A Large Ion Collider Experiment

Vertical Slice and Data Distribution

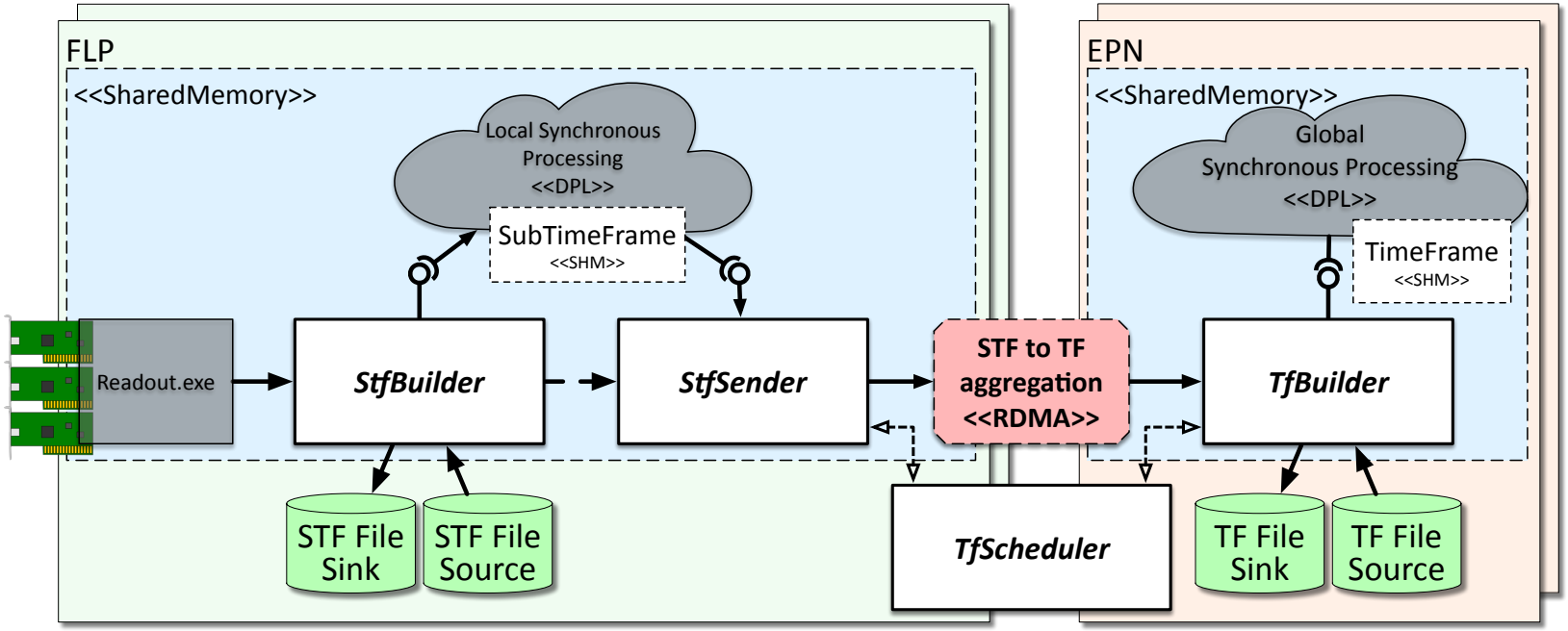
Gvozden Nešković
neskovic@compeng.uni-frankfurt.de

Frankfurt Institute for Advanced Studies

29.11.2019

Data distribution and TimeFrame building

- Scope: take data from readout and build TimeFrames
  - Deliver TimeFrames to a process accepting data, or store to file
- No: data inspection, filtering, reordering, …



Integrations

- Readout
  - Skeptical about usefulness of the setup without actually having proper data sources (i.e. Detectors)
  - We already observed issues only on a setup with the full detector chain
- ECS
  - No development on the interface since discussions
  - Not a big effort to support envisioned set of commands
    - Define messages and service. gRPC already in use internally
- DPL and online processing
  - Work ongoing with TPC data, soon other detectors
- Storage (EOS)
  - What is our data warehousing strategy? File/object semantics?
  - EOS client? Erasure coding?

Whole chain testing

- Definition of all “links in the chain”
  - What is the input data?
    - /dev/zero, simulation, detector?

- What component are actually being tested
  - Where is the data injected?
  - Purpose of tests: performance, scalability, robustness, …?
  - More importantly: what data/code paths we will not be testing?

- Backups

StfBuilder (FLP)

- Input data from readout.exe or (Sub)TimeFrame files
  - CRU data in SHM; only pointers to valid data communicated on-node
  - Collection of HBFrames with the source information (CRU and Link ID)
    - Handles 6.8 GB/s for 2 CRUs (empty pages in TPC trigger mode)

- Output to StfSender or DPL gateway (for local processing)
  - If there's no local processing on FLP, data is forwarded for sending
  - Standalone use to inject recorded STFs or TFs into DPL reconstruction workflows

StfSender (FLP) and TfBuilder (EPN)

- Component discovery and connections
  - Each StfSender in the partition connects to each TfBuilder
  - TfBuilders (EPNs) can join or leave during the run
    - Number of EPNs can be adjusted according to load of EPNs during the run
- Metadata communication to TfScheduler
  - StfSender: New STF information: TF_ID, size, detector, …
  - TfBuilder: Current utilization, free memory for receiving TFs, …
- Time Frame building
  - TfScheduler assigns each TF to new TfBuilder once complete TF-metadata is received
  - TfBuilders requests the STFs with the same ID from all StfSenders
  - Incomplete TFs are dropped or built based on specified policy

TimeFrame scheduling

- Balanced resource utilization
  - EPNs process multiple TFs in parallel
  - Select a least utilized EPN to receive the new TF
  - Support adding and removing EPN nodes
- Application-level traffic shaping
  - Spread in-progress TF transfers evenly over EPN ToR switches
    - Core to EPN-ToR congestion avoidance
  - EPNs pull data from FLPs on different FLP-ToR switches
    - FLP-ToR to Core congestion avoidance
  - InfiniBand HDR adaptive and fallback routing
- TF Scheduler component
  - Facilitate discovery and connecting of FLPs and EPNs
  - Dynamic scheduling of TFs to available EPNs
  - Support operation of multiple partitions of FLPs and EPNs
- Scheduling policies
  - TF aggregation (global processing)
  - STF distribution (local processing)
    - Use EPNs to analyze STFs independently
    - Dynamic or static EPN selection
    - Calibration and commissioning

Deployment for commissioning setups

- Data Distribution components are started using Ansible playbooks
  - Enable fast iteration and tailoring of the workflow to the setup
  - Guaranties repeatable processes and consistent environment
  - Parallel execution on all selected nodes

- FLP playbook
  - Checks if all components FLPs are in “good” state to start the run
  - Prepares the directory structure for the new run (data, logs, misc)
  - Starts all data flow software components on the FLP

- EPN playbook
  - Make sure one instance of TfScheduler is running and configured
  - Starts one TfBuilder per EPN and configure the output: file or DPL workflow

Detector-independent integration testbed

- Many (*all?*) components in the chain under active development
  - CRU fw/sw, data layout, CTP-LTU, TF building, DPL, O^2 reconstruction, …
  - Detector setups are currently required for testing and debugging
    - TPC: O2-905, O2-890

- Possibility to verify the chain without detectors
  - Run tests with a single CRU with emulated link data
  - Tests with multiple CRUs with CTP-LTU and emulated data

- Place to test components independently of detectors
  - Usually small systems of 2 FLPs were enough to solve issues
  - CTP-LTU required for controlling and synchronizing data flow (CRUs)
  - But, also should be possible to run at arbitrary scale and arbitrary detector setup