Object Sharing Demonstrator Plans

Paolo Calafiura

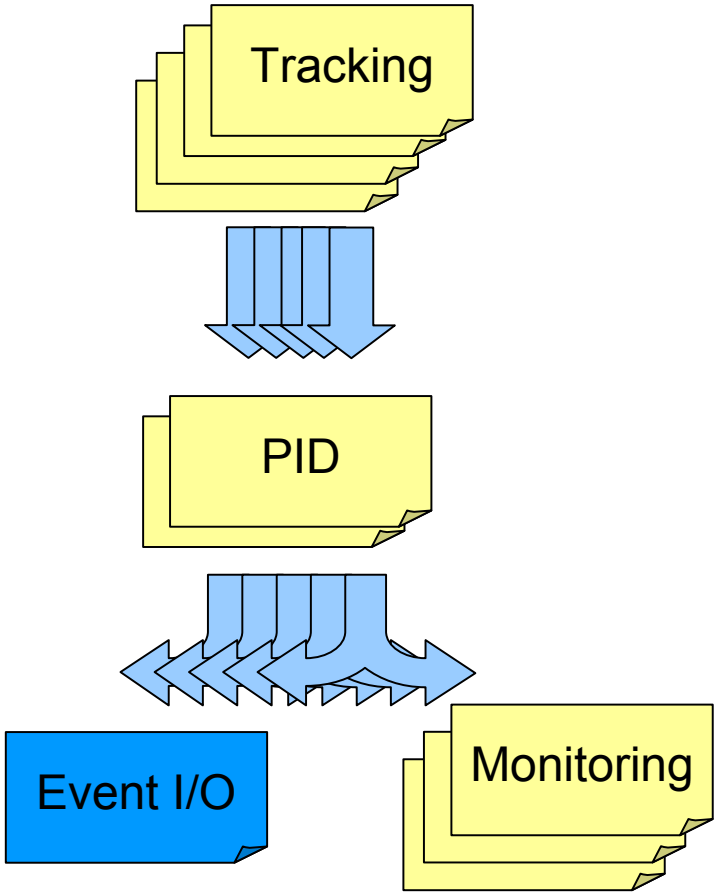
Motivation: Multi-processing Beyond Event Parallel

Manycore pushing us beyond event parallel

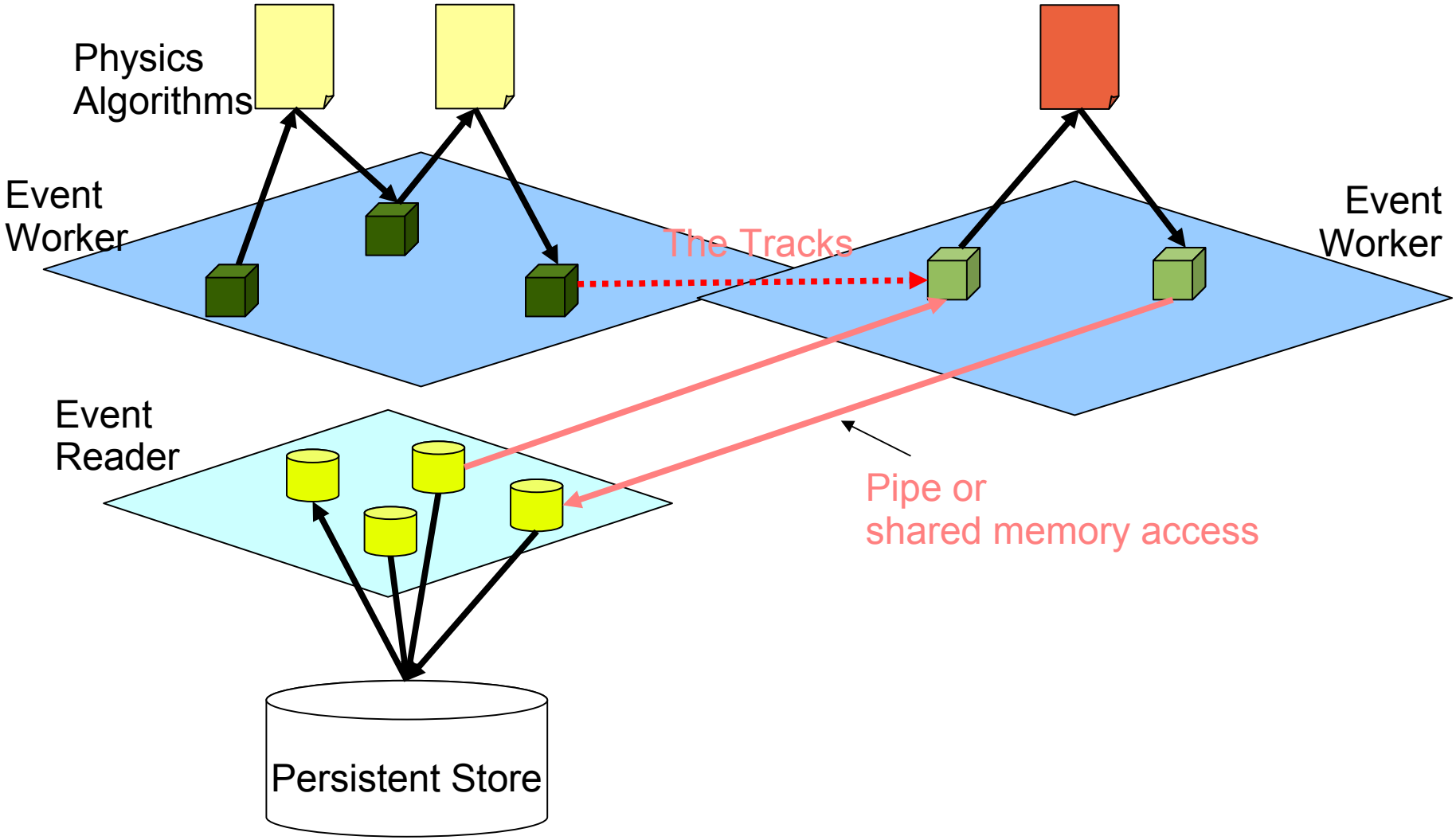
- Smaller processes
- Improved locality

Event I/O first use case

- ROOT dict and buffer sharing
- Better caching, less (un)zipping



Object Sharing will be Crucial



What do we want to do?

- Demonstrate sharing of C++ objects among processes
  - Issue:
    - In general objects created in one C++ process are not usable by another
      - vtable contents are process-specific
  - Two possible solutions so far
    - Stream objects as PODs using T/P conversion
      - See Peter Van Gemmeren talk at FNAL concurrency workshop
        https://indico.fnal.gov/getFile.py/access?contribId=14&sessionId=3&resId=1&m
    - Force linker/loader to make objects interchangeable
      - See Roberto Vitillo's talk at FNAL concurrency workshop
        https://indico.fnal.gov/getFile.py/access?contribId=21&sessionId=3&resId=0&m
- Demonstrate simple access synchronization scheme
  - Focus on event reader/worker/writer use case
    - Read-only objects (at least for now)
    - Low bandwidth o(MB/s/process)

Plans for the next six months

- Contrast the two sharing approaches
  - e.g. framework specific vs platform specific

    → determine best sharing approach for event I/O use case
- Investigate synchronization mechanism
  - e.g. Unix pipes, boost interprocess,...

    → determine best synchronization approach for event I/O
- Who will work on this?
  - Peter Van Gemmeren and LBL athena group
  - Input/help sought from
    - the “Whiteboard group”
    - C++ language/portability experts (on pitfalls of sharing live objects)
    - Scheduling/synchronization experts