Gaudi and ROOT in LHCb Offline

- Gaudi: Data processing framework used by LHCb
- ROOT is a core component:
  - All files in ROOT format except for RAW data
  - ~7PB of ROOT production files on tape
- Successful but facing challenges:
  - ROOT6 integration
  - Optimization/vectorization
  - Many core machines...
The Gaudi Framework

Event/Data processing framework used by LHCb, ATLAS, and other non-LHC experiments
Used by both LHCb Offline and Online

• Design principles:
  - Clear separation of Algorithms and Data
  - Clear interfaces between User code and framework
  - Separation between transient and persistent data

• Single process application
  - C++ core
  - Run/Configured from Python from greater flexibility

• Relies on ROOT for:
  - Persistency with ROOT I/O, ROOT trees
  - Reflection via Reflex library
  - Reflex plugin manager for Gaudi Plugins
  - Histograms, Mathematical libraries
**ROOT I/O**

- One single converter in Gaudi Framework
  - Used to use POOL (itself serializing to ROOT files)
  - Direct ROOT API used since 2011

- Data loaded using ROOT plugins
  - Use of Xrootd in production

- Reflection necessary to store LHCb Data in ROOT files...
Gaudi and reflection

- Gaudi uses Reflex dictionaries
  - `genreflex` used at build time
  - `cintex` needed to use the dictionaries

- GaudiPython uses PyCintex/PyROOT
Reflex plugin system

- Reflex also provides a plugin system
  - Gaudi algorithms/Tools defined as Reflex plugins
  - `genmap` used to generate the rootmap file
  - Reflex uses ROOT to load the plugins

- This will need reviewing for ROOT6...
ROOT 6 Impact

- Reflex disappears
  - genreflex (or equivalent) needed for migration
  - Would rather move to the native API

- The need for cintex/pycintex disappears...
  - PyROOT should be enough...

- Simpler architecture
- Eager to start testing as soon as possible
- Need to find alternative for Reflex plugin system
- Deadline imposed by LHC restart
Optimization and code vectorization

• All experiments need to maximize resource usage
  – LHCb code base underuses the SIMD registers
• Code review is starting... we need to identify the bottlenecks
  – Math primitives are the ideal level to tackle the problem
  – Many ROOT::Math primitives used...
    • SMatrix
    • SVector
    • Vector3D/4D
    • CholeskyDecomp
    • ...

➢ Improved Vectorization ?
The future of Gaudi

- LHCb jobs need several GB of Memory
  - ~1.5 GB for Reconstruction
  - More than 2GB for Stripping...

- Current productions will not run on many core machines...

- Many efforts to solve this issue:
  - GaudiMP
  - GaudiHive
Near future: Gaudi MP

- Multi Process Gaudi (P.Mato, E.Smith)
  - Multiprocessing in Python
  - Shows benefits in memory usage (late forking)

- Ready for use by LHCb
  - C.f. Work by N.Rauschmayr

- Dedicated Writer process
  - Could investigate the Parallel Merger
Longer term: Gaudi Hive

• Integration with Whiteboard developed by PH/SFT
  – Multithreaded event processing Gaudi
  – Uses Intel Threading Building Blocks (TBB)

• Potential Issue for Parallel I/O and Zipping
  – All I/O currently done in a single thread
  – Thread safety required
    • Without affecting (too much...) performance

• Moving towards task based programming
  – We need ROOT to integrate with TBB
Conclusion

- ROOT 6 is an opportunity to simplify the system
  - BUT with the reflection system, the devil is in the details...
  - Amplitude of the changes still to be determined
  - Gaudi/ROOT6 needs to be in place by the beginning of 2014...

- The challenge is to improve performance ...
  - Vectorization of Mathematical primitives?

- And to deal with multi-threaded frameworks
  - Need to integrate with new task schedulers
  - Thread safety of operations required