TABLE I
ONLINE GAME SYSTEM ARCHITECTURES [19]

Peer-to-Peer	Peer-to-Peer (P2P) systems where game players interact directly with each other. They have not been very popular in recent systems.
Client-Server	Client-Server systems where a set of logically central servers store the game state and provide most operations except player interaction and complex view rendering (in a local client). This architecture has been very popular recently.
P2P Client-Server Hybrid	Peer-to-peer*n*Client-Server hybrid (lockstep) Hybrid types of implementations.
Cloud Gaming	Cloud Gaming where the client is a very thin piece of software are almost all functions are realized by the server. Like all cloud platforms, services or software, its advantages are in easy deployment, no installation for the players etc. But its disadvantages lie in heavier computation and communication. In [19] bandwidth requirements multiplied by 10 to 1000 for cloud games by comparison with traditional client-server games.

games networks. Indeed Saldana and Suznjevic [19] in *QoE and Latency Issues in Networked Games* have summarized existing literature that, among other things, highlights the insensitivity of most online games to bandwidth while confirming that latency is critical to most games except perhaps games of strategy.

In the rest of this section we summarize those concepts and results of [19] that define precise measurement and modelling objectives for GPNPerf2.

a) Game genres i.e. types of online games:

First Person shooters (FPS): Games such as *Call of Duty* where the user sees himself as an armed warrior evolving alone or among a team of a few dozen members to eliminate virtual enemies. Average time between firing and death of the enemy is about 161 ms for the most popular FPS games. Studies have confirmed that such games require very low latencies.

Massively Multiplayer Online Role Playing Games (MMORPG) are games where thousands of players, merge with artificial entities into a complex virtual world. They cooperate or fight each other so that, (simulated) firearm exchanges happen and put a constraint on latency as in FPS games but at a lesser frequency by the nature of the game. The element of tactics is also important in such games so virtual-world coherence is critical.

Real Time Strategy (RTS) games where a dozen players share a virtual worlds where they build "civilizations" i.e. geometric and slowly-dynamic structures.

Multiplayer Online Battle Arena (MOBA) games, a special-case of RTS where two teams try to conquer a battlefield.

Sports games that simulate car racing or team sports. In vehicle racing it is possible that latency can be critical while team- or other realistic sports involve the simulation of balls, running humans and other objects that are very slow relative to the "firearms" and "bullets" of FPS games.

In [19] surveys that for FPS games, a one-way delay of 80 ms can be acceptable for most game users. Low latency convinces users to join the game, which confirms its importance for game-related business that is aimed at latency-reduction or latency-stabilization.

For MMORPG games, players started rating the game quality from "excellent" to "good" when one-way latency raised above 120 ms. When it rose further above 150 ms up to 200 ms, players started leaving their game sessions. The same phenomenon has been observed when latency in RTS games rose from 200 ms to 500 ms.

Studies have also shown that experienced players are more sensitive to those factors than ordinary players.

b) Connection Types: Network games are implemented with a variety of network connection types.

c) Geographical location: Geographical location of servers is correlated with latency for obvious reasons of transmission delays. Many games report the geographical location of their servers so players connect to the closest ones. Mapping IP addresses to geographical location is necessary for experimental and mathematical investigation of the above questions.

d) Latency reduction: Once empirical and mathematical tools are built to analyze and predict message latency and its variance, the results will be used to predict human- and business-effects in various games. For example player QoE in specific situations defined by combinations of the above factors (game genres, connection types, etc). In turn, the effect of this perceived or real QoE can be correlated to game session durations, popularity of the game service etc, and in the end of business objectives and economic factors for game players and GPN providers.

Once this analysis is put in place it will be natural to apply latency-reduction techniques such as *zoning* and *mirroring*. Zoning partitions the virtual world into geographical areas called zones, handled independently by separate machines. Mirroring, targets parallelization of game sessions with a large density of players located and interacting within each other's geographical vicinity.

e) Methods to enhance QoE: Scalability:

Logical Sharding: A virtual world spans over multiple servers.

Sharding:

TABLE II
NETWORK GAME SCALING

Peer-to-Peer	Connection Type: TCP or UDP Quality of Experience (QoE) considerations: No server authority (cheating/hacking). No hardware considerations for game developer. Game Example(s): StarCraft 1 (1998) Game Genre(s): RTS
Client-Server	Connection Type: TCP or UDP Description: Clients synchronize to a single source which propagates the synchronization outwards. QoE considerations: Central server authority for anti-cheating, pay-gate, etc. Ability to deploy dedicated servers if included with game (LAN parties w/ low latency). Individual players game-client may also act as server (Game interruption if player leaves). Game Example(s): Counter Strike, Warcr aft 3, Call of duty-350(3,)-350(pa(arcr)750(pa(arcc(pe:)-350(TCPII)FPS)-350(duMORPG)-350(dSar

data update was able to handle 3,959 requests (20 update queries) over 15 seconds. The update request's average latency was 62.9 ms with a standard deviation of 62.3 ms and a maximum of 893.7 ms. These results as well as the plaintext test results were the closest to resembling what we wanted for performance. Further information regarding test frameworks utilized are shown below in Table III.

TABLE III
TECHEMPOWER TEST ENVIRONEMENT

Hardware	**Dell R720xd dual Xeon E5-2660 v2** (40 HT cores) with 32 GB memory; database servers equipped with SSDs in RAID; switched 10-gigabit Ethernet **i7: Sandy Bridge** Core i7-2600K workstations with 8 GB memory (early 2011 vintage); database server equipped with Samsung 840 Pro SSD; switched gigabit Ethernet **EC2: Amazon EC2 c3** large instances (2 vCPU each); switched gigabit Ethernet (m1.large was used through Round 9)
Operating System	Ubuntu Linux 12.04 64-bit Windows Server 2012 64-bit
Databases	MySQL MongoDB PostrgeSQL
LoadSimulator	Wrk
Tests	JSON serialization Single query Multiple queries Fortunes Data updates Plaintext

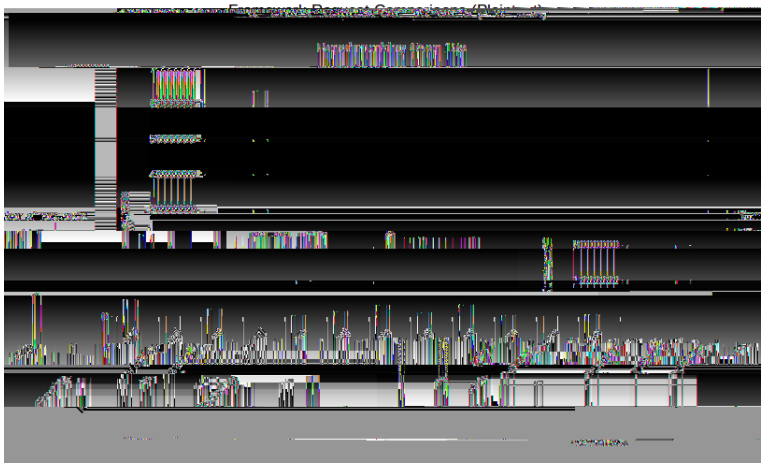

Fig. V.1. Plaintext Requests Comparisons

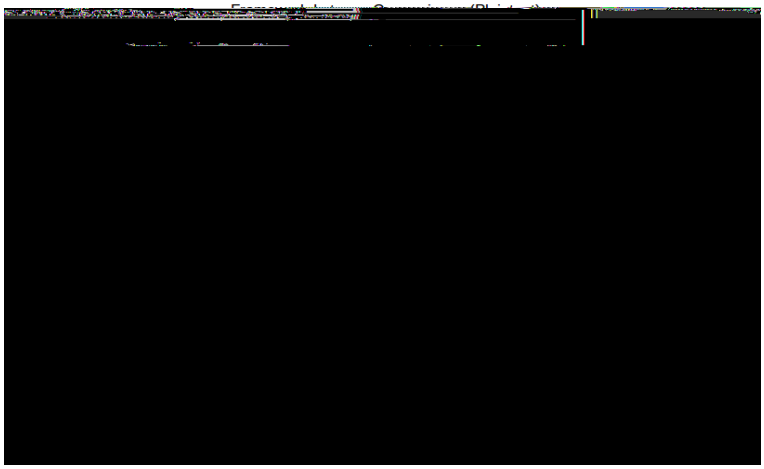

Fig. V.2. Plaintext Latency Comparisons

We implemented our own tests to validate and benchmark our fasthttp application. The results of our tests are shown in Table IV and Table V (see the first line with the GPerf2 results), where we have almost 30 times more requests # in 15 seconds and much less average latency to compare with the other frameworks. The Plaintext Test in Table V is shown just for the comparison only.

Fig. V.1 shows the number of requests the frameworks are able to handle for plaintext HTTP GET requests. The results of our test can be seen in the tables. Our collector web application is shown to be considerably slower than fasthttp's implementation of 6 million requests per 15 seconds by a factor of 10. This can be explained by using the *wrk* benchmarking tool on a lower powered computer over a 100 Mbps switch. Fig. V.2 further shows that the latency is quite low much like fasthttp. The system is actually faster however the tests were done on the same network that the collector which explains how much faster the system is. Fig. V.3 shows the number of database queries that the frameworks can handle. Due to utilizing a buffered method to accept database queries we are able to accept queries at a higher rate as compared to the frameworks shown here. This however is a trade off as we run into buffer bloat (latency created due to network hardware buffering too much data) which can slow down the system if the collector receives requests at a faster rate than it can execute them. This is shown in Fig. V.4 where the overall latency is lower but the standard deviation is 50%.

Future tests of the system will utilize remote benchmarking machines to further emulate the test environment executed by TechEmpower as well as performing longer tests to show the long term functionality and determine what affects bufferbloat will have on the networked machines.

VI. CONCLUSION

In this work we have built the elements and general structure of a web application that accepts requests to insert data to a database from clients. The framework utilized was determined

TABLE IV
U

REFERENCES

[1] T. Alstad, J. R. Dunkin, S. Detlor, B. French, H. Caswell, Z. Ouimet, and Y. Khmelevsky., "Game network traffic emulation by a custom bot." in *2015 IEEE International Systems Conference (SysCon 2015) Proceedings*, ser. 2015 IEEE International Systems Conference. IEEE Systems Council., April 13-16 2015.

[2] W. Doherty and A. Thadhani. (1982) The economic value of rapid response time (IBM Technical Report GE20-0752-0). [Online]. Available: http://www.vm.ibm.com/devpages/jelliott/evrrt.html

[3] D. H. Sitrick, "Video game network. United States Patent number 4,572,509," Feb. 25, 1986.

[4] S. G. Perlman, "Network architecture to support multiple site real-time video games. United States Patent number 5,586,257," Dec. 17, 1996.

[5] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned federation of game servers: A peer-to-peer approach to scalable multi-player online games," in *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, ser. NetGames '04. New York, NY, USA: ACM, 2004, pp. 116–120. [Online]. Available: http://doi.acm.org/10.1145/1016540.1016549

[6] J. Jardine and D. Zappala, "A hybrid architecture for massively multiplayer online games," in *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, ser. NetGames '08. New York, NY, USA: ACM, 2008, pp. 60–65. [Online]. Available: http://doi.acm.org/10.1145/1517494.1517507

[7] J. D. Pellegrino and C. Dovrolis, "Bandwidth requirement and state consistency in three multiplayer game architectures," in *Proceedings of the 2Nd Workshop on Network and System Support for Games*, ser. NetGames '03. New York, NY, USA: ACM, 2003, pp. 52–59. [Online]. Available: http://doi.acm.org/10.1145/963900.963905

[8] P. Ghosh, K. Basu, and S. K. Das, "Improving end-to-end quality-of-service in online multi-player wireless gaming networks," *Computer Communications*, vol. 31, no. 11, pp. 2685–2698, 2008.

[9] Q. Zhou, C. Miller, and V. Bassilious, "First person shooter multiplayer game traffic analysis," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, May 2008, pp. 195–200.

[10] M. Claypool and K. Claypool, "Latency and player actions in online games," *Commun. ACM*, vol. 49, no. 11, pp. 40–45, Nov. 2006. [Online]. Available: http://doi.acm.org/10.1145/1167838.1167860

[11] P. A. Branch, A. L. Cricenti, and G. J. Armitage, "An ARMA (1, 1) prediction model of first person shooter game traffic," in *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*. IEEE, 2008, pp. 736–741.

[12] A. L. Cricenti and P. A. Branch, "A generalised prediction model of first person shooter game traffic," in *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*. IEEE, 2009, pp. 213–216.

[13] Y. Wu, H. Huang, and D. Zhang, "Traffic modeling for massive multiplayer on-line role playing game (MMORPG) in GPRS access network," in *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, vol. 3, June 2006, pp. 1811–1815.

[14] B. Hariri, S. Shirmohammadi, and M. R. Pakravan, "A hierarchical HMM model for online gaming traffic patterns," in *Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE*. IEEE, 2008, pp. 2195–2200.

[15] J. Färber, "Traffic modelling for fast action network games," *Multimedia Tools and Applications*, vol. 23, no. 1, pp. 31–46, 2004.

[16] T. Alstad, J. R. Dunkin, R. Bartlett, A. Needham, G. Hains, and Y. Khmelevsky, "Minecraft computer game simulation and network performance analysis," in