DISTRIBUTED KEY-VALUE STORE FOR PETASCALE HOT STORAGE IN DATA ACQUISITION SYSTEMS

Presenter: Grzegorz Jereczek
on behalf of the FogKV team

CHEP 2018, Sofia, Bulgaria
Legal Notices & Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

© 2018 Intel Corporation

Intel, the Intel logo, Intel Xeon, Intel, Optane are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.
LHC experiments will be producing hundreds of petabytes a day
Intel-CERN collaboration targeting Trigger and Data Acquisition (TDAQ) upgrades

Development of a **storage system** for decoupling real-time data acquisition from asynchronous event selection

- Distributed over $O(100)$ nodes
- Large, temporary storage of $O(100)$ PB
- Total throughput of $O(10)$ TB/s with $O(100000)$ clients.
Maximizing data taking efficiency by decoupling real-time DAQ from event selection

Temporary storage
- Make use of the inter-fill/no-beam time for data selection
- Reduce dead-time when adapting trigger menu
- Use latest calibration
- Exploit commonalities across different DAQ architectures

Data acquisition

Data selection
Data AcQuisition (DAQ) with Key-Value Store (KVS)

Large key-value store

- Insert data from each fragment keyed with (run_id, event_id, subdetector_id)
- Potentially stored for several hours/days with replication for fault tolerance
- Distributed storage may be local to the readout system or remote

Data selection

- Query data when needed
- Cooprocessors for subsets of data
- Event assembly after acceptance
FogKV – a KVS for DAQ

- First-line buffer for fast pre-computing and second-line buffer for longer term
- Data structure built on partially pre-allocated Adaptive Radix Trie
- Optimized data locality

FogKV nodes ~500

- Key/Value
- Persistent Memory
- Replication
- Request router

- NVMe
- O(10) TB
- O(100) TB
- O(100) Gbps
- O(100) Gbps

Compute farm ~2k nodes

Offline storage

6TB/s
Enabling emerging technologies

**Memory & Storage**
- [Intel Optane DC Persistent Memory](https://www.intel.com/content/www/us/en/optane-dc.htm)

**Compute**
- [Intel Xeon Platinum Inside](https://www.intel.com/content/www/us/en/xeon-platinum.html)

**Connectivity**
- Ethernet (RDMA)

**Software**
- **Persistent Memory Development Kit (PMDK)**
  - Optimal performance of persistent memory
  - [http://pmem.io](http://pmem.io)
- **Storage Performance Development Kit (SPDK)**
  - User-mode access to NVMe devices (SSDs)
  - [http://spdk.io](http://spdk.io)
- **libfabric**
  - High-performance and scalable networking
  - [https://github.com/ofiwg/libfabric](https://github.com/ofiwg/libfabric)
## DAQ-specific API

<table>
<thead>
<tr>
<th>Feature</th>
<th>Code</th>
</tr>
</thead>
</table>
| User-defined key structure                   | struct MinidaqKey {
|                                              |  uint64_t eventId; uint16_t subdetectorId; uint16_t runId; }        |
| Range queries with compound keys             | kvPairVector = kvs->GetRange(keyMin, keyMax)                         |
| Asynchronous mode for even higher performance| kvs->GetRangeAsync(keyMin, keyMax, cb)                               |
| Distributed locking for next event retrieval | eventKey = kvs->GetAny(options=(lock))                               |
| FogKV memory allocator minimizing copy operations | value = kvs->Alloc(key, 10 * 1024)                                  |
Preliminary performance on emulated first-line buffer

- Linear scaling thanks to lockless design reaching around 200 Gbps
- Current bottleneck is memory allocation, especially near capacity limits
- Will change with second-line buffer
- Similar performance for GET

Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz, persisten memory emulated with DDR4 2666MHz
Conclusions

Asynchronous data selection to extract the best physics potential

An optimal balance between storage and fast data rejection to optimize overall system costs

FogKV
Multi-TB/s hot storage solution of a petascale capacity for DAQ
Outlook

Public repo @github soon
Distributed mode
Pilot Q1’19

FogKV
Multi-TB/s hot storage solution of a petascale capacity for DAQ