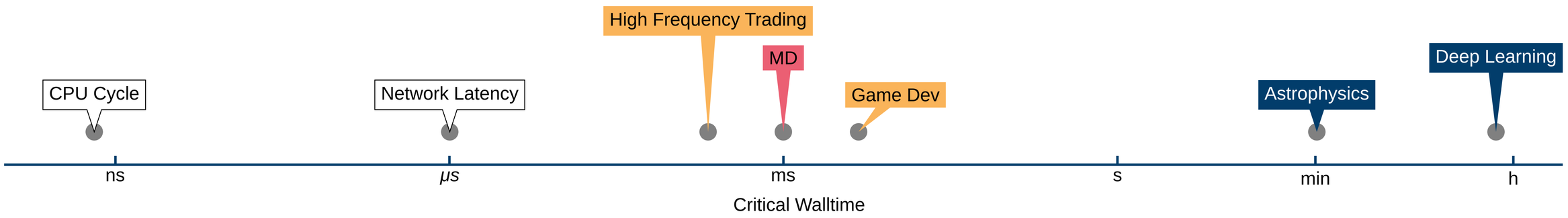
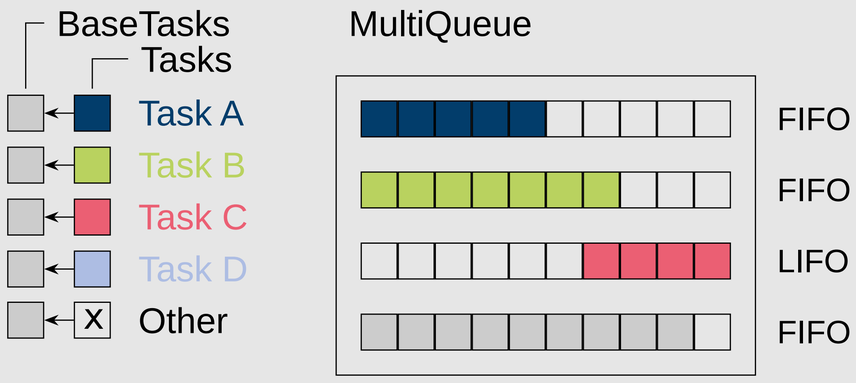
# A C++ Tasking Framework with Compile-Time Dispatching and Type-Driven Priority Scheduling for HPC

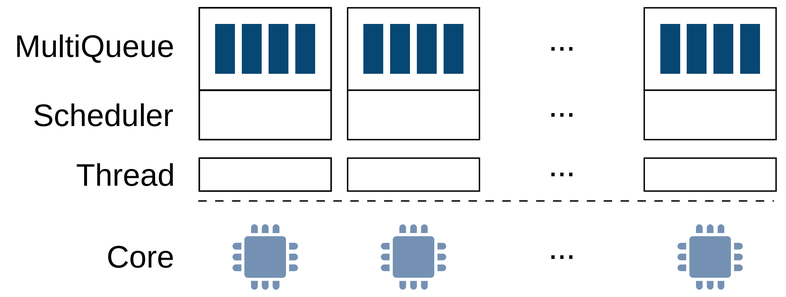
D. Haensel, A. Beckmann, L. Morgenstern, I. Kabadshow



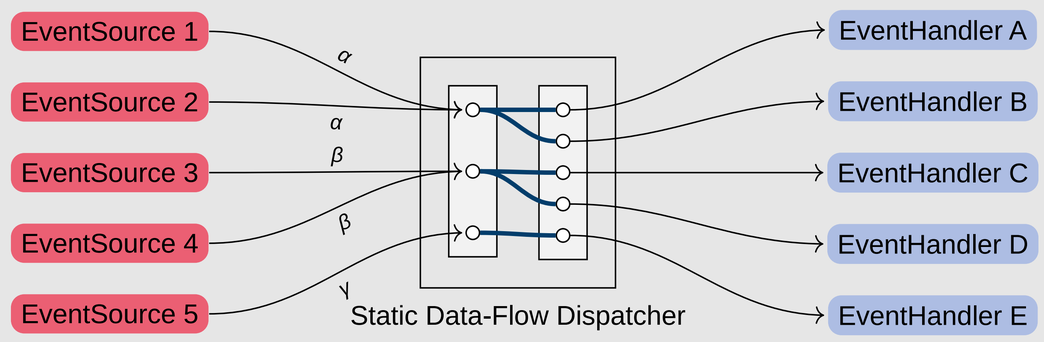

## Type-Driven Priority Scheduling




- Task type is used as priority
- Typed sub-queues can be configured by the user
- Order of typed sub-queues defines prioritization
- Sub-queues are stored in std::tuple
- Selection of corresponding sub-queue via TMP



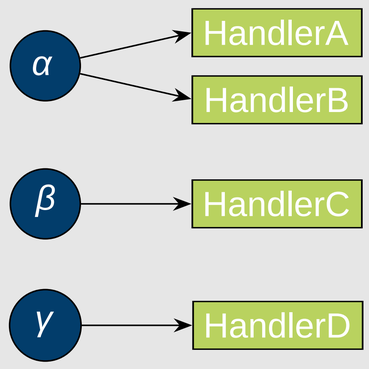

## Static Data-Flow Dispatcher




- Dispatch events to corresponding event handlers
- Template pack for collecting event listeners and handlers
- Dispatch method iterates over event listeners and handlers
- Filtering of events using sfinae
- Resolution of wiring between events and handlers at compile-time

```
template <typename searched_queue_t, int cur = 0>
static constexpr
    typename std::enable_if<cur != std::tuple_size<tuple_type>::value,
                            int>::type
    TuplePos() {
  // compare current type with wanted type
  return std::is_same<searched_queue_t,
                      typename std::tuple_element<cur, tuple_type>::
                          type>::value
             ? cur                                        // found
             : TuplePos<searched_queue_t, cur + 1>(); // try next
}
```

```
using DataFlowDispatcher = EventListenerContainer<
      EventListener<
          ALPHA,
          Handler< HandlerA >,
          Handler< HandlerB >>,
      EventListener<
          BETA,
          Handler< HandlerC >>,
      EventListener<
          GAMMA,
          Handler< HandlerD >>>;
```




## Results

2 x Intel Xeon Platinum 8170 (2 x 26 cores)