<table>
<thead>
<tr>
<th>PR</th>
<th>Title</th>
<th>Status</th>
<th>Components</th>
<th>Impact</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>fix: GdmlGeantinoRecording does not need DD4hep</td>
<td>✔</td>
<td>Component - Core</td>
<td>Impact - Minor</td>
<td>next</td>
</tr>
<tr>
<td>2</td>
<td>ci: Disable codecov-patch check</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>perf: Add track summary root writer and standalone script for reco performance plots</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>fix: Search of last measurement index</td>
<td>✗</td>
<td>Component - Core</td>
<td>Impact - Minor</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>fix: trackAverage when surface has zero binning</td>
<td>✗</td>
<td>Component - Core</td>
<td>Impact - Minor</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>fix: Root readers ensure event ordering is consistent</td>
<td>✗</td>
<td>Component - Examples</td>
<td>Impact - Minor</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>feat: Allow the writing of Json object to Cbor</td>
<td>✔</td>
<td>Component - Examples</td>
<td>Component - Plugins</td>
<td>Feature</td>
</tr>
<tr>
<td>8</td>
<td>fix: Remove Geant4 data archive after extraction</td>
<td>✗</td>
<td>Bug</td>
<td>Component - Examples</td>
<td>Impact - Minor</td>
</tr>
</tbody>
</table>

**Performance writing update**

- **Fix to CKF**
- **Work on ROOT reading for performance tuning**
- **Possibility to write binary JSON format**
- **Cleanup of G4 data folder**
- **Decouples Gdml building from DD4hep**
- **Codecov-check is switched off**

**PRs | 08/06/21 - 22/06/21**
**BREAKING CHANGES**

- Refactor of the template for the input measurements in the CKF findTracks invocation. ([b84d305](#), [8830](#))
  It implements a source link accessor concept which has:
  - type members including `Container`, `Key`, `Value`, `Iterator`
  - lookup methods including `count`, `range` and `at` to help access the source link container

  The CKF findTracks is changed to be templated on the source link accessor. In this way, the input measurements can be directly used by the CKF Actor without further internal converting as long as proper accessor is provided.

- The constructor and public members of `Acts::Navigator` change ([b4f988c](#), [892b](#))
  When before it could be created like
  ```cpp
  Acts::Navigator nav(tGeometry);
  nav.resolveSensitive = true;
  nav.resolveMaterial = true;
  nav.resolvePassive = true;
  ```
  it must now be created like
  ```cpp
  Acts::Navigator::Config cfg;
  cfg.trackingGeometry = tGeometry;
  nav.resolveSensitive = true;
  nav.resolveMaterial = true;
  nav.resolvePassive = true;
  ```
  Since `trackingGeometry` is the first member of `Acts::Navigator::Config`, if you don’t want to change the `resolve*` values, you can also write
  ```cpp
  Acts::Navigator nav(cfg);
  ```
The instantiation of an interpolated B field changes. (ba385e3, #828)

Old:
using Grid_t = detail::Grid<detail::EquidistantAxis, detail::EquidistantAxis>;
using Mapper_t = InterpolatedBFieldMapper<Grid_t>;
using BField_t = InterpolatedBFieldMap<Mapper_t>;

Grid_t g(std::make_tuple(std::move(r), std::move(z)));
Mapper_t mapper(transformPos, transformBField, std::move(g));
BField_t::Config config(std::move(mapper));
config.scale = 1;

BField_t b(std::move(config));

Now:
using Grid_t = detail::Grid<Vector3, detail::EquidistantAxis, detail::EquidistantAxis>;
using BField_t = InterpolatedBFieldMap<Grid_t>;

Grid_t g(std::make_tuple(std::move(r), std::move(z)));
BField_t::Config cfg;
cfg.transformPos = transformPos;
cfg.transformBField = transformBField;
cfg.grid = std::move(g);
cfg.scale = 1;

B field access returns Result (ba371e2, #828)

- The signature of field query methods in MagneticFieldProvider changes from
  
  virtual Vector3 getField(const Vector3& position, Cache& cache) const = 0;

  virtual Vector3 getFieldGradient(const Vector3& position, ActMatrix<3, 3>& derivative, Cache& cache) const = 0;

  to

  virtual Result<Vector3> getField(const Vector3& position, Cache& cache) const = 0;

  virtual Result<Vector3> getFieldGradient(const Vector3& position, ActMatrix<3, 3>& derivative, Cache& cache) const = 0;

- Acts::MagneticFieldProvider loses two pure virtual overloads (ba38872, #819)

  virtual Vector3 getField(const Vector3& position) const = 0;

  virtual Vector3 getFieldGradient(const Vector3& position, ActMatrix<3, 3>& derivative) const = 0;

Clients of generic magnetic field providers need to be adapted.
Finally …

... on the brink of submission!
Upcoming meetings | 22/06/21 - 29/06/21

Acts Parallelization Meeting
Friday 25 Jun 2021, 16:00 → 17:15 Europe/Zurich

Videoconference Rooms
Acts Parallelization Meeting

16:00 → 16:10 Introduction
Speaker: Paul Geesinger (DERN)