**OSTEP Research Team**

70 The Pond Road

Toronto, ON M3J 3M6

(416) 491 - 5050

BigBlueButton Profiling & Scaling (HTML5 Client)

**July 26, 2018**

# Overview

As reported by the Blindside Networks/BigBlueButton team, the current implementation of their web conferencing system can support up to 250 simultaneous users in a web meeting at one time. There are currently two implementations of the web conferencing system -- Flash and HTML5 -- with HTML5 being the successor to Flash as it is being deprecated in the near future. As such, the HTML5 client has been selected as the focal point for optimization in this project.

Tests will be conducted for different use cases to identify the current bottlenecks on the server, which will help us to determine which components need to be improved, changed, or scaled to improve performance and ultimately achieve 1000 simultaneous users in a meeting at one time.

# Goals

1. **Determine top resource consuming processes:** Through a variety of profiling tools such as htop, iftop, iotop, strace, and perf, identify whether the server’s bottlenecks are with CPU, RAM, Network, or Storage and also the cause of high resource usage.
2. **Decouple and scale:** Once the most demanding component of the server is identified, decouple it from the rest of the application, scale it through Docker & Kubernetes and reintegrate into the server.

# Specifications

* One t2.xlarge AWS instance (4 vCPU, 16GB), running a production BigBlueButton server at [https://bbbaws.cdot.systems](https://bbbaws.cdot.systems/) (running the latest version as of July 5 2018).
* One t2.medium AWS instance (2 vCPU, 4GB) for the juju controller machine
* Eighteen c3.4xlarge AWS instances (16 vCPU, 30GB) to load the BBB server with users
* One laptop logged into the HTML5 meeting to provide a user sharing audio and video

# Initial Observations

Base Server Resource Consumption:

Without BBB running:  
**CPU:** 0.7% total across cores, 0.175% core average, 0 load average  
**RAM:** 160MB  
**DISK:** 0B/s Read & Write  
**NETWORK:** 1.1Kb/s peak

With BBB running:  
**CPU:** 3.4% total across cores, 0.85% core average, 0 load average  
**RAM:** 2GB  
**DISK:** 0B/s Read & Write  
**NETWORK:** 4.95Kb/s peak

With BBB running and one meeting (1 user, no audio/video):  
**CPU:** 4.6% total across cores, 1.15% core average, 0 load average  
**RAM:** 2.24GB  
**DISK:** 0B/s Read & Write  
**NETWORK:** 20.37Kb/s peak

Additional Notes and Observations

* Some ogres fail randomly for unknown reasons, with no errors in selenium/ogre logs
* Headless chrome instances success/failure rate is inconsistent
* Sometimes when an ogre fails, it is not reusable (‘docker’ not found)
* Some headless chrome instances fail to leave the server after sleep time has expired
* Ogres crash at a high rate in the HTML5 client when trying to join a large amount of users (250) within a small time frame (30 sec or less)
* Ogre scripts fail altogether on the latest release of BBB as of Jul. 17, 2018 (doesn’t call enter API when trying to join a meeting)

# Test Scenarios

**Scenario A:** top consuming resources without users in audio (18 ogres @ 15 chrome instances each)

**Scenario B:** top consuming resources with users in audio (18 ogres @ 15 chrome instances each)

**Resources are monitored at 3 points:**

1. When the server is idle
2. While users are joining
3. After all users have joined

**Resources being monitored:**

1. CPU utilization (htop)
2. RAM utilization (htop)
3. Network traffic (iftop)
4. Disk I/O (iotop)
5. client machine cpu/memory/network/disk

# Results

The [BigBlueButton docs page](http://docs.bigbluebutton.org/install/install.html#minimum-server-requirements) recommends the following server specifications:  
Minimum:   
**CPU:** quad-core 2.6GHz  
**RAM:** 4GB with swap enabled  
**DISK:** 500GB storage (for recordings - no read or write speeds specified)  
**NETWORK:** 100Mb/s (symmectrical)

Recommended:  
**CPU:** > quad-core 2.6GHz  
**RAM:** 8GB  
**DISK:** 500GB storage (for recordings - no read or write speeds specified)  
**NETWORK:** 100Mb/s (symmectrical)

[Recall](#_oqwsre3bx3t2) our server’s resource usage without BBB running in comparison to the usage with BBB running and one meeting. We’ll compare all results going forward to the total resource usage from the server running BBB and one meeting in order to isolate how much load is being caused by users and audio/video:

**CPU:** 0.7% total across cores, 0.175% core average, 0 load average  
**RAM:** 160MB  
**DISK:** 0B/s Read & Write  
**NETWORK:** 1.1Kb/s peak  
  
**CPU:** 4.6% total across cores (+3.9%), 1.15% core average (+0.975%), 0 load average (+0)  
**RAM:** 2.24GB (+2.08GB)  
**DISK:** 0B/s Read & Write (+0B/s)  
**NETWORK:** 20.37Kb/s peak (+20.27Kb/s)

### Server Idle

When the server is idle with one user sharing audio & video, freeswitch is the top resource consuming process, however the resources used are negligible with no significant demand on CPU, RAM ([Fig. 2](#_8n4wxytj2wy9)), disk I/O, or network traffic ([Fig. 3](#_cjqvdt7mrp1b)). The client machine also shows normal resource usage ([Fig. 4](#_ia5l56sdbop4)).

**CPU:** 8.6% total across cores (+4%), 2.15% core average (+1%), 0.11 load average (+0.11)  
**RAM:** 2.36GB (+0.12GB)  
**DISK:** 0B/s read & write (+0B/s)  
**NETWORK:** 476Kb/s peak (+455.63Kb/s)

\*numbers shown in parenthesis are indicative of the increase in resource consumption compared to running BBB with 1 meeting created with no audio or video and no additional users

## **Scenario A - Without Audio**

### Server Under Load (users not in audio - approx. 200)

When the server is in the process of users joining, the Node process (Meteor) CPU usage spikes to ~150% and RAM sees a small increase ([Fig. 5](#_6icymt5wpdy5)), while disk I/O and network traffic see some increases but are still well within the minimum server requirements ([Fig. 6](#_yvnoshg2dtt2)). CPU demand at this stage is the primary resource of concern, averaging 150% for the node process and 6.77 load average. The client machine shows a significant increase, jumping from 12% to 52% in CPU usage ([Fig. 7](#_cq384m5q8yvk)).

**CPU:** 229.2% total across cores (+224.6%), 57.3% core average (+56.15%), 6.77 load average (+6.77)  
**RAM:** 3.04GB (+0.8GB)  
**DISK:** 15K/s read (+15K/s), 3M/s write (+3M/s)  
**NETWORK:** 80.5Mb/s peak (+80.48Kb/s)

### Server After Stabilizing (users not in audio - approx. 250)

Once all users have finished joining the server, Kurento media server takes over as the top process at around 67% CPU usage and 2.84 overall load average. RAM usage has increased by about 1.2GB compared to idle ([Fig. 8](#_uvvfkamxheal)) and is approaching the minimum recommended amount at this point (4GB). Disk I/O shows no read/write operations and network traffic increases from idle, but decreases from when the server is adding users in the previous test ([Fig. 9](#_l21io3txkgi0)). The client machine remains at roughly 55% CPU usage ([Fig. 10](#_vo18nd728pjy)).

**CPU:** 105.4% total across cores (+100.8%), 26.35% core average (+25.2%), 2.84 load average (+2.84)  
**RAM:** 3.53GB (+1.29GB)  
**DISK:** 0B/s read & write (+0B/s)  
**NETWORK:** 1.8Mb/s peak (+1.78Mb/s)

## **Scenario B - With Audio**

### Server Under Load (users in audio - approx. 90)

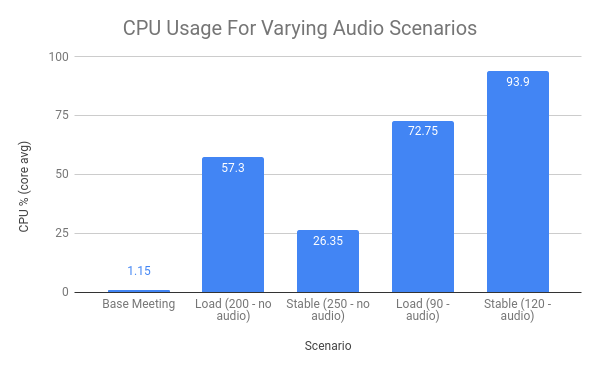
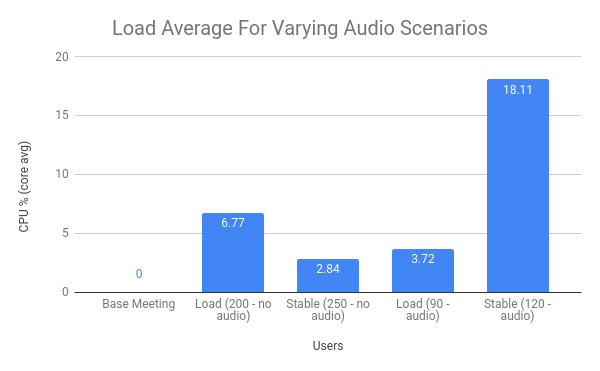
Having the users join the meeting in audio drastically changes the resource consumption. The FreeSWITCH process can be seen jumping to 165% CPU usage and 3.72 overall load average, while RAM usage has nearly doubled compared to idle values ([Fig. 11](#_9ah2fo9pmaek)). Disk I/O is minimal and network traffic increases but still stays within the 100 Mb/s recommended range ([Fig. 12](#_636ly0quhovj)). The client machine averages 50% CPU usage ([Fig. 13](#_asrfhmwd2zq2)).

**CPU:** 291% total across cores (+286.4%), 72.75% core average (+71.6%), 3.72 load average (+3.72)  
**RAM:** 4.21GB (+1.97GB)  
**DISK:** 0B/s read (+0B/s), 2.53M/s write (+2.53M/s)  
**NETWORK:** 27.3Mb/s peak (+27.28Mb/s)

### Server After Stabilizing (users in audio - approx. 120)

The server appears to reach it’s limit here, with FreeSWITCH occupying 345% CPU ([Fig. 14](#_qw77bqc318rq)) and an overall load average of 18.11. It’s worth noting that there were supposed to be ~250 total users in this meeting based on the load generation parameters, but at around 120 users in the meeting, they can be seen dropping and joining repeatedly in the user list. RAM hits 4.64GB which exceeds the minimum server specs, but if using recommended specs is only about 50% usage. Disk I/O remains unchanged and network traffic shows little usage ([Fig. 15](#_jfiqnzqis3py)) -- bizarre considering each user should be receiving audio and video. The client machine appears stable at 50% CPU usage ([Fig. 16](#_2f4e33teiytk)).

**CPU:** 375.6% total across cores (+371%), 93.9% core average (+92.75%), 18.11 load average (+18.11)  
**RAM:** 4.64GB (+2.4GB)  
**DISK:** 0B/s read and write (+0B/s)  
**NETWORK:** 13Mb/s peak (+12.98Mb/s)

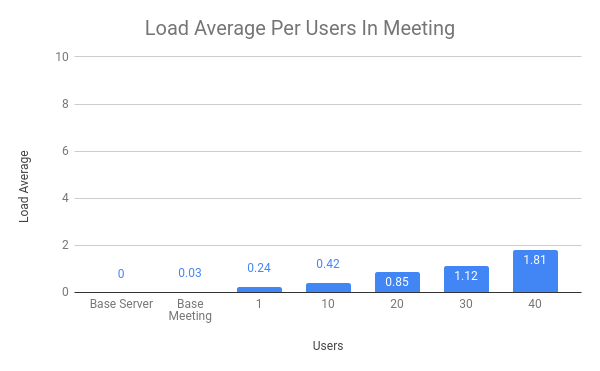
  


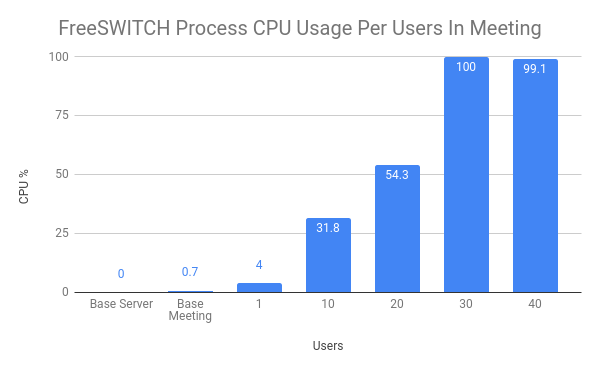
# Additional Tests

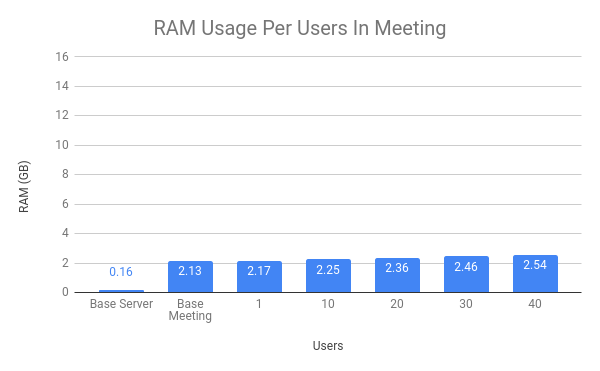
**FreeSWITCH threads/user:** 2 threads  
By performing a grep for “freeswitch” on the process list, determined that the server creates 37 initial FreeSWITCH threads, then depending on how the first user in the meeting joins audio adds 3-4 threads, and each additional users adds 2 threads each.

**CPU/user:** 0.89%  
**RAM/user:** 9.125MB  
CPU and RAM per user values were calculated by joining a meeting sharing audio and video, with each additional user joining in listen only. We can subtract the CPU/RAM values before any users have joined from the value at 10, 20, 30, 40 users and divide by the number of users at each stage to estimate a value/user.

## Chart

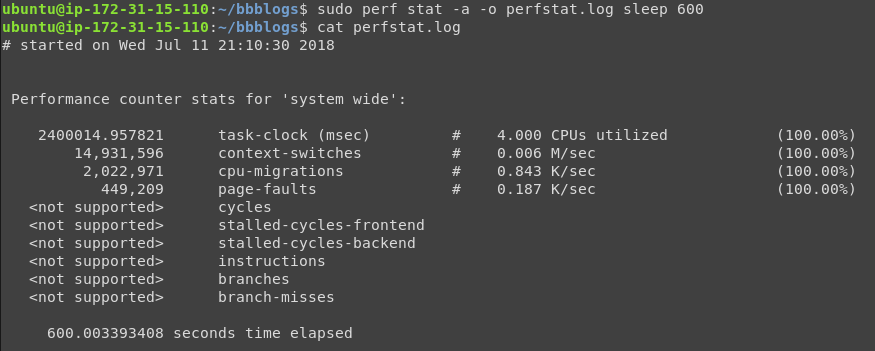


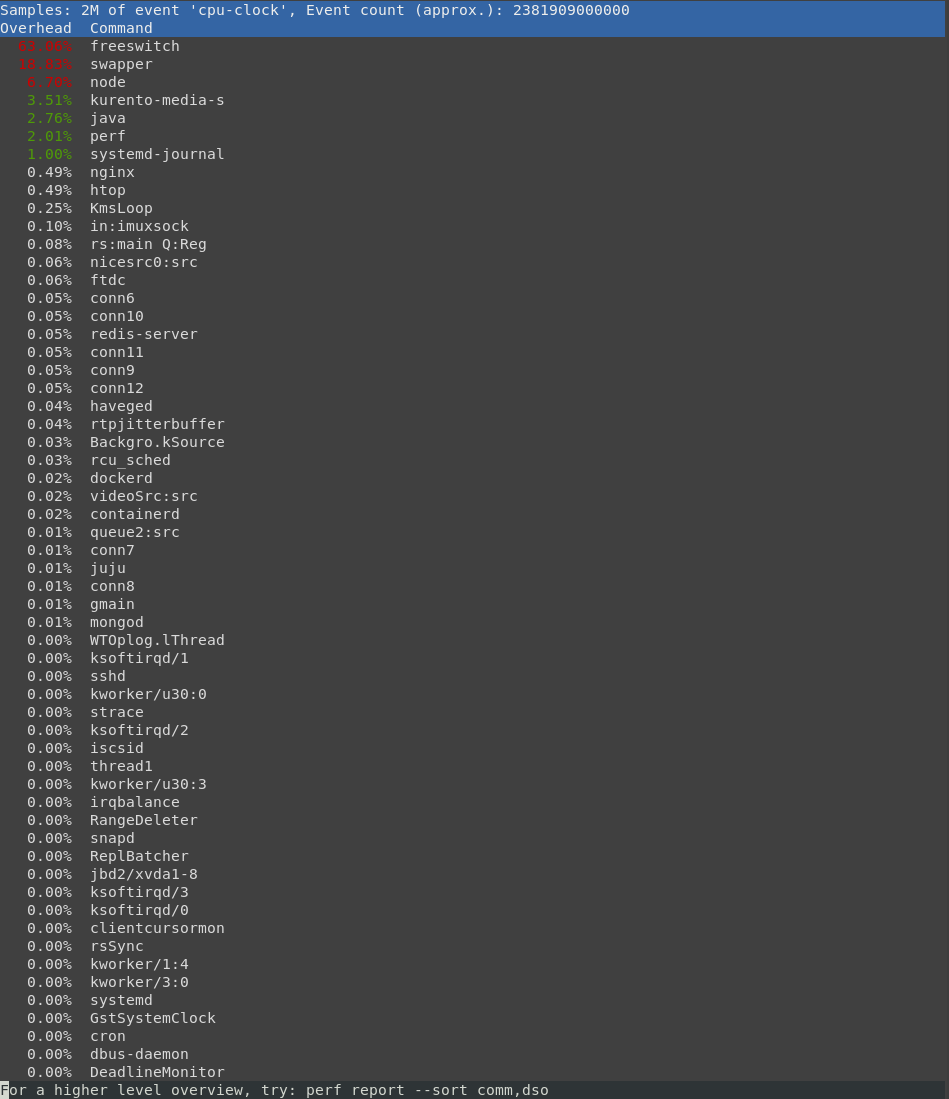




## Perf

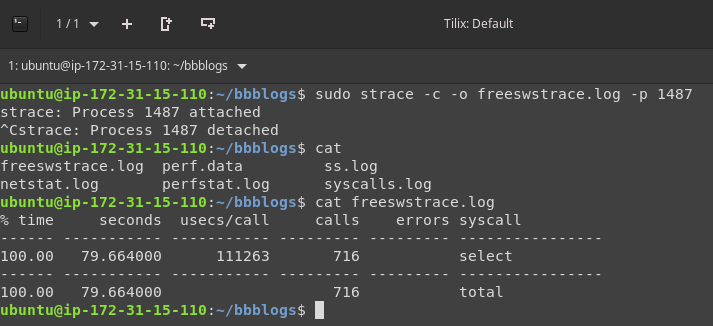
System wide perf stat collection from all CPUs, collected over 10 minutes, generated by:  
**$ sudo perf stat -a -o perfstat.log sleep 600**



Perf report, sorted by command. The perf report was generated by:  
**$ sudo perf record -a -F 1000 sleep 600  
$ sudo perf report -s comm**

## Strace

The process being traced here is FreeSWITCH, we can see that 716 ‘select’ system calls are being made; strace report generated by:  
**$ sudo strace -c -o freeswstrace.log -o 1487**



# Conclusion

From the results seen above, it becomes evident that RAM, Disk I/O, and Network traffic are not the current bottlenecks in the BigBlueButton server. This leaves us with CPU, which we see spikes with 3 processes in our two scenarios:

* **Node/Meteor** when users are joining or leaving within a small time frame
* **Kurento Media Server** when a large number of users exist in the meeting at one time
* **FreeSWITCH** when a large number of users exist in the meeting in audio (most typical use case)

Of the two scenarios tested, having users join the meeting in audio was the most taxing on the CPU by a significant amount. It is worth noting that the ‘no audio’ scenarios were able to join 250 users within a 30 second window, while the ‘in audio’ scenarios had to spread joining 250 users (CPU limit reached at 120) over a 3 minute window to prevent users from instantly dropping/failing. This could explain why the ‘no audio’ scenario saw high load average values compared to some of the ‘in audio’ scenarios, as more users were joining the server at once.   
At this time if we are to proceed with scaling BBB server as-is, the primary candidate for docker scaling should be FreeSWITCH for the following reasons:

* Node/Meteor high CPU only occurs under specific conditions; during the joining/leaving process -- which is temporary -- and can be relieved by spreading out how quickly users join the meeting
* Kurento CPU spike only occurs at very high user numbers, and Kurento is not currently handling audio. Once FreeSWITCH is containerized and higher user numbers can be reached in audio, or if Kurento is fixed to handle the listen-only users, Kurento will be the biggest bottleneck and candidate for containerization.

**Additional notes on Kurento:** When a user joins audio in listen only mode in the HTML5 client, it is a ‘faked’ listen only in that it still joins the user in a full-duplex audio mode with a dead microphone object; Kurento is not properly handling listen-only users in the way that Red5 does on the Flash client. Each additional HTML5 user is being connected to FreeSWITCH when they should only be connecting to Kurento, causing unnecessary FreeSWITCH processing.

# Appendix

## Fig. 1: AWS Instance CPU Information



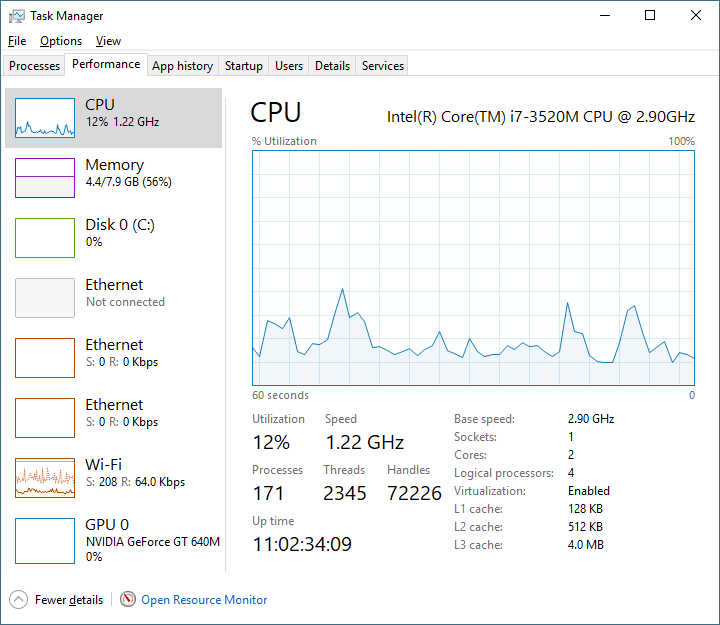
## Fig. 2: Server Idle - CPU & RAM - 1 user in meeting with audio & video:

## 

## Fig. 3: Server Idle - Disk & Network - 1 user in meeting with audio & video:

## 

## Fig. 4: Client Idle - Machine Resources - 1 user in meeting with audio & video:



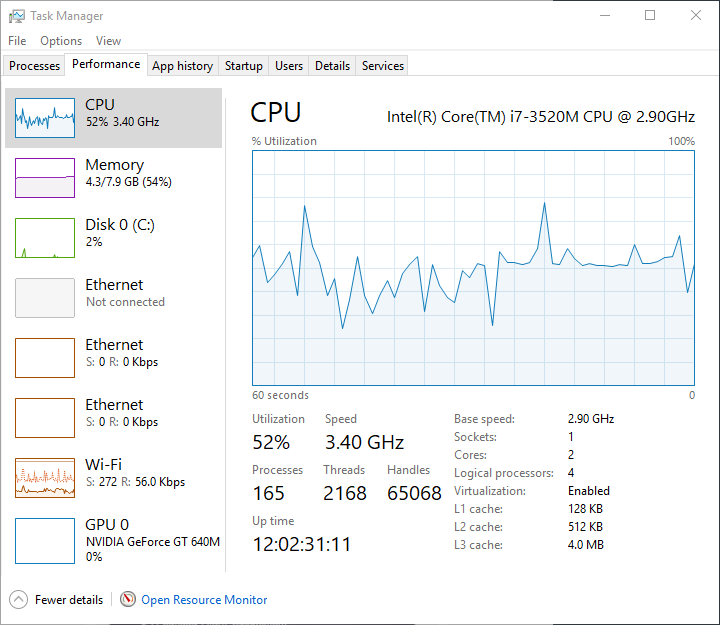
## Fig. 5: Server Under Load - CPU & RAM (no audio):

## 

## Fig. 6: Server Under Load - Disk & Network (no audio):

## 

## Fig. 7: Client Under Load - Machine Resources (no audio):



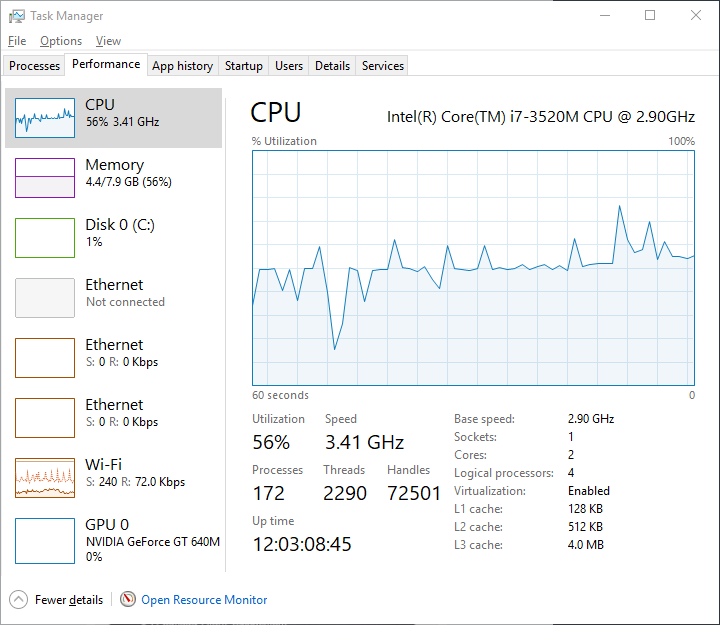
## Fig. 8: Server After Stabilizing - CPU & RAM (no audio):

## 

## Fig. 9: Server After Stabilizing - Disk & Network (no audio):

## 

## Fig. 10: Client After Stabilizing - Machine Resources (no audio):



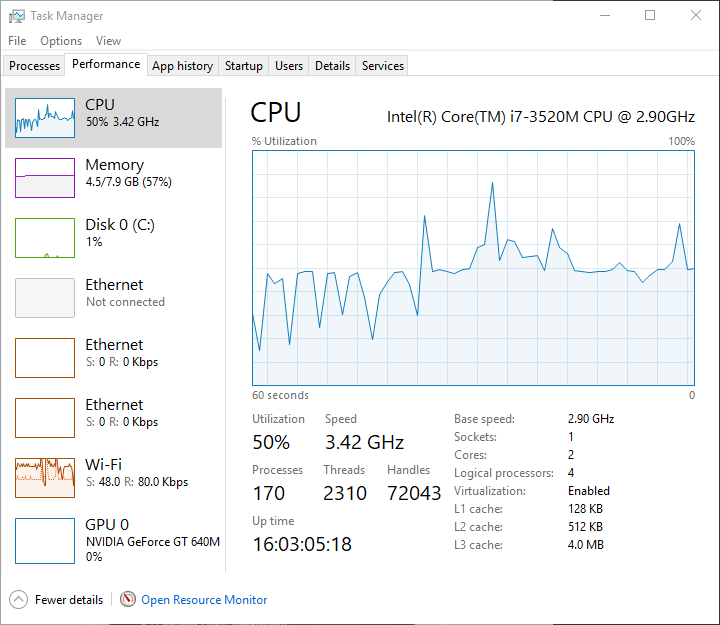
## Fig. 11: Server Under Load - CPU & RAM (with listen only audio):

## 

## Fig. 12: Server Under Load - Disk & Network (with listen only audio):

## 

## Fig. 13: Client Under Load - Machine Resources (with listen only audio):



## Fig. 14: Server After Stabilizing - CPU & RAM (with listen only audio):

## 

## 

## Fig. 15: Server After Stabilizing - Disk & Network (with listen only audio):

## 

## Fig. 16: Client After Stabilizing - Machine Resources (with listen only audio):

## 

## 