**CPU:**

In this experiment we measure processor performance in terms of GIOPS & GFLOPS ;

Program Design:

The program computes Integer/Floating point operation in a loop which iterates over a million times. Each thread created has its own loop and performs similar operations except with different variables values.

Trade-off:

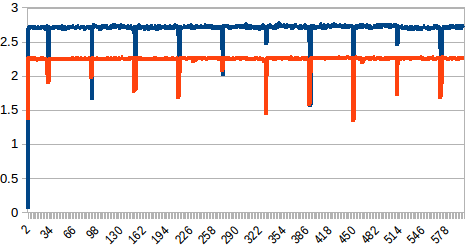
As operations done are kept simple at best no synchronizations or thread communication is made.

Improvements:

the operations could be made bigger & complex and have to do with thread synchronization to achieve higher efficiency.

Results:

Analysis: we can observer that there is silight increase in performance with number of threads. Threading helps to use cpu ideal time better.



x-axis: time Y-axis Giga operations blue: iops orange: flops

Analysis: we can see there is a drop in performance occasionally but the processor performance remains fairly constant.

**c.**  Theoretical peak performance = GHz\*Cores\*IPC

Cpu model name: Intel ® Xenon E5-2676 v3

Ghz:2.4

Cores: 1

IPC: 8

Theoretical peak performance= 19.2 Gflops

reference:<https://aws.amazon.com/ec2/instance-types/>

http://browser.primatelabs.com/geekbench2/2577759

And cmd on linux

**d.**

Effiency = 11.97%

Analysis : there are many factors that important for performance wise: benchmark code, OS, processor make & in this case even cpu credit.

Aws processor performance does depend on availability of underlying hardware & credit limit set. However they do promise 10% base performance all times.

Reference: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/t2-instances.html

**f.**

Linpack pack results were run. And can be found “lin\_Xenon.txt” under cpu directory we peek result of 31.70 Gflops for 4 threads and average of atleast 25 Gflops

Linpack is a great tool to measure processor proformance, uses martices to solve linear algebra solutions. It makes the processor throthell to reach its limit.

**DISK:**

The Experiment runs to find disk latency (ms) and throughput(mb/s)

Program Design:

The program attempts to read or write (in sequential & random fashion) blocks of predefined size into a huge file; we play with of data for each configuration.

Trade-off:

We synchronize threads created to our advantage, but repeatedly have to flush buffers explicitly or have to set seek to change file pointer (in random access). But this cost is nullified by huge amount of data we handle in loops which run huge number of time.

The fastest api in java-streams is used to write to or read from files directly in bytes.

Improvements:

A larger dataset can be chosen to work with.

Results:

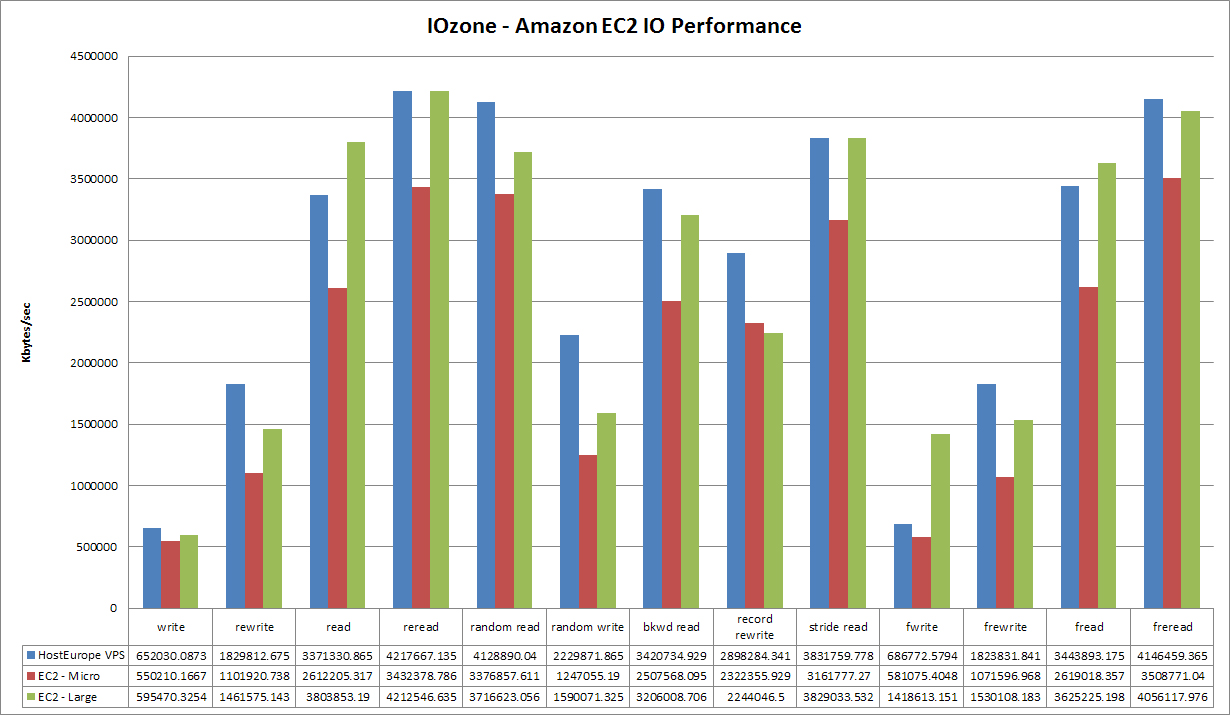
Benchmark done produces results in “results.txt”. One done on EC2 t2.micro is present in disk directory.

We can observe that for random access the latency is very high, but throughput is more. This is cause random access take time to chnage the file pointer for the first time. Later stream of bytes are quick to access.

Note: infinity dosent mean in literal sense java get this value when long division value happens with very low denomenator value (like time here) we just need a better precission to get excate value.

**f.**

iozone benchmark results found online:

reference: [https://www.krenger.ch/blog/amazon-ec2-io-performance/](https://www.krenger.ch/blog/amazon-ec2-io-performance/)

And AWS claim for there general purpose SSD-(gp2) use for ec2-t2.micor is

160 mb/s (for write)

[http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html#EBSVolumeTypes\_gp2](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html" \l "EBSVolumeTypes_gp2)

Effiency:

(taken for higest result config)

sequence write; 60.79% read: 86.23%

**NETWORK:**

The Experiment tries to find out latency(ms) & Throughput(mb/s) of network.

Program Design:

The program send data and receives it to measure the above mentioned metrics at client-side.

Trade-off:

No synchronized is done between threads, as communication port for each thread is different.

Improvements:

More threads can be made in such a manner as variant a to network performance monitored at small intervals.

A larger dataset can be chosen to work with.

Results:

-

ipref was done results can be found in images in network directory.

The benchmark was performed on private sub-net involving two t2.micro instances to get results of 980 mb/sec