GPU-based reconstruction and data compression at ALICE during LHC Run 3

David Rohr
drohr@cern.ch, CERN

CHEP 2019
Adelaide, 5.11.2019
ALICE in Run 3: 50 kHz Pb-Pb

Record large minimum bias sample.
- All collisions stored for main detectors → no trigger.
- Continuous readout → data in drift detectors overlap.
- 50x more events, 50x more data.
- Cannot store all raw data → online compression.
→ Use GPUs to speed up online processing.

- Overlapping events in TPC with realistic bunch structure @ 50 kHz Pb-Pb.
- Timeframe of 2 ms shown (will be 10 – 20 ms in production).
- Tracks of different collisions shown in different colors.
Online / Offline Computing in ALICE in Run 3

- ALICE computing strategy for Run 3
  - On-site server farm for **synchronous** (online) processing.
  - When not taking data \(\rightarrow\) used for **asynchronous** (offline).

![Diagram showing data processing flow](image_url)

- **Data links from detectors**
  - **Run 3 farm**
  - **Readout nodes**
    - **Synchronous processing**
      - Local processing
      - Event / timeframe building
      - Calibration / compression
    - **Asynchronous processing**
      - Reprocessing with full calibration
      - Full reconstruction
  - **Disk buffer**
  - **Permanent storage**
    - Compressed Raw Data
    - Reconstructed Data

- **Readout nodes**
  - **Run 3 farm**
  - **Data links from detectors**
    - **> 3.5 TB/s**
    - **> 600 GB/s**
    - **< 100 GB/s**

- **Reconstruction and storage**
  - During Data taking
  - During no beam
  - Compressed Raw Data
  - Reconstructed Data

5.11.2019
David Rohr, drohr@cern.ch

2 / 9
Online / Offline Computing in ALICE in Run 3

- ALICE computing strategy for Run 3
  - On-site server farm for synchronous (online) processing.
  - When not taking data → used for asynchronous (offline).

- Partial ITS + TPC + TRD tracking for TPC calibration
  - reduced statistics sufficient
    (TPC calibration based on matching of TPC / ITS / TRD tracks)
  - Other detectors without significant CPU load

- Full TPC tracking for TPC compression
  - cluster to track residuals → better entropy coding
  - removal of tracks not used for physics
  - Entropy coding for other detectors

Final reconstruction pass with final calibration

Data links from detectors

- Synchronous processing
  - Local processing
  - Event / timeframe building
  - Calibration / compression
- Asynchronous processing
  - Reprocessing with full calibration
  - Full reconstruction

Run 3 farm

Readout nodes

Disk buffer

Permanent storage

Reconstructed Data

Compressed Raw Data

> 3.5 TB/s

> 600 GB/s

< 100 GB/s
Tracking in ALICE in Run 3

- **Bulk of computing workload:**
  - Synchronous
    - >90% TPC tracking / compression
    - Low load for other detectors
  - Asynchronous
    - TPC among largest contributors
    - Other detectors also significant
Baseline solution (almost available today):
TPC + part of ITS tracking on GPU

- **Mandatory** solution to keep up with the data rate online.
- **Defines** number of servers / GPUs.

Asynchronous phase should make use of the available GPUs:
- Available in the O² farm anyway.
- Future HPC / grid sites may have GPUs.

Optimistic solution (what could we do in the ideal case):
Run most of tracking + X on GPU.

- Extension of baseline solution to make best use of GPUs.
  - Ideally, **full barrel tracking** without ever leaving the GPU.
  - In the end, we will probably be somewhere in between.

• Bulk of computing workload:
  - **Synchronous**
  - >90% TPC tracking / compression
  - Low load for other detectors
  - TPC among largest contributors
  - Other detectors also significant

• **ALICE GPU processing strategy**
  - Mandatory solution to keep up with the data rate online.
  - Defines number of servers / GPUs.
  - TPC among largest contributors
  - Other detectors also significant

- ALICE GPU processing strategy
  - **Mandatory** solution to keep up with the data rate online.
  - Defines number of servers / GPUs.
  - Extension of baseline solution to make best use of GPUs.
    - Ideally, **full barrel tracking** without ever leaving the GPU.
    - In the end, we will probably be somewhere in between.
Reconstruction steps on GPU (Barrel Tracking)

- **Status of reconstruction steps on GPU:**
  - All TPC steps during synchronous reconstruction are **required** on the GPU.
  - Synchronous ITS tracking and TPC dE/dx in good shape, thus considered **baseline** on the GPU.
  - Remaining steps in tracking chain part of **optimistic scenario**, being ported step by step to GPU.
    - Porting order follows topology of chain, to avoid unnecessary data transfer for ported steps – current blocker is **TPC ITS matching**.

---

**GPU barrel tracking chain**

- **In operation**
- **Nearly ready**
- **Being studied**
- **Development not started**

**Common GPU Components:**

- **GPU API Framework**
- **Sorting**
- **Material Lookup**
- **Memory Reuse**

---

TRD tracking / TPC calibration: see poster of Ole Schmidt
Space point calibration of the ALICE TPC with track residuals
Reconstruction steps on GPU (Barrel Tracking)

- Status of reconstruction steps on GPU:
  - Baseline scenario: all steps almost ready

New requirement arose few months ago, since clusterizer does not fit in FPGA.
Reconstruction steps on GPU (Barrel Tracking)

- **Status of reconstruction steps on GPU:**
  - Different reconstruction steps enabled in *synchronous* and *asynchronous* reconstruction.

---

**GPU barrel tracking chain**

- **TPC Cluster Finder**
  - In operation
  - Nearly ready
  - Being studied
  - Development not started

- **TPC Track Finding**
- **TPC Track Merging**
- **TPC Track Fit**
- **ITS Track Finding**
- **ITS Track Fit**
- **ITS Vertexing**
- **TPC ITS Matching**
- **TRD Tracking**
- **TOF Matching**
- **Global Fit**
- **TPC Cluster removal**
- **TPC Entropy Compression**

**Common GPU Components:**

- GPU API Framework
- Sorting
- Material Lookup
- Memory Reuse

---

**Synchronous chain**

- *all events*
- *only few % of events*

- **TPC Track Model Compression**
- **TPC dE/dx**
- **TRD Tracking**
- **ITS Afterburner**
- **V0 Finding**
- **TPC Calibration**

**GPU barrel tracking chain**

- Part of baseline scenario
- Part of optimistic scenario

---

**Space point calibration of the ALICE TPC with track residuals**

**TRD tracking / TPC calibration:** see poster of Ole Schmidt

**5.11.2019, David Rohr, drohr@cern.ch**
5.11.2019

David Rohr, drohr@cern.ch

Status of reconstruction steps on GPU:
- Different reconstruction steps enabled in synchronous and asynchronous reconstruction.

GPU barrel tracking chain
- Part of baseline scenario

Asynchronous chain
- Part of optimistic scenario

TPC Cluster Finder
- In operation
- Nearly ready
- Being studied
- Development not started

Common GPU Components:
- GPU API Framework
- Sorting
- Material Lookup
- Memory Reuse

Material Lookup
GPU API Framework
Sorting
Memory Reuse

TPC Track Finding
TPC Track Merging
TPC Track Fit
TPC dE/dx
TRD Tracking
TOF Matching
Global Fit
TPC Cluster
TPC Entropy Compression

TPC Track Model Compression

TRD tracking, TPC calibration; see poster of O. Schmidt
Space point calibration of the ALICE TPC with track residuals
Reconstruction steps on GPU (Barrel Tracking)

**Status of reconstruction steps on GPU:**
- All TPC steps during synchronous reconstruction are required on the GPU.
- Synchronous ITS tracking and TPC \(dE/dx\) are in good shape, thus considered baseline on the GPU.
- Remaining steps in跟踪链 are part of an optimistic scenario, being ported step by step to GPU.

**Strategy:**
- Start with standalone TPC and ITS tracking.
  - Standalone ITS tracking needed since TPC tracks lack absolute time.
  - ITS tracking uses vertexer as first step.
  - TPC tracking has no vertex constraint, starts with segment tracking in individual TPC sectors, than merges the segments and refits.
- ITS and TPC tracks are matched, fixing the time for the TPC.
- The afterburner propagates unmatched TPC tracks into the ITS and tries to find matching hits of short tracks not found in ITS standalone tracking.
- Tracks are extrapolated outwards into the TRD, once the time is fixed.
  - TRD standalone tracking and matching (like for ITS) is less efficient due to many fake TRD tracklets.
- Optionally, after TRD tracks can be extrapolated to TOF.
- Global refit uses the information from all detectors.
- V0 finding
- In the synchronous phase, the TPC compression chain starts after the TPC standalone tracking in parallel:
  - Clusters not used in physics are removed, depending on the strategy (see later) this might require extra steps for identification and rejection of very low \(p_T\) clusters below 10 MeV/c.
  - Track model (and other steps) reduce the entropy for the final entropy encoding.
  - Final entropy encoding using ANS. Not clear yet whether this will run on GPU efficiently. Alternatively, transport entropy-reduced clusters to host and run entropy encoder there.
TPC Tracking performance

- TPC Tracking most critical for synchronous phase.
  - GPU replaces ~40 CPU cores in TPC track finding.
  - This defines size of O² farm.
  - Online tracking efficiency / resolution match Run 2 offline tracking.
  - Factor 40 will change after modifications for continuous readout.

  ![TPC Tracking Performance Diagram](image)

  - Speed-up normalized to single CPU core.
    - Red curve: algorithm speed-up.
    - Other curves: GPU v.s. CPU speed-up corrected for CPU resources.
      - How many cores does the GPU replace.
  - Significant gain with newer GPU (blue v.s. green).
  - GPU with Run 3 algorithm replaces > 800 CPU cores Running Run 2 algorithm (blue v.s. red),
    (at same efficiency / resolution).
  - We see ~30% speedup with new GPU generation (RTX 2080 v.s. GTX 1080).

See CHEP 2018 talk: https://indico.cern.ch/event/587955/contributions/2935757/
TPC Tracking performance

- TPC Tracking **most critical** for synchronous phase.
  - GPU replaces ~40 CPU cores in TPC track finding.
    -> This defines size of O² farm.
  - Online tracking efficiency / resolution match Run 2 offline tracking.
  - Factor 40 will change after modifications for continuous readout.

- TPC Tracking best example for GPU offloading.
  - Other algorithms show 12x – 35x v.s. single CPU core.
  - Depending on CPU / CPU price, still favorable.
  - In particular for asynchronous phase of ALICE:
    -> Without offloading more algorithms → CPUs throttle, GPUs idle

See CHEP 2018 talk: https://indico.cern.ch/event/587955/contributions/2935757/

See CHEP 2018 talk: https://indico.cern.ch/event/587955/contributions/2935757/

- Speed-up normalized to single CPU core.
  - Red curve: algorithm speed-up.
  - Other curves: GPU v.s. CPU speed-up corrected for CPU resources.
    -> How many cores does the GPU replace.
  - Significant gain with newer GPU (blue v.s. green).
  - GPU with Run 3 algorithm replaces > 800 CPU cores Running Run 2 algorithm (blue v.s. red),
    (at same efficiency / resolution).
  - We see ~30% speedup with new GPU generation (RTX 2080 v.s. GTX 1080)

- Online tracking efficiency / resolution match Run 2 offline tracking.
  - Factor 40 will change after modifications for continuous readout.
- TPC Tracking best example for GPU offloading.
  - Other algorithms show 12x – 35x v.s. single CPU core.
  - Depending on CPU / CPU price, still favorable.
  - In particular for asynchronous phase of ALICE:
    -> Without offloading more algorithms → CPUs throttle, GPUs idle

See CHEP 2018 talk: https://indico.cern.ch/event/587955/contributions/2935757/

See CHEP 2018 talk: https://indico.cern.ch/event/587955/contributions/2935757/

- Speed-up normalized to single CPU core.
  - Red curve: algorithm speed-up.
  - Other curves: GPU v.s. CPU speed-up corrected for CPU resources.
    -> How many cores does the GPU replace.
  - Significant gain with newer GPU (blue v.s. green).
  - GPU with Run 3 algorithm replaces > 800 CPU cores Running Run 2 algorithm (blue v.s. red),
    (at same efficiency / resolution).
  - We see ~30% speedup with new GPU generation (RTX 2080 v.s. GTX 1080)
Memory requirements

- **ALICE reconstructs timeframes (TF) independently (~10 – ~20 ms; 128 – 256 orbits; ~500 – ~1000 collisions).**
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 1 % of statistics lost (< 0.5 % for 20 ms).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator:** grabs all GPU memory, gives out chunks manually, memory will be reused when possible.
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.

---

**Memory**

- **Persistent data**
  - TPC Hits 1
  - TPC cluster finder
  - TPC hits must persist, needed for final refit.

- **Non-persisting input data**
  - TPC Raw 1
  - TPC raw data can be removed after clusterization, memory will be reused.
Memory requirements

- **ALICE reconstructs timeframes (TF) independently (~10 – ~20 ms; 128 – 256 orbits; ~500 – ~1000 collisions).**
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) \( \rightarrow < 1 \% \) of statistics lost (< 0.5 % for 20 ms).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.

- **Custom allocator:** grabs all GPU memory, gives out chunks manually, memory will be reused.
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.

**TPC cluster finder**

![Memory diagram](image-url)
Memory requirements

- ALICE reconstructs timeframes (TF) independently (~10 – ~20 ms; 128 – 256 orbits; ~500 – ~1000 collisions).
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 1 % of statistics lost (< 0.5 % for 20 ms).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- Custom allocator: grabs all GPU memory, gives out chunks manually, memory will be reused when possible.
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.

### Memory

<table>
<thead>
<tr>
<th>Persistent data</th>
<th>Non-persistent scratch data for algorithms</th>
<th>Non-persisting input data</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC Hits 1</td>
<td>Scratch</td>
<td>TPC Raw 4</td>
</tr>
<tr>
<td>TPC Hits 2</td>
<td>Scratch</td>
<td>TPC Raw 3</td>
</tr>
<tr>
<td>TPC Hits 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPC Hits 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TPC cluster finder

Can run multiple instances, in parallel...
Memory requirements

- ALICE reconstructs timeframes (TF) independently (~10 – ~20 ms; 128 – 256 orbits; ~500 – ~1000 collisions).
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 1 % of statistics lost (< 0.5 % for 20 ms).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- Custom allocator: **grabs all GPU memory**, gives out chunks **manually**, memory will be **reused** when possible.
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.

### Memory

<table>
<thead>
<tr>
<th>TPC Hits 1</th>
<th>TPC Hits 2</th>
<th>TPC Hits 3</th>
<th>TPC Hits 4</th>
<th>ITS Hits</th>
<th>TPC Tracks</th>
<th>ITS Tracks</th>
</tr>
</thead>
</table>

- **Persistent data**
  - TPC Hits
  - ITS Tracks

- **Non-persistent scratch data for algorithms**
  - Scratch

- **Non-persisting input data**
  - Scratch

*Input data may also be persistent, ITS hits are reused in the final fit.*

... or run multiple algorithms in parallel
Memory requirements

- **ALICE reconstructs timeframes (TF) independently** (~10 – ~20 ms; 128 – 256 orbits; ~500 – ~1000 collisions).
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 1 % of statistics lost (< 0.5 % for 20 ms).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator:** *grabs all GPU memory, gives out chunks manually, memory will be reused when possible.*
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.

### Memory

<table>
<thead>
<tr>
<th>Persistent data</th>
<th>Non-persistent scratch data for algorithms</th>
<th>Non-persisting input data</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC Hits 1</td>
<td>TPC Hits 2</td>
<td>TPC Hits 3</td>
</tr>
<tr>
<td>TPC Hits 4</td>
<td>TPC Tracks</td>
<td>ITS Tracks</td>
</tr>
<tr>
<td>ITS Hits</td>
<td>Compressed TPC Hits</td>
<td>Scratch</td>
</tr>
<tr>
<td>TPC Tracks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Gaps can appear when size is not known exactly in advance.
- Minor problem with time frames since most fluctuations average out.
- Could compact the memory but probably not needed.

Some output can be moved to the host immediately, and the memory reused.

Available memory for scratch buffers decreases, but most memory is needed at the beginning for TPC clustering and tracking.
5.11.2019

Memory requirements

- **ALICE reconstructs timeframes (TF) independently** (~10 – ~20 ms; 128 – 256 orbits; ~500 – ~1000 collisions).
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) \( \rightarrow \) < 1 % of statistics lost (< 0.5 % for 20 ms).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator:** grabs all GPU memory, gives out chunks manually, memory will be reused when possible.
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.

### Memory allocation

<table>
<thead>
<tr>
<th>Persistent data</th>
<th>Non-persistent scratch data for algorithms</th>
<th>Non-persisting input data</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC Hits 1</td>
<td>TPC Tracks</td>
<td>TPC Raw 1</td>
</tr>
<tr>
<td>TPC Hits 2</td>
<td>ITS Tracks</td>
<td></td>
</tr>
<tr>
<td>TPC Hits 3</td>
<td>Matches</td>
<td></td>
</tr>
<tr>
<td>TPC Hits 4</td>
<td>Scratch</td>
<td></td>
</tr>
<tr>
<td>ITS Hits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPC Tracks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITS Tracks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matches</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scratch</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TPC ITS Matching**

Preload TPC raw data of next TF before current TF is finished.
Memory requirements

- **ALICE reconstructs timeframes (TF) independently** (~10 – ~20 ms; 128 – 256 orbits; ~500 – ~1000 collisions).
  - One TPC drift time of data not reconstructible at TF border (~ 90 us) → < 1 % of statistics lost (< 0.5 % for 20 ms).
  - Timeframe should fit in GPU memory. If not, could use kind of ring buffer, or reduce TF length to 128 orbits.
  - Trying to avoid the ring buffer approach, could be added later if needed.
- **Custom allocator:** grabs all GPU memory, gives out chunks manually, memory will be **reused** when possible.
  - Classically: reuse memory between events, collisions are not that large.
  - ALICE reuses memory between different algorithms in a TF, possibly also between independent collisions.
  - Some memory must persist during timeframe processing.

**Estimated maximum memory needed during important for 10 ms TF (*2 for 20 ms):**
- TPC Cluster finder: ~ 3 GB (+ input / scratch data, which is pipelined)
- TPC Transformation: 12.1 GB
- TPC Sector tracker: ~ 14.6 GB (including persistent memory from previous steps)
- TPC Merger / track fit: 14.1 GB
- TPC Compression: 12.9 GB
  - Later steps do not scale their scratch memory with TPC input → less memory intensive.

→ **16 GB GPU will suffice for 10 ms TF (unclear for 12 GB after optimizations).**
- 8 GB insufficient for 10 ms TF, 20 ms TF needs 32 GB, alternatively ring buffer.
Compatibility with several GPU frameworks

- **Generic common C++ Code compatible to CUDA, OpenCL, HIP, and CPU** (with pure C++, OpenMP, or OpenCL).
  - OpenCL needs clang compiler (ARM or AMD ROCm) or AMD extensions (TPC track finding only on Run 2 GPUs and CPU for testing).
  - Certain worthwhile algorithms have a vectorized code branch for CPU using the Vc library.
  - All GPU code swapped out in dedicated libraries, same software binaries run on GPU-enabled and CPU servers.
Compatibility with several GPU frameworks

- Generic common C++ Code compatible to CUDA, OpenCL, HIP, and CPU (with pure C++, OpenMP, or OpenCL).
  - OpenCL needs clang compiler (ARM or AMD ROCm) or AMD extensions (TPC track finding only on Run 2 GPUs and CPU for testing).
  - Certain worthwhile algorithms have a vectorized code branch for CPU using the Vc library.
  - All GPU code swapped out in dedicated libraries, same software binaries run on GPU-enabled and CPU servers.

- Screening different platforms for best price / performance.
  (including some non-competitive platforms for cross-checks and validation.)
  - CPUs (AMD Zen, Intel Skylake)
    C++ backend with OpenMP, AMD OCL
  - AMD GPUs
    (S9000 with OpenCL 1.2, MI50 / Radeon 7 / Navi with HIP / OCL 2.x)
  - NVIDIA GPUs
    (RTX 2080 / RTX 2080 Ti / Tesla T4 with CUDA)
  - ARM Mali GPU with OCL 2.x
    (Tested on dev-board with Mali G52)

- Optimize TCO (faster GPUs → less latency → smaller buffers).
Data compression

- Data compression mandatory to store minimum bias data.
  - TPC most critical as largest data contributor (lossy + lossless compression).
    - **Online cluster finding** (basis for entropy compression, entropy coding of raw data insufficient).
    - All cluster properties stored in individually optimized fixed / floating point format.
    - Coordinates of unattached clusters sorted, and stored as differences.
    - Entropy-compressed via Huffman (Run 2).

- Several steps already used in Run 2.
  - Achieved 8x compression in **Run 2** (4x in Run 1).
  - Aim for **20x** in **Run 3** (wrt. Run 2 raw data size).
Data compression

- Data compression mandatory to store minimum bias data.
- TPC most critical as largest data contributor (lossy + lossless compression).
  - Online cluster finding (basis for entropy compression, entropy coding of raw data insufficient).
  - All cluster properties stored in individually optimized fixed / floating point format.
  - Coordinates of unattached clusters sorted, and stored as differences.
  - Entropy-compressed via ANS (reaching ideal entropy).
  - Correlated properties encoded together.

For ANS, see poster of Michael Lettrich:
Fast and Efficient Entropy Compression of ALICE Data using ANS Coding
Data compression

- Data compression mandatory to store minimum bias data.
  - TPC most critical as largest data contributor (lossy + lossless compression).
    - Online cluster finding (basis for entropy compression, entropy coding of raw data insufficient).
    - All cluster properties stored in individually optimized fixed / floating point format.
    - Coordinates of unattached clusters sorted, and stored as differences.
    - Entropy-compressed via ANS (reaching ideal entropy).
    - Correlated properties encoded together.
    - Coordinates of clusters attached to tracks stored as residual to extrapolated track.

Track model compression:
Needs track refit in distorted (uncorrected) coordinates for best residuals in raw TPC coordinates.

For ANS: see poster of Michael Lettrich: Fast and Efficient Entropy Compression of ALICE Data using ANS Coding
For track model: see CTD2019 talk: https://indico.cern.ch/event/742793/contributions/3274344/
Data compression

- Data compression mandatory to store minimum bias data.
  - TPC most critical as largest data contributor (lossy + lossless compression).
    - Online cluster finding (basis for entropy compression, entropy coding of raw data insufficient).
    - All cluster properties stored in individually optimized fixed / floating point format.
    - Coordinates of unattached clusters sorted, and stored as differences.
    - Entropy-compressed via ANS (reaching ideal entropy).
    - Correlated properties encoded together.
    - Coordinates of clusters attached to tracks stored as residual to extrapolated track.
    - Clusters of tracks not used for physics are removed.

Candidates for removal (~50% of total TPC hits):
  - Tracks below 50 MeV/c
  - Additional legs of looping tracks below 200 MeV/c
  - Track segments with high inclination angle
  - Noisy pads
  - Charge clouds of low $p_T$ protons
Data compression

- Data compression mandatory to store minimum bias data.
  - TPC most critical as largest data contributor (lossy + lossless compression).
    - Online cluster finding (basis for entropy compression, entropy coding of raw data insufficient).
    - All cluster properties stored in individually optimized fixed / floating point format.
    - Coordinates of unattached clusters sorted, and stored as differences.
    - Entropy-compressed via ANS (reaching ideal entropy).
    - Correlated properties encoded together.
    - Coordinates of clusters attached to tracks stored as residual to extrapolated track.
    - Clusters of tracks not used for physics are removed.

TPC data rejection alternatives

A. Reject only clusters of identified background / tracks (loopers).
   Rejects: 12.5% - 39.1%

B. Keep only clusters attached or in proximity of identified signal tracks.
   Rejects: 37.3% - 52.5%

For ANS: see poster of Michael Lettrich: Fast and Efficient Entropy Compression of ALICE Data using ANS Coding

For track model: see CTD2019 talk: https://indico.cern.ch/event/742793/contributions/3274344/
Data compression

- Data compression mandatory to store minimum bias data.
  - TPC most critical as largest data contributor (lossy + lossless compression).
    - Online cluster finding (basis for entropy compression, entropy coding of raw data insufficient).
    - All cluster properties stored in individually optimized fixed / floating point format.
    - Coordinates of unattached clusters sorted, and stored as differences.
    - Entropy-compressed via ANS (reaching ideal entropy).
    - Correlated properties encoded together.
    - Coordinates of clusters attached to tracks stored as residual to extrapolated track.
    - Clusters of tracks not used for physics are removed.
  - Strategy B already yields sufficient data reduction, still working on strategy A.

TPC data rejection alternatives

A. Reject only clusters of identified background / tracks (loopers).
   Rejects: 12.5% - 39.1%

B. Keep only clusters attached or in proximity of identified signal tracks.
   Rejects: 37.3% - 52.5%

Current challenges:

- Incomplete merging of legs of looping tracks (strategy A & B).
- Strategy A needs in addition identification of low $p_T$ hits below 10 MeV/c (15% of hits), which is inaccessible to tracking.

For ANS: see poster of Michael Lettrich: Fast and Efficient Entropy Compression of ALICE Data using ANS Coding

For track model: see CTD2019 talk: [https://indico.cern.ch/event/742793/contributions/3274344/](https://indico.cern.ch/event/742793/contributions/3274344/)
Data compression

- Data compression mandatory to store minimum bias data.
  - TPC most critical as largest data contributor (lossy + lossless compression).
    - Online cluster finding (basis for entropy compression, entropy coding of raw data insufficient).
    - All cluster properties stored in individually optimized fixed / floating point format.
    - Coordinates of unattached clusters sorted, and stored as differences.
    - Entropy-compressed via ANS (reaching ideal entropy).
    - Correlated properties encoded together.
    - Coordinates of clusters attached to tracks stored as residual to extrapolated track.
    - Clusters of tracks not used for physics are removed.
  - Strategy B already yields sufficient data reduction, still working on strategy A.

Data compression 8 / 9

For ANS: see poster of Michael Lettrich: Fast and Efficient Entropy Compression of ALICE Data using ANS Coding

For track model: see CTD2019 talk: https://indico.cern.ch/event/742793/contributions/3274344/

Tracking needed for data compression
Data compression

- Data compression mandatory to store minimum bias data.
  - TPC most critical as largest data contributor (lossy + lossless compression).
    - Online cluster finding (basis for entropy compression, entropy coding of raw data insufficient).
    - All cluster properties stored in individually optimized fixed / floating point format.
    - Coordinates of unattached clusters sorted, and stored as differences.
    - Entropy-compressed via ANS (reaching ideal entropy).
    - Correlated properties encoded together.
    - Coordinates of clusters attached to tracks stored as residual to extrapolated track.
    - Clusters of tracks not used for physics are removed.
  - Strategy B already yields sufficient data reduction, still working on strategy A.

- Other detectors also use ANS coding.
  - Optionally use additional steps: zero suppression, cluster shape hashing (ITS), …
  - Less critical than TPC

- Total data rate estimate with strategy B: 71.7 – 89.9 GB/s (TDR: 88 GB/s).
Summary

• ALICE will take 50 kHz of Pb-Pb data in Run 3: 3.5 TB/s of raw data into online farm.
  • No triggers, full minimum bias sample stored, full online processing in software for data compression.

• During data taking: > 90% of synchronous reconstruction will run on GPU.
  • Offloading the remaining <10% not worth it.

• The optimistic scenario moves a significant fraction of the asynchronous reconstruction onto GPU.
  • Make full use of online GPU farm, and of possible GRID GPU sites.

• Memory reusage allows to process full 10 ms TF on GPU with 16 GB.

• Generic common C++ code runs on CPU and GPU (CUDA, HIP, OpenCL).

• 1 GPU replaces about 40 CPU cores in TPC tracking.
  • 12 – 35 cores in other steps.
  • Processing time goes linearly with data size.

• General TPC compression implemented.
  • Strategy B already yields sufficient reduction (< 90 GB/s stored to disk).
  • Strategy A needs better identification of clusters to reject (in particular below 10 MeV/c).
  • Better merging of loopers needed for best data reduction.