RNTuple: AoS and SoA

Jonas Hahnfeld, for the RNTuple development team

RNTuple Workshop – December 3, 2024

Context: Array of Structs / Struct of Arrays

```
struct Point {
  int x;
  int y;
};

using PointsAoS = ROOT::RVec<Point>;

struct PointsSoA {
  ROOT::RVec<int> x;
  ROOT::RVec<int> y;
};
```

see also last year's contributions: https://indico.cern.ch/event/1303499/

On Disk: Storage Efficiency

- Want one column for the sizes and one column per member
  - ie not one size per member

- Comes natural with columnar storage of an AoS:
  `model->MakeField<PointsAoS>("points");`

```
*******************************************************************************
* Field 1           : points (ROOT::VecOps::RVec<Point>)                      *
*   Field 1.1       : _0 (Point)                                              *
*     Field 1.1.1   : x (std::int32_t)                                        *
*     Field 1.1.2   : y (std::int32_t)                                        *
*******************************************************************************
```

from `reader->PrintInfo(kStorageDetails)`

```
# Fields:           5
# Columns:          3
```

On Disk: Storage Efficiency

- Want one column for the sizes and one column per member
  - ie not one size per member

- Comes natural with columnar storage of an AoS:
  `model->MakeField<PointsAoS>("points");`

```
*****************************************************************************
* Field 1           : points (ROOT::VecOps::RVec<Point>)                    *
*   Field 1.1       : _0 (Point)                                            *
*     Field 1.1.1   : x (std::int32_t)                                      *
*     Field 1.1.2   : y (std::int32_t)                                      *
*****                                                                   *****
```

from

Proposal: declare this the canonical on-disk representation in RNTuple

```
# Fields:           5
# Columns:          3
```

On Disk: Projected Fields

- Concept in RNTuple to present columns as different fields
  - For example used in the `RNTupleImporter` to convert TTree leaf count arrays

- Can be used to add array projections on disk:

```
auto points_x =
  std::make_unique<RField<decltype(PointsSoA::x)>>("points_x");
model->AddProjectedField(std::move(points_x),
                         [](const std::string &s) {
  if (s == "points_x") {
    return "points";
  } else {
    return "points._0.x";
  }
});
```

On Disk: Projected Fields

```
*****************************************************************************
* Field 1           : points (ROOT::VecOps::RVec<Point>)                    *
*   Field 1.1       : _0 (Point)                                            *
*     Field 1.1.1   : x (std::int32_t)                                      *
*     Field 1.1.2   : y (std::int32_t)                                      *
* Field 2           : points_x (ROOT::VecOps::RVec<std::int32_t>)           *
*   Field 2.1       : _0 (std::int32_t)                                     *
* Field 3           : points_y (ROOT::VecOps::RVec<std::int32_t>)           *
*   Field 3.1       : _0 (std::int32_t)                                     *
*****************************************************************************
```

from `reader->PrintInfo(kStorageDetails)`

```
# Fields:          9
# Columns:         3
# Alias Columns:   4
```

On Disk: Projected Fields

- Would it be desirable to project the whole SoA?

```
auto points_soa =
  std::make_unique<RField<PointsSoA>>("points_soa");
model->AddProjectedField(std::move(points_soa),
                         [](const std::string &s) {
  if (s == "points_soa") {
    return "points";
  } else if (s == "points_soa.x" || "points_soa.y") {
    return "points"; // The index columns of the RVecs
  } else if (s == "points_soa.x._0") {
    return "points._0.x"; // The item column of the RVec x
  } else if (s == "points_soa.y._0") {
    return "points._0.y"; // The item column of the RVec y
  }
});
```

Doesn't work at the moment, mismatch of field structural roles

RNTuple Reading: Array of Structs

- Reading trivial for AoS, since data is stored that way:

```
auto reader = RNTupleReader::Open("ntpl", "points.root");
```

`reader->Show(0)`:

```
{
  "points": [{"x": 0, "y": 1}, {"x": 2, "y": 3}, {"x": 4, "y": 5}]
}
```

`reader->Show(1)`:

```
{
  "points": [{"x": 6, "y": 7}, {"x": 8, "y": 9}]
}
```

RNTuple Reading: With Projections

- Projected fields allow reading as individual arrays

- `reader->Show(0)`:

```
{
  "points": [{"x": 0, "y": 1}, {"x": 2, "y": 3}, {"x": 4, "y": 5}],
  "points_x": [0, 2, 4],
  "points_y": [1, 3, 5]
}
```

- Currently projected fields can only be defined while writing
  - Is there a use case for adding projected fields on-the-fly when reading?

RNTuple Reading: Struct of Arrays with Views

```
auto reader = RNTupleReader::Open("ntpl", "points.root");

PointsSoA soa;
auto points_x =
  reader->GetView<decltype(PointsSoA::x)>("points_x", &soa.x);
auto points_y =
  reader->GetView<decltype(PointsSoA::y)>("points_y", &soa.y);

points_x(0);
points_y(0);
print("soa.x", soa.x); // -> soa.x: [0, 2, 4]
print("soa.y", soa.y); // -> soa.y: [1, 3, 5]
```

RNTuple Reading: `SoAView`

- Can build a view that feels like SoA
  - Should we add the implementation as a tutorial or to RNTuple proper?

```
auto reader = RNTupleReader::Open("ntpl", "points.root");

auto view = SoAView(*reader, "points");
auto view_x = view.GetColumnView<decltype(Point::x)>("x");
auto view_y = view.GetColumnView<decltype(Point::y)>("y");

print("view_x", view_x(0)); // -> view_x: [0, 2, 4]
print("view_y", view_y(0)); // -> view_y: [1, 3, 5]
```

RNTuple Reading: Bulk Reading

- Can use bulk reading to read all arrays into a single memory buffer
  - See Andrea's presentation from last year on CMSSW implementation (with TTree)

```
auto view = reader->GetCollectionView("points");
auto view_x = view.GetView<int>("_0.x");
auto bulk_x = view_x.CreateBulk();
auto view_y = view.GetView<int>("_0.y");
auto bulk_y = view_y.CreateBulk();

auto range = view.GetCollectionRange(/*globalIndex=*/0);
std::unique_ptr<int[]> buffer(new int[2 * range.size()]);
std::unique_ptr<bool[]> mask(new bool[range.size()]);
std::fill(mask.get(), mask.get() + range.size(), true);

bulk_x.AdoptBuffer(buffer.get(), range.size());
bulk_x.ReadBulk(*range.begin(), mask.get(), range.size());
bulk_y.AdoptBuffer(buffer.get() + range.size(), range.size());
bulk_y.ReadBulk(*range.begin(), mask.get(), range.size());
```

RNTuple Writing: Array of Structs

- Writing again trivial for AoS
  - (this is how I produced the ntuples discussed so far)

```
auto model = RNTupleModel::Create();
auto points = model->MakeField<PointsAoS>("points");
// Add projections if wanted...

auto writer = RNTupleWriter::Recreate(std::move(model),
                                      "ntpl", "points.root");
*points = {{0, 1}, {2, 3}, {4, 5}};
writer->Fill();
*points = {{6, 7}, {8, 9}};
writer->Fill();
```

RNTuple Writing: Struct of Arrays

- Is there a use case for writing SoA data into AoS on-disk format?
- Could be supported via a custom field taking multiple vectors
  - Not implemented and currently not on priority list – is it needed?

Any Other Business

- Currently not considered:
  - Combining data from multiple entries
  - Having AoS / SoA data inside other collections
- Anything else?