Core SW
From LHC to Higgs Factories
Mostly about frameworks, but not only ...

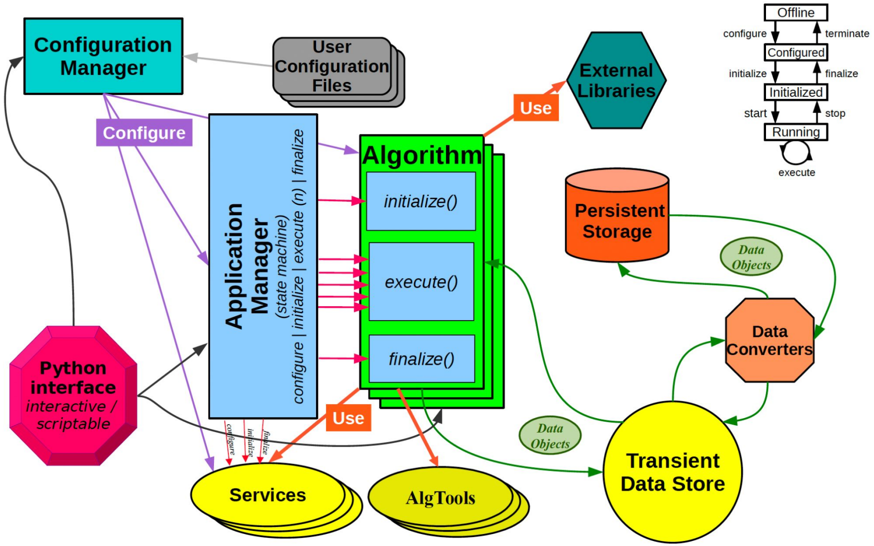
Paolo Calafiura, Charles Leggett and *Vakho Tsulaia*
LBNL

Physics Software and Computing Session
US Higgs Factory Planning Meeting
SLAC, December 19, 2024

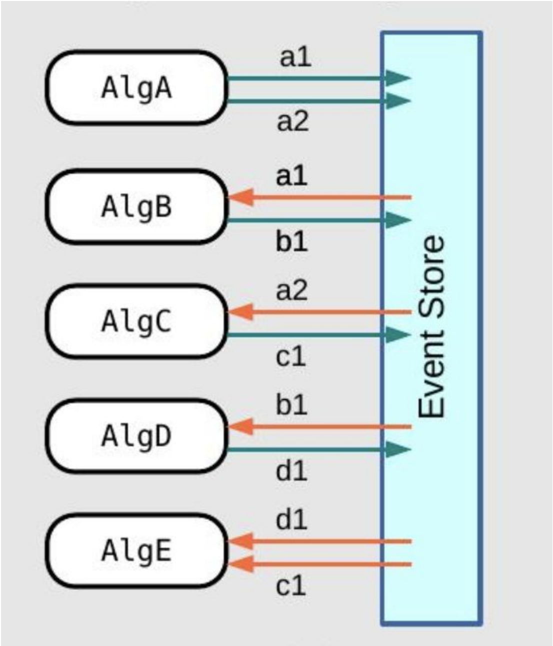
Part I
Gaudi: Experiment-Agnostic Software Framework

Gaudi Framework

- Has been developed since **1998**
- Component model of Gaudi:
  - Algorithm. Main building block of the *Event Loop*.
    Called once every event. Developed by experts in relevant domains (e.g., *Tracking*)
  - AlgTool. A plugin that helps an *Algorithm* to perform some action
  - Service. A plugin providing a common functionality to multiple components (e.g., *Detector Geometry*, *Calibration Data*, *Data Persistence*)
- State machine:
  configure / initialize / execute / finalize



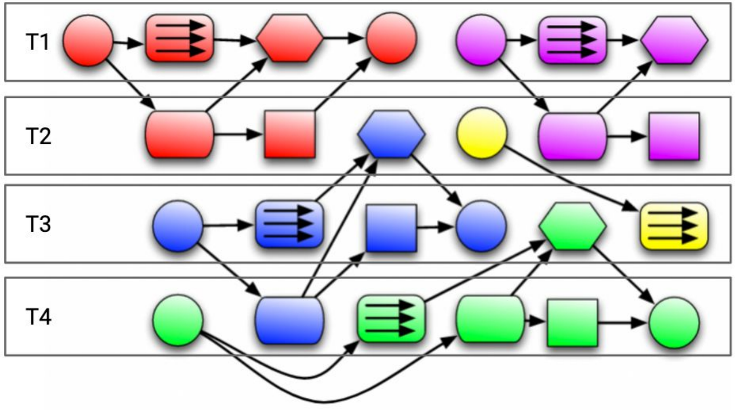
From Serial to Multithreaded Gaudi



- Gaudi was originally designed for event processing in a **single execution thread**
  - Algorithms are executed sequentially every event in the order defined at configuration time
  - Algorithms use Transient Event Store for reading and publishing data objects
- For the start of the LHC Run2 ATLAS switched to the process-parallel version of the framework: **AthenaMP**
  - Allows for sharing memory pages between event processors by leveraging Linux Copy-on-Write technology
- ~10 years ago to better utilise modern computing architectures the Gaudi team started the process of transitioning from serial to **multithreaded** framework
  - Originally named *GaudiHive*. Today is known as **GaudiMT**
  - ATLAS switched to the MT framework during the LHC LS2

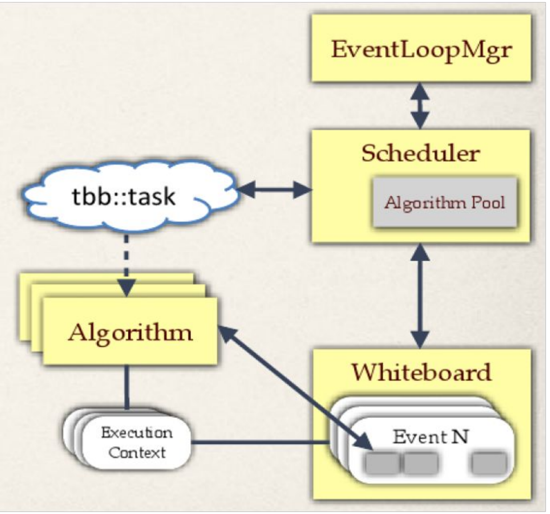
Execution of Algorithms in Multithreaded Gaudi

- GaudiMT uses Intel Threaded Building Blocks (TBB) for thread management
    - The TBB layer is hidden from Algorithm developers
- Each Algorithm executes in its own thread
- Different levels of thread safety in algorithms: *single copy, clonable, reentrant* (execute() can be called concurrently from multiple threads)
- Multiple Algorithms can concurrently work on the same event from multiple threads
    - Intra-event parallelism
- Multiple events can be executed concurrently
    - Inter-event parallelism



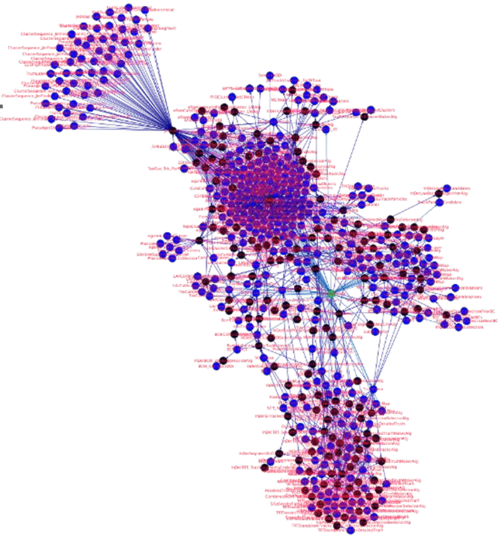
Event processing in GaudiMT:
- Each event is a different color
- Each shape is a different algorithm
- T1-T4: threads #1-4

Task Scheduler in Multithreaded Gaudi



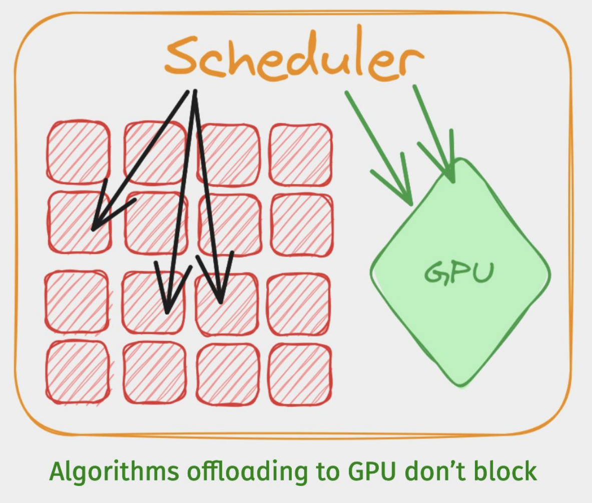
- Scheduler is a central framework component that orchestrates the execution of multithreaded applications
- The main responsibility of the scheduler is to efficiently assign algorithms to available threads from the thread pool, with the goal to optimize the overall event throughput
- *Developed and supported by the LBNL ATLAS Software Group*

ATLAS MC Reconstruction Precedence Rules (Data Flow)



- Algorithms
- Control-Flow Decision Hubs
- Data

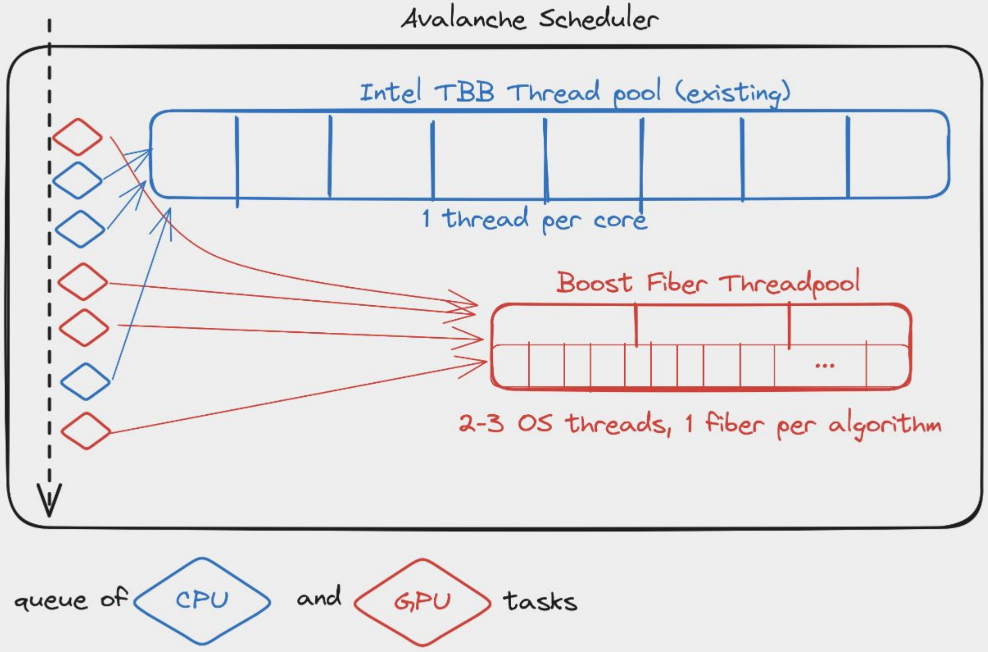
Gaudi in Heterogeneous Environment



Talk by ***Beojan Stanislaus*** (LBNL) at CHEP 2024

- New component: **Asynchronous Algorithm**
  - Algorithm with asynchronous *execute()* member function
- Implementation based on the usage of Boost Fiber
  - Fibers are lightweight user mode threads
  - Stackful coroutines + scheduler + convenience features
- Asynchronous Algorithms are scheduled on separate thread pool
- Algorithm yields after submitting work
  - OS thread freed for next fiber whenever you *yield*
- Comes with CUDA support …
  - Create CUDA streams; Use CUDA *Async* functions with streams
- … and easy to extend to other GPU languages (e.g., HIP, SYCL)

Heterogeneous Gaudi



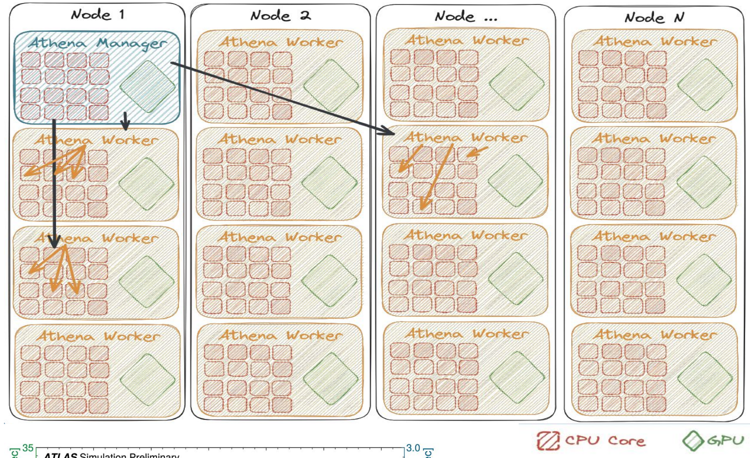
Synchronous → TBB   Asynchronous → Fiber

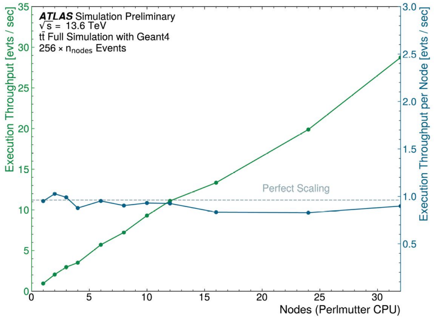
Beojan Stanislaus

Distributed applications

- Over the past several years the LBNL ATLAS software group has been exploring the mechanisms for extending experiment framework to allow running over multiple compute nodes simultaneously
    - Initial implementation based on the Ray distributed execution platform (***Raythena***)
- The latest version based on MPI (**AthenaMPI**)
    - MPI handles event scheduling across nodes
    - Within a node the scheduling is still handled by TBB
- Currently still a prototype
    - Getting ready to start Physics Validation runs
- Early tests demonstrated nearly ideal scaling of the event throughput with number of compute nodes

AthenaMPI





Advantages of using Gaudi

- Gaudi is an **experiment-agnostic** software framework
  - Designed for collision-based experiments with a concept of an “event” (**ATLAS**, **LHCb**)
  - Extended to use moving “time windows” for experiments that record data continuously (**LZ**, **DayaBay**)
  - It is also now an experiment framework component of the **Key4hep** software stack
- The **component model** of Gaudi allows for building experiment-specific extensions on top of core functionality of the framework
  - E.g., ATLAS-specific extensions built in the Athena framework
- Gaudi components can be configured at runtime through the Python-based configuration layer developed by the LBNL group
  - Allows for setting values to component properties and building relationships between components (e.g., assigning private *AlgTools* to *Algorithms*)

Advantages of using Gaudi (contd.)

- The functionality of Gaudi is currently being extended to support running in **heterogeneous environments** (*Asynchronous Algorithms*)
  - Non-blocking offloading of computations to accelerators

- Support for **Python Algorithms** in Gaudi has been available for many years

- Some recent exploratory work on how one can leverage interoperability of C++ with other programming languages (e.g., **Rust**) in the framework
  - See the talk by Marco Clemencic at CHEP 2024
  - Similar plans in DUNE: C++ to Python and Julia (the project led by the LBNL group)

Evolution of Gaudi over the next 20 years?

- A speculative look in the future listing some possible directions based on current trends in software development
    - The list was generated with the help of our friendly ChatGPT

- Increased modularity and flexibility
- Enhanced performance
- AI and ML integration
- Improved usability and user experience
- Interoperability with other scientific frameworks
- Cloud and distributed computing
- Security and data privacy
- Sustainability and energy efficiency
- Community and collaboration
- Support for new experiments and technologies

Gaudi at LBNL

- The LBNL ATLAS Software Group has been actively involved in Gaudi core development and support since year 2000
  - Proceeding for CHEP 2001
- Recently, the members of our group made critical contributions to the transition of Gaudi from serial to multithreaded operation mode
  - E.g., Gaudi Avalanche Scheduler was developed at LBNL
- Our group is a driving force behind the implementation of new mechanisms for supporting heterogeneous platforms (*Asynchronous Algorithms*)
- We plan to continue working on critical projects for extending and supporting Gaudi framework in the future

Part II
GeoModel: Detector Description Software Toolkit

GeoModel in One Slide

- **GeoModel** is a toolkit for describing geometries of large and complex detector systems with **minimal memory footprint**
- GeoModel has been used by the ATLAS experiment **since 2004**
  - Single source of the ATLAS geometry description for *Simulation, Digitization, Reconstruction, etc.*
  - Battle-tested by ATLAS for ~20 years
- In 2019 GeoModel was repackaged as an **independent, experiment-agnostic API** with lightweight external dependencies
- **Kernel library** with classes for building **material geometry trees** (*physical and logical volumes, solids, identifiers, materials, transformations*), tools for **building readout geometries** and for **applying alignment corrections** (supports multiple alignments in fly for multithreaded running)
- **Tool suite**: *Visualization, Standalone Detector Simulation, Clash Detection, Mass Calculation, Conversion to/from GDML, etc.*
- Developed and supported by **US ATLAS** teams: *University of Pittsburgh and LBNL*

ATLAS Geometry described in GeoModel