Parallel Unpacking

Felix Weiglhofer (weiglhofer@fias.uni-frankfurt.de)

29. Juni 2023

Group Lindenstruth, Frankfurt Institute for Advanced Studies (FIAS)

- Parallel Unpacker by Sebastian Heinemann
- Unpack MS in parallel
- Adjustment by me: Unpack entire TS in parallel, not only components
- MR !1219
- Algorithm:
    1. Parallel for all MS: Unpack MS
    2. Count all Digis and allocate output vector
    3. In parallel: Copy Digis to output vector

- Scheme is detector independent
- Only requires changes to Unpacker, to make them thread-safe (very easy in case of STS and TOF)
- Should be easy to adapt most other detectors as well

- Step 2 uses `vector::resize`
- `vector::resize` initialises underlying memory
- **Resize takes up 70% of unpacking**

Solution:

- Custom allocator `PODAllocator` that doesn't initialise memory (wrapper around malloc and free)
- Typedef for vector with new allocator:

```
template<typename T>
using PODVector = std::vector<T, PODAllocator<T>>;
```

- Replace usage of `std::vector` with `PODVector`
- MR !1221

- Small change with wide-ranging consequences ...
- To the STL `std::vector` and `PODVector` are two separate types
- So not possible to use `std::move`, changing types requires copy
- Additional requirement: Digis need trivial constructor (Don't initialise members)
- Avoid copy: Requires changing type in `CbmDigiData` to `PODVector`
- As convention: We should use `gsl::span` as input when possible (allocator agnostic)
- Open question: Does Root schema evolution work when changing allocator?