Game Servers Deployment Automation Case Study

Zane Ouimet, Heath Caswell* and Youry Khmelevsky*
Computer Science, Okanagan College, Kelowna, BC Canada
Emails: zane.ouimet@outlook.com, heath.caswell@gmail.com and ykhmelevsky@okanagan.bc.ca

Rob Bartlett and Alex Needham
WTFast, Kelowna, BC Canada
Emails: {rob, alex}@wtfast.com

*Also Affiliated with Mathematics, Statistics, Physics, and Computer Science Department
Irving K. Barber School of Arts and Sciences, UBC, BC Canada

Abstract **T** [illegible]

T [illegible]

I. Introduction

Minecraft [28], [1] is a popular video game played worldwide, and is built simply enough to be used for network analysis and research. This paper describes a software system prototype for automated game servers deployment and configuring customized game servers on demand. The described system has a web interface which allows customers to create accounts, purchase, services, and gain access to and configure their purchased web servers. This service consists of a website which acts as an interface for customers to purchase subscriptions, and gain access to and configure their purchased servers. Behind the website is a system which dynamically deploys virtual machines with the requested configurations, handles all of the networking details, and provides information back to the customer on how to connect to their server.

The project prototype development was the next step in the design, construction and test of a new layer of game server software that can optimize and monitor in real-time game services [3], [4]. It stems from the observation that game servers place demands on computing resources - hardware and network - that can vary with user behaviour and whose optimization is the key to customer satisfaction. Virtualized servers provide new flexibility in hardware reservation and allocation but their use can make resource optimization difficult by making it context sensitive i.e. dependent on the allocation of virtual machines (VMs) to hardware.

The main project's objective was to study predictive monitoring and optimization for game server clusters. The first phase of the project was to gather performance data about game servers, then analyze its time behaviour to allow the creation of a performance-prediction software module [5]. The initial module version applied virtualized game servers in various configurations, and later versions were tested with physical servers as well as parallel (cluster) game servers. Later, the project investigated performance optimization based on short-term predictions.

Our main contributions are: (1) a unique network and gaming servers infrastructure created for the emulation experiments, which is also being used to perform stress testing and data analysis of network game applications, as well as to monitor the performance of game servers within a proprietary Gamers Private Network (GPN) [13]; (2) an automated software system prototype for creating and configuring customized game servers on demand, and (3) using this automated software system prototype we are able to improve the game servers' utilization. Two networking and servers optimization research projects GPN-Perf1: "Investigating performance of gamers Cbntovga-597(gamers)]TJ048(aCbnto)o spikes in packet traffic [13]. Our goal was to generate realistic network traffic related to video game environment on the Internet by ready made game emulation applications, available online.

”A *video game network* is a distributed set of apparatus which are capable of exhibiting an interactive single identity game” [26]. Response time and network latencies are very important video game parameters, which can be a reason for the gamers’ frustrations and dissatisfaction, especially in the multi-user environment. On the other hand, ”the online service’s computers themselves introduce latencies, typically increasing as the number of active users increases” [24].

In [12] and in [14] multiuser online video game architectures been discussed in order to reduce the bandwidth and the servers processing time. Their approach may improve scaling, but it ”opens the game to additional cheating, since players are responsible for distributing events and storing state”. A

Fig. 1. Domain Model of the Developed Prototype

Fig. 2. System Architecture

Fig. 3. Web Subsystem Architecture

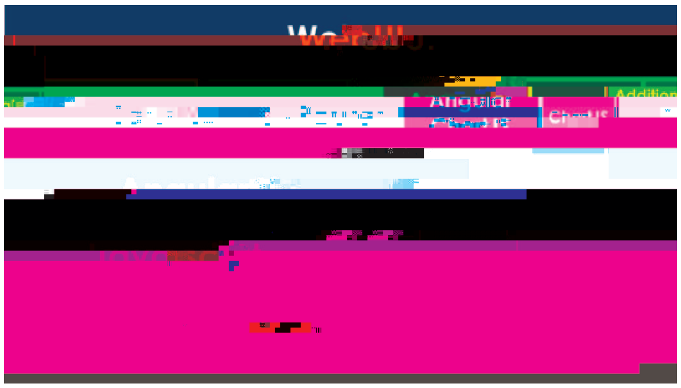

Fig. 4. WebUI

Web API
Laravel
PHP
Beanstalk

Fig. 5. WebAPI

as well as the API polling the Cloudstack API to ascertain the status of asynchronous tasks.

IV. Development

The entirety of the project consisted of two main areas the ”web” and ”infrastructure” parts. The prototype development part of the project was broken into two parts: a RESTful API (written in PHP) and a JavaScript UI. The diverse range of technologies being used can create issue while integrating systems and may require additional effort to either maintain or replace parts with a singular language or technology such as replacing PHP with NodeJS. At this time the PHP API is interacting directly with Cloudstack to create and control VMs while utilizing Puppet to install and configure software. The prototype makes use of PHP’s ability to access and write to the file structure allowing for the automated creation of the Puppet Node files, or more simply put, machine configuration

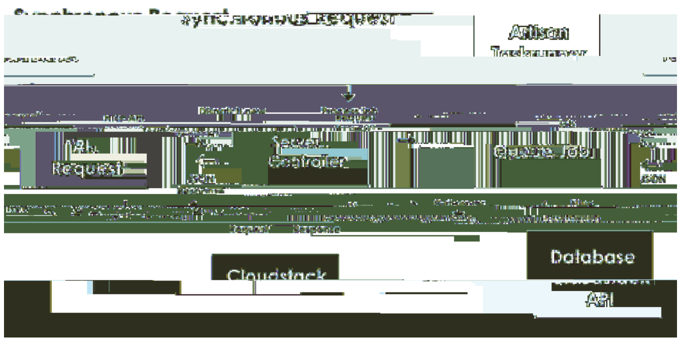

Fig. 6. Synchronous Request.

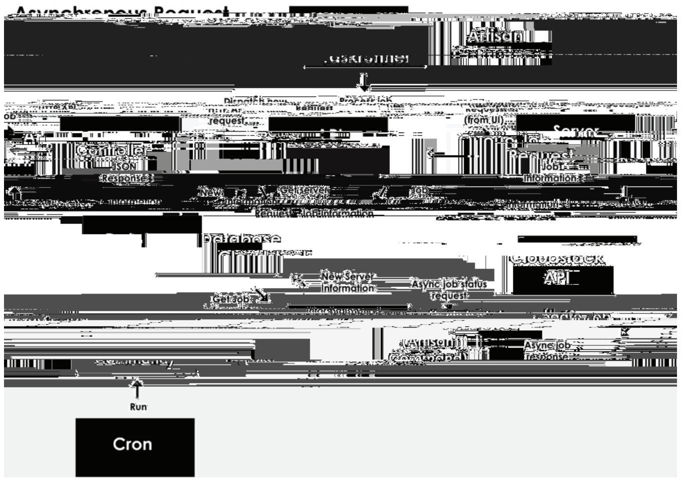

Fig. 7. Asynchronous Request.

files. Both the UI and API are incomplete; however given a functioning Cloudstack [8] install the system is able to: create, destroy, start, stop, reboot, and configure a game server on the VM through puppet files. The current issue that is blocking

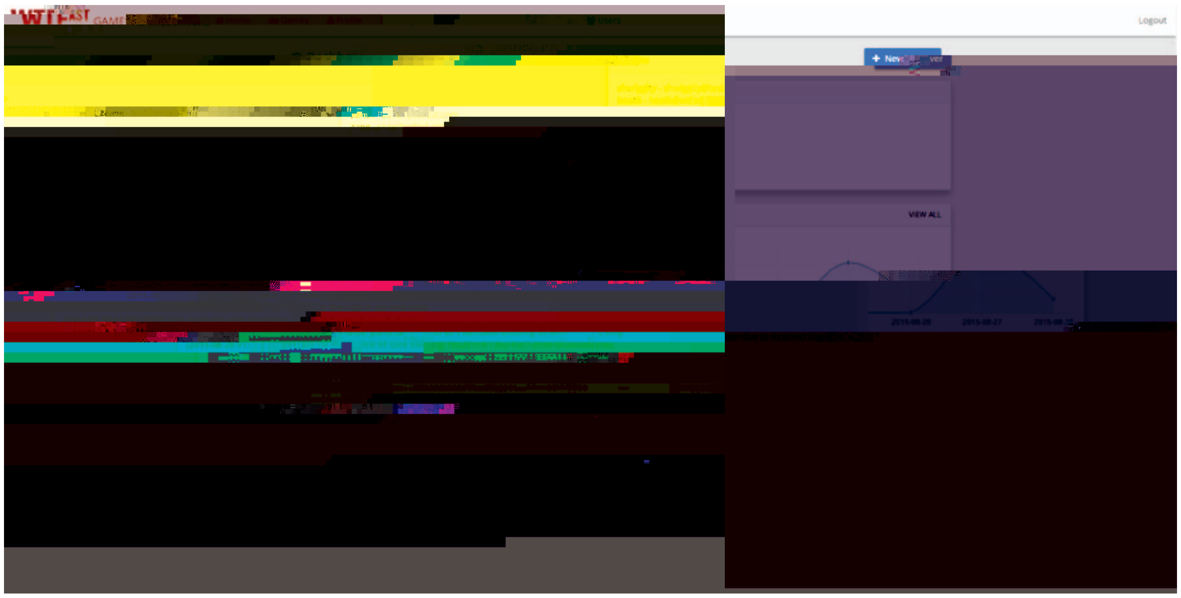

Fig. 9. Admin Dashboard Showing Error Graph.

- Viewing error logs [admin]
- Changing user role [admin]
- Updating profile
- Deleting server [admin]

Our initial attempt to deploy a gaming server was with Minecraft. Using foreman and puppet we have automated deployment and configuration conceding a purchase from a user via an external purchasing web system. The server page is shown at Fig. 10 and process of virtual machines deployment is shown at Fig. 11.

The interface was built on a two tier system that separated administrators and users. It was meant to be extensible in the event that further user roles where required. A user was granted control over their servers and virtual machines but no other whereas an admin was capable of modifying and accessing any users accounts or their servers. Additional account levels will be added as needed to accommodate new staff or users.

V. Future Work

Our future research will be related to utilizing the infrastructure created in the first phase of the project as the framework on which to test and build game server software that can optimize and monitor WTFast game services in real time. An effort will also be made to create and utilize predictive models for handling anticipated demand for a particular server. This infrastructure will allow new servers to be automatically deployed and configured for use as private game servers, while also monitoring their performance and usage statistics, including the following additional features:

- Authentication with WTFast
- Integration with the GPN
- Load balancing for both the API and UI
- Advanced resource allocation
- Failsafes and issue handling
- Payment and store
- Improved UI design
- Better user profiles

By using predictive models, new servers may be automatically added when traffic levels require more resources to maintain optimal performance. These models will likely be altered on a case by case basis and require operator interaction to deal with high variance events such as a new game launch or an unpredictable flood to a particular game.

Testing of the network infrastructure and the network software will be done using a program that emulates many players connecting to multiple game servers on several virtual machines in the network. The goal of these network tests is to identify the point at which the network software can no longer keep up to the flow of traffic, i.e. the point where the network software becomes a cause of latency. The tests will also serve to identify the capacity of the game servers and their host virtual machines. Initially the tests will be performed under ideal conditions, that is, on a local network with almost no wire latency. Later, network factors such as latency and jitter can be artificially added to the network to emulate real conditions of the game being played over the Internet on a geographically remote server in order to confirm our tests in a more realistic environment.

In this paper we discussed latency for gaming, which is vital and it is discussed qualitatively both singularly and via networks. In our future research we will try to capture the latency issues quantitatively. Then we will try to do relevant comparisons under ideal versus the point where it ’gracefully degrades’ or the point where we can provide a consistent performance across the spectrum of users. On the other hand, will try to go more in depth qualitatively as to those measures

video games led us to experimentation with Minecraft clients and servers due to its simplicity and the nature of Java. The need for a controlled environment to ensure consistent and accurate results led to the requirement of multiple automated bots [3], [4], that could interact with the servers in a predictable way and for the automated game servers deployment system prototype and configuring customized game servers on demand. This project was successful in creating just that, meeting all originally set criteria. We anticipate that the automated deployment system prototype will greatly help out future work in this field. Based on our experiments, our infrastructure gives to us a suitable tool for the network performance investigation and to generate a stress test for the game servers.

ACKNOWLEDGMENTS

The described WTFast Gateway was developed by Computer Science students at Okanagan College within COSC 470 Software Engineering and COSC 471 Software Engineering Project courses in the Fall 2014 and in the Winter 2015 (Bachelor of Information Systems program, Computer Science department). Our thanks to the students Marc Schroth, Mike Adkins, J. Riley Dunkin, Simon Detlor and Brad French for their work on the WTFast Gateway.

We would like to thank NSERC's CCI Engage College program for supporting our "GPN-Perf1: Investigating performance of gamers private networks" in 2014 and GPN-Perf2 research project application in 2015. Our thank to Amazon Web Services, Inc. for supporting our capstone student software engineering and our student research projects by AWS (Amazon Web Services) Grants for Research and Education.

We would like to thank both reviewers for their insightful comments on the paper, as these comments led us to an improvement of the work. We thank the reviewers because these points were useful to clarify the main focus of the paper.

REFERENCES

[1] Mojang Synergies AB./Microsoft. Minecraft home page: https://minecraft.net/., June 2014.

[2] Simonas Gildutis Dimitris Bozelos. Alice Chen, Salim Alami. Satellizer - token-based angularjs authentication: https://satellizer.herokuapp.com.

[3] T. Alstad, J.R. Duncan, S. Detlor, B. French, H. Caswell, Z. Ouimet, Y. Khmelevsky, G. Hains, R. Bartlett, and A. Needham. Minecraft computer game performance analysis and network traffic emulation by a custom bot. In *S ien e and In ormation Con eren e (SAI), 2015*, pages 227–236, July 2015.

[4] T. Alstad, J. Riley Dunkin, S. Detlor, B. French, H. Caswell, Z. Ouimet, Y. Khmelevsky, and G. Hains. Game network traffic simulation by a custom bot. In *Systems Con eren e (SysCon), 2015 9t Ann a IEEE Internationa* , pages 675–680, April 2015.

[5] Trevor Alstad, J. Riley Dunkin, Rob Bartlett, Alex Needham, Gaétan Hains, and Youry Khmelevsky. Minecraft computer game simulation and network performance analysis. In *Se ond Internationa Con eren es on Com ter Gra i s, is a i ation, Com ter ision, and Game e no ogy (isioGame 2014)*, Bandung, Indonesia, November 2014. Accepted for publication.

[6] Mark Claypool and Kajal Claypool. Latency and player actions in online games. *Comm n. ACM*, 49(11):40–45, November 2006.

[7] Mark Claypool, David Finkel, Alexander Grant, and Michael Solano. Thin to win?: Network performance analysis of the onlive thin client game system. In *Pro eedings o t e 11t Ann a orks o on Network and Systems S ort or Games*, NetGames '12, pages 1:1–1:6, Piscataway, NJ, USA, 2012. IEEE Press.

[8] Apache CloudStack^{TM}. Open source cloud computing^{TM}: https://cloudstack.apache.org.

[9] Johannes Färber. Traffic modelling for fast action network games. *M timedia oo s and A i ations*, 23(1):31–46, 2004.

[10] Google. Angularjs - html enhanced for web apps: https://angularjs.org.

[11] Aymen Hafsaoui, Navid Nikaein, and Christian Bonnet. Analysis and experimentation with a realistic traffic generation tool for emerging application scenarios. In *Pro eedings o t e 6t Internationa ICS Con eren e on Sim ation oo s and e ni es*, SimuTools '13, pages 268–273, ICST, Brussels, Belgium, Belgium, 2013. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[12] Takuji Iimura, Hiroaki Hazeyama, and Youki Kadobayashi. Zoned federation of game servers: A peer-to-peer approach to scalable multi-