DARSHAN

Shane is sharing his slides from the plenary session.

Shane: HDF5 can be used now but the DAOS may be sort of long term.
DAOS already has proto type model. Right now working on object API.
Probably the API is not much work but capturing the DAOS object takes some effort.
DARSHAN captures one record per object, for DAOS it could be thousands per object and maybe cannot capture the granularity.

No good answer since Shane is not sure about how things are written on top of libDAOS.

Philippe: Reconnect Javier with Shane for DAOS regarding Rntuple.
Shane: Shane is already working on the HDF5 Outputer for DARSHAN.

Patrick: TFileBufferWrite implemented in the DARSHAN for the ROOT (refer the plot he sent around earlier)

When writing to Buffer Burst (PFS–Parallel File System), using BufferWrite is a better thing. Doing instantaneous writing.

Philipe: Improvement makes sense since it is a remote file system.

Patrick: This is for Default ROOT file format and writing out AOD. What is it reading??? Based on what reading/writing, the event size might be relevant for information.
Factor of two speed up using the BufferWrite.
Compression is LZMA4

What's the mechanism to change cache size etc?
-->There are configuration parameters

8 threads (Running) using the ROOT IMT
Chris: Factor of 3 factors compared to single threaded if using the same ROOT File maybe.

Caching works better for remote file systems but not for the local file system maybe.
Maybe we can test with the CMS system.

CMS can be in good shape to test with the DARSHAN framework. ATLAS has lost a bit of effort for now and we don't know where DUNE is yet.

Chris: Take the CMS file to take the actual workflow and get the DARSHAN log.
Patrick: Cori has the default version of DARSHAN which is the MPI version.

Shane: Need to find the data of reasonable size to look at the log from Darshan without relying too much on the collaborators.

Chris: DARSHAN can be used to test the multi-threading within each node.

Shane/Patric:Probably, tests can be done in the CORI.

Keep focusing on the ROOT stuff but maybe need to do tests with the HDF5 but experiments don't have implemented the HDF5 yet or no decision on data format etc to test with experiment data.

Next test would be to test with different ways of ROOT outputers.

1. ROOT-Serialization with Parallel (Users do compression and ask ROOT to take the blob and store).
2. HDF5 also does the same thing (after serialization).

   Both show good scaling.

1. Maybe good to have DARSHAN log books for parallel HDF5.
2. What about for the Collective IO?
   a. DARSHAN can be easily implemented for the collective IO once it is in the ROOT-Serialization Framework.

Long term plan:

Philippe:ROOT/HDF5 choices to be made by experiments. And need to have parameters that can be tuned and could be experiment/hardware specific.
For example: For X framework/machine, here are the parameters that can be tuned in such and such order to get the optimal result.

Peter: Need to feed back the reading information to the writer.

Peter: Learn during reading the files and and take lessons learnt from those feed back the information for the next cycle of production.

Kyle: How much of this feedback implementation is library specific?

Peter: It is a layer on top of the framework. ATLAS has comprehensive monitoring on the top of the workforce. There is information that we can learn from a current data campaign and use that in the future campaign.

Paolo: Issue for DUNE: IO performance in terms of writing of as little data as possible.

Compression is well known and probably not much to do
What else can be done/invented to use less storage.

Peter: That goes away from that has been done but there are some things to be tried. Duplication of data (currently we write the same data multiple times in ATLAS for example) can be minimized. Write something flexible in the central place and that can be extracted in multiple places, maybe one way to reduce the stored data-size.

Why duplication?
Because events use the access time. Also needs the optimization of the distribution model since duplication is done to decrease the access time.

Chris: Before the LHC came, the original POOL had the idea of using a central flexible system that could be extracted. But it sort of imploded.

Why imploded?
Because it couldn't transport data well, lost pieces of information etc and went to the model where the data is redundant (i.e. you get a file, you get everything).

But it was 20 years ago so maybe things are different.
Failure rate in data access still exists even though it is much lower.

Derived data (few derived data but make the system know where rest of the info is)

Paolo: We already do that.

Paolo: What we are talking about, NETflix etc might be doing much better.

Philippe: But they don't care if a frame is lost or not. That is not our case.

Salman:
Lossy compression.
Whatever validation is done, depends on analysis used for validation. Need to validate and in ideal situations, don't want to throw away things in case it affects an analysis in the future.

But probably it is a direction going forward but we need to know how much info is compromised.

Maybe a recoverable loss. Save space by not duplicating the lost information but storing it in some central system.

Chris: Saving lost information context. Lots of space is taken by things that are never used. In the end, experiments need to know what they need. Should remove the unused files. But maybe even go further in looking at the part of the data that is not used and maybe can use that information for the future campaign (or offline storing).

Salman:
A part of it is psychological because people don't know if the hardly used data might be used at some point.

Philipe: How do they detect which data members are never used? Makes sense for files or branches but not sure how it is done for the data members.

Chris: Maybe based on how fast/often people complain when a part of the code is broken. Certainly there is no automated way to find the usage frequency of the data members.

Peter/Paolo: DUNE challenge is a unique challenge and also important.

Salman: It is something everyone is thinking about (storage vs. performance issue).
Storage volume is the huge limiting factor.

Philipe: Storage with the embedded compute system in future for HPC?

Salman: Bound to happen but maybe not in next 5 years.

Parallel File system: Vendor specific and users have to deal with it.

Philipe: How likely is it to bypass the file-system and do the streaming?

Salman: Maybe able to just ignore the file system and just do streaming because of the increasing file size. Technology is there but the concern is security.
Isolate the two end points (source where the files are stored and destination where files are processed) and do the streaming right now (maybe?)

Data-Streaming: This is the future thing to do.
CMS/ATLAS: Bring job to the data (that means computation site also needs to have enough storage resources)

Kenneth:
DUNE: Quite a few sites that have reasonably decent levels of computation resources but not storage. Streaming them the data, they can still compute and continue to be involved in the DUNE.

(Regarding the DARSHAN, Amit had asked if DARSHAN knows that type of IO calls are being made…for example there are IO calls related to just accessing the ROOT libraries or other system libraries that are independent of how frameworks are designed. Shane responded that in the early years, DARSHAN had this issue but now it is resolved…i.e. Now the logs only reflect the IO calls related to the framework which is being tested.)

Data Models

Tomorrow we will have a data model parallel session to discuss this in more detail.

HDF5

Saba is sharing the future plan slide (to give context to this part of the minute).
Future work: Immediate and near term future.

Completing performance evaluation in CORI (also work in progress)

Saba: 3 items realistically doable:

CORI performance
Tuning HDF5 tuning
Parallel PP/O integration into the testing framework
Compression
+ Maybe we need to write a technote in the end.

Compression: Kyle: What do you mean by compression?

Saba: How Compression work when we do parallel I/O? Compression in the HDF5 side. Could turn out unnecessary.
Parallel IO and compressions are the things that are not explored yet.

Peter: Can we apply compression to the blobs and then give it to the HDF5 modules?

Chris: Yes. Two ways that Chris tested:

1. Used compression on the user side and then tool ROOT not to compress.
2. Another was ROOT did compression and not the user.

Kyle: The proposal is to turn off explicit compression and enable implicit compression in HDF5?

Saba: Yes.
HDF5 has a plugin to allow which compression system we want.

Peter: Maybe interesting to see how each compression algorithm works and compare it with ROOT.

Philippe: ROOT is way smarter…. :)

Peter: Need to have some sort of note/documentation

Elizabeth: Idea of tuning HDF5 for serial I/O is that a good idea?

Chris: Serial mode means in a single process model. And parallel is multi process mode. Maybe I need to change the terms.

Elizabeth: In the PPS, MPI could be used with tbb as long as you tell MPI you are in single node.

Chris: That is how the test framework with MPI also works. We still control all the threads with tbb and MPI helps to communicate through processes.

Peter: Similar in ATLAS. In each node, multiple threads.

Next bullet: Different HDF5 layout if needed
What we are doing is probably logical. Saba not sure if looking at compound data types.
Another scenario for example if the DUNE uses simpler data but in compound data format.

Elizabeth: Maybe can be tested with DUNE RAW Format.
Why did DUNE decide to take the decision to store in HDF5?
We can test/benchmark with the DUNE raw data and provide the feedback for the DUNE FD.

Kyle: Ability to write to into same HDF5 file from different nodes (from Mike)

Elizabeth: ATLAS is already doing that with SharedWriter.
Philippe: HDF5 does collective IO by reserving spaces during reading and then write collectively in the end.

Kyle: HDF5 layout-->Data model or how data can be stored (in X data sets in Y groups etc)
Saba: How data can be stored (second)

Compound data types: What do you mean by compound data type– C++ objects or using complex data models that won't use serialization?

Saba: Having one data type representing– blob+offset for example. Hence still using the Serialization but just how data is stored.
Peter thinks that it can be done immediately.

Peter: Store objects that don't need serialization or different ways of storing meta-data (token method that was tested with h5demos toy framework) in test framework (ROOT-Serialization).

Peter: The Serialization issue that DUNE Computing brings up usually.

Peter: ROOT serialized blob and store into HDF5 (What we currently do)

Peter:Next step is just design data into HDF5 coupled format and then just write directly.

Peter: Challenge has been the Streamer.

Chris: If we are explicitly doing ROOT serialization then we won't have versioning control that comes with ROOT. One thing to do is how we could go about adding this information.

Elizabeth: Good point since the data has to store for a long time like 10 years (in case of DUNE). Without any data model evolution, they might need such info ….like info about serialization tools. CMS/ATLAS usually evolve over time.

Peter: If direct coupling to HDF5, then there is better control with schema etc.

Philippe: If the data is simple and if the info is saved somewhere (because it is raw data), 10 yrs from when the data is written and stored, the data can be used easily. If the saved data is serialized, then they need to keep track of the object since it is hidden by the serialization tool.

Peter: Complex data means the info on the data is also complicated and probably needs to use the serialization.
So, simple data---> Maybe directly coupled to HDF5 that can be persisted.
Complex -->Still use ROOT serialization.

Peter: Nice extension to the fourth bullet.
Simple data structure that can be stored directly in the HDF5 and maybe with the option of storing serialized data for complicated data structure that can be persisted easily.
Maybe a long term issue.

Elizabeth: RAW Data is simple (matrix). Trigger data on top of RAW Data (complex data)

Peter: ATLAS puts byte stream (complicated) and then serialized object(which are more complicated than byte stream data)

Using node local storage for storing intermediate HDF5 Files:

Examples like CMS's step chain where everything happens in the same node.

Phillipe: Do we still need to directly store the file?

Chris: What is the vision that Saba is thinking when she is saying using node local storage

Saba: This bullet point was added at some point and now she cannot remember the context. Maybe it is something we already do and maybe redundant.

Chris: We do it for ROOT.

Elizabeth: By doing with HDF5, we could tune it to the HPC file system and make the intermediate access better. Maybe that was why it was added?

Peter: This is for the intermediate stage.

Elizabeth: One thing I realized.. The HPC system has no local storage but a shared file system. So, probably this might not be a huge point (using node local storage).
Yay so we decided something we don’t have to do.

MT HDF5:
Peter: Probably Serhan could have some input

Saba: What is the direction or if this is going to be a viable thing to do in the context of MT framework on top of MP framework. Right now, not even initial study is not being done to say or suggest what needs to be done.

Peter: Would be a nice investigation. Could be somewhere where Serhan or Quao could help.

Elizabeth: Did MT since ROOT supported MT (shared-pool) but what does MT in HDF5 look like? For example how TBB would perform with TBB?

Peter: ROOT advantage is that HEP is the primary customer and things are optimized for HEP. But the MT aspect for HDF5 is important.

Data Storage Access from GPUs: Should be the next goal for HEP-CCE 24-26. This also implies that data should be stored directly and not ROOT serialized.

Philippe: Yes and no. Rntuple access through GPU.

Peter: When we design persistent data format, we should keep data format in find.

Philippe: Any potential application in the current or short term plans of the frameworks.

Peter: Not for CMS/ATLAS.

Elizabeth: For the DUNE, it is relevant and they have raw data in HDF5 and raw data can be parallelized.

Elizabeth: Could Bret's kernel can work with the data in HDF5 because of how the framework is built?

Kyle: Wirecell actually has an HDF5 front end but there is a layer that avoids direct communication between HDF5 and wirecell. If we use HDF5 as input, no one has done it yet. That is what Barnali is planning to do but not yet.

Elizabeth: Wirecell is a kernel and uses random numbers to simulate the raw data (so no input from persisted data)

Kyle: Trying to understand why DUNE decided to use HDF5 format (looking at docdb). Being able to write into the same files through multiple processes (mentioned in docdb). Not sure if they can do it

Philipe: Also what is the tangible benefit?

Peter: Also the challenge of compression for collective IO. How you compress data when it comes to collective IO (not ROOT specific or storage format specific).

Elizabeth: Maybe that is why Proto-DUNE (raw data) is not compressing their data but they cannot get away with that in the long term. Maybe repackage to compress the raw data later on like CMS. Send in in one format and send out in another format. Performance bounded by IO. Can use the remaining CPU (CPU resources left after doing the primary job) resources to repack.
In the case of DUNE whether the event size is challenging to do the repackaging ?

Peter: We can be helpful for experiments like DUNE who might decide to store data in HDF5. We can extract as much information from the data-model (persistent) and give feedback.

Elizabeth: We all are moving from Ttree to Rntuple. So, all the existing frameworks have to deal with the transition. DUNE's early adoption of HDF5 could mean they can bypass that transition effort.

RNtuple

Peter: Maybe switch to ROOT-Rntuple and how much effort to adopt Rntuple for existing frameworks.
Transitioning to Rntuple:

Philipe:
Code wise, much more clear ownership model.
No document yet on how to translate the code but should not be difficult.
CMS had done some tests with a simpler format (NanoAOD).
In CMS it is mostly integrated and was done by an undergrad/master student.

Peter:
ATLAS has done similar work with DAOD. But it would be nice to have effort there. It is going to be replaced for the HL-LHC. That means all kinds of data has to be written in that format (raw to final analysis format)

Elizabeth:
Time scales are off.

Peter: For the next cycle (beyond 2024).

Philippe: Rntuple should be ready for these efforts for the next cycle of HEP-CCE .

****

ONE SUGGESTION FROM THE MINUTE WRITER

I realized that both DARSHAN and HDF5 related work (and also the main talk) didn’t have slide numbers. Maybe good to have slide numbers so they could be referenced easily.