may be transferred to the nearest server.

III. Game Architecture

To understand, how the database is used in a game we need to see the modern games architecture. The use of a database depends on from game to game. In some games, the database is only used once every session to store the player's progress and in other cases, it is used every second to update the data. In the case of MMOs, the database is an integral part of the design and it can be used to store the character behavior.

Describing a video game architecture is very difficult due to the fact that every game has a unique design and working. While most of the game design lies in creating the game content, game characters, sound effects, and animations, most game studios spend ample of time in creating the game architecture for their game. This is done in order to tune the architecture of the game according to the market standards a smooth functioning of every aspect of the game. The database of a game can be anywhere in the architecture and is largely dependent on the type and genre of the game. Describing a game architecture is only possible if it is discussed according to the genre of the game because different types of games have absolutely unrelated designs and architectures.

The classification of the games can be done into three categories on the basis of how the players are connected to each other. The non-networked games are those games in which there is no network in between different players and player interacts with the local elements of the game. The architecture of this category of games is rather simple as compared to the others. There is no need of a distributive database, a local database will work properly. Non-persistent games are those which are connected to each other through a network but works only for a session. The session details are absolutely wiped out as soon as the session ends. In this kind, when players return after a session, they are the initial state as others who are playing for the first time. A famous example of this category is Counter Strike where users are at initial state whenever a new session is started. Persistent Games are those in which the user actions are preserved and the user can evolve its character according to his wish. This kind of games gives the benefit to the experienced players.

The usage of the database is done for various purposes like entering or updating players' details and progress, updating players' various actions, getting new updated information, etc. The position and role of the database system are varied according to the game. The database system plays an active and important role in Massive Multiplayer Online Games. It is updated continuously to provide seamlessly unrestricted access and feeling of playing in a virtual environment. The database also stores every player detail and their characters' behavior. A database system should be able to satisfy all the requirements of the game like scaling, backup, good performance, and fault tolerance. The database will also provide the opportunity of monitoring the data and analyze it. This task is very difficult to perform and sometimes, it is impossible to collect results for usage. This data can help in future actions and ways to improve the gameplay and the whole game.

IV. Basic Requirements of Video Game

Modern Video Games produces a large amount of data and this increases the complexity of handling this data. The data generated in video games require a high level of scalability, fault tolerance and performance. As all the data has to be processed in real time, traditional approaches like MySQL do not support such extensive usage and data transfer with periodic updates. Moreover, with the introduction of Artificial Intelligence in modern games and the best ever system configurations, more instructions can be processed per second and thus, efficient and reliable data management techniques have to be followed. The basic features which are required in a database system used in a video game are discussed below.

Scalability: In general, scalability refers to the ability to improve the systems performance by adding more resources to it. Distributed systems are designed for horizontal scaling, that means it can be made to handle more workload by adding more machines to the system. Considering an example in which 80% of the disk space is occupied by the database, if another machine is added, then each will balance the data load and thus both the machines will have more free disk space. Similar is the case with RAM and CPU functioning. Scalability in Distributed database also offers minimum shared resources. This is most needed for Massive Multiplayer Online Games where most players play during the day time and load is much higher than any other time. The extra resources will only be needed at this peak time and this will decrease the total handling cost.

Fault Tolerance: It is the property of a system to continue its operation in the case of failure of a single or multiple components. In some cases, it is not of prime importance, but in business applications and gaming, the crashing of the system or unresponsive application cannot be ignored. Fault tolerance is achieved by replication and redundancy and the basic requirements include no single point of failure and fault isolation of the failing component. After the fault has occurred, the system must be able to recover to the last possible state. This is most needed when the system has been recently updated and there is a high possibility of a fault. A single fault can crash down the whole system if it is not handled properly.

Performance: The performance in distributed database is achieved by allowing the transaction to process on more than one machine. As it is known that the response time of memory (hard disk and other mass storage devices) is less as compared to that of a CPU; having multiple machines to process a huge number of accesses, generally 10,000 to 1,00,000 per second, is best achieved by distributed databases. Each game has a certain standard of performance which must be achieved in order to enjoy. If this performance is compromised, this will take a toll on the business of the game. A shooting game which requires at least 30fps for smooth running, if played in 20fps will not be able to provide that level of accuracy which is expected by the player. Performance can be affected by the hardware used, database system, network connection and bugs.

V. Databases in Non-Networked Games

To understand the requirements of databases in video games, we need to understand the architecture of games. Almost every game has its unique elements and assets. But the basic working of every video game is almost the same. The architecture for every game starts from the designing part. The game working, characters, levels design and the concept is decided. After the game is designed, it needs to be provided the input which will be provided by the user. The user will make changes to the virtual game world. These changes can be of any size and vary according to the user and the flexibility the game provides to the user.

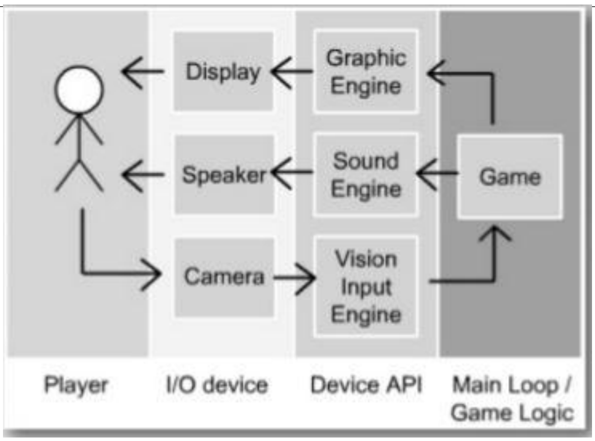

Fig. 1: Offline Game Architecture

In modern games, the player progresses and explore the world, tasks, and missions. The game needs to store all the player's data and progress at a place which can be accessed at any time. This is only possible by the use of a database and this database needs to be regularly updated and saved. In offline games, the database need not be updated constantly and can be updated at a regular interval of time. This database will be stored on the user's device and can be accessed without the internet or network. The database is also used to store the game data like the scenes, characters, elements, assets and other data. This database need not be updated, but this should not be altered to avoid any malfunctioning in the game. The database suitable for offline games is a centralized database.

The centralized database keeps all the data in one place. It is perfect for keeping data which is of big size, and which is needed by only a single user. This database cannot work on a network and is not shareable. A centralized database is one which is located, stored and maintained in a single location. This kind of database is easier to handle and the security is maximum. The user has all the control with him and this provides better data security. This type of database also provides easier data portability and database administration. A computer game like Grand Theft Auto 5 which is played offline most of the times and uses the internet just verifying the user account and progress, a centralized database will work. But this type of database creates a problem when the online functioning and gameplay features come into play. This type of database fails to handle and often creates problems. We need a different database which can offer to handle the networked architecture of the game.

One area in which databases can definitely help the game to run is the game engine. In a game, the world is created using the game engine and every game has a main loop that animates the game. The animation changes according to the players' movement and his actions. Every time player does something in the game, the main loop is passed. During a single pass, the game engine does multiple tasks. It computes the behavior and status of the elements and assets of the game. It updates the state of the simulated world according to the changes made to the world. At last, it redraws the world with the updates elements. During this process, the database is used to store the old data and the new updates. It is being constantly used to make the game work.

VI. Databases in Non-Persistent Games

In these games, the players are connected to each other through a network. This network is usually a local area network and the game is played among a small number of players. The player and details are created for the session and every detail is deleted after the session. If a player leaves the game in between the session, all of his data will vanish and the player can rejoin only as a new player. These games are similar to non-networked games except in this type, players are connected to a network.

The database is used to store the data when the session is up and running. It is also used for the players' data but it is wiped out whenever the session ends or terminated. The example of this type is the game "Counter Strike". It is a multiplayer non-persistent game where players can play against each other as two teams named Terrorist Forces and Counter-Terrorist Forces. It is First Person Shooting Game and its real MVP is its speed and low requirements. This game is quite famous and old. The players play in a session and the data is stored in a common database which is created on the host's computer. The database stores the team score, players' personal score, the health of a player, players' guns and other assets, etc. This database is deleted as soon as the session ends. Next time, when a session is created, the database will be again created on the new hosts' computer. This category of games contains single level games where interaction between the different layers is most important. In this game category, centralized databases will not work properly due to the fact that multiple sources of data are present and data needs to be transmitted to them back again and again. Distributive databases will work perfectly for this game category.

In this, the database is, usually, divided into several parts and these parts are used to store a different kind of data. One part may be used for players' progress and character details. Another part may be used for storing the messages between the players. Another part may be used to store the network details and players' computers details. The database is used for various purposes and it will work properly on creating the right queries and design. It should be made sure that there is no data loss or faults.

VII. Databases in Persistent Games

Persistent Games are those which provides the environment in which the players' actions are preserved and can be used later. These games use a database which has a central role to play in the game architecture. These games almost always have a client-server architecture. The most common example of persistent games is any MMO game. These games use the database as central and use it actively. These games are not much developed in graphics quality due to the fact that these works on a dedicated network and millions are players are playing together on the same network and using the game servers. The data from each player should be less so that it can be handled and should not overwhelm the servers. The margin of error in these games architecture is very less because time is an important factor and players should be able to connect anytime. Modern MMOs has included various other features like messaging, lobbies, communities, etc. to make games more interactive and interesting. Massive Multiplayer Online Game is the most famous category for smartphones and handheld devices. One of the most famous game for smartphones is Clash of Clans. On the Android play store alone, there are more than 100 million downloads of this game. This game is based on attacking and looting resources from other players. This game has a lot of detailing like gold resources, elixir resources, a number of troops, troops health, etc. which needs to be maintained every second. A single second delay can lead to a bad gameplay and losses to the developer. This game connects the player to other players from all over the world. A bad connection or no connection is not an option for MMOs.

The database has a central role to play in Persistent Games. The use of the database will vary from game to game but it is used for every purpose and every second. There may be more than one database to maintain. There can also be a local database which is used to provide the necessary speed to the game. It can also be used to store the temporary information from the main database which is on the server side. In MMOs, the traditional database will not work because of its requirements. Players are connected to the server using a network and a single server cannot provide service to millions of players at the same time. The architecture needs to be able to provide services without any hiccups and able to tackle any kind of situation.

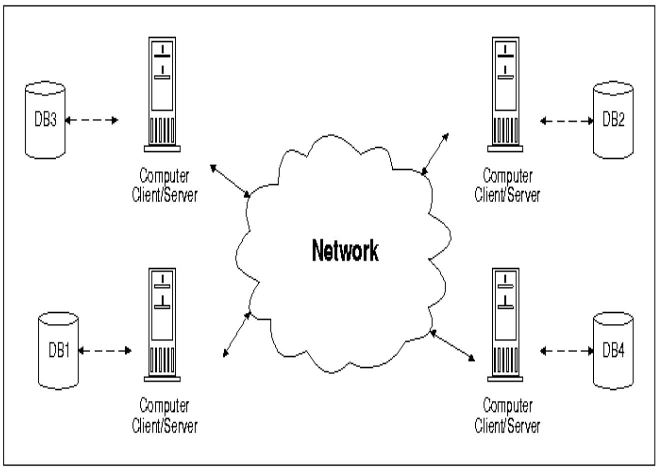

Fig. Distributive Database

The best possible database system for MMOs would be a distributive database system. In this system, a number of databases will work together to provide the services. Different databases can be used to provide different information and database can also be a local database. A local database is preferred to store the basic information regarding the player and game characters to make the game faster. This kind of database system also has an upper hand as compared to the traditional databases when it comes to backup. The fault in a single server will not affect the other servers and may not be able to affect the whole game. Distributive databases also have better performance as compared to a centralized database due to the fact that multiple servers are in use to provide the information and data which will definitely lead to better speed. Scaling is not a problem at the peak hours because we can add extra servers whenever needed and this will save the extra costs of running unused servers all the time. These servers will only be used when the number of players is more than the existing servers are able to handle.

All the prior information leads us to the merits of distributed database but each technological innovation has certain disadvantages as well. The most important in this case is the security of the data as it is to be managed and secured at various places instead of just one. Not to forget that the communication network that interconnects these distributed servers have to be secured individually. The cost of these systems is also huge because of the need of additional hardware as compared to the traditional database. In the case of a technical glitch, a specialist is needed on board, thus poses extra expenditure. Concurrency and consistency of the data have to be managed and for that locking and time stamping methods are used. The complexity of the systems is also of major concern as the transfer of data from one node to the other is expensive in terms of time consumed.

VIII. Conclusion

In this paper, we have discussed how the database systems are used in video games. The type of system needed is found out only be studying the architecture of the game and the requirements of the game. The type of the game can also affect the needs of the game. A centralized database is best for use in the non-networked games. For the persistent and non-persistent games, a distributive database is a better choice. The architecture of the database system will still vary according to the requirements of the game. The system used will also vary according to the resources the game developer has. A simple and less complicated system are always preferred over the more complicated system due to the problems which can occur later. The game architecture should be able to provide basic requirements of the scalability, fault tolerance, and satisfying performance. All this will make sure that the game’s success is not restricted by the hardware and technical problems.

References

1. S. Posniewski. Massively modernized online: MMO technologies for next-gen and beyond. In Proc. Austin GDC, Austin, TX, September 2007
2. P. Bettner and M. Terrano. 1500 archers on a 28.8: Network programming in Age of Empires and beyond. In Proc. GDC, 2001.
3. J. Lee, R. Cedeno, and D. Mellencamp. The latest learning - database solutions. In Proc. Austin GDC, Austin, TX, September 2007.
4. D. McGrath, M. Ryan, and D. Hill. Simulation interoperability with a commercial game engine. In European Sim. Interop. Workshop, 2005.
5. M. Zyda. Introduction: Creating a science of games. Communications of the ACM Special Issue: Creating a science of games, 50(7):26–29, July 2007.
6. P. A. Boncz and M. L. Kersten. Monet: An Impressionist Sketch of an Advanced Database System. In Proc. Basque Int. Workshop on Information Technology, San Sebastian, Spain, July 1995
7. Various. Artificial intelligence in computer games. Roundtable Discussion at GDC, San Francisco, CA, March 2007
8. Entertainment Software Association. 2006 sales, demographic and usage data: Essential facts about the computer and video game industry. http://www.theesa.com/.
9. Harteveld, Casper. Triadic Game Design: Balancing Reality, Meaning and Play. Springer Science & Business Media. P. 71. ISBN 1849961573, February 2011.
10. A. Mulholland and T. Hakala. Programming Multiplayer Games. Wordware Publishing, 2004.
11. B. Dalton. Online gaming architecture: Dealing with the real-time data crunch in MMOs. In Proc. Austin GDC, Austin, TX, September 2007.