Cassandra on Armv8 - A comparison with x86 and other platforms

Sankalp Sah, Manish Singh
MityLytics Inc

Why ARM for Cassandra ?

- RISC architecture as opposed to x86
- Lower Cost
- Thermals
- Power and it’s management
- Cost per operation
- High number of CPUs on each board
- Memory throughput
- Lots of simple instructions executed in parallel

Caveats

1. Bleeding edge
2. Performance not yet tuned
3. Efforts on to tune for ARM via AdoptJDK and Linaro distributions

ARMv8 - Specifications of platform

Each machine :

1. 96-core Cavium ThunderX @2GHz
2. 128GB RAM
3. 1 x 340GB Enterprise SSD
4. 2 x 10Gbps Bonded Ports

Evaluation - The operator view

- Cost - $0.50/hour at Packet.net 96 core ThunderX from cavium at 2.0GHz
- Thermals
- Power consumption
- Dollar cost per-operation
- Utilization - Workload fit

Evaluation of Performance - micro perspective

- Write operations
- Read-write mix
- Max achievable
- Latency
- Co-tenanted applications - should not evaluate in isolation.

1 million writes with default cassandra config in a 3-node cluster

1. Throughput
   a. Max operations per sec - 192,449
   b. Sustained Throughput - 129,170
2. Latency
   a. Latency mean : 1.5 [WRITE:1.5]
   b. latency median : 0.8 [WRITE:0.8]
   c. latency 95th percentile : 2.6 [WRITE:2.6]
   d. latency 99th percentile : 7.3 [WRITE:7.3]
   e. latency 99.9th percentile : 170.9 [WRITE:170.9]
   f. latency max : 321.6 [WRITE:321.6]

10 million writes with default cassandra config in a 3-node cluster

1. Throughput
   a. Max operations per sec - 220,000
   b. Sustained Throughput - 137,689
2. Latency
   a. latency mean : 1.4 [WRITE:1.4]
   b. latency median : 0.8 [WRITE:0.8]
   c. latency 95th percentile : 1.3 [WRITE:1.3]
   d. latency 99th percentile : 4.3 [WRITE:4.3]
   e. latency 99.9th percentile : 45.4 [WRITE:45.4]
   f. latency max : 397.0 [WRITE:397.0]

20 million writes with default cassandra config in a 3-node cluster

1. Throughput
   a. Max operations per sec - 193,220
   b. Sustained Throughput - 124,784
2. Latency
   a. latency mean : 1.5 [WRITE:1.5]
   b. latency median : 0.8 [WRITE:0.8]
   c. latency 95th percentile : 1.4 [WRITE:1.4]
   d. latency 99th percentile : 4.3 [WRITE:4.3]
   e. latency 99.9th percentile : 41.1 [WRITE:41.1]
   f. latency max : 567.4 [WRITE:567.4]

50 million writes with default cassandra config in a 3-node cluster

1. Throughput
   a. Max operations per sec - 206k
   b. Sustained Throughput - 129,000
2. Latency
   a. latency mean : 1.5 [WRITE:1.5]
   b. latency median : 0.8 [WRITE:0.8]
   c. latency 95th percentile : 1.3 [WRITE:1.3]
   d. latency 99th percentile : 2.1 [WRITE:2.1]
   e. latency 99.9th percentile : 72.3 [WRITE:72.3]
   f. latency max : 584.0 [WRITE:584.0]

1 million Read-Write mixed workloads -75%read 25% writes

1. Throughput
    a. Max operations per sec - 124k
    b. Sustained Throughput - 123k
2. Latency
    a. atency mean : 2.1 [READ:2.5, WRITE:1.2]
    b. latency median : 0.7 [READ:0.7, WRITE:0.7]
    c. latency 95th percentile : 6.2 [READ:6.4, WRITE:2.2]
    d. latency 99th percentile : 7.6 [READ:8.1, WRITE:2.7]
    e. latency 99.9th percentile : 51.5 [READ:54.7, WRITE:25.3]
    f. latency max : 124.0 [READ:124.0, WRITE:113.0]

10 million Read-Write mixed workloads -75%read 25% writes

1. Throughput
    a. Peak 150,842
    b. Sustained 122,000
2. Latency
    a. latency mean : 4.9 [READ:5.1, WRITE:4.1]
    b. latency median : 2.2 [READ:2.3, WRITE:1.7]
    c. latency 95th percentile : 6.2 [READ:6.7, WRITE:5.4]
    d. latency 99th percentile : 25.2 [READ:88.8, WRITE:85.3]
    e. latency 99.9th percentile : 125.9 [READ:128.7, WRITE:127.2]
    f. latency max : 256.2 [READ:256.2, WRITE:247.4]

20 million Read-Write mixed workloads -75%read 25% writes

1. Throughput
    a. Peak 147k
    b. Sustained 138k
2. Latency
    a. latency mean : 6.6 [READ:6.8, WRITE:5.8]
    b. latency median : 3.1 [READ:3.3, WRITE:2.7]
    c. latency 95th percentile : 9.6 [READ:10.5, WRITE:8.5]
    d. latency 99th percentile : 97.7 [READ:104.3, WRITE:99.6]
    e. latency 99.9th percentile : 138.6 [READ:142.0, WRITE:140.5]
    f. latency max : 429.4 [READ:429.4, WRITE:421.9]

50 million Read-Write mixed workloads -75% Read 25% Writes

1. Throughput
    a. Peak 155k
    b. Sustained 135k
2. Latency
    a. atency mean : 6.7 [READ:6.9, WRITE:6.0]
    b. latency median : 3.2 [READ:3.4, WRITE:2.7]
    c. latency 95th percentile : 8.6 [READ:9.5, WRITE:8.0]
    d. latency 99th percentile : 101.3 [READ:117.9, WRITE:107.8]
    e. latency 99.9th percentile : 140.0 [READ:142.4, WRITE:141.6]
    f. latency max : 229.2 [READ:229.2, WRITE:186.4]

Perf counters for ARM - while running cassandra stress

Overall CPU at 44%, Memory usage at 60GB

711069.602520	task-clock (msec)	#	96.046 CPUs utilized
14,802	context-switches	#	0.004 K/sec
137	cpu-migrations	#	0.000 K/sec
7,207	page-faults	#	0.002 K/sec
7,422,259,052,720	cycles	#	2.000 GHz
3,929,716,281	stalled-cycles-frontend	#	0.05% frontend cycles idle
7,384,719,523,004	stalled-cycles-backend	#	99.49% backend cycles idle
43,938,297,479	instructions	#	0.01 insns per cycle
		#	168.07 stalled cycles per insn
6,114,998,824	branches	#	1.648 M/sec
375,388,710	branch-misses	#	6.14% of all branches

Performance counter stats for the JVM

```
 1.        285.560310      task-clock (msec)         #    1.239 CPUs utilized
 2.               359      context-switches          #    0.001 M/sec
 3.               231      cpu-migrations            #    0.809 K/sec
 4.             2,855      page-faults               #    0.010 M/sec
 5.       565,162,728      cycles                    #    1.979 GHz
 6.       114,307,459      stalled-cycles-frontend   #   20.23% frontend cycles idle
 7.       280,646,883      stalled-cycles-backend    #   49.66% backend  cycles idle
 8.       205,551,207      instructions              #    0.36  insns per cycle
 9.                                                  #    1.37  stalled cycles per insn
10.        28,882,484      branches                  #  101.143 M/sec
11.         4,453,137      branch-misses             #   15.42% of all branches
```

Packet.net Type-1 node

- Intel E3-1240 v3 - 4 physical Cores @ 3.4 GHz
- 32GB
- 2 x 120GB Enterprise SSD
- 2 x 1Gbps Bonded Ports
- $0.40/hr - on demand pricing

1 million writes

Throughput - 3 node
Peak : 174877
Sustained : 154738
Latency:
latency mean : 1.3 [WRITE:1.3]
latency median : 0.7 [WRITE:0.7]
latency 95th percentile : 2.7 [WRITE:2.7]
latency 99th percentile : 5.0 [WRITE:5.0]
latency 99.9th percentile : 44.7 [WRITE:44.7]
latency max : 82.5 [WRITE:82.5]

1 million Read-Write mixed workloads -75%read 25% writes

1. Throughput
    a. Max operations per sec - 117k
    b. Sustained Throughput - 117k
2. Latency

    a. latency mean : 1.5 [READ:1.6, WRITE:1.3]
    b. latency median : 1.5 [READ:0.7, WRITE:0.6]
    c. latency 95th percentile : 4.2[READ:4.5, WRITE:3.6]
    d. latency 99th percentile : 9.9 [READ:10.6, WRITE:9.6]
    e. latency 99.9th percentile : 86.5 [READ:86.7, WRITE:51.6]
    f. latency max : 88 ms [READ:88.0, WRITE:86.2]

10 million Read-Write mixed workloads -75%read 25% writes

1. Throughput
    a. Max operations per sec - 86k
    b. Sustained Throughput - 80k
2. Latency

    a. latency mean : 5.0 [READ:5.1, WRITE:4.9]
    b. latency median : 1.8 [READ:1.8, WRITE:1.7]
    c. latency 95th percentile : 15.5 [READ:16.4, WRITE:14.8]
    d. latency 99th percentile : 43.0 [READ:49.2, WRITE:43.5
    e. latency 99.9th percentile : 87.4 [READ:97.4, WRITE:86.1]
    f. latency max : 377.3 [READ:377.3, WRITE:299.7]

Performance counters - while running Cassandra-stress - Type 1

243304.786828	task-clock (msec)	# 7.994 CPUs utilized
4,770,619	context-switches	# 0.020 M/sec
533,669	cpu-migrations	# 0.002 M/sec
32,955	page-faults	# 0.135 K/sec
823,721,139,097	cycles	# 3.386 GHz
793,542,050,783	instructions	# 0.96 insns per cycle
139,500,426,441	branches	# 573.357 M/sec
1,239,316,562	branch-misses	# 0.89% of all branches

Idle perf counters - Type 1 cluster on Packet

75615.159586	task-clock (msec)	#	7.995 CPUs utilized
1,504	context-switches	#	0.020 K/sec
23	cpu-migrations	#	0.000 K/sec
2	page-faults	#	0.000 K/sec
102,605,745	cycles	#	0.001 GHz
17,332,602	instructions	#	0.17 insns per cycle
3,499,854	branches	#	0.046 M/sec
412,360	branch-misses	#	11.78% of all branches

Characterisation for Type-1

1. Peak CPU hit 81%
2. Peak Memory hit ~100
3. Performance starts to degrade after 1 million read-writes.

Amazon i3.4xlarge - 1million writes - 3 nodes

Throughput:
Peak 135,700
Sustained 114,906 op/s [WRITE: 114,906 op/s]

Latency:
Latency mean : 1.7 ms [WRITE: 1.7 ms]
Latency median : 1.1 ms [WRITE: 1.1 ms]
Latency 95th percentile : 4.0 ms [WRITE: 4.0 ms]
Latency 99th percentile : 10.5 ms [WRITE: 10.5 ms]
Latency 99.9th percentile : 90.8 ms [WRITE: 90.8 ms]
Latency max : 218.8 ms [WRITE: 218.8 ms]

Amazon i3.4xlarge - 10 million writes - 3 nodes

Throughput	:	126,064 op/s [WRITE: 126,064 op/s]
Latency mean	:	1.6 ms [WRITE: 1.6 ms]
Latency median	:	1.0 ms [WRITE: 1.0 ms]
Latency 95th percentile	:	3.8 ms [WRITE: 3.8 ms]
Latency 99th percentile	:	8.5 ms [WRITE: 8.5 ms]
Latency 99.9th percentile	:	99.4 ms [WRITE: 99.4 ms]
Latency max	:	268.7 ms [WRITE: 268.7 ms]

AWS i3 4xlarge - 20 million writes - 3 nodes

Throughput	:	123,022 op/s [WRITE: 123,022 op/s]
Latency mean	:	1.6 ms [WRITE: 1.6 ms]
Latency median	:	1.0 ms [WRITE: 1.0 ms]
Latency 95th percentile	:	3.8 ms [WRITE: 3.8 ms]
Latency 99th percentile	:	8.4 ms [WRITE: 8.4 ms]
Latency 99.9th percentile	:	82.9 ms [WRITE: 82.9 ms]
Latency max	:	195.3 ms [WRITE: 195.3 ms]

AWS i3 4xlarge - 50 million writes - 3 node

Throughput	:	121,055 op/s [WRITE: 121,055 op/s]
Latency mean	:	1.6 ms [WRITE: 1.6 ms]
Latency median	:	1.0 ms [WRITE: 1.0 ms]
Latency 95th percentile	:	3.8 ms [WRITE: 3.8 ms]
Latency 99th percentile	:	8.8 ms [WRITE: 8.8 ms]
Latency 99.9th percentile	:	80.8 ms [WRITE: 82.9 ms]
Latency max	:	156 [WRITE: 156.3 ms]

AWS i3 4xl perf counters

JVM

1309.462273	task-clock (msec)	#	16.294 CPUs utilized
1,296	context-switches	#	0.990 K/sec
87	cpu-migrations	#	0.066 K/sec
3,304	page-faults	#	0.003 M/sec

AWS i3.4xlarge - 3 node - with external client

Peak throughput - 310k ops/sec

```
Op rate                   :  272,544 op/s  [WRITE: 272,544 op/s]
Partition rate            :  272,544 pk/s  [WRITE: 272,544 pk/s]
Row rate                  :  272,544 row/s [WRITE: 272,544 row/s]
Latency mean              :    3.3 ms [WRITE: 3.3 ms]
Latency median            :    1.9 ms [WRITE: 1.9 ms]
Latency 95th percentile   :    5.7 ms [WRITE: 5.7 ms]
Latency 99th percentile   :   71.2 ms [WRITE: 71.2 ms]
Latency 99.9th percentile :   92.5 ms [WRITE: 92.5 ms]
Latency max               :  254.9 ms [WRITE: 254.9 ms]
```

AWS i3.4xlarge - single node - write workload

Peak Throughput	:	133k ops/sec - 20 million operations.
Sustained throughput	:	114k ops/sec
Latency mean	:	1.7 ms
Latency median	:	1.2 ms
Latency 95th percentile	:	3.2 ms
Latency 99th percentile	:	6.6 ms
Latency 99.9th percentile	:	94.0 ms
Latency max	:	584.1 ms

AWS i3.4xlarge - single node - mixed read 75%-write 25%

Peak Throughput	:	112.5k ops/sec - 100 million operations.
Sustained throughput	:	102k ops/sec
Latency mean	:	8.9 ms [READ: 8.9 WRITE: 8.9ms]
Latency median	:	6.4 ms [READ: 6.4 WRITE: 6.4 ms]
Latency 95th percentile	:	19.3 ms [READ: 19.4 WRITE: 19.6 ms]
Latency 99th percentile	:	65.4 ms [READ: 65.4 WRITE: 65.5 ms]
Latency 99.9th percentile	:	103.5 ms [READ: 103.4 WRITE: 103.9 ms]
Latency max	:	276.0 ms [READ: 276.0 WRITE: 271.3 ms]

GPU - g2.8xlarge node - 1 million writes

Throughput :

Peak throughput : 118,973 ops/sec
Sustained : 102,840 ops/sec

Latency:

Latency mean	:	1.8 ms	[WRITE: 1.8 ms]
Latency median	:	1.4 ms	[WRITE: 1.4 ms]
Latency 95th percentile	:	2.7 ms	[WRITE: 2.7 ms]
Latency 99th percentile	:	4.7 ms	[WRITE: 4.7 ms]
Latency 99.9th percentile	:	125.3 ms	[WRITE: 125.3 ms]
Latency max	:	204.7 ms	[WRITE: 204.7 ms]

GPU - g2.8xlarge node - 10 million writes

Throughput :

Peak throughput : 131,658 ops/sec
Sustained : 109,202 ops/sec

Latency:
Latency mean : 1.8 ms [WRITE: 1.8 ms]
Latency median : 1.5 ms [WRITE: 1.5 ms]
Latency 95th percentile : 2.6 ms [WRITE: 2.6 ms]
Latency 99th percentile : 3.7 ms [WRITE: 3.7 ms]
Latency 99.9th percentile : 118.0 ms [WRITE: 118.0 ms]
Latency max : 551.0 ms [WRITE: 551.0 ms]

GPU - g2.8xlarge node - 20 million writes

Throughput :

Peak throughput : 120,640 ops/sec
Sustained : 99,776 ops/sec

Latency:
Latency mean : 2.0 ms [WRITE: 1.8 ms]
Latency median : 1.5 ms [WRITE: 1.5 ms]
Latency 95th percentile : 2.8 ms [WRITE: 2.6 ms]
Latency 99th percentile : 3.9 ms [WRITE: 3.7 ms]
Latency 99.9th percentile : 109.6 ms [WRITE: 118.0 ms]
Latency max : 1201.7 ms [WRITE: 551.0 ms]

GPU - g2.8xlarge node - 75% Read- 25%write mixed workload

Throughput :

Peak throughput : 117, 759 ops/sec - For 10 million operations
Sustained : 107, 049 ops/sec

Latency:
Latency mean : 8.5 ms [READ 8.8 ms, WRITE: 7.6 ms]
Latency median : 6.6 ms [READ 6.8, WRITE: 5.8 ms]
Latency 95th percentile : 15.5 ms [READ 15.5 ms WRITE: 14.4 ms]
Latency 99th percentile : 66.5 ms [READ 67.0 ms WRITE: 64.1 ms]
Latency 99.9th percentile : 91.3 ms [READ 91.8 WRITE: 90.0 ms]
Latency max : 213.1 ms [READ 213.1 WRITE: 134.3 ms]

ARMv8 - Type 2a node from Packet - 10 million writes

Throughput :

Peak throughput : 140,247ops/sec
Sustained : 91,675 ops/sec

Latency:

Latency mean	:	2.1 ms	[WRITE: 2.1 ms]
Latency median	:	1.3 ms	[WRITE: 1.3 ms]
Latency 95th percentile	:	3.5 ms	[WRITE: 3.5 ms]
Latency 99th percentile	:	8.1 ms	[WRITE: 8.1 ms]
Latency 99.9th percentile	:	83.8 ms	[WRITE: 83.8 ms]
Latency max	:	470.3 ms	[WRITE: 470.3 ms]

ARMv8 - Type 2a node from Packet - 75% Read- 25%write mixed workload

Throughput :

Peak : 59,110 ops/sec - For 10 million operations

Sustained : 54,909 ops/sec

Latency:

Latency mean	:	2.2ms [READ: 2.5 ms WRITE: 1.0 ms]
Latency median	:	1.7 ms[READ: 1.8 ms WRITE: 0.7 ms]
Latency 95th percentile	:	2.4 ms [READ: 2.5 ms WRITE: 1.0 ms]
Latency 99th percentile	:	66.5 ms [READ 3.5 ms WRITE: 1.2 ms]
Latency 99.9th percentile	:	91.3 ms [READ: 128.2 WRITE: 22.9 ms]
Latency max	:	301.0 ms [READ 301.0 ms WRITE: 286.1 ms]

Some key considerations

- How many bytes can you push through the network.
- How much memory throughput can you get?
- How many instructions can you execute? Several different measures
- How much storage bandwidth can you get?

Characterisation for ARMv8

1. Scales well as writes go up to 50 million on a 3-node cluster with peak CPU and memory at 40-50% even with large number of client threads on the machines.
2. Performs favorably when compared to other bare-metal co-tenanted clusters for eg packet type-1 nodes - where peak write performance for 3-node cluster was 117k write operations/sec and memory usage was 90% while peak CPU usage was 88%.
3. Performs favorably to VM environments such as the recommended Amazon i3 nodes where peak performance is 126k write operations/sec for a 3-node multi-tenanted system.

Cassandra

- Massive number of deployments
- JVM performance
- What is optimizable in Cassandra for ARM based server
- What kind of servers should be used
- What settings?
- Cassandra is CPU bound before being memory bound
- Local storage vs Block storage
- I/O
- Kernel settings
- Network settings
- Configuration

Performance affected by several additional factors

1. Workload mix
2. Co-located clusters
3. Network design

Applications running in your ecosystem

- What other apps or frameworks or clusters are running?
- How you can carve up infrastructure resources?
- How to observe where you have spare capacity?
- Scheduling appropriately
- Maintaining performance

An example - where Cassandra was running in a containerized environment

1. CoreOS
2. Converged infrastructure
3. Mesos
4. Docker
5. Single flat 10Gig network
6. Co-tenant applications other than Cassandra present on machine - fairly common given that a lot of deployments use Spark co-located with Cassandra.

Performance numbers for Containerized environment

At a customer site ten node cluster gave a sustained throughput of 550,000 ops/sec in a mixed 80/20 read/write workload where each node has 40 cores

Finding:

Cassandra, when co-tenancy of multiple high network usage containers was blocked, was able to attain over 550,000 writes/second in a 10 node cluster using 55,000 writes/second per node is good and clearly meets or exceeds. CPU utilization never exceeded 25% of the available 400 cores. If the applications were not blocked peak throughput fell to 40% and 95% latency was 10 milliseconds while MAX latency was 500 milliseconds

Example Client to cluster on Remote Rack - Mixed load

Throughput
10,653 op/s [READ: 7,989 op/s, WRITE: 2,664 op/s]

Latency median : 80.4 ms [READ: 80.5 ms, WRITE: 80.4 ms]
Latency 95th percentile : 112.9 ms [READ: 114.0 ms, WRITE: 114.7 ms]

Latency 99.9th percentile : 148.6 ms [READ: 159.5 ms, WRITE: 153.8ms]
Latency max : 313.6 ms [READ: 313.6 ms, WRITE: 313.5ms]

Thank you and we would like to acknowledge

1. Support from Packet.net
2. Support from IBM GEP
3. Support from AWS through the activate program
4. Questions?

Contact info :

mksingh@mitylytics.com

www.mitylytics.com

Appendix

Contains - Pricing for AWS from AWS

Jvm profile - Branch prediction misses on ARMv8

```
 1.65%  java     libjvm.so           [.] ClassFileParser::parse_method
1.29%  java     libjvm.so           [.] CodeHeap::find_start
1.26%  java     libjvm.so           [.] Rewriter::rewrite_bytecodes
1.26%  java     libjvm.so           [.] SymbolTable::lookup_only
 1.18%  java     libjvm.so           [.] SymbolTable::basic_add
1.11%  java     libjvm.so           [.] Monitor::lock
1.11%  java     libpthread-2.23.so  [.] pthread_getspecific
1.08%  java     libjvm.so           [.] constantPoolHandle::remove
1.05%  java     libjvm.so           [.] ClassFileParser::parse_constant_pool_entries
1.05%  java     libjvm.so           [.] InterpreterRuntime::resolve_i
0.90%  java     [kernel.kallsyms]   [k] _raw_spin_unlock_irqrestore
0.88%  java     libjvm.so           [.] Rewriter::compute_index_map
```

a. 0.52% java [kernel.kallsyms] [k] __vma_link_list
b. 0.52% java libjvm.so [.] methodHandle::remove
c. 0.79% java libjvm.so [.] Symbol::operator new
d. 0.78% java libjvm.so [.] binary_search
e. 0.74% java libc-2.23.so [.] memset
f. 0.73% java libjvm.so [.] SpaceManager::allocate_work
g. 0.71% java libjvm.so [.] ConstMethod::allocate
h. 0.67% java libjvm.so [.] TypeArrayKlass::allocate_common
i. 0.66% java [kernel.kallsyms] [k] link_path_walk
j. 0.62% java libjvm.so [.] vmSymbols::find_sid
k. 0.62% java libjvm.so [.] Symbol::equals
l. 0.62% java libc-2.23.so [.] _IO_getc
m. 0.61% java libjvm.so [.] AdapterHandlerLibrary::get_adapter
n. 0.61% java [kernel.kallsyms] [k] do_group_exit
o. 0.60% java libc-2.23.so [.] strnlen
p. 0.60% java [kernel.kallsyms] [k] mprotect_fixup
q. 0.60% java libjvm.so [.] icache_flush
r. 0.59% java libjvm.so [.] Monitor::ILock
s. 0.56% java libjvm.so [.] InterpreterRuntime::resolve_get_put
t. 0.55% java libjvm.so [.] Metaspace::allocate
u. 0.54% java libjvm.so [.] klassVtable::compute_vtable_size_and_num_mirandas
v. 0.52% java [kernel.kallsyms] [k] __vma_link_list
w. 0.52% java libjvm.so [.] methodHandle::remove

1. 0.79% java libjvm.so [.] Symbol::operator new
2. 0.78% java libjvm.so [.] binary_search
3. 0.74% java libc-2.23.so [.] memset
4. 0.73% java libjvm.so [.] SpaceManager::allocate_work
5. 0.71% java libjvm.so [.] ConstMethod::allocate
6. 0.67% java libjvm.so [.] TypeArrayKlass::allocate_common
7. 0.66% java [kernel.kallsyms] [k] link_path_walk
8. 0.62% java libjvm.so [.] vmSymbols::find_sid
9. 0.62% java libjvm.so [.] Symbol::equals
10. 0.62% java libc-2.23.so [.] _IO_getc
11. 0.61% java libjvm.so [.] AdapterHandlerLibrary::get_adapter
12. 0.61% java [kernel.kallsyms] [k] do_group_exit
13. 0.60% java libc-2.23.so [.] strnlen
14. 0.60% java [kernel.kallsyms] [k] mprotect_fixup
15. 0.60% java libjvm.so [.] icache_flush
16. 0.59% java libjvm.so [.] Monitor::ILock
17. 0.56% java libjvm.so [.] InterpreterRuntime::resolve_get_put
18. 0.55% java libjvm.so [.] Metaspace::allocate
19. 0.54% java libjvm.so [.]klassVtable::compute_vtable_size_and_num_mirandas

AWS - GPU instance pricing - Very expensive

Name	GPUs	vCPUs	RAM (GiB)	Network Bandwidth	Price/Hour*	RI Price / Hour**
p2.xlarge	1	4	61	High	$0.900	$0.425
p2.8xlarge	8	32	488	10 Gbps	$7.200	$3.400
p2.16xlarge	16	64	732	20 Gbps	$14.400	$6.800

1-YEAR TERM

Dedicated host pricing for i3

Payment Option	Upfront	Monthly*	Effective Hourly**	Savings over On-Demand	On-Demand Hourly
No Upfront	$0	$3048.48	$4.176	24%	$5.491 per Hour
Partial Upfront	$15633	$1303.05	$3.570	35%	

Model	vCPU	Mem (GiB)	Networking Performance	Storage (TB)
i3.large	2	15.25	Up to 10 Gigabit	1 x 0.475 NVMe SSD
i3.xlarge	4	30.5	Up to 10 Gigabit	1 x 0.95 NVMe SSD
i3.2xlarge	8	61	Up to 10 Gigabit	1 x 1.9 NVMe SSD
i3.4xlarge	16	122	Up to 10 Gigabit	2 x 1.9 NVMe SSD
i3.8xlarge	32	244	10 Gigabit	4 x 1.9 NVMe SSD
i3.16xlarge	64	488	20 Gigabit	8 x 1.9 NVMe SSD

I3 4xlarge pricing

STANDARD 1-YEAR TERM					
Payment Option	Upfront	Monthly *	Effective Hourly**	Savings over On-Demand	On-Demand Hourly
No Upfront	$0	$692.77	$0.949	24%	$1.248 per Hour
Partial Upfront	$3553	$296.38	$0.812	35%	
All Upfront	$6964	$0	$0.795	36%	