SNIA DEVELOPER CONFERENCE

SDC 24

BY Developers FOR Developers

September 16-18, 2024
Santa Clara, CA

Scaling out IPUs to create a server cluster

Ian Wigmore, Intel

Abstract

Infrastructure Processing Units (IPUs) have traditionally been deployed as PCIe endpoints in servers and used to offload networking/storage/security duties from the CPU. An alternative deployment for IPUs is a stand-alone mode where the IPU acts as a server, presenting PCIe root-port interfaces to downstream NVMe SSDs either directly or via a PCIe switch. Along with the IPU’s high-speed networking capabilities and the ability to run a Linux OS on its embedded Arm cores, the IPU has all the necessary ingredients to be a highly capable low-cost, low-power, and compact Linux server. As with traditional servers, IPU servers may be clustered together to form an extensible platform allowing scale-out applications to run on it.

Apache Cassandra NoSQL database is one such application. Cassandra can scale to thousands of nodes making any per-server reduction in cost, power and size have a significant magnifying effect on datacenter sustainability. Another characteristic of Cassandra and which makes it advantageous to IPU server clusters is that it works better with many thin nodes, i.e., low storage capacity nodes rather than fewer fat ones. This in turn minimizes both blast radius and compaction/garbage-collection overhead. The low storage capacity requirement negates the need for an intervening PCIe switch between the IPU and the SSDs further reducing cost and complexity. Other scale-out applications such as ScyllaDB or Ceph block/object/file storage could also be considered as candidates to run on an IPU-based server cluster.

This presentation covers the development and path-finding work required to build and manage a multi-node IPU based cluster and details performance tuning techniques for lowering database tail latency whilst keeping throughput high. As AI/ML applications and datasets drive ever-increasing storage capacity in datacenters already hitting power capacity limits, clustered IPUs could provide a timely solution.

Starting with the end in mind…

Datacenters running AI and HPC workloads are hitting up against power and cooling limits

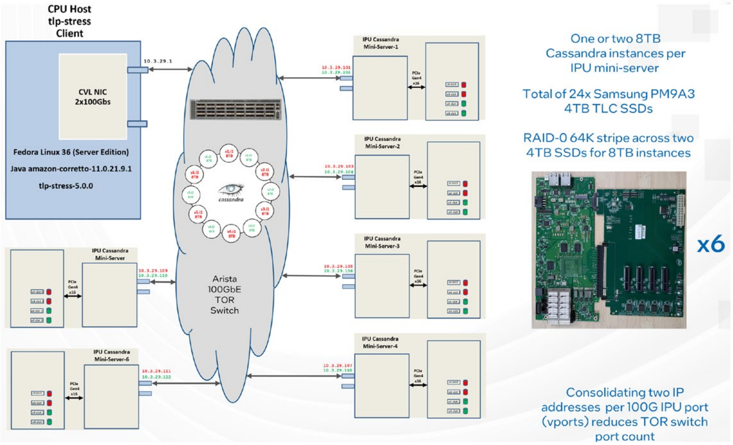
In response, we have built a cluster of inexpensive, low powered, compact servers to run distributed and scale-out applications

Is there value here?

If so, which distributed, and scale-out applications should we consider?

Agenda

- Intro to building an IPU-based Cassandra cluster
- IPU storage use cases
- IPU SoC E2100 and boards
- Software
- A summary of PoC development progress
    - Performance tuning
- What’s next?

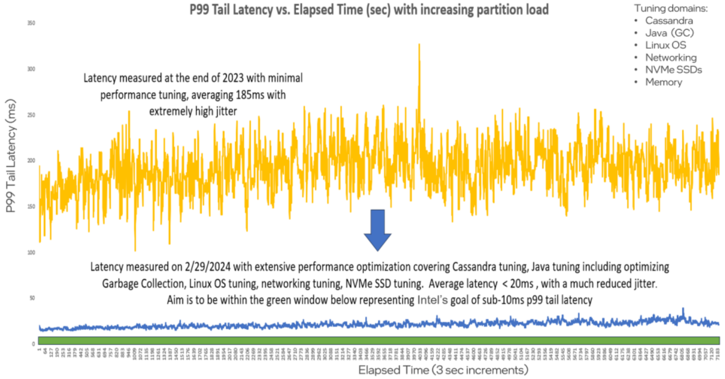


One or two 8TB Cassandra instances per IPU mini-server

Total of 24x Samsung PM9A3 4TB TLC SSDs

RAID-0 64K stripe across two 4TB SSDs for 8TB instances

Consolidating two IP addresses per 100G IPU port (vports) reduces TOR switch port count



Tuning domains:
- Cassandra
- Java (GC)
- Linux OS
- Networking
- NVMe SSDs
- Memory

Latency measured at the end of 2023 with minimal performance tuning, averaging 185ms with extremely high jitter

Latency measured on 2/29/2024 with extensive performance optimization covering Cassandra tuning, Java tuning including optimizing Garbage Collection, Linux OS tuning, networking tuning, NVMe SSD tuning. Average latency < 20ms , with a much reduced jitter. Aim is to be within the green window below representing Intel’s goal of sub-10ms p99 tail latency

If it smells of infrastructure, the IPU will handle it

- Traditional IPU use-case:
  - Infrastructure Processing Units (IPUs) deployed as PCIe endpoints in x86 hosts for offloading infrastructure work such as networking/storage/security from the CPU

- Alternative IPU use-case:
  - A stand-alone mini-server, presenting PCIe root-port interfaces to downstream NVMe SSDs either directly or via a PCIe switch

IPU benefits

- With the IPU’s high-speed networking capabilities and ability to run a Linux OS on its embedded Arm cores, the IPU can become a highly capable low-cost, low-power Linux mini-server

- And as with traditional CPU-based servers, IPU mini-servers may be clustered together to form an extensible platform allowing scale-out applications to run on it

Cassandra

- One such scale-out application is Apache Cassandra NoSQL database
- It can scale-out to thousands of nodes making any per-server reduction in cost, power and size improve datacenter sustainability
- The PoC features six IPU HW nodes each running two Cassandra instances

Cassandra usage

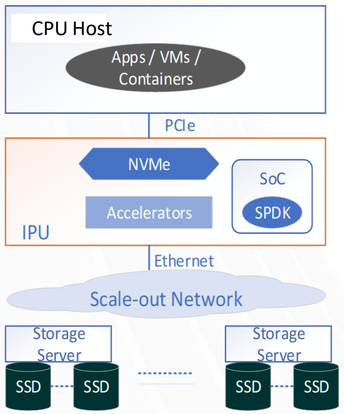
- Some 6500+ companies use Cassandra in production in areas such as:
  - Social media back-end services
  - Time series data storage
  - Distributed databases, etc.

- 40% of Fortune 100 use Cassandra and is ranked in the top 10 of all databases used

IPU storage use cases

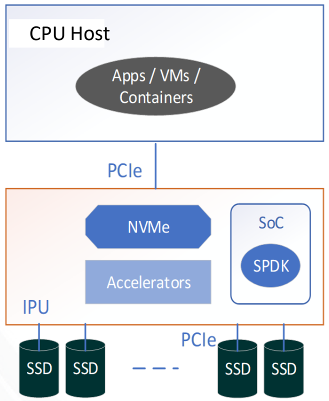
IPU PCIe End Points | **IPU PCIe Root Ports (Headless)**

Initiator/Client side of the wire | **Target/Server side of the wire**



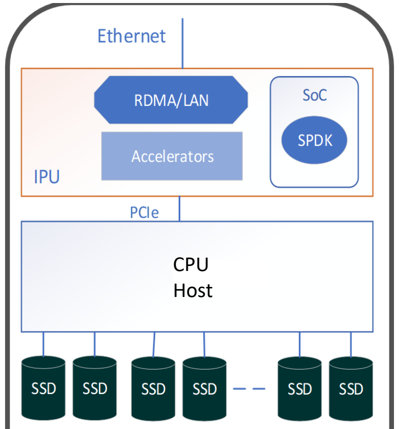
NVMe-oF initiator

- Storage disaggregation at cloud scale
- Bare-metal hosting
- NVMe/TCP



Local storage disaggregation

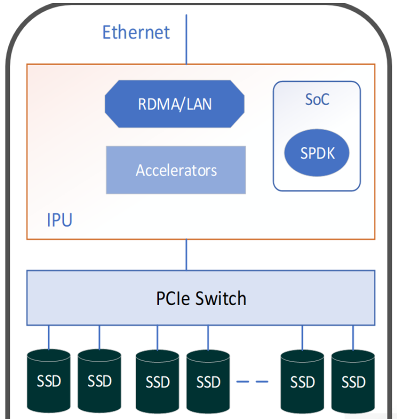
- Nitro SSD like design
- IPU provides NVMe virtualization layer
- Inline crypto and DIF/CRC



IPU+CPU storage server

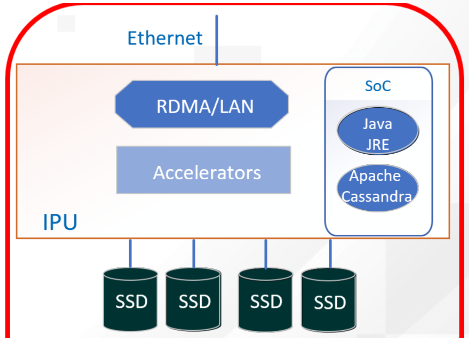
- IPU for storage data path acceleration

*Note: IPU also supports HCI usage. Both initiator and target flows can run concurrently.



IPU-based JBOF with PCIe switch

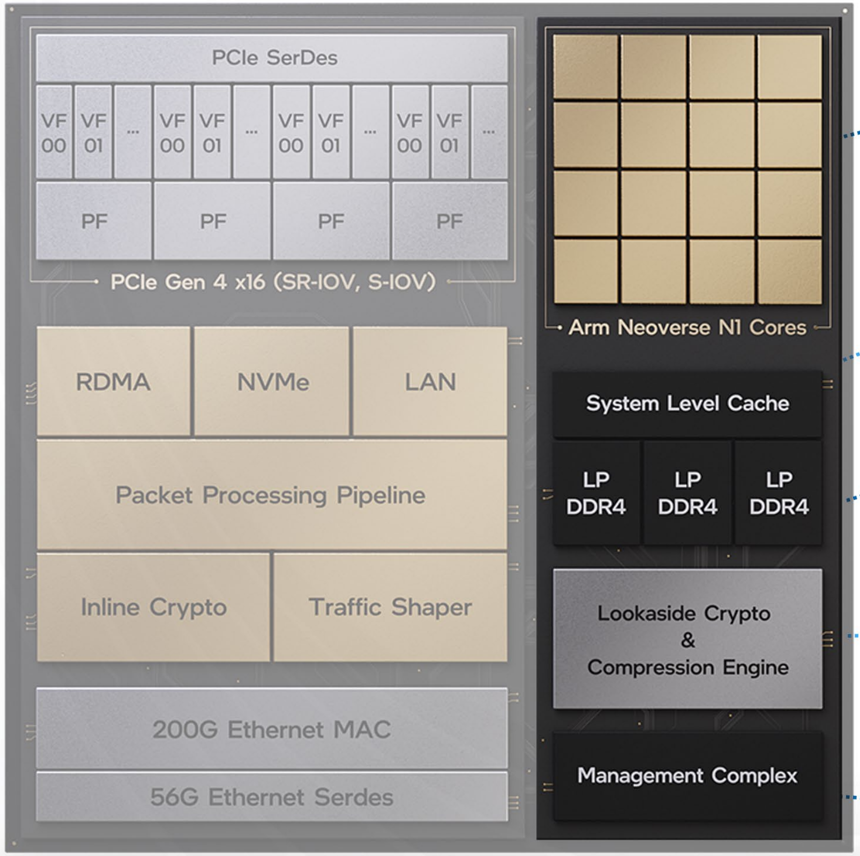
- Storage data path acceleration
- High SSD fanout



IPU server or JBOF with direct-connect SSDs

- Low cost, low power, high bandwidth, low latency
- Compact Arm-based Linux server
- Networked together to form large, scale-out clusters

Intel IPU SoC E2100 – compute complex



Compute Complex Highlights

16 ARM N1 cores @ up to 2.5GHz. Common tasks might be vSwitch exception processing, Storage backend, TLS proxy and VMM offloads.

32MB System Level Cache for greater IO processing, Storage bounce buffers, NSS cache extensions and general purpose compute

3 LPDDR4 controllers for up to peak BW 100 GB/s

Lookaside crypto accelerates workloads (e.g.TLS) for the host or Compute Complex. Advanced ZStandard compression for best-in-class storage and database use

Dedicated management processor and subsystem secure boot, maintenance, and upgradeability

Software roles & characteristics

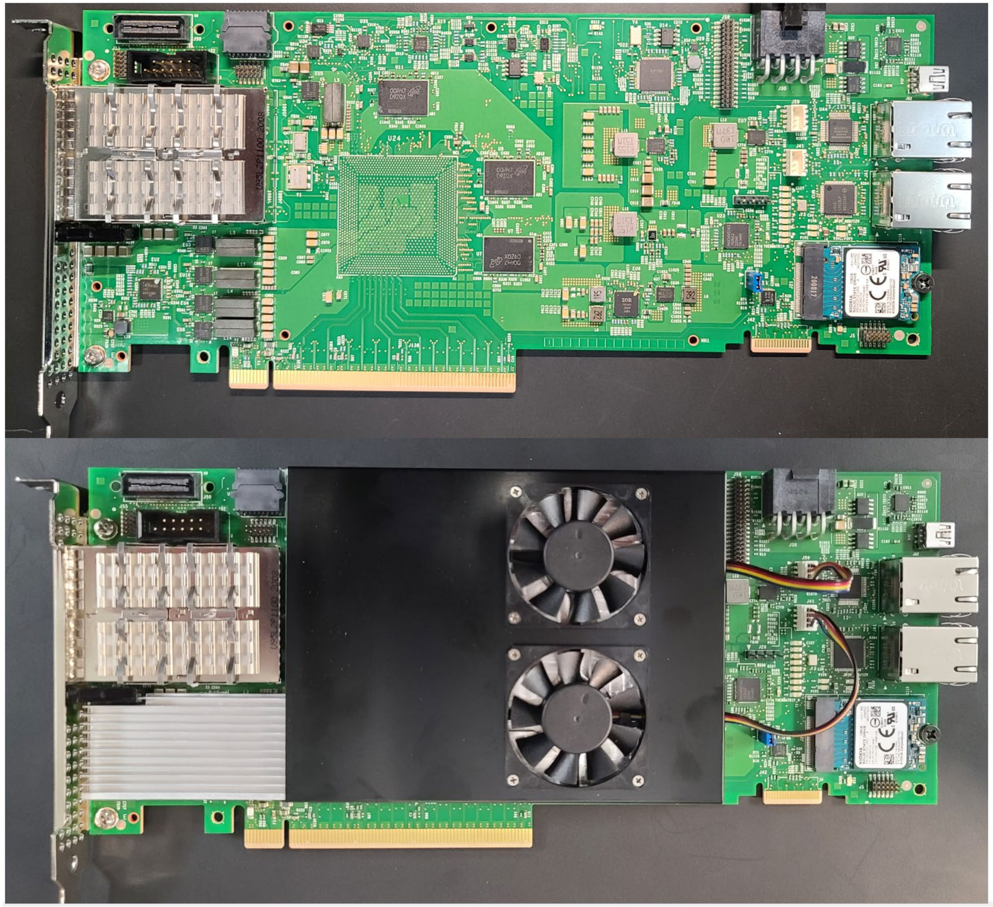
Feature	Integrated Management Complex	ARM Compute Complex
OS	Rocky Linux / Customer Choice	Customer Choice
Role	• Device boot and reset handling • Device level security and lifecycle • System resource partitioning • Manageability protocols & integration with DC management frameworks	• Customer networking dataplane & dataplane offload orchestration • Customer storage offload & storage orchestration • Customer debug, telemetry, provisioning and other customer-specific lifecycle operations • Many more use-cases…

- **IMC**
  - Stored on SPI flash
    - Preboot components
      - Early-stage boot loaders
      - ARM trusted firmware
      - uBoot
    - Recovery image (full OS stack + tools)
    - Static configuration elements
      - Autoload blobs (boot-time device configuration)
      - Mutable content (board specific content, customer mutable KV content, etc.)
  - Stored on M.2 SSD
    - IMC OS image + app stack (supports active/backup model)
- **ACC**
  - Stored on M.2 SSD
    - Preboot components
      - ARM trusted firmware
      - UEFI
    - ACC OS image + app stack

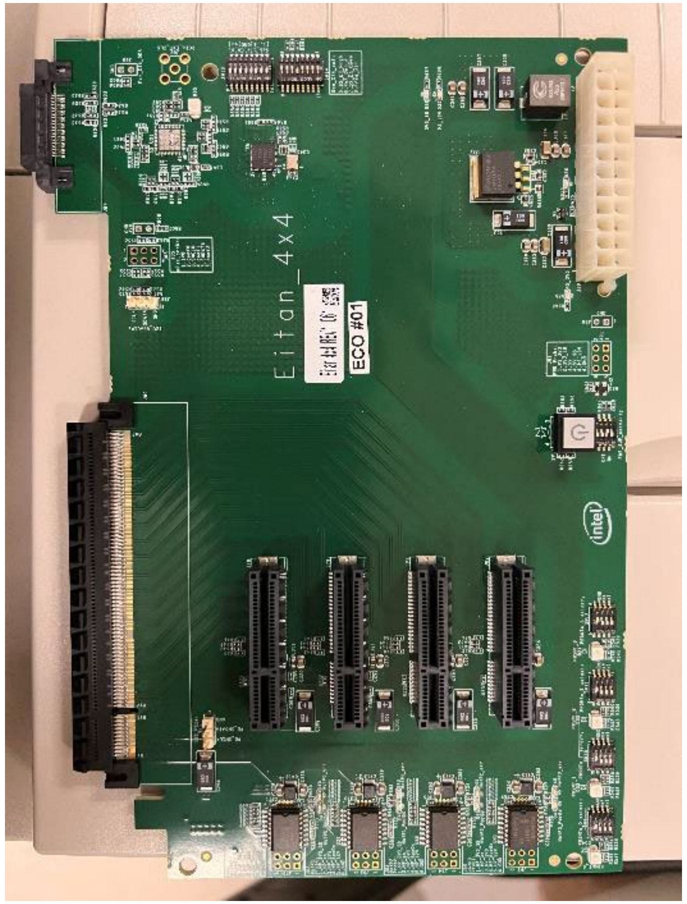
IPU Network Reference Board (NRB) and Root Port (RP4x4) adapter

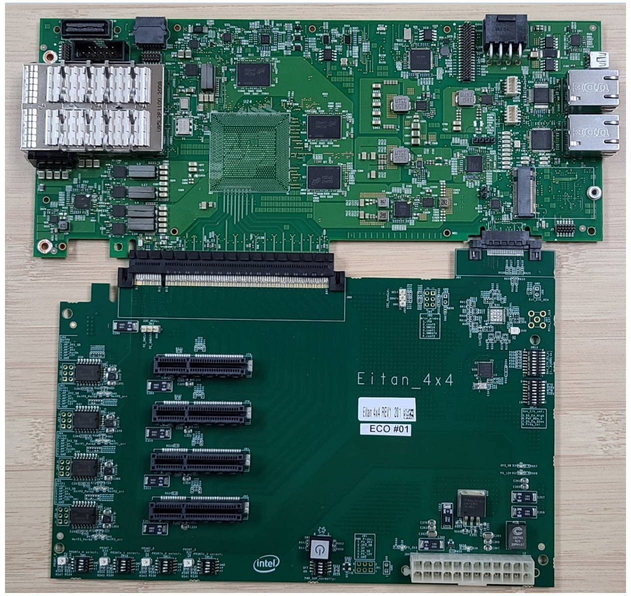
PCIe CEM card, full height, ¾ length (9.8 in x 4.37 in)

Dual QSFP28 LPDDR4x NRB, PCIe x16 Gen4

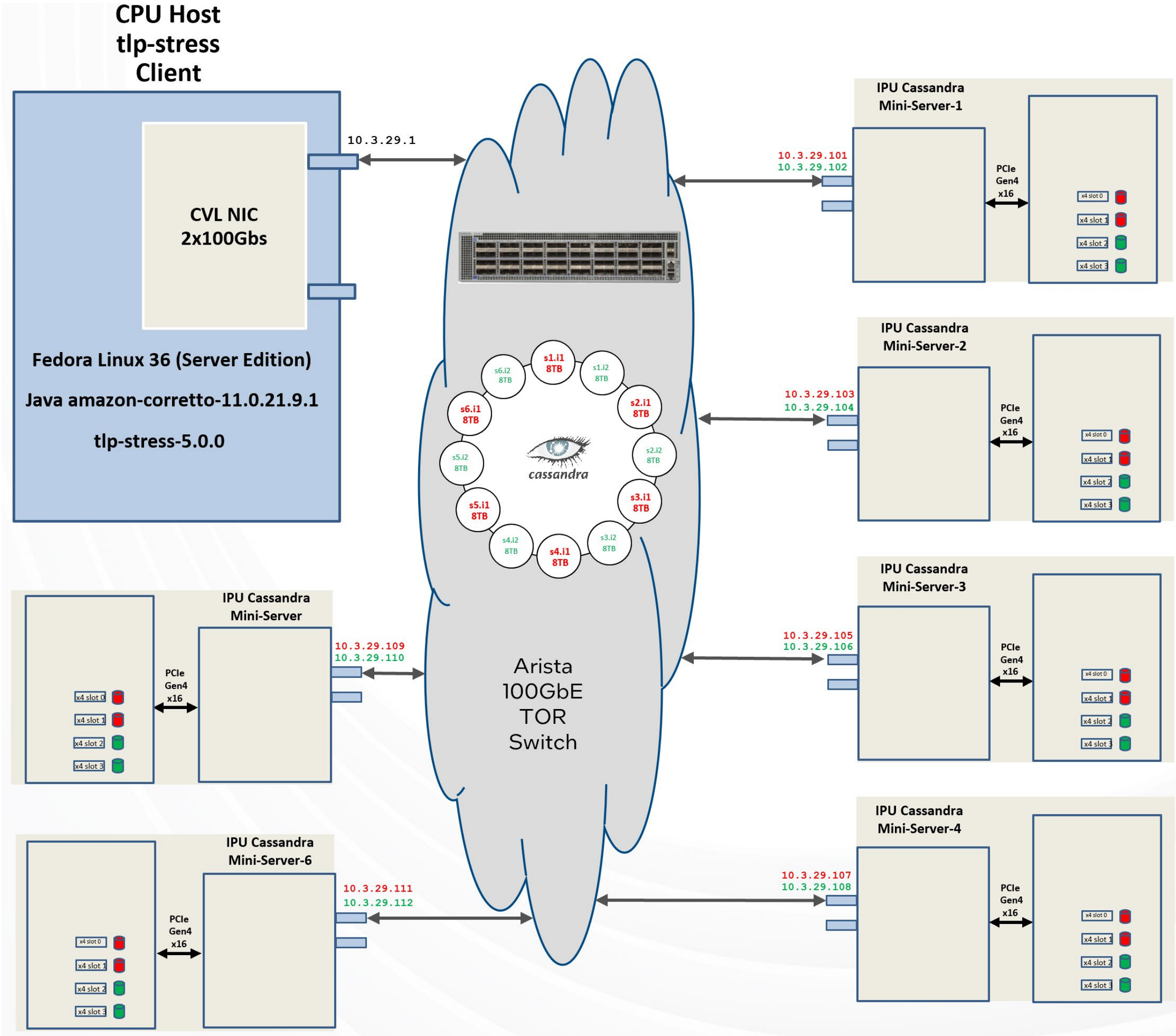


RP4x4 Adapter – 4 x PCIe Gen4 4-lane slots
for 4 x 4TB NVMe SSDs





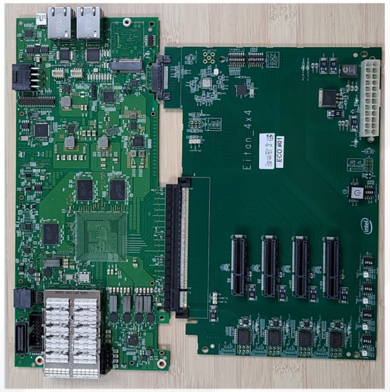
PoC: IPU mini-server cluster with 12 Cassandra instances



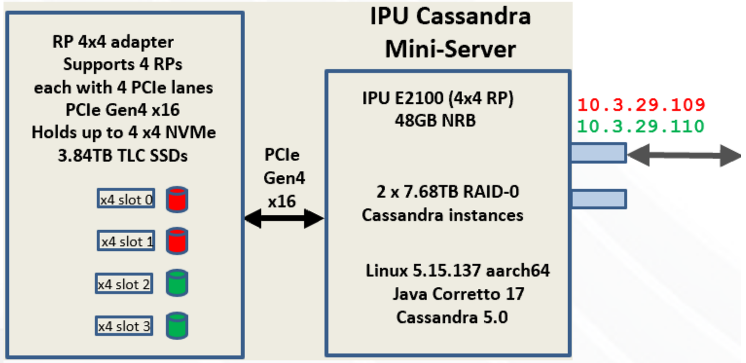
One or two 8TB Cassandra instances per IPU mini-server

Total of 24x Samsung PM9A3 4TB TLC SSDs

RAID-0 64K stripe across two 4TB SSDs for 8TB instances



x6



Consolidating two IP addresses per 100G IPU port (vports) reduces TOR switch port count

The Cassandra cluster lives – it's a real thing!

IMC control-path configuration

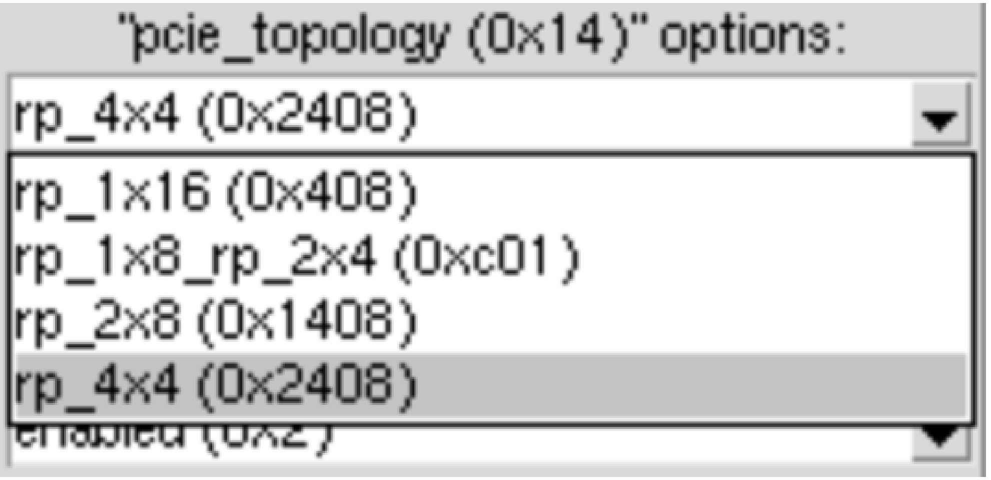
- cp_init.cfg

  vports = 2

  uplink_vports = ([5,1,0],[5,2,0]) - [<ACC>,<vport>,<physical 100G port>]

  pf_mac_address = "00:00:00:00:03:14“ - unique for each Cassandra IPU node

- ANVM tool GUI used to set the required Root Port mode (rp_4x4)



"pcie_topology (0x14)" options:

rp_4x4 (0x2408)

rp_1x16 (0x408)
rp_1x8_rp_2x4 (0xc01)
rp_2x8 (0x1408)
rp_4x4 (0x2408)
enabled (0x2)

IMC = Integrated Management Complex

ACC = Arm Compute Complex

ANVM = Adaptive Non-Volatile Memory

Prepare SSDs for one Cassandra instance

- Prepare two SSDs for a Cassandra instance using /dev/nvme0n1 and /dev/nvme1n1
- RAID-0 the two SSDs together using mdadm with 64KB stripes -> /dev/md0

```
./mdadm --create --verbose /dev/md0 --level=0 --raid-devices=2 /dev/nvme1n1 /dev/nvme0n1 -c 64K
```

- fdisk format and create partitions on /dev/md0
- Write ext4 file system on /dev/md0 (xfs is also valid)
- Create mountpoint /mnt/nvme0 from /dev/md0
- Use the mountpoint for Cassandra configuration …

IPU ACC access to outside world

- To behave as a bonafide Linux server for setting up required tools and utilities, the IPU's ACC is required to be able to pull dnf packages, run git clone, etc…
- This is achieved by setting up a port forwarding mechanism to ssh tunnel via the IMC and the x86 client which already has external network access
- On IMC:

```
ssh -fgCNL 192.168.0.1:911:proxy-us.intel.com:911 root@100.0.0.1
```

- Proxies configured on the ACC
- On ACC:

`dnf install chrony -y` (to time sync each IPU node to x86 client…)

Time sync each IPU node to x86 client

- Network Time Protocol (NTP) required to time synchronize Cassandra nodes to each other
- The x86 server will act as the NTP server and the IPU’s ACCs are the NTP clients
- chronyd is a daemon that synchronizes a NTP server’s clock with the NTP clients
- As Cassandra's log-based storage is timestamp-driven, it is critical that all nodes in the cluster have synchronized clocks
- Cassandra uses a concept called 'Last Write Wins' to resolve conflicts and determine which mutation represents the most correct up-to date state of data

Mutation is just another word for a database write - includes INSERT, UPDATE and DELETE statements from the application. The word "mutation" is used because these statements change the data in the database.

Cassandra and Java configuration

cassandra.yaml

```
hints_directory: /mnt/nvme0/cassandra/hints
data_file_directories:
    - /mnt/nvme0/cassandra/data
commitlog_directory:
/mnt/nvme0/cassandra/commitlog
cdc_raw_directory: /mnt/nvme0/cassandra/cdc_raw
saved_caches_directory:
/mnt/nvme0/cassandra/saved_caches
          - seeds: 10.3.29.101:7000
concurrent_reads: 32
concurrent_writes: 64
listen_address: 10.3.29.101
rpc_address: 10.3.29.101
```

1/cassandra-env.sh

```
JMX_PORT="7199"
```

2/cassandra-env.sh

```
# Required for the 2nd C* instance on IPU HW node
JMX_PORT="7299"
```

jvm-server.options

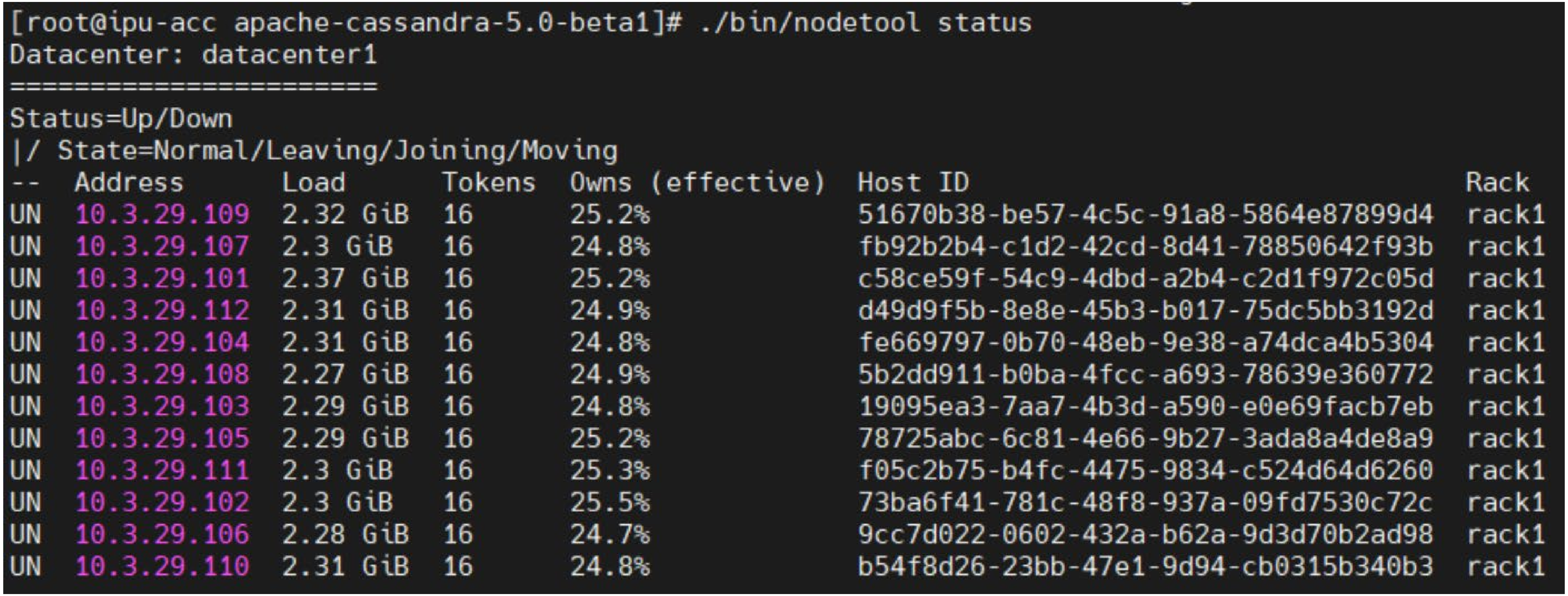
```
# Increased to Xms16G with one C* instance per IPU HW node
-Xms8G
-Xmx8G
```

jvm17-server.options

```
### ZGC Settings
-XX:+UseZGC
-XX:ConcGCThreads=4
-XX:ParallelGCThreads=10
-XX:+UseTransparentHugePages
-XX:SurvivorRatio=6
-XX:MaxTenuringThreshold=4
-XX:+AlwaysPreTouch

#-XX:-UseNUMA
#-XX:+UnlockExperimentalVMOptions
#-XX:+UseLargePages
#-XX:+ZGenerational
```

Cassandra node status from IPU ACC showing 12 instances online



```
[root@ipu-acc apache-cassandra-5.0-beta1]# ./bin/nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load      Tokens  Owns (effective)  Host ID                               Rack
UN  10.3.29.109  2.32 GiB  16      25.2%             51670b38-be57-4c5c-91a8-5864e87899d4  rack1
UN  10.3.29.107  2.3 GiB   16      24.8%             fb92b2b4-c1d2-42cd-8d41-78850642f93b  rack1
UN  10.3.29.101  2.37 GiB  16      25.2%             c58ce59f-54c9-4dbd-a2b4-c2d1f972c05d  rack1
UN  10.3.29.112  2.31 GiB  16      24.9%             d49d9f5b-8e8e-45b3-b017-75dc5bb3192d  rack1
UN  10.3.29.104  2.31 GiB  16      24.8%             fe669797-0b70-48eb-9e38-a74dca4b5304  rack1
UN  10.3.29.108  2.27 GiB  16      24.9%             5b2dd911-b0ba-4fcc-a693-78639e360772  rack1
UN  10.3.29.103  2.29 GiB  16      24.8%             19095ea3-7aa7-4b3d-a590-e0e69facb7eb  rack1
UN  10.3.29.105  2.29 GiB  16      25.2%             78725abc-6c81-4e66-9b27-3ada8a4de8a9  rack1
UN  10.3.29.111  2.3 GiB   16      25.3%             f05c2b75-b4fc-4475-9834-c524d64d6260  rack1
UN  10.3.29.102  2.3 GiB   16      25.5%             73ba6f41-781c-48f8-937a-09fd7530c72c  rack1
UN  10.3.29.106  2.28 GiB  16      24.7%             9cc7d022-0602-432a-b62a-9d3d70b2ad98  rack1
UN  10.3.29.110  2.31 GiB  16      24.8%             b54f8d26-23bb-47e1-9d94-cb0315b340b3  rack1
```

- Three-way replication
- Data capacity nicely distributed over the 12 instances
- All 12 instances are online: UN=Up/Normal
- Two instance IP addresses per single IPU 100GbE port (2 vports)

IPU active power consumption

- During tlp-stress loading on the IPU-based Cassandra cluster
- Via scripts running on the IMC, power values from each voltage rail within the IPU were extracted:
- IPU NRB power: ~40W
- SSD power: ~8W per NVMe TLC SSD (~32W total)
- IPU NRB + 4 x SSDs: 72W total power to support two Cassandra instances

- Traditional x86 server : 1200-1500W – supporting four Cassandra instances

- Per Cassandra instance: IPU 36W versus x86 server 300-375W

Performance tuning

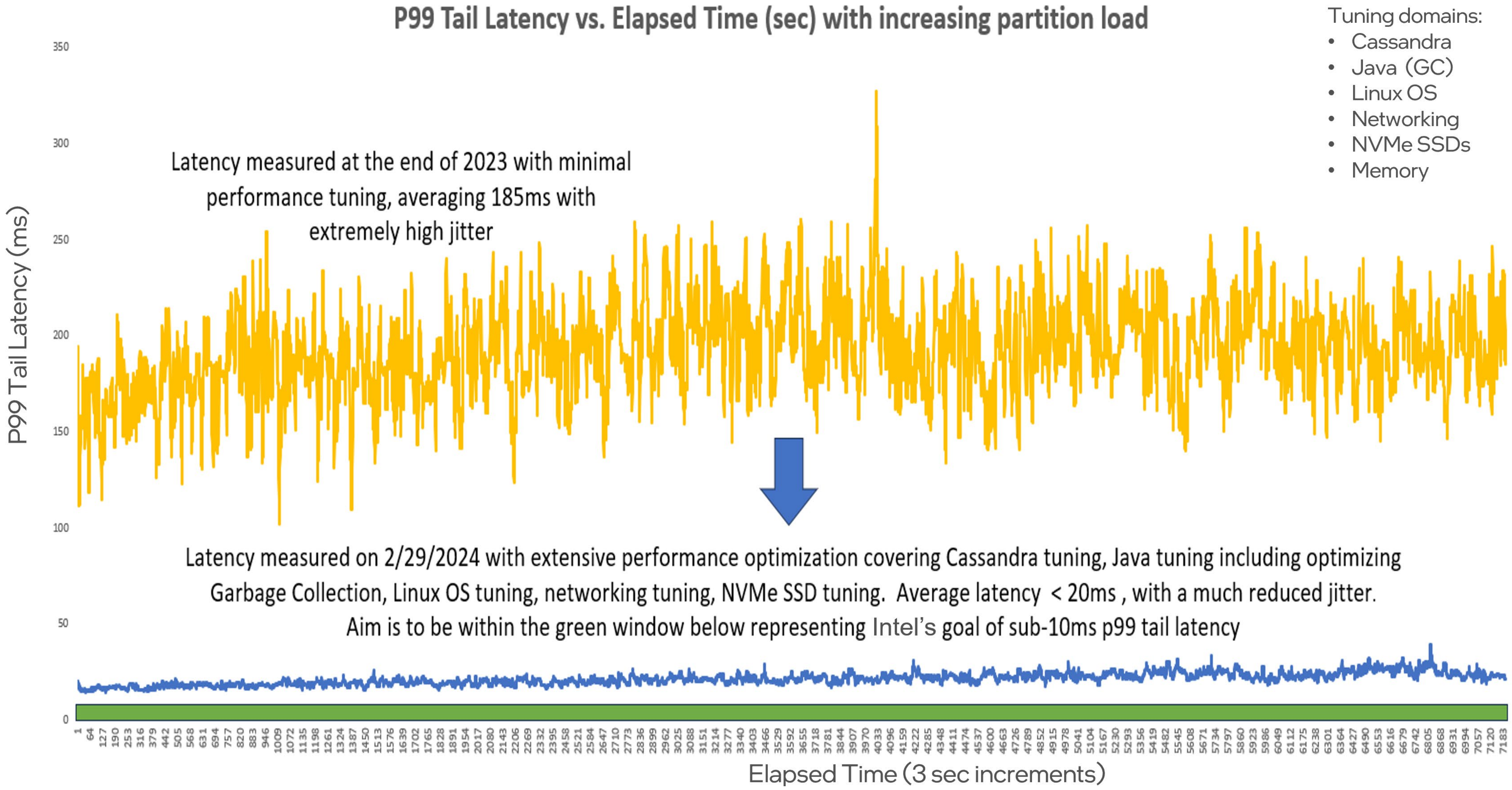
- The current IPU/Cassandra nodes are compute and memory bound rather than network and SSD bound
  - The nextgen IPU will be less so with more compute, more memory and greater memory bandwidth

- Processor cores:
  - One C* instance: Arm core utilization is 80-90%, with two instance it increases to 90-95%. Ideally want to see 75-80% to provide headroom for compaction and GC
  - Cassandra is highly threaded, so it can take advantage of the ACC's 16 N1 Arm cores

- Memory:
  - Increased ACC memory from 29GB to 37GB by re-directing DRAM allocation away from the FXP (flexible packet processor)
    - Increases Cassandra's memtable and row/counter/key cache size

- Cassandra/Java:
  - Seeing best performance so far - in terms of both latency and throughput - with Cassandra 5.0-beta, OpenJDK17, ZGC garbage collection, a 16GB heap (from total of 37GB on ACC) and with a single tlp-stress workload

Results of months of performance tuning



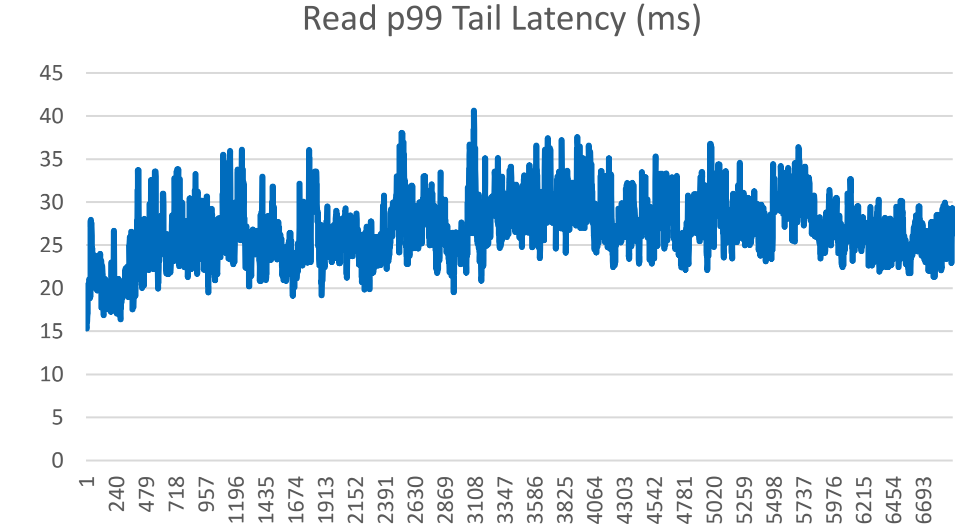
Latency measured at the end of 2023 with minimal performance tuning, averaging 185ms with extremely high jitter

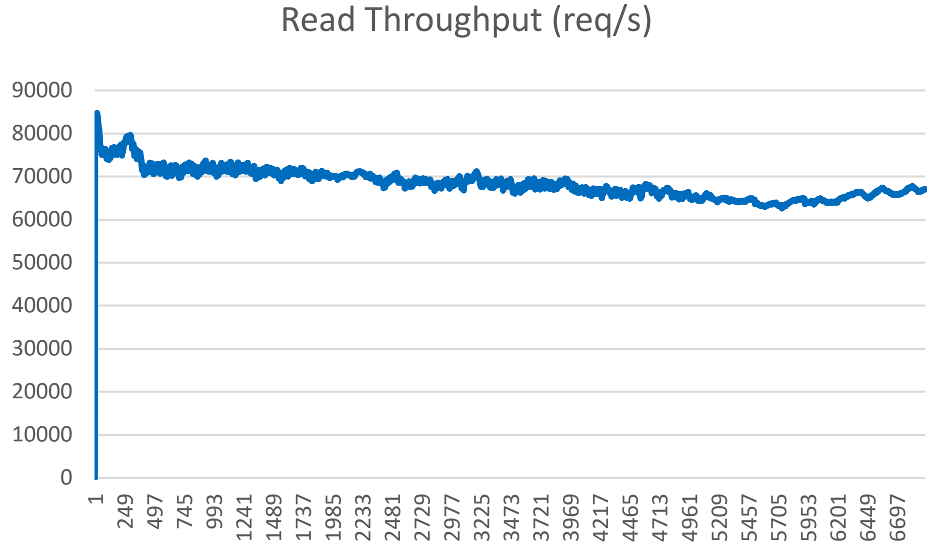
Latency measured on 2/29/2024 with extensive performance optimization covering Cassandra tuning, Java tuning including optimizing Garbage Collection, Linux OS tuning, networking tuning, NVMe SSD tuning. Average latency < 20ms , with a much reduced jitter. Aim is to be within the green window below representing Intel's goal of sub-10ms p99 tail latency

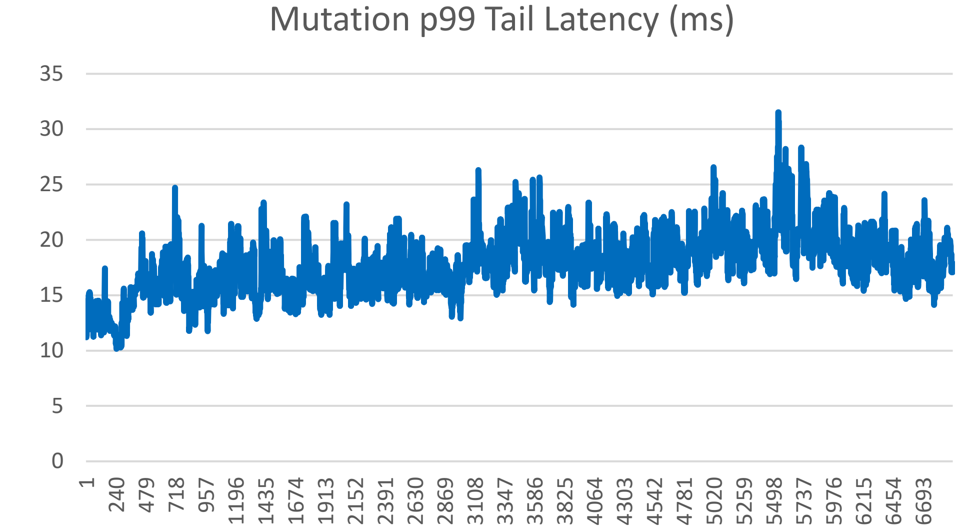
Tuning domains:

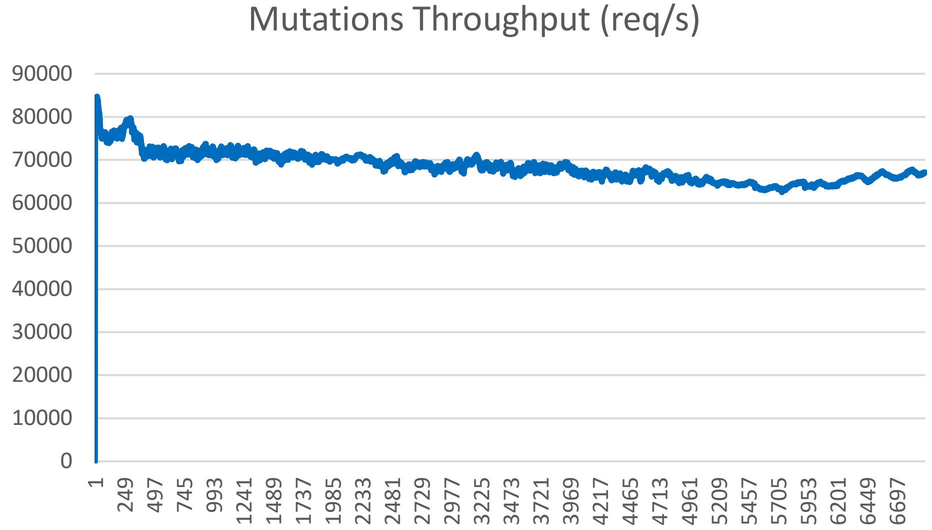
- Cassandra
- Java (GC)
- Linux OS
- Networking
- NVMe SSDs
- Memory

tip-stress KV: p99 tail latency and throughput (req/s) and results

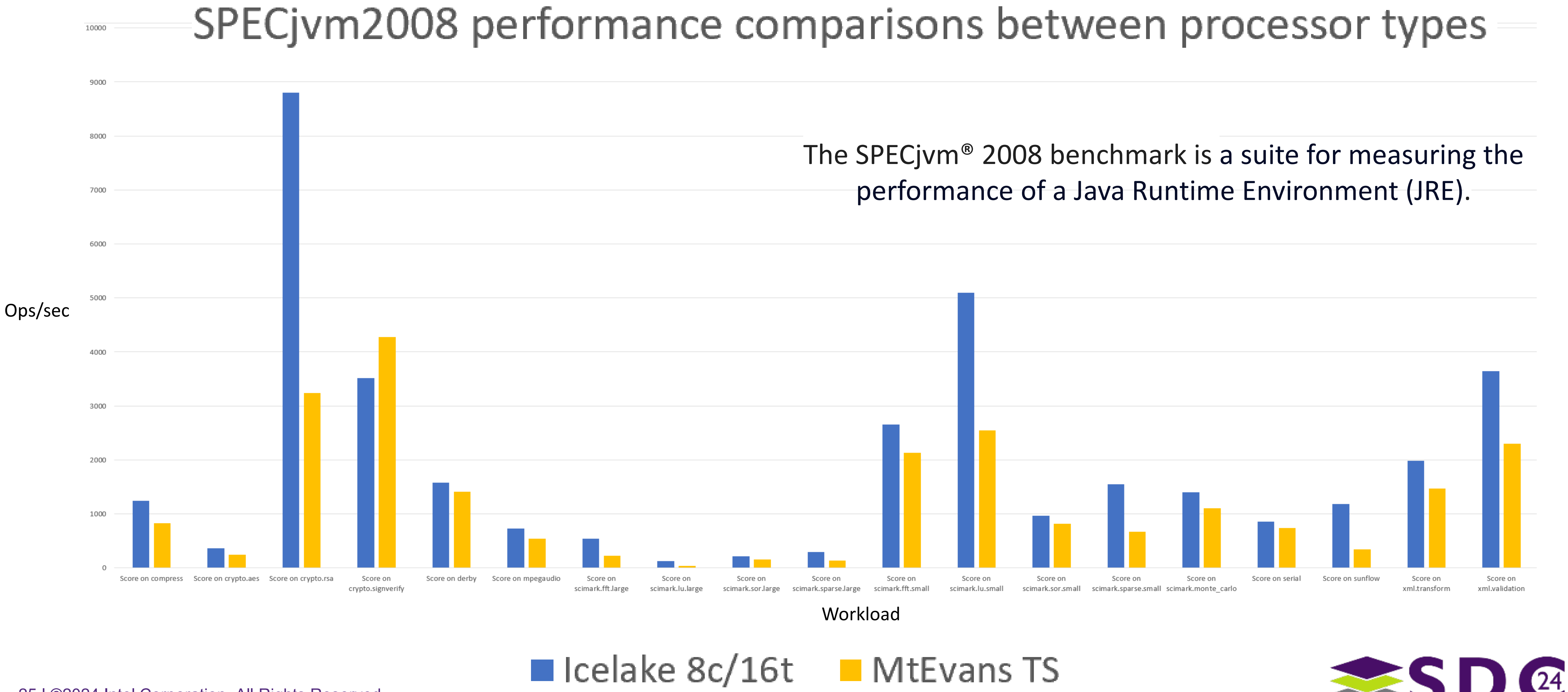

Read p99 Tail Latency (ms)


Read Throughput (req/s)


Mutation p99 Tail Latency (ms)


Mutations Throughput (req/s)

SPECjvm2008 Ice Lake 8c/16t versus IPU



The SPECjvm® 2008 benchmark is a suite for measuring the performance of a Java Runtime Environment (JRE).

What’s next?

- Transition from the current 200G IPU to the next-gen Mount Morgan IPU 400G SoC
  - Increase Cassandra instances from two to four

- RDMA support
  - Increase overall performance by using a HW based transport to replace SW based TCP stack

- Add other scale-out applications such as ScyllaDB or Ceph block/object/file storage

Ending with the start in mind…

Is there value here supporting a low-cost, low-power compact server?

And, if so, which distributed, and scale-out applications should we consider?

Please take a moment to rate this session.

Your feedback is important to us.