

Firestore – Performance Analysis

2008-08-05

Fredrik Johansson, Cubeia Ltd

Scope

The aim of this document is to evaluate the performance and scalability of Firebase as a game server. This document presents the results of running a long term load test of the poker game on single and multi-node setup of Firebase.

Setup

The tests were performed on Cubeia Ltd's load test environment in Hammarby, Stockholm. All servers participating in the test were configured as follows:

- Dual Core, Intel Xeon 2.4 GHz
- Linux (CentOS), Kernel 2.6.9 (EPOLL enabled)
- Sun Java HotSpot™ Server VM (build 1.5.0_12-b04, mixed mode)
- Poker 1.0-SNAPSHOT
- Firebase (FB) 1.5-SNAPSHOT (nightly build 2008-08-02)

The system was profiled using the following topologies:

- Singleton (One server running all roles)
- Dual Singletons (Two servers running all roles enabling fail over)
- 4 Nodes (2 Master & Client + 2 Game)

The sampling values were:

- CPU idle (percentage of available CPU)
- Events per Second (Events executed by FB per second)
- Hands per Minute (poker hands per minute as per Poker JMX values)

It can be noted that all values are approximate (i.e. average over all similar nodes) except for the values for the Singleton.

Game Setup

The poker game has multiple tables configured. Sizes of the tables vary between 2, 6 and 10 seats with the majority being 10 seats.

Different tables will run with different timings as well. The tables are configured to normal, express and super express. Normal runs like a normal table, express is noticeably faster and super express is much faster than any real life table. About 50% are normal, 25% express and 25% super express.

Using faster than real life tables ensures that we are not running lenient load tests. The bots will respond to the table timing and answer within time unless they decide to timeout.

All bots subscribe to the entire lobby in all load tests. This is also usually not the case in a real life scenario where you would partition the lobby between play money, real money tables etc. in order to minimize network and CPU usage for handling and communicating the lobby.

The bots were also chatting to each other at the tables with a chat frequency that is much higher than you would normally see at a real life table. Once again we are trying to model an aggressive scenario.

Below is a screenshot of the poker client (flash). The chat can be noted in the event box.



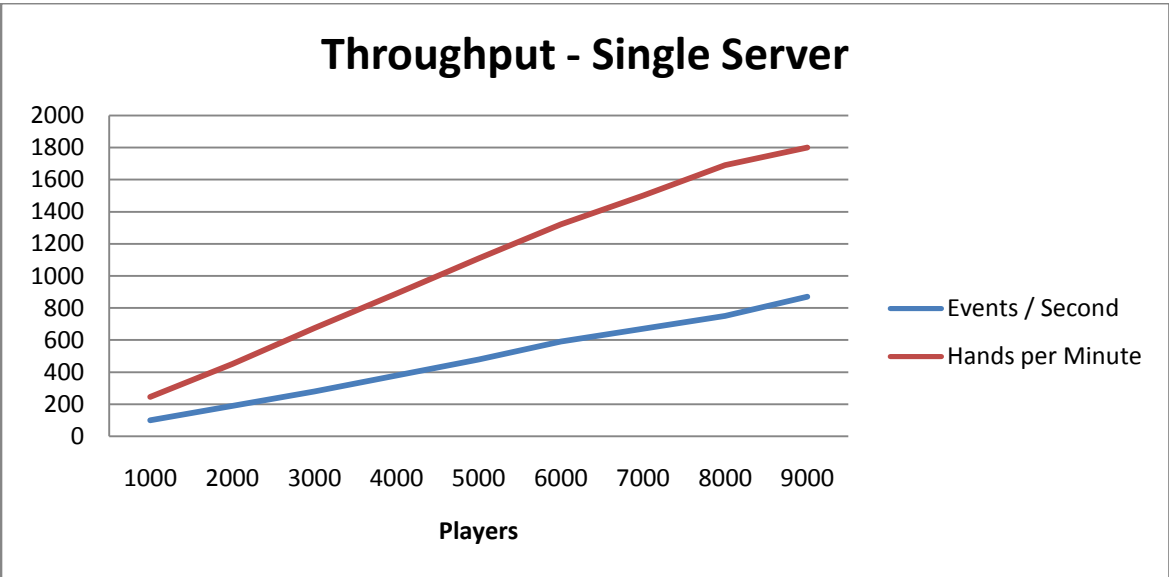
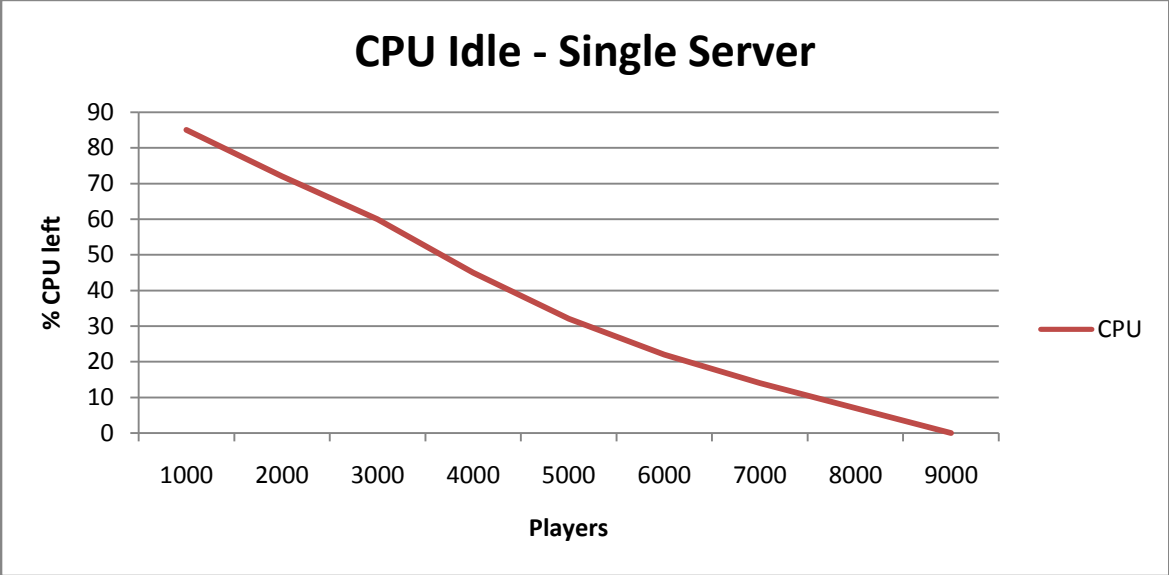
The profiling data is collected through the JMX attributes exposed by Firebase and the poker implementation.

Since we are load testing Firebase as a game server and not external systems we have disconnected all external dependencies, i.e. we do not write hand history etc. to a database.

Performance

Single Server

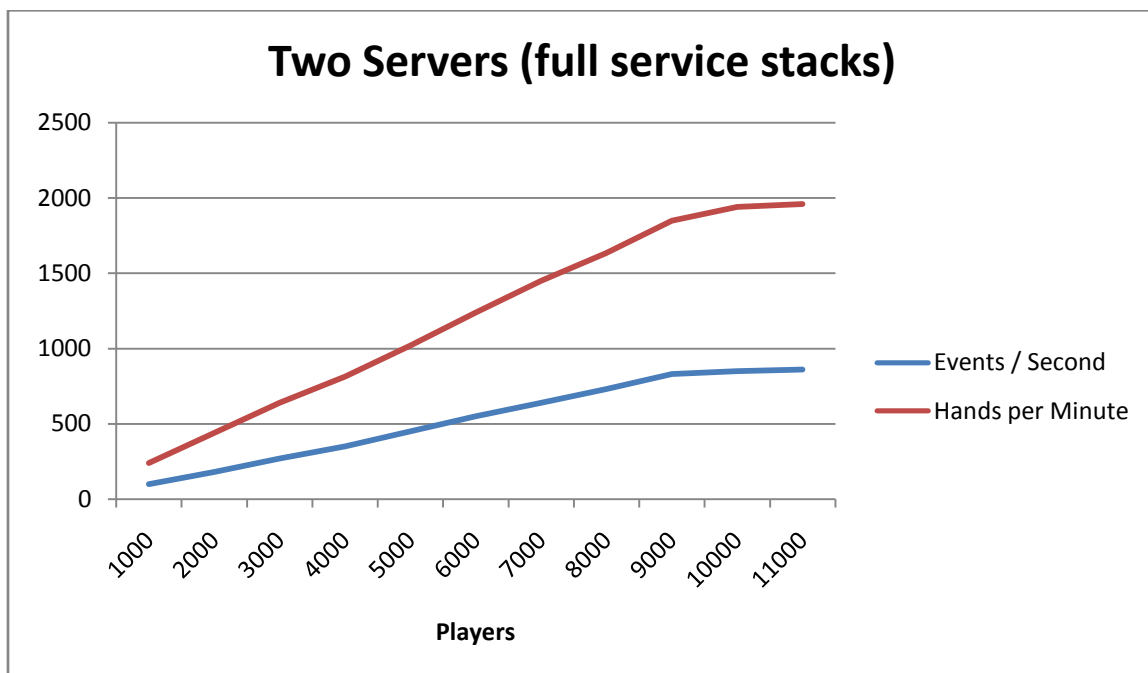
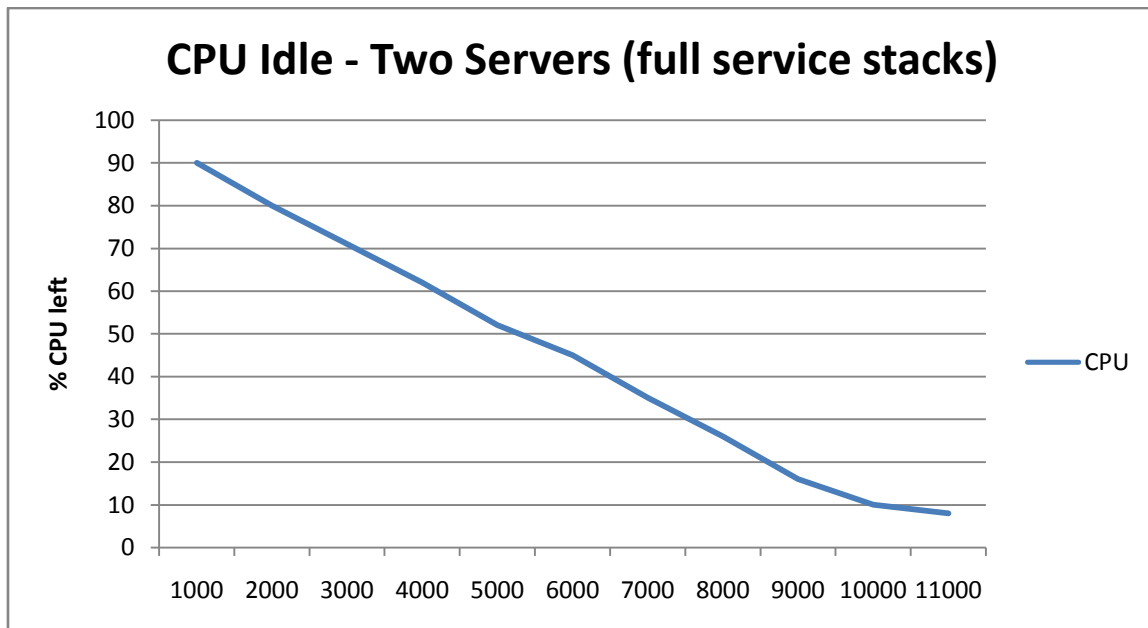
The single server executes all roles (master, client, game and tournament) in a single server process. This topology will not support any fail over.



Running all roles in a single server stack gives a maximum number of concurrent players of 9.000 before we are exhausting the CPU. It can be noted that the hands/minute trails off a little bit when the available CPU goes under 10%. This is expected since the latency will start to increase due to contention on the executing thread pools when running lots of tables.

Dual Servers

The dual server setup includes two servers executing all roles. This topology supports fail over, i.e. if one server goes down then the other server will take over.

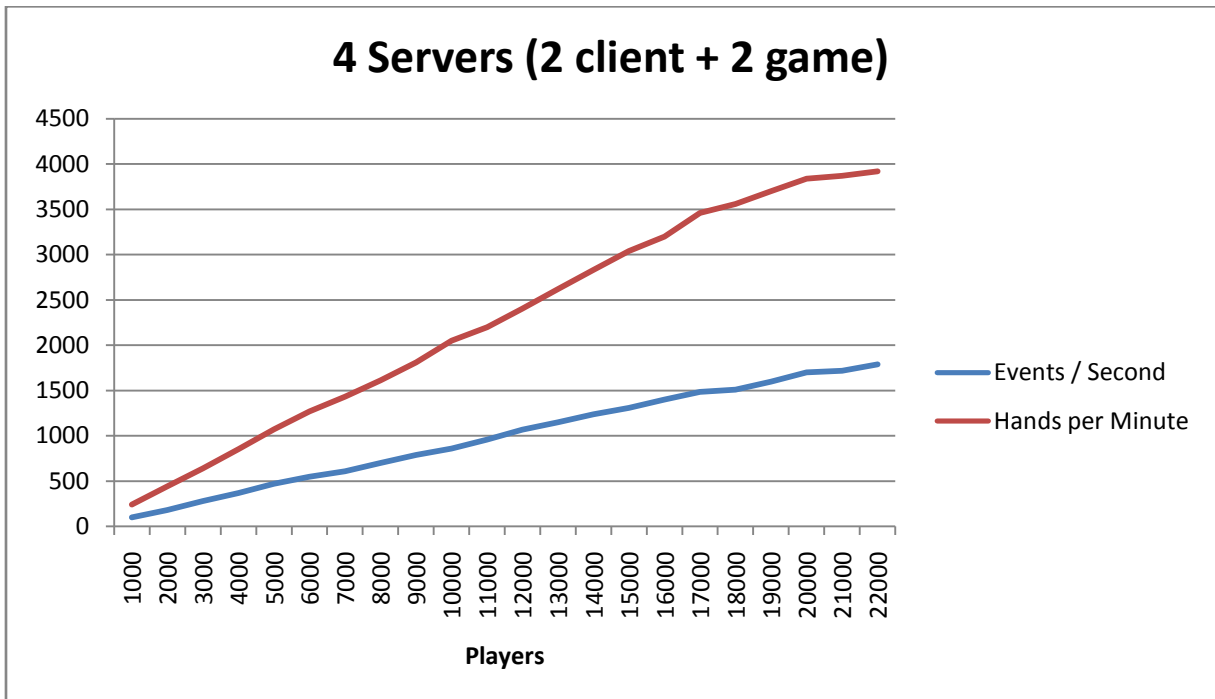
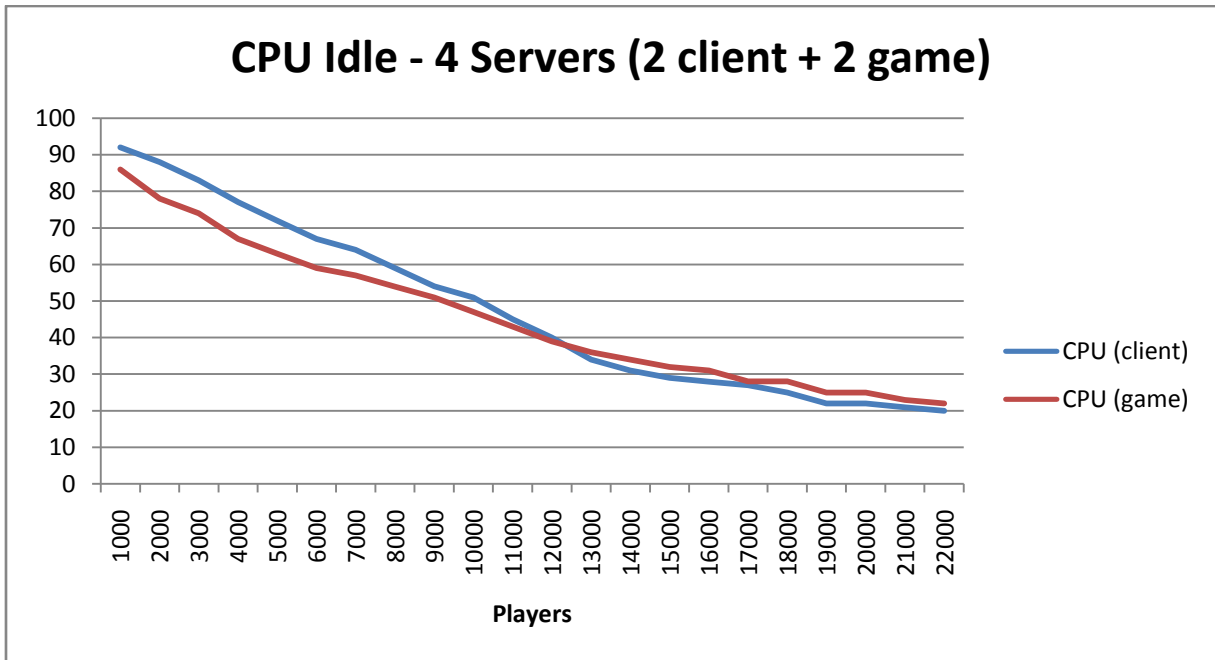


Running two servers with fail over enabled will not double the capacity of the system. This is expected as the gain is redundancy and not capacity. The capacity for two servers is still slightly higher than a single server.

The event handling trails off harder when the CPU goes under 10% than for a single server. This is due to the fact that state replication involves a remote call.

Four Servers

This topology setup separates roles in the cluster. We are running two servers that executes master and client roles and two servers that focus on the game role (tournament is excluded since we don't run any tournaments in this scenario).

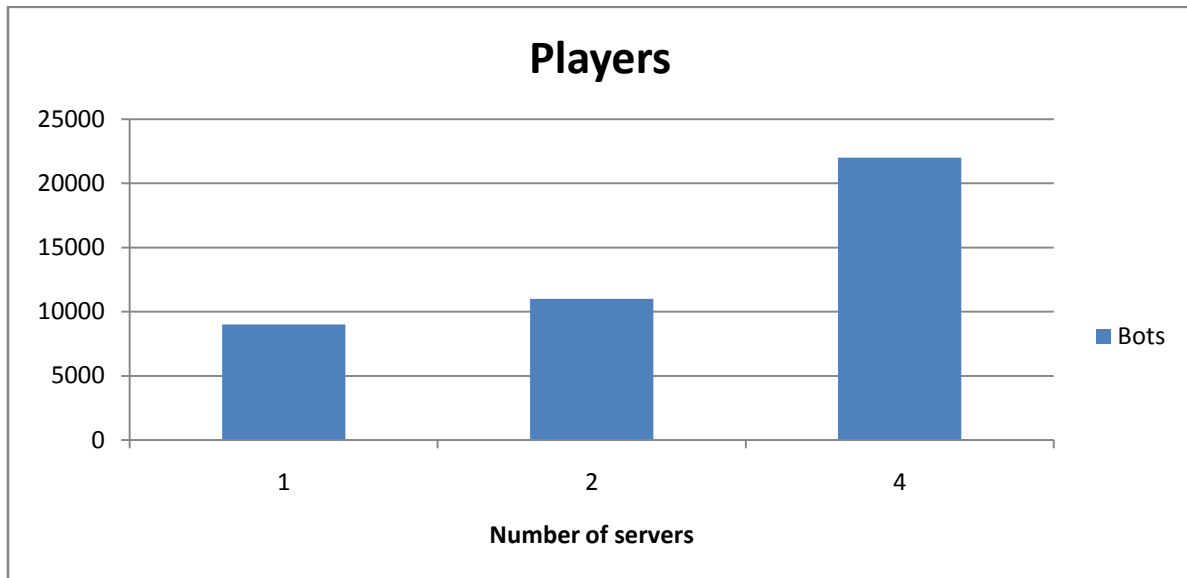


Here we see a linear scaling of capacity when compared to two servers. By adding two client servers we offload the game execution servers from handling client and lobby communication.

We can see that the hands per minute is trailing off somewhat although we still have 20% CPU available. At this rate of messaging our switch is getting overloaded and has a hard time to cope with the number of packets.

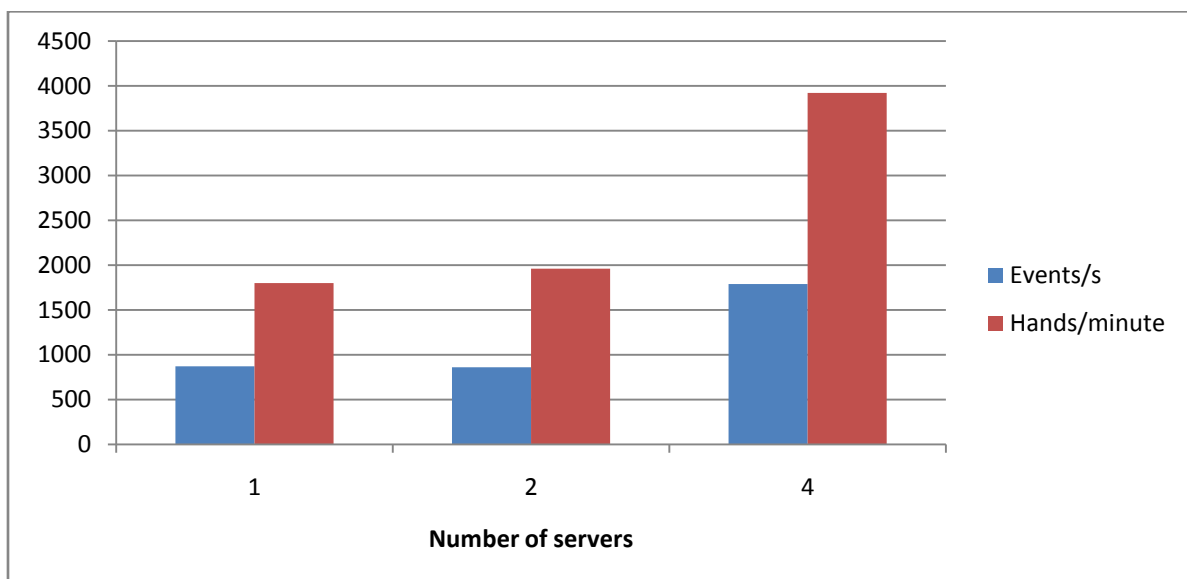
Scalability

Below is the maximum number of poker players plotted versus the number of servers in the cluster. We can see that the capacity is marginally increased between one and two servers; this is because we added redundancy and fail-over to the system at this point. When adding doubling the size of the fail over enabled system we can see a doubling of the capacity (roughly).

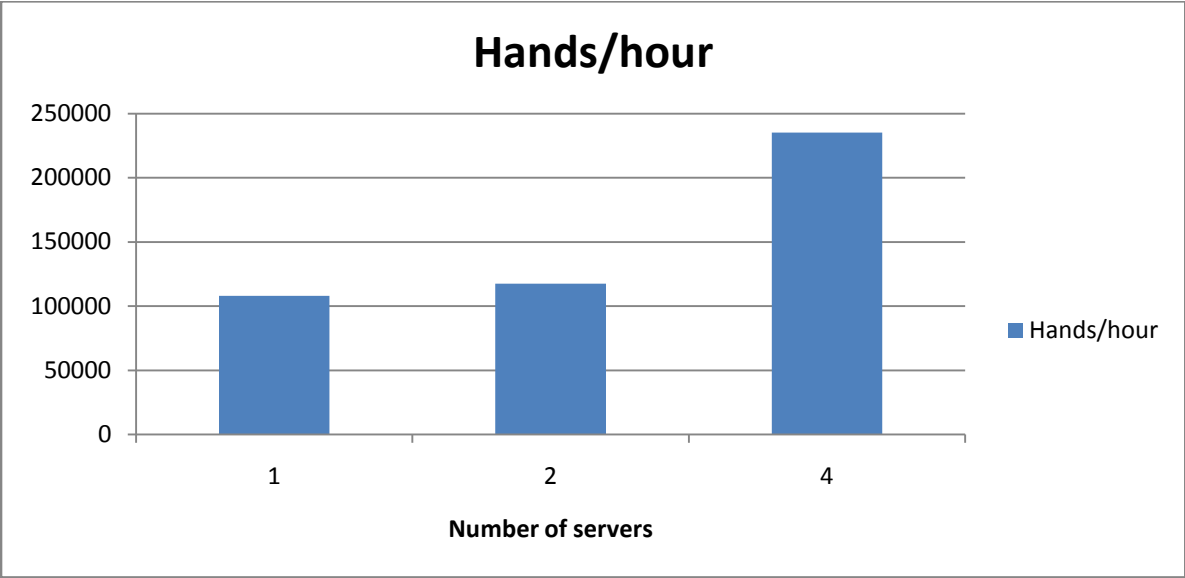


Below are the events through put plotted versus the number of servers in the cluster. We are looking for the same increase in event through put as increase in number of players, otherwise we might be adding players but are not executing more hands.

We can clearly see that we are effectively increasing event and hand through put by adding more servers.



Below is the poker hands per hour plotted versus number of servers in the cluster. Hands-per-hour is a much more used statistical figure than hands-per-minute. When running four servers at full capacity we are generating a whopping 235.000 poker hands per hour.



Summary

Since the topology of a Firebase cluster can be configured with different roles and each server can be configured differently it is not easy to present a one and true capacity test. This document has focused on running the default configuration with fail over enabled, further tweak and/or disabling fail over would most likely increase the capacity.

Scalability is proven by adding more servers to system and observing an increase in overall capacity.

It should also be noted that these tests are done on low end servers, the total cost for the four server setup (hardware wise) is well below 3.000€.