Assignment of Games to Servers in the OnLive Cloud Game System

David Finkel, Mark Claypool, Sam Jaffe, Thinh Nguyen, Brendan Stephen
Computer Science Department, Worcester Polytechnic Institute, Worcester, MA 01609, USA
(dfinkel | claypool} @wpi.edu}

***Abstract*— OnLive is a thin-client, cloud-based, video game streaming service. As part of their service, OnLive must distribute game content to their servers in a manner that provides good performance for the players while using hardware resources effectively. In order to evaluate and improve the effectiveness of their game distribution strategies, we built a simulator for OnLive's service and modeled their current game distribution strategy. Using a hill-climbing algorithm, we constructed game distribution strategies with improved abilities to meet customer demand while significantly reducing disk space requirements.**

Keywords—thin client, cloud-based gaming, server distribution

I. Introduction

OnLive [4] is a cloud-based game streaming service. In order to provide low latency, high bitrate connections to game players, OnLive maintains several server farms at different geographic locations. Due to limited hardware resources, each server has a subset of the games in OnLive's catalog installed, and a server only streams to one client at a time in order to provide for a good playing experience.

Clients contact the nearest server farm, and connect to a server running the Game Selection Portal (GSP), from which the client selects a game to play. The GSP identifies a server in the same farm which is available (i.e., is not currently serving another player) and has the requested game installed, the client is connected to that server and then the player is able to play the game. If no server with the requested game is available, the client is offered the opportunity to select another game. We call the inability of the GSP to find an available server with the player's desired game a *miss*. If a client suffers too many misses and leaves the system, we call that occurrence a *failure*.

The current distribution strategy at OnLive manually assigns a weight to each game based on the expected popularity. Games are then distributed to servers in an attempt to respect those weights using a worst-fit bin-packing algorithm. In addition to the worst-fit algorithm, the distribution to servers must respect hardware requirements for the different games (e.g., some games have specific operating system requirements and so cannot run on every game server).

978-1-4799-6882-4/14/$31.00 (c)2014 IEEE

The goal of our study was to find a strategy for distributing games to the servers to minimize the number of misses and failures and improve the use of disk resources.

In order to explore distribution methodologies, we built a simulator that modeled the distribution of games to servers and players requesting games. As the basis for our model, we examined actual OnLive server logs that provide details on player title requests with accompanying request times and durations. Using the data from the logs, we constructed a discrete event simulator to model the OnLive game distribution, selecting and launch system. Improvements to the distribution algorithm use a hill-climbing algorithm to perform the optimization, providing better performance than OnLive's current distribution method.

II. Analysis and Modeling of OnLive Player Data

In order to evaluate game distribution policies, the first step was to analyze and model OnLive player data with appropriate probability distributions. Specifically: player game selection (Section II.A), player interarrival times (Section II.B), and game session lengths (Section II.C). Our analysis is based on OnLive logs representing approximately 1.6 million player sessions. We used EasyFit software [1] to help us identify probability distributions to represent the data extracted from the logs.

A. Player Game Selection

As of January 2014, the OnLive catalog contained several hundred games players could select. We determined the probability that each game is selected based on the occurrence in the log files.

Figure 1 shows the popularity of the fifty most popular games, ordered in decreasing popularity from left to right. In the figure, each bar represents a single game, and the height of the bar is the probability that a client selects that game. For reasons of confidentiality, the names of the games are omitted.

Using EasyFit [1], the best fit probability distribution to the popularity data is a Pareto 2 distribution, with parameters $\alpha = 0.621$ and $\beta = 34.8$. The Pareto distribution indicates that the data follows Zipf's law [3], where the popularity of a game is inversely proportional to its rank.

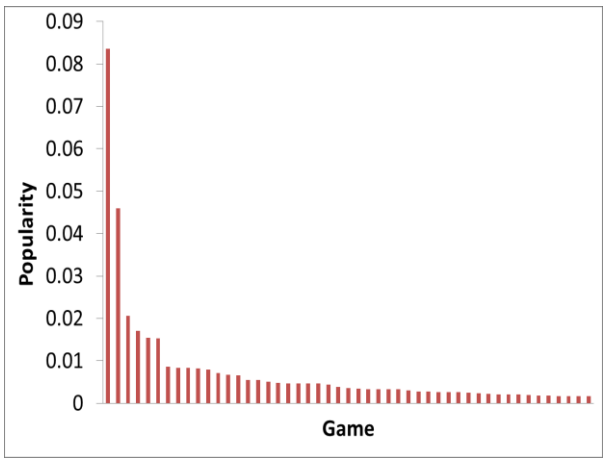

Figure 1: Popularity of Games

B. *Player Interarrival Times*

Figure 2 shows the interarrival times for players obtained from analysis of the OnLive logs for a single game. The x-axis is time shown in hour:minute:second format. The y-axis is the probability. The solid line shows the game data and the dashed line shows an exponential fit of the data determined by EasyFit with parameter $\alpha = 0.00664$.

C. *Game Session Lengths*

We determined the game session lengths based a randomly selected sample of 1500 user session lengths extracted from the OnLive logs. Figure 3 shows the session lengths. The x-axis is time shown in hours:minutes:seconds format, and the y-axis is the probability. The solid line shows the data from the OnLive logs, and the dotted line shows a Weibull distribution with parameters $k = 0.9637$ and $\lambda = 2504.8$ determined by the EasyFit [1] to best fit the data.

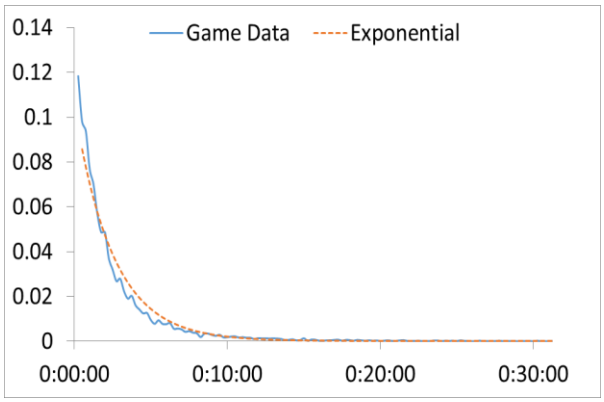

Figure 2: Distribution of player interarrival times to OnLive for a single game

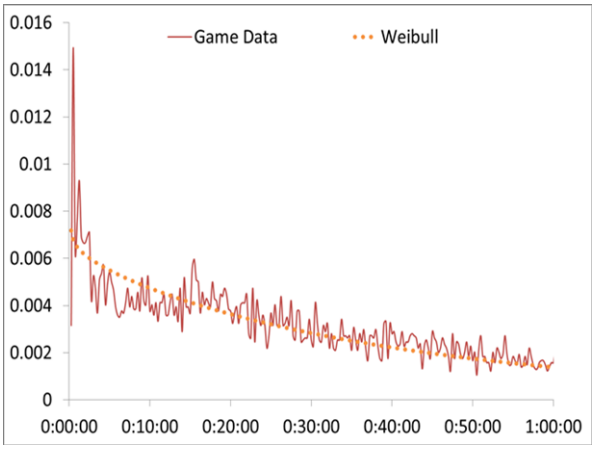

Figure 3: Distribution of Session Lengths

III. Simulation Methodology

With player game selection, interarrival times and session length modeled, our next step was to construct a discrete event simulator that used the models as inputs..

A. *Modeling*

To develop our discrete event simulation, we used MASON, a fast discrete event multi-agent simulation library core in Java [2].

For the popularity of games, we used the observed probability for each game. For player interarrival times and session lengths, we used the fitted exponential and Weibull distributions, respectively.

We reverse-engineered the OnLive code in order to accurately implement OnLive's distribution method in our simulation. Their method is a worst-fit bin-packing algorithm, taking into account the different hardware configurations of the servers and the hardware requirements of the games.

B. *Validation*

We applied various validation tests to make sure the simulation accurately modeled the OnLive system.

We ran some small-scale simulation models where we could compute the results by hand and obtained the same result from the simulation. We ran the simulation with weights used by OnLive for games and obtained the same assignment of games to servers as the current (as of January 2014) OnLive system.

We also ran the simulation with the current weights and player parameters and obtained essentially the same number of misses and failures as the current system.

In addition, we worked closely with OnLive engineers who verified that our simulation model accurately reflected the way their system operated.

C. *Optimization*

Once the simulation model was developed and validated, the model allows exploration of different game distribution configurations as well as player parameters so as to reduce the number of misses and failures. Under current conditions, there are few misses and no failures. To allow OnLive to prepare for future growth and test the usefulness of our model, we studied the operation of the system under increased load.

To improve the distribution of games, the simulator uses a hill-climbing algorithm to modify the weights for game placement. The simulator distributes games to servers based on the revised weights and evaluates performance. This process is repeated until an end condition is met (e.g., a low number of misses and/or failures).

IV. Results

A comparison of the results for different assignment of weights are shown in Figure 4. The x-axis is the demand multiplier and the y-axis is the number of misses. The trend lines depict three different configurations: OnLive, the original hand-made assignment of weights originally used by OnLive; Popularity, the weight for each game is the popularity of the game as indicated in Figure 1; and Hill-Climbing, the weights produced by our simulation and optimization algorithm. The curves for OnLive and Hill-Climbing essentially overlap. Assigning weights simply by using priority is much less effective at 3x and higher demands. Both Hill Climbing and OnLive produce essentially no misses up to 7 times current demand. At higher levels of demand, the number of misses quickly becomes unacceptable.

We ran the simulation with different numbers of servers and computed the disk space used. Figure 5 depicts the results. The y-axis is the minimum, maximum, and average disk space used across all servers with the height of the bar providing the number of GBytes used. For each region in the graph, the bars on the left are the Popularity weights, the bars in the middle are the Hill-Climbing weights and the bars on the right are the original OnLive weights. The Hill-Climbing weights produce substantial savings of disk space over the original OnLive weights. The savings from the Hill-Climbing method is an indication that the OnLive weights substantially over-provisions the games on more servers than necessary. This saving of disk space could be used to install additional games, or to respond to unexpected demand for games in the current catalog.

V. Conclusion

We developed a trace-driven simulation of the OnLive game streaming system that models user demand and game distribution. Comparison of several optimization techniques shows the best performance is provided by a hill-climbing algorithm, best satisfying player demand and reducing resource use.

References

[1] EasyFit, http://www.mathwave.com/, Accessed July 24, 2014

[2] MASON, http://cs.gmu.edu/~eclab/projects/mason/, Accessed July 24, 2014.

[3] M.E.J. Newman, "Power laws, Pareto distributions, and Zipf's Law, Contemporary Physics vol. 46 (5), pp. 323–351, http://arxiv.org/PS_cache/condmat/pdf/0412/0412004v3.pdf, Last modified: May 29, 2006.

[4] OnLive, www.onlive.com. Accessed July 24, 2014.

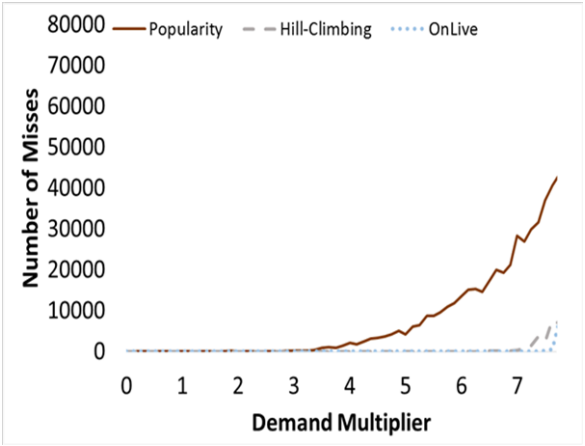

Figure 4: Number of misses for different configurations

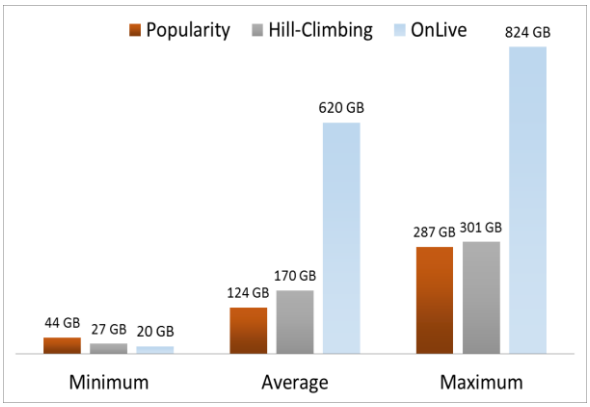

Figure 5: Disk space used for different server configurations