# A Service Platform for On-Line Games

Debanjan Saha Sambit Sahu Anees Shaikh

Network Services and Software
IBM TJ Watson Research Center
Hawthorne, NY 10598

{dsaha,sambits}@us.ibm.com, aashaikh@watson.ibm.com

## ABSTRACT

Providing a satisfying experience for players of on-line first-person and multiplayer role-playing games is greatly influenced by the distribution and scalability of the game server infrastructure. Deploying a large, dedicated server infrastructure to support a single game title, however, is an expensive approach that does not make the best use of available resources. In this work-in-progress report, we propose a shared, on-demand service platform for hosting on-line games based on grid technology. A standards-based grid infrastructure provides economies of scale and high availability, necessary prerequisites for a successful on-line gaming service. We give an overview of the service architecture, and a detailed description of a number of middleware services that comprise the game hosting platform. In addition, we describe existing standards in grid technology and show how standard grid toolkit services can be leveraged to realize a gaming grid.

## 1. INTRODUCTION

On-line gaming is a rapidly growing segment of the video games industry and has the potential to be a killer Internet application. There are three primary classes of on-line games: first-person shooters (FPS), massively multiplayer on-line role-playing games (MMORPG), and peer-to-peer games. On-line FPS games provide a virtual world in real-time. A central server maintains the global state of the world and periodically distributes updates to the clients, and clients frequently communicate their own local state to the server. The frequency of state update is about 50ms. On-line FPS games stress both the server and the network infrastructure due to their strict interactivity requirements. Consequently, FPS game servers can support a fairly small number of players, typically no more than a few tens [3]. MMORPG games also make use of central servers, which are each responsible for a portion of the overall game world (*i.e., realms*). These games do not have as strict interactivity requirements as FPS games, and a single server can support hundreds of players at a time. However, in order to maintain persistence across large numbers of players, the server has to process a large number of database transactions that ultimately limit the number of players that a given server can support. Peer-to-peer games, as the name suggests, do not need any servers except for a directory service for players to locate opponents.

It is readily apparent that FPS and MMORPG games require a significant amount of computing and networking resources. During the nascent phases of the gaming industry, many of the FPS and MMORPG games were hosted on individual servers with virtually no global management. This resulted in a fragmented gaming community with no guarantees on server or network performance and often led to poor game-playing experience. Over the last several years, there has been a shift in the on-line gaming landscape in which a number of vendors and service providers have recognized the huge commercial potential of on-line games. They have also recognized that in order to make a successful business out of on-line games, they need to create an infrastructure that consistently provides a superior game playing experience. Consequently, we have seen a move from individually hosted game servers to full production gaming services hosted by game publishers (*e.g.*, EA.com), game console manufacturers (*e.g.*, Xbox Live), and third-party infrastructure providers (*e.g.*, Butterfly.net).

In this paper, we propose to use on-demand grid technology as the underlying infrastructure for hosting on-line games. While our architecture is geared initially toward FPS games, several of the gaming services are applicable to MMORPG games as well. FPS games require a tremendous amount of server resources, and, due to interactivity requirements, they also require that the server resources be placed closed to the players. Since concentration of player populations can vary dramatically (*e.g.*, with time of day), intelligent on-demand allocation of computing resources is critical to optimize server utilization while maximizing game playing satisfaction. We believe that grid technology provides a standards-based platform well-suited for such an on-demand computing infrastructure. In addition, a grid-based shared infrastructure provides economies of scale and industrial scale resiliency, necessary prerequisites for a successful on-line gaming industry.

Our specific objective is to develop middleware based on existing grid components that will facilitate an on-line games hosting platform. The middleware targets the following services:

- Player services, including service provisioning, player


*NetGames 2003* May 22–23, 2003, Redwood City, California, USA.

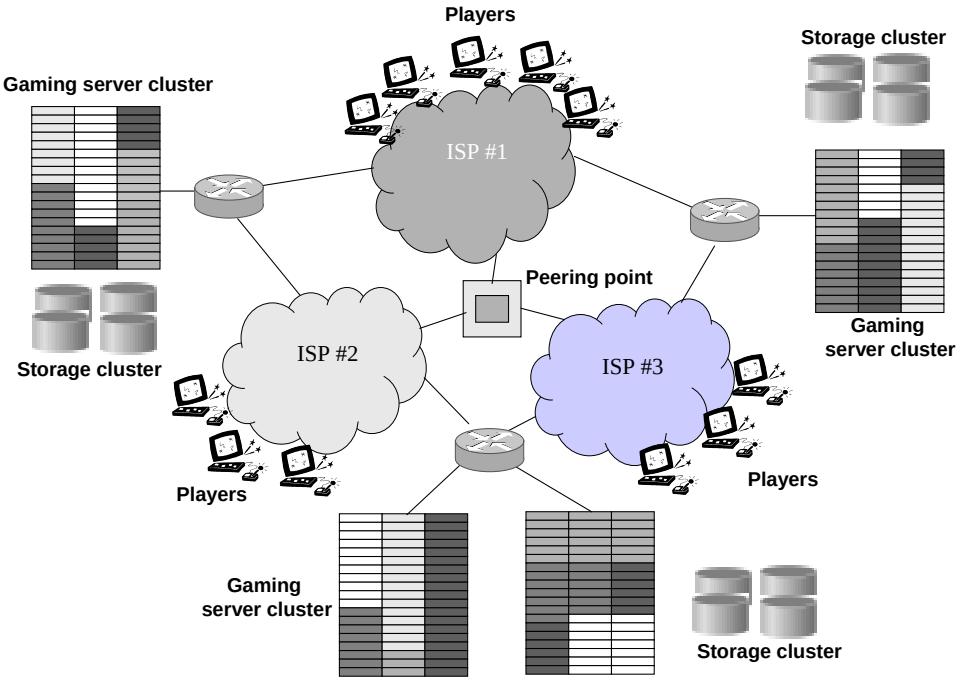


**Figure 1: Service architecture for hosting on-line games**

authentication, account management, game news, game statistics, player ranking, expert advice, chat rooms, etc.

- Game publisher services, including different modes of billing, self-service game publishing, game software update distribution, performance monitoring, and service level management.
- System services, including player communication services, on-demand server resource management, and directory services to help players find each other.

Note that we are primarily interested in services that minimize alterations to the way game developers architect and write their games (in contrast to recent middleware proposals, *e.g.* [1, 2, 5, 8]). Our objective is to develop "control and management plane" services to facilitate on-line game hosting on an Internet scale.

Though our intent is that this middleware layer is as transparent to game developers as possible, there are admittedly some functions that would likely require developer awareness of the service platform. For example, proposed APIs to support data management for MMORPG realms spread across multiple servers may require knowledge about game servers as they are dynamically provisioned. Our approach is to focus on the services described above and include, if possible, support for data management middleware as it becomes standardized and gains wide adoption among game developers.

In Section 2 we describe the service architecture in more detail, along with details of the specific gaming-related services provided by the platform. Section 3 provides some background on grid technologies and standards. We also describe specific middleware components available in a widely-used grid implementation, and how they can be leveraged to support the game hosting architecture. Section 4 summarizes the paper and discusses next steps in realizing the architecture.

## 2. ARCHITECTURE AND SERVICES

In this section we describe the gaming service platform architecture and the middleware services that it provides. We propose three sets of services to allow players, game publishers, and service providers the ability to deploy games in a grid-based architecture. These are described in more detail below.

### 2.1 Architecture

Figure 1 shows an example of a gaming grid. As shown in the figure, the gaming grid consists of a number of server clusters located in geographically distributed data centers. The data centers are connected to the Internet via one or more ISPs. This federation of server clusters is a resource that is shared across a number of game publishers and is used to host different games. In the figure, different shades in the server clusters represent server resources allocated to different publishers or game titles at a certain point in time. Note that a given publisher may have server resources allocated at different clusters in order to handle players located in different geographic locations. Furthermore, the allocation of server resources may change over time as the number and distribution of players change. In fact, one of the primary reasons for using a shared grid infrastructure is to take advantage of statistical multiplexing gains by exploiting dynamic changes in player population.

As mentioned before, one of our objectives is to develop software services that hide the complexity of managing this

shared grid infrastructure from the publishers and players of on-line games. In order to achieve this goal, we do make use of standard grid services such as resource monitoring, scheduling, and task dispatching. However, we also develop more advanced features specifically for the on-line gaming environment on top of these basic grid services. For example, the decision to spawn a new instance of a game server in a specific server cluster may depend on a number of game-specific inputs such as maximum number of players a server can handle, location of the players, and player preferences regarding different gaming communities. These decisions are handled by the gaming grid middleware. However, the task of spawning a server instance makes use of the standard mechanisms provided by grid middleware. Hence, architecturally our middleware sits on top of generic grid services middleware and handles tasks that are more specific to game hosting.

## 2.2 Player services

The player services component of the middleware provides game players with the functions to locate games, connect to a game server, and play the desired game hosted on a grid infrastructure. It hides the details of the underlying architecture, including the fact that the game is hosted on a grid infrastructure. The basic functionality of this service includes managing account information, authenticating the player, providing game related announcements, and most importantly, finding an appropriate game server. The middleware service allows a player to specify his or her game-related preferences, *e.g.*, in which game community or team the player wishes to participate. To the player, these functions can be presented as a simple Web-based portal. Game developers could then replace integrated game finders with a simple interface to a Web browser which contacts the portal.

Server selection is also a key component of the gaming platform. Current server load on the set of game servers currently hosting the game, the inferred geographic location about the player, and player preferences are taken into account to choose a suitable server for the player. It may be the case that there is no such server that meets the requirement of the player. In that case, more instances of the game are created by using the functions provided by the game publisher and system services of the middleware. The game server selection process in our architecture differs from traditional network-aware server selection in that our middleware (i) accounts for player preferences (e.g., teams), and (ii) on-demand resources may be allocated while making the server selection. By clustering players into regions, network and server monitoring can also be made more scalable, since it is not necessary to track the entire network of servers for a given set of players.

## 2.3 Game publisher services

These services collectively provide the required functions and interface for a game publisher to participate in the grid infrastructure. The various functionalities include deploying a game, automatically updating game software, monitoring game server performance, managing service level agreements, and handling various modes of billing and settlement. The automatic software update service should scale to a large number of servers. One approach would be to use overlay multicast on a per-title basis to automatically handle the often very large updates, and includes required authentication mechanisms to detect any unauthorized or malicious update requests.

The game publisher middleware also allows a game publisher to specify an SLA for its players. For example, the game publisher may request that not more than 10% of its players suffer more than 150 ms delay and not more than 1% players are turned away due to lack of resources. In addition to performance related SLAs, the publisher can also specify network-related SLAs such as game servers should be allocated so players connect to the servers on the same ISP whenever possible. These specifications are then used by the player services when assigning a player to a game server.

## 2.4 System services

This service is central to overall resource management in the grid infrastructure for game hosting. It provides on-demand server resource management, directory services for player and game discovery, and resource-specific performance and availability monitoring. Note that these system services are tailored to the gaming service platform, and hence, are built on top of base grid resource management services. These system services help in facilitating compliance with SLAs specified by the game publishers and, indirectly, through players preferences.

The objective of this resource management is to maximize the revenue of the grid infrastructure provider while meeting the objectives of the game publishers and players. Using the monitoring infrastructure, it detects when the current allocation of resources is not sufficient to meet the desired objectives of a game publisher. In that case, it allocates additional resources to meet the desired SLAs on a per game publisher basis. Similarly, if the allocated resources exceed what is required to meet the SLAs, it can automatically release the resources back to the grid infrastructure. This on-demand grid resource allocation introduces several interesting problems. An important issue is the choice of a suitable timescale for this resource allocation/deallocation process to ensure system-wide stability. Also the platform must scale to handle a large number of players and changes in players across the games. One approach is to do *a priori* allocation of resources based on stated, or historical, estimates of the load, and then incrementally adapt to handle changes in the load.

The system services also include directory services for players and game publishers. A player may be interested in finding other players, or currently running games. In order to provide an up-to-date view, the platform must handle a highly dynamic player population in a scalable manner. We are exploring the use of distributed hash table-based lookup schemes [9], or existing hierarchical grid directory services (as described in Section 3).

# 3. LEVERAGING THE GRID

Two primary advantages of developing a grid-based services architecture are i) the existence and ongoing development of open standards that allow many services to interoperate, and ii) the rich set of base services, implemented as a general middleware layer, available for use in developing new services. In this section, we provide an overview of grid functionality and the current direction of grid standardization efforts. We also describe how the existing middleware available can be leveraged to implement some of the func-

tionality described in Section 2.

## 3.1 Current and evolving grid standards

The gaming service architecture is essentially a computational grid service, in which basic infrastructure such as the hardware and software game server platform is virtualized to deliver a utility-like abstraction to publishers and players. Computational grids are currently built around several different models. For example, enterprises can use computational grids to harness unused CPU cycles to better utilize their existing hardware resources. In the case of the gaming service grid, however, it is more likely that the service provider (or grid provider) follows a dedicated resource model, in which servers, storage, and bandwidth, are deployed specifically for the computational grid [4].

To support these models, a number of vendors have released software that provides base features needed to construct the grid. For a CPU-harnessing grid, these might include a server which schedules programs, collects data, and provides a management interface, and a desktop client that executes programs and reports results and status information to the server. One such package is the Globus Toolkit, an open source software project which provides a number of component services in the areas of resource management, information services, and data management (described further below) [10]. Version 2 of the Globus Toolkit, with its C language-based APIs, is currently considered the *de facto* standard grid implementation [4].

Similar to the Internet community's Internet Engineering Task Force (IETF), the Global Grid Forum (GGF) is a standards organization charged with documenting best practices for Grid technology, including technical specifications and implementation guidelines [6]. One of the key GGF working groups is the Open Grid Services Architecture (OGSA), which is developing a conceptual architecture for future grid services built around interoperable interfaces. In the OGSA architecture, Grid-based services are implemented as Web Services with some grid-specific interfaces. Web Services define a standards-based (*e.g.*, XML, HTTP, SOAP) communications model to access software components and discover existing services in the network. Within OGSA, Open Grid Services Infrastructure (OGSI) defines in detail the interfaces that a Grid Service must support to be part of OGSA. The next version of the Globus Toolkit is intended to support OGSI so that, for example, the existing resource management services described below are OGSI-compliant. Additional details on OGSA, OGSI, and Globus development is available at [6, 10].

## 3.2 Applying grid middleware

In this section we provide an overview of the way in which existing grid middleware can be leveraged to realize key functions of the gaming service architecture. We focus the discussion around the components of the widely used Globus Toolkit 2.2 [10].

**Resource management**: The Grid Resource Allocation Manager (GRAM) provides services to support remote execution of client jobs. Clients specify jobs using a standard Resource Specification Language (RSL) which describes the executable program, arguments, and resource requirements (*e.g.*, minimum memory requirements). The RSL specification is shipped over a secure channel to a Gatekeeper daemon at the remote location (similar to an inetd). The Gatekeeper starts a local Job Manager process which parses the RSL and directs the request to a local resource manager, which in turn may invoke a local job scheduler. The Job Manager also supports callbacks to clients, status/cancel requests from clients, and sends output results to clients. In addition, Globus supports jobs that require multiple, separately controlled resources, for example at different grid locations. This is done using the Dynamically-Updated Request Online Coallocator (DUROC) mechanism which parses RSL containing DUROC syntax and sends requests to different Job Managers.

This support for task execution can be used directly for automatically provisioning platform resources on the grid for running different game servers. In addition, using DUROC it is possible to start several game servers simultaneously with a single command. The local Job Manager may be used with platform-specific priority schedulers that can give preference to certain games when server resources are constrained.

**Information services**: Globus provides a set of information services collectively called the Monitoring and Discovery Service (MDS). MDS consists of three primary components, the Grid Resource Information Service (GRIS), Grid Index Information Service (GIIS), and local Information Providers (IPs). Access to MDS information is based on the Lightweight Directory Access Protocol (LDAP) [7], and can be done using a Web browser interface implemented with a set of publicly available PHP scripts. The components are arranged in a flexible hierarchy similar to DNS, with GIIS providing an aggregated view of data from a *collection* of resources. This collection may be defined over all of the grid sites, or a single site. Each GRIS instance serves as a repository of local resource information, and registers itself with a GIIS index. GIIS instances can themselves register with higher-level GIIS directories.

IPs collect data from any local data source and translate information into the appropriate schema for GRIS. For information about a resource to to be visible to MDS, it must have an IP created for it. The Globus toolkit includes a set of core IPs for several platforms which provide such basic information as CPU load, platform description (*e.g.*, processor type, operating system, file system), available memory and disk space, and network connectivity.

MDS can be used in a game hosting environment for many core functions. The directory services are essential for locating currently instances of operational game servers for a specific game, discovering the server to which a player or group is assigned, and keeping track of where additional resources are available for provisioning new game server instances. In addition, MDS is well-suited to provide dynamic platform-level information, such as server load and availability or network bandwidth and delay statistics, in order to support adaptive provisioning of additional resources when needed. The hosting environment can also leverage MDS to index application- or game-level information such as player performance or other statistics, if an IP is developed to interface with the game server.

**Data management**: The Global Access to Secondary Storage (GASS) component is used by GRAM to transfer job output back to clients. In general, GASS provides a standard API (*e.g.,* a `gass-copy` function) to securely read and write data between remote locations. In addition, GridFTP, currently in development, provides a uniform protocol and

client/server implementations for all data transfers on the grid. Based on FTP, it features additional facilities such as multistreamed transfer and Globus-based security [4]. These data transfer facilities are naturally useful in the context of the gaming service grid in order to transfer server executables and data on-demand to various hardware clusters.

**Security**: The Grid Security Infrastructure (GSI) provided by Globus enables secure and authenticated communication over public networks. GSI is largely based on public key cryptography, where users and services have standard X.509 certificates which are used for mutual authentication. GSI leverages the SSL (or TLS) protocol for its mutual authentication. GSI also supports confidential communication through encryption, and support for single sign-on (through delegation to a proxy) and coarse-grained access control are currently in development. Clearly, the GSI plays an important role in the gaming service provider environment where access to games, server platforms, and game information must be available only to appropriate parties (*e.g.*, paying players and publishers and the service provider).

## 4. CONCLUSION

In this paper, we present work-in-progress on developing a grid-based hosting infrastructure for on-line games. We believe that a grid-based on-demand computing environment is particularly suitable for server intensive FPS games and, with some additional support for data management, MMORPG games as well. It provides a scalable and robust infrastructure for hosting games and allows service providers to exploit the economies of scale through intelligent sharing and adaptive provisioning of server resources.

In order to make the best use of the grid infrastructure for gaming, we are developing a set of software services based on the open source Globus Toolkit grid implementation. Our gaming middleware provides player management, publisher management, and system management services designed to facilitate hosting of on-line games in a grid environment. Our ongoing work lies in defining these services further and implementing them using components of the Globus Toolkit. We also plan to demonstrate the architecture on a grid testbed with instances of popular on-line game server software.

## 5. REFERENCES


[1] D. Bauer, S. Rooney, and P. Scotton. Network infrastructure for massively distributed games. In *Proceedings of Workshop on Network and System Support for Games (NETGAMES)*, Braunschweig, Germany, April 2002.

[2] A. R. Bharambe, S. Rao, and S. Seshan. Mercury: A scalable publish-subscribe system for internet games. In *Proceedings of Workshop on Network and System Support for Games (NETGAMES)*, Braunschweig, Germany, April 2002.

[3] W. Feng, F. Chang, W. Feng, and J. Walpole. Provisioning on-line games: A traffic analysis of a busy counter-strike server. In *Proceedings of Internet Measurement Workshop (IMW)*, November 2002.

[4] L. Ferreira et al. *Introduction to Grid Computing with Globus*. IBM Corporation, 2002. `http://www.ibm.com/redbooks`.

[5] S. Fiedler, M. Wallner, and M. Weber. A communication architecture for massive multiplayer games. In *Proceedings of Workshop on Network and System Support for Games (NETGAMES)*, Braunschweig, Germany, April 2002.

[6] Global Grid Forum. `http://www.ggf.org`.

[7] J. Hodges and R. L. Morgan. Lightweight directory access protocol (v3): Technical specification. Internet Request for Comments (RFC 3377), September 2002.

[8] D. Levine, M. Wirt, and B. Whitebook. *Practical Grid Computing For Massively Multiplayer Games*. Charles River Media, 2003.

[9] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, San Diego, CA, August 2001.

[10] The Globus Project. `http://www.globus.org`.