BUILDING A MOBILE-FIRST SOCIAL GAME PLATFORM

Driven by a belief that player-to-player dynamics in social casino games could be dramatically improved, Asylum Labs set out to build its own mobile-first platform. However, this didn't come without technical challenges, as CTO **Albert Mack** explains.

Despite their name, most social casino games are just not very social. Our company was founded on that simple yet powerful observation, driven by a belief that the player-to-player dynamics in digital casino games could be dramatically improved. To these ends, we set out to develop our proprietary mobile-first multiplayer platform, called ArcLight, built upon NodeJS, Mysql, Memcache, and Redis.

We decided to use Redis as the core of the platform in order to be fast and scalable. Redis is a NoSQL database that has some unique feature-sets not found in other database technologies. NoSQL databases have gained in popularity in recent years as a replacement to SQL databases, where the complexity of relational data and queries are unnecessary. NoSQL databases tend to be faster and easier to scale than SQL databases but are usually limited to key-value stores, lack transactional capabilities, and have eventually-consistent reads (similar to SQL replication databases being behind master).

Since Redis natively supports sets (sorted/unsorted/hash), it allows for unique solutions to problems that would otherwise be very difficult or inefficient to do in other systems. In ArcLight, the friend graph for each user is stored as Redis sets. In addition, the online status for all users is stored as Redis sets. This allows us to very efficiently determine who is online by doing a simple set intersection. Determination of online users sounds like a simple thing but doing it efficiently without the use of Redis sets is difficult, especially at scale.

The ability to determine which of a user's friends are online drives our Join-With system. With a single click, any player can jump into the room their friend is playing in, a feature almost never seen in mobile games, enabling a highly intuitive user experience.

Redis is not a magic bullet, however. It complements our use of Mysql to provide capabilities that Mysql lacks. Redis does have a relatively important drawback in that it is an in-memory solution that requires the dataset to all fit in memory. It's also not particular fault tolerant in that it doesn't, in its default configuration, distribute data to multiple servers. There is a clustered version that removes some of these limitations but that version has limited support for Lua scripting and set operations, features that make Redis attractive in the first place. In our use cases, the advantages of its unique feature set outweigh its drawbacks.

"Determination of online users sounds like a simple thing but doing it efficiently without the use of Redis sets is difficult, especially at scale."

But despite the technical challenges, building in this way allowed us to develop ArcLight based on three fundamental concepts which give players all kinds of fun opportunities to interact with each other, as evidenced by our recent Wild Party Bingo. These concepts are as follows:

Client server communication

In WPB, our aim was for all users to play together live in a single integrated bingo universe, connected by whatever device they want to play on. Many core elements of our technology flowed from this design goal. It meant an architecture with a cross platform, interoperable client connected to a highly (dynamically) scalable server. All game logic and data lives server-side so all players are sharing the same datasets.

ArcLight uses client initiated polling as its primary means of communication with the game servers. While this may seem inefficient compared to sockets based communication where servers can push updates to clients, the biggest advantage is that polling via HTTPS requests using standard ports means players will not have to alter any firewall settings in order for ArcLight communication to work properly. This is critical in the casual game space where players may not be as savvy about computers as core gamers. Using standard sockets-based communication requires the use of custom network ports which are often blocked by firewalls. Other solutions such as long polling or web sockets can still get blocked by corporate and ISP level firewalls and packet sniffers, especially in countries outside of the United States. To ensure the broadest reach, ArcLight uses standard HTTPS requests to minimize any possible problems with network communications between clients and servers.

Ranked matchmaking

ArcLight's Matchmaking System is a good example of technical complexity under

the hood that's completely invisible to the players. At any one time, there many be hundreds or thousands of different bingo games going on simultaneously at different stages of play. We call each of these a 'room'. Each game has ~20 people split into four teams of five players. Games are organized by difficulty level which governs speed of play, # of bingo cards allowed and buy-ins & payouts. A player can join a game in one of two ways. In the first case, the player just picks game level and clicks "Play". In this circumstance, the client app asks the server for an appropriate game given the player's game history and social graph. This crucial matchmaking process evaluates all of the available games by calculating the best fit for the player based on the range and degree of skill levels of players already playing. Weight is also given to pairing a player with their friends as well as insuring that wait time is minimized. These factors resolve into a single result; the room that the player automatically joins.

The second way of joining a game is by touching a friend's profile pic and clicking "Join-With". In this situation a player directly joins the same game (and team) that their friend is playing, no matter what level they are playing or at what point is the round. The client app needs a constant stream of up-to-date information about all of one's friends, whether they are online, whether they are actively playing in a room and whether that game is even joinable. All this information is in constant flux and potentially out-of-date at any given moment. Therefore, the client must be able to handle joining a game in many different states. Ultimately, what the player experiences is seamlessly connecting with their friend and playing together.

Figure 1: Wild Party Bingo user interface

Proprietary friends system

Developing our own Friend System was crucial to realizing our goals. While we do leverage other social media platforms (like Facebook), we do not want to rely on them. Building our own allows us to have more efficient access to real-time information about the status of our players, i.e. how exactly they are playing WPB at any given moment. We are also able to create a more nuanced friend invite system that tells us who invited whom, who become friends and whether their invitations were mutual. Finally, having our own friends database lets us 'grow' as we add new partners internationally.

Conclusion

Fundamentally our technical efforts are focused on two key aspects of social: improving the way players are matched together and facilitating a myriad of ways for them to interact. Our approach puts our customers front-and-center and mirrors more recent broader trends in mobile app growth, such as messaging. When players can find their friends (and easily make new ones) inside our ecosystem, we find it drives both retention and monetization dramatically.

Albert Mack is CTO, Asylum Labs Inc. Albert has been programming and designing games since junior high when he got his first C64 computer. Albert is a 20-year game industry veteran who has worked on numerous console, PC, and social game titles including TIE Fighter CD, X-Wing vs. TIE Fighter, X-Wing Alliance, Secret Weapons Over Normandy, Star Trek: Bridge Commander, Alien Syndrome, Monster Apps (Zombies, Vampires, Werewolves, Slayers), and many others. He has served in various development roles including Technical Director and Lead Designer, and specializes in UI, multiplayer, and social games. Before AL, Albert was co-founder of Forty Humans LLC, a company that created social and mobile titles on Facebook, iOS, Android, and Kindle platforms. For more information: www.asylumlabsinc.com.