Massively parallel, fine grained event processing on HPCs with the ATLAS Yoda system

Vakho Tsulaia (LBNL)

HPX Meeting
CERN, May 19, 2016

Event Service and Yoda @ CHEP2015

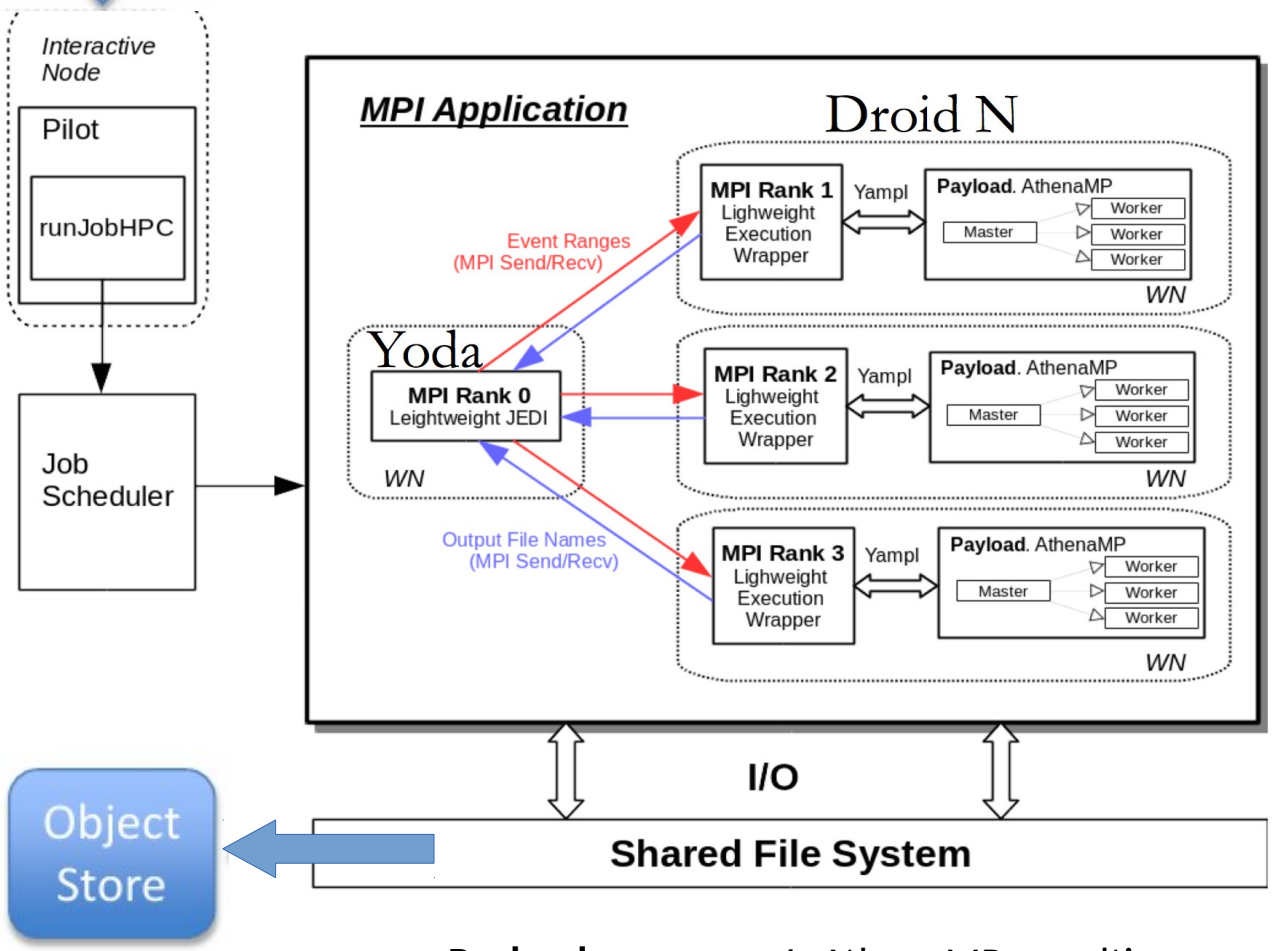
- "The ATLAS Event Service: A new approach to event processing"
  - http://iopscience.iop.org/article/10.1088/1742-6596/664/6/062065

- "Fine grained event processing on HPCs with the ATLAS Yoda system"
  - http://iopscience.iop.org/article/10.1088/1742-6596/664/9/092025

Event Service

- **ATLAS Event Service:** designed and implemented for efficient running of ATLAS production workflows on a variety of computing platforms
  - ✔ Conventional Grid sites
  - ✔ Opportunistic, often short-lived, resources: **Clouds, HPC, Volunteer Computing**
- Complex distributed heterogeneous system
  - ✔ **PanDA/JEDI** (distributed workload manager) delivers fine-grained workload to many compute nodes
  - ✔ **Payload application** on the compute node processes the events in chunks
    - ➔ Runs in "**daemon mode**": does its work when data becomes available
- Payload application uses all available CPU resources on the node (whole node)
  - ✔ **Multi-Proces** for now, **Multi-Threaded** in the future

Yoda. Schematic view



- **MPI application implementing master – slave architecture**
- **Rank 0 (Yoda, master).** Distributes workload between slave ranks
- **Fine grained workload:** individual events or event ranges
- **Rank N (Droid, slave).** Occupies entire compute node;
  Processes assigned workload;
  Saves outputs to the shared file system;
  Asks for the next workload …

- **Payload component:** AthenaMP – multi-process version of the ATLAS simulation, reconstruction and data analysis framework Athena

Yoda: Challenges

- Efficient delivery of the software to HPC compute nodes
  - A single ATLAS software release occupies ~10GB on the disk
  - Large number of configuration (and other) files accessed during the initialization step
  - The solution must allow efficient scaling for thousands of concurrent starts
- Efficient delivery of input data to compute nodes and efficient collection of output data from compute nodes
  - Current implementation works well for **CPU-intensive** workflows
  - May not scale for **I/O-intensive** tasks
  - Work is underway to implement **specialized I/O processes/threads**