Building a simple yet powerful MMO game architecture, Part 1: Introduction

A simple, elegant implementation that delivers the functionality needed by any MMO game

Hyun Sung Chu

September 08, 2008

There is growing excitement among business, education, and government institutions in massive multiplayer online (MMO) virtual-world games and how they can be applied to business and educational needs. MMO games offer tantalizing new ways to learn, entertain, collaborate, socialize, visualize information, and do business. In this series, learn about an architecture based upon the first 3D MMO game from IBM, PowerUp. This first article will begin to show you how to build a flexible and powerful MMO game architecture that is quick and easy to implement.

View more content in this series

Introduction

Massive multiplayer online (MMO) games are Internet-based video games that can accommodate hundreds—or even thousands—of concurrent users. A defining characteristic of most multiplayer online games is that they present a single, integrated, persistent gaming world. *World of Warcraft* and *Second Life* are examples of MMO games.

The *Halo* and *Counter-Strike* series of games are examples of basic multiplayer online games (MOGs). Even an online game of Yahoo! chess can be considered an MOG. However, these examples are not MMO games because they lack an integrated, persistent gaming world. MOGs typically consist of a list of separate, nonintegrated, multiplayer matches, and the matches continuously recycle through a series of rounds. For example, in Yahoo! chess there is no persistence: the game ends and starts anew each time the match ends. And of course, there is no semblance of a single, integrated, chess game "world."

There are several MMO game genres. Massive multiplayer online role-playing games (MMORPGs) are the most popular.

Part 2 will explore technical details of the architecture, including the functions and calls for integrating game clients and servers with back-end systems.

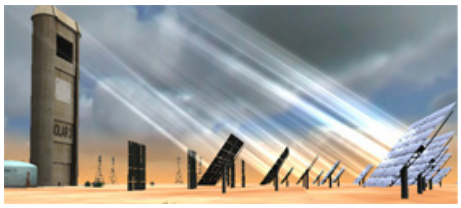
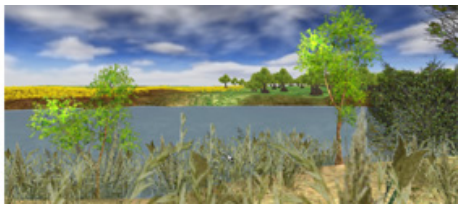
Part 3 will assess whether the architecture addresses the overarching goals (as shown here in Table 1), reviews the capabilities and limitations of the architecture, and includes lessons learned.

Others examples include massive multiplayer first-person shooter games, MMO racing games, and MMO real-time strategy.

In this series, learn about an MMO game architecture that accommodates almost any MMO game genre. The architecture underlies several IBM projects, including PowerUp, The first 3D MMO game from IBM available to the public.

This first article outlines the architecture, specifications, and intended functions of the architecture. You'll learn about the high-level and detailed architecture design.

Figure 1. IBM MMO games (PowerUp at left)

Overall goals

An MMO game architecture should address the specifications and goals in Table 1.

Table 1. Specifications and goals

For...	The architecture should...
Scalability	Allow for any potential number of concurrent users. It should also scale with relative ease.
Flexibility in deployment	Allow for a wide range of gaming server configurations and updates. Ideally, the server configurations should be relatively easy to reconfigure. Updates should be transparent to the end user.
Flexibility in gaming design	Minimize constraints on game designers, and facilitate the ability to design an expansive, integrated gaming world.
Performance	Perform smoothly and quickly. Provide performance-related functions, such as game-server load balancing and monitoring.
Functions	Provide a means to perform functions for an MMO game. A feature might be the ability to persist and retrieve game data, such as player and game-related data.
Security	Be secure and not expose any security risks.

High-level architecture

The MMO game architecture described in this article consists of four major components:

- A gaming client to render the game for the user.
- Gaming servers to interact with the gaming client.
- A Web application server to integrate with the gaming servers and clients.
- A database server to persist and retrieve data.

Any Web application server (such as IBM WebSphere®, BEA WebLogic, or Apache Tomcat) will suffice for the architecture described in this article. Any database server will also suffice (IBM DB2®, Oracle, MySQL).

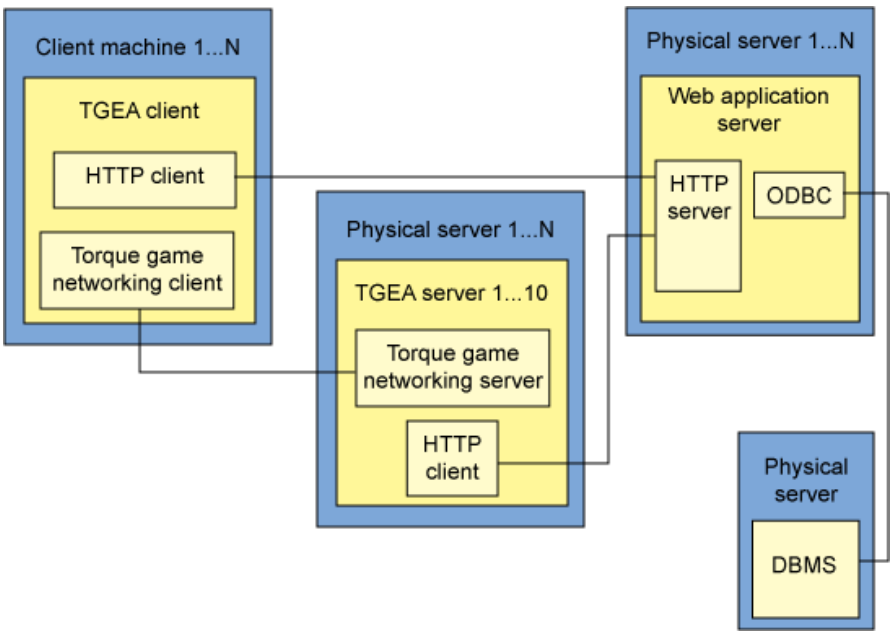
This article focuses on the integration between the gaming server and client code with the Web and DB2 back end, and not on the gaming client and server or the back end themselves. It doesn't address how to create a building in Torque Game Engine Advanced (TGEA) or what networking protocol TGEA uses for the gaming client-and-server networking. There is ample documentation on TGEA at GarageGames.

The architecture in this article is somewhat predicated on the Torque Game Engine Advanced (TGEA) from GarageGames. However, other multiplayer gaming engines provide similar networking functions to TGEA, such as:

- The integration between the TGEA client and server to the Web and database back end described in this article is performed with HTTP requests. Other gaming engines also provide this function.
- All multiplayer-capable gaming engines provide gaming client and server networking, although each gaming engine implements the actual gaming client-and-server networking in its own manner. Therefore, there will be some idiosyncrasies with the TGEA gaming client-and-server networking code that cannot be applied to other gaming engines.

Figure 2 shows the high-level MMO game architecture. Several IBM MMO games use this architecture.

Figure 2. High-level MMO game architecture



The architecture consists of four major components: the game client, game server, Web application server, and database.

TGEA client (game client)

The client machine 1...N represents the user's machine running the TGEA client. 1...N indicates that there are more than one concurrent machines (users).
With the HTTP client, TGEA can send GET requests to a Web server and process the returning page sent by the Web server.

The Torque Game Networking Client is the default game-networking client code that allows the game client to talk to the TGEA game server.

TGEA server 1...N (game server)

1...N indicates that multiple TGEA servers can be run on a single physical server. Multiple TGEA servers can also be run on a particular processor within a server. By default, TGEA allows for 10 instances of a TGEA server per processor.
Physical server 1...N is the actual physical server that runs the TGEA server. 1...N indicates the architecture accounts for multiple physical servers running TGEA servers.

Torque Game Networking Server is the default game-networking server code that allows the TGEA servers to talk to the TGEA clients.

With the HTTP client, TGEA can send GET requests to a server, so this indicates that the TGEA server can also make these HTTP requests.

Web application server

Physical server 1...N are the physical servers that run the Web application server.
HTTP server is the Web server that processes the HTTP requests from the TGEA client and server.

Database connection is the application programming interface (API) that connect the Web application server to the database, such as Open Database Connectivity (ODBC) or Java™ Database Connectivity (JDBC).

Database management system

The database server to persist game data.

The design is a standard three-tiered server architecture, where the client, the Web business-logic server, and database components are modularized. The benefits of a modularized three-tiered model are well established. A modularized design provides standardized interfaces and makes it easy to update any particular module. The gaming servers can also be considered clients of the Web server.

The TGEA client and TGEA server are a standard MOG that TGEA (and most other multiplayer gaming engines) provides out of the box. The Web application server and database management server provide the functions to implement an MMO game, or a *persistent, integrated gaming world*.

The Web server serves up Web services that can retrieve and persist data. This loose, straightforward coupling of the gaming clients and servers with a Web and database back end along with HTTP allows for an elegant, simple, yet powerful MMO game architecture.

Conceptually, it might help to think of the architecture as composed of two overarching components: the gaming server and client providing MOG functions, and the additional Web and database back end that adds MMO game functions.

MMO game architecture implementation

Starship IBM?

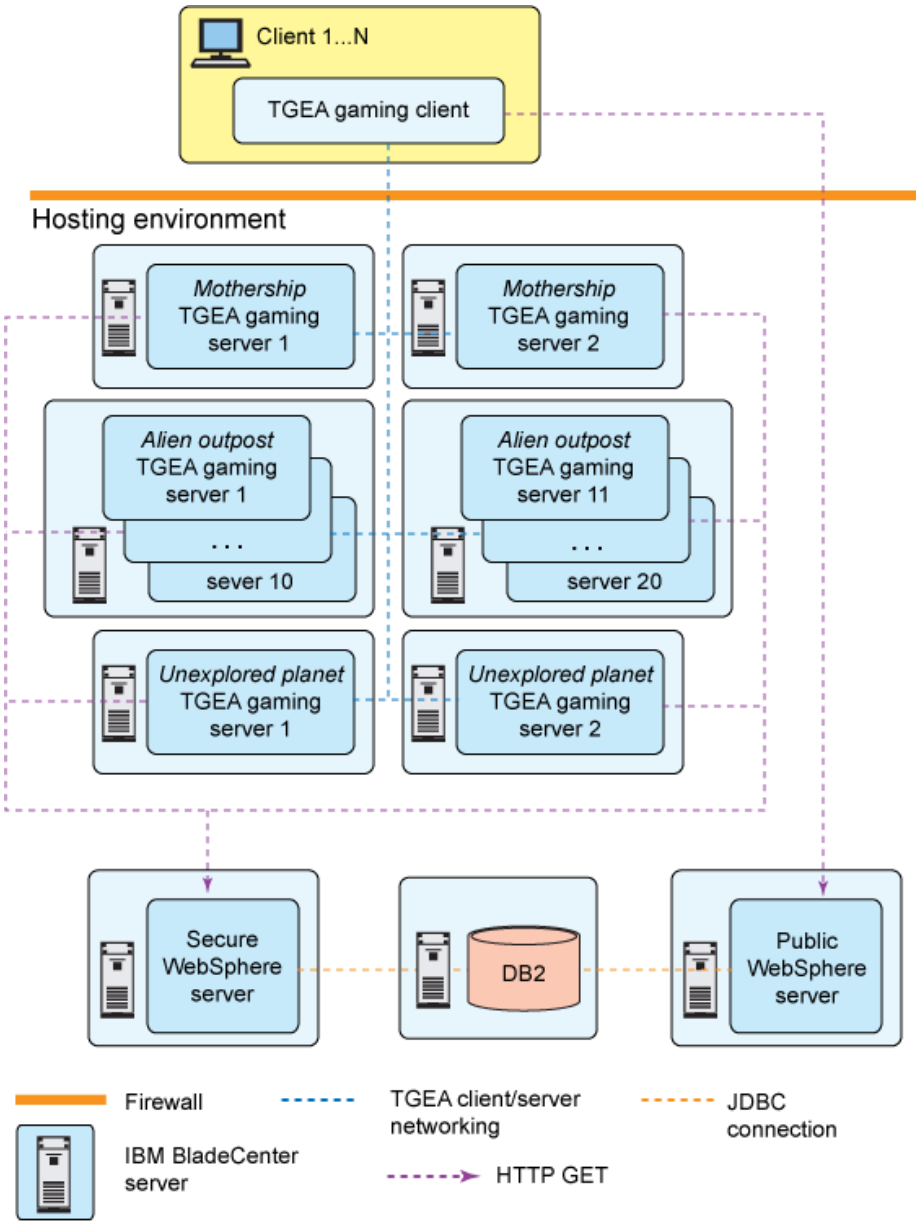
By explaining the architecture using a familiar game theme of a sci-fi space setting, we can avoid cumbersome terminology such as "initial spawn area game server" and "cloned side mission game server A." Instead, such terms as *mothership* server and *alien outpost* server are used in the Starship IBM example game.

Imagine a fictitious MMO type game called Starship IBM. It is in a science fiction space setting consisting of:

- The mothership "Starship IBM," which is the home ship where players initially enter when they log in to the game. Players can explore the ship and interact with other crew members (players). From there, players can visit other planets to perform missions and return to the mothership after completing a mission.
- An alien outpost overrun by hostile aliens. Players can form small "away teams" with other players and travel to this planet to fight the alien monsters.
- An unexplored planet, which players can explore in a vehicle.

Using the imaginary game as an example, Figure 3 shows an MMO game architecture based on an actual IBM MMO game.

Figure 3. MMO game architecture implementation



The objects in Figure 3 are:

Client 1...N
The user client machines.

TGEA gaming client
The game client installed on the user's machine.

Mothership TGEA gaming server
The mothership game server is where a player initially logs in when they join the game. From here, players can then play the alien outpost and unexplored planet missions.
There are two mothership game servers on two separate physical servers. There are only a certain number of concurrent clients a server can handle due to the finite amount of server

processing power and bandwidth. The additional cloned mothership server was added to handle more clients.

Alien outpost

TGEA gaming server. Serves up the alien outpost missions. Players can form small teams and "travel" to this server on a mission from the mothership.

There are 10 alien outpost game servers on each physical server. TGEA allows for multiple instances of TGEA servers per processor, with each server listening on a unique port for TGEA client connections. Starship IBM is designed so that each alien outpost gaming server runs a mission with a maximum of eight players per mission.

Unexplored planet

TGEA gaming server. From the mothership, players can travel to this server (planet), and explore the world in a vehicle.

Secure WebSphere server

Handles all the HTTP GET requests from the TGEA gaming servers.

Public WebSphere server

Handles all the HTTP GET requests from the TGEA clients. Because the requests are publicly accessible, only secure, appropriate functions should be exposed.

For example, any requests that would make an update to the database should not be exposed. (This function belongs in the secure WebSphere server).

DB2

Stores the game-related data.

Firewall

The hosting environment firewall, which allows access to the gaming servers using the TGEA networking protocol. Allows HTTP requests to access the public WebSphere server.

IBM BladeCenter® system

The physical server running the various gaming, WebSphere, and DB2 servers.

The TGEA server requires Windows.

TGEA client-and-server networking

The game networking connection. Each TGEA client connects to a particular TGEA server. TGEA allows the client to disconnect and then connect to a different TGEA server.

HTTP GET

The TGEA clients make HTTP GET requests to the public WebSphere server, while only the TGEA servers make requests to the secure WebSphere server.

JDBC connection

The connection between the WebSphere servers and DB2.

The architecture in Figure 3 is a sharded architecture, which means that scalability is provided by distributing servers onto multiple physical servers. In this case, the gaming servers are distributed onto six physical servers. So theoretically, they handle six times the number of concurrent users as a single physical server.

Looking at the architecture in Figure 3, you might wonder what number of concurrent players the gaming system can handle. There is no set answer. By default, a TGEA server is configured to handle a maximum of 64 players, but this configuration can be easily updated to any number. At some point, though, when a certain number of concurrent users are on a game server, performance will degrade to the point of making the game unplayable. There is no set limit to this number. Performance is dependent on a number of factors, such as:

- The processing power and bandwidth available to a server.
- The design of the world. A game consisting of simple avatars walking around a featureless plain, devoid of any objects, in a thick fog, can perform better than a highly interactive first-person shooter set in a richly detailed world full of numerous objects.
- The game type. A virtual world or turn-based MMORPG does not require the networking fidelity and performance of a first-person shooter game.

It has been our experience in IBM that approximately 200 concurrent players can be accommodated on a single TGEA server for an MMORPG or virtual world game. Again, this largely depends on the factors mentioned above. Using this approximate number, the MMO architecture in Figure 3 can handle a maximum of:

Two mothership servers = 400 players.
20 alien outpost servers, limited to 8 players each = 160 players.
2 unexplored planet servers = 400 players.

Therefore, the maximum total would equal 960 players.

Summary

In this article you learned about the MMO game architecture design based on the IBM MMO game PowerUp. You walked through the architecture of an example sci-fi space game.

Stay tuned for Part 2, which will explore the technical details of the architecture: how the game server and clients and back-end WebSphere and DB2 servers are integrated, and what MMO-type functions are provided by the back end. Explore how traveling from the mothership to the alien outpost is accomplished, how game-related data (such as points) are stored, and what functions are provided by the back end to manage the gaming servers.

Part 3 will assess whether the architecture met the initial goals and will examine future capabilities and limitations.

Acknowledgments

Thanks to Igor Gershfang and Richard J. Young from IBM Global Business Services for their advice on how many concurrent users a Torque Gaming Engine server can handle.

Related topics

- Check out the other parts of this series:
  - Part 2 explores technical details of the architecture, including the functions, and calls for integrating game clients and servers with back-end systems.
  - Part 3demonstrates how the MMO game architecture meets the high-level architectural goals of scalability, flexibility in deployment, flexibility in gaming design, performance, functions, and security.
- Read about MMO games on Wikipedia.
- GarageGames specializes in gaming engines targeted for small and independent game developers.
- Torque Game Engine Advanced (TGEA) is GarageGame's flagship 3D game engine.
- Torque Game Engine (TGE), another game engine from GarageGames, has less advanced graphical capabilities than TGEA. However, its networking capabilities are very similar to TGEA. The architecture described in this article does not use any particular feature that is TGEA-specific and not part of TGE. TGE also runs on Linux® and Macintosh.
- The GarageGames forum covers Torque client/server networking.
- Read about how Torque interacts with a Web server.
- For comparison purposes, Unity is a 3D gaming engine that has features comparable to Torque.
- Check out Unity's game networking, particularly the section on "WWW functions," to learn how Unity talks to Web servers.
- Learn more about the various IBM products described in this article, including those from WebSphere , DB2 , and BladeCenter.
- Download, explore, and play PowerUp.