# Acts parallelization

04.06.2021

# traccc - CPU parallelization v1

Sequential algorithm

```
for event in events:
  for module in modules:
    clusters=F1(cells)
    meas=F2(clusters)
    sp=F3(meas)
```

Parallel algorithm (v1)

```
parallel_for event in events:
  modules = from_events()
  parallel_for module in modules:
      sp=F3(F2(F1(cells))
```

## Observations

- v1 code in available in acts/traccc PR#40
- **5x** faster than the sequential example (tested on 10 events!)

# traccc - CPU parallelization v2

Sequential algorithm

```
for event in events:
  for module in modules:
    clusters=F1(cells)
    meas=F2(clusters)
    sp=F3(meas)
```

Parallel algorithm (v1)

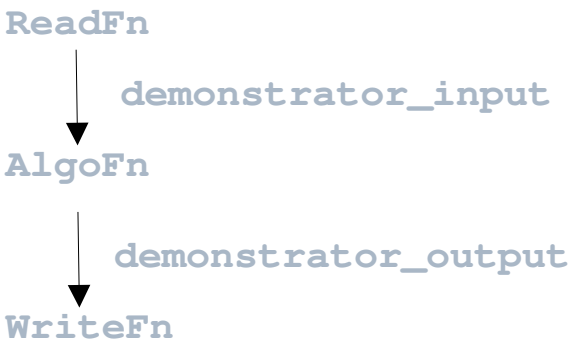
```
parallel_for event in events:
  modules = from_events()
  parallel_for module in modules:
      sp=F3(F2(F1(cells))
```

Parallel & functional & IO decoupled algorithm (v2)

```
(WriteFn
    (AlgoFn
       (ReadFn (x)))
```

## Observations

- v2 is **still 5x** faster than the sequential example (tested on 10 events!)




- v2 is almost stateless!!!
- **Issue**: memory leaks if std containers are used for aggregating data (~3M bytes marked as “definitely” or “indirectly lost” by memcheck tool)
- **Fix**: Use vecmem data structures & memory resource defined as global variable

# Observations on v2

```
namespace traccc {

    vecmem::host_memory_resource resource;

    struct result {
        traccc::host_measurement_container measurements;
        traccc::host_spacepoint_container spacepoints;
        ...
    };

    using geometry = std::map<traccc::geometry_id, traccc::transform3>;
    using demonstrator_input = vecmem::vector<traccc::host_cell_container>;
    using demonstrator_result = vecmem::vector<traccc::result>;
}
```

- The code sets this resource as default resource

```
set_default_resource(&traccc::resource);
```

- The resource is passed to the constructors

```
traccc::demonstrator_input input_data(events, &traccc::resource);

traccc::demonstrator_result aggregated_results(input_data.size(),
                                               &traccc::resource);
```

(No PR yet due to further decoupling in progress, but online in my github [1])

# Question

Is this the intended way to use the vecmem memory resource, as a global resource?

Other alternatives:

- Define it at the beginning of the algorithm and pass it along as an argument to the calling functions
- Use std containers in conjunction with the memory resource (not sure if that is possible)