Running Concurrent Gaudi in Real Life: Status Update on MiniBrunel

B. Hegner for the Concurrent Gaudi Team

ATLAS S&C Workshop 13-6-2013

Concurrency at What Level?

- It is obvious that we have to "go parallel" rather sooner than later
- How do we do that in concrete?
- Multiple jobs
  - Huge memory consumption
  - Job and output file management a problem
  - Huge number of resources needed (open files, DB connections, …)
- Multiple processes
  - Helps on memory consumption
  - File merging a problem
  - Number of required resources not addressed
- Concurrent framework
  - Helps greatly on memory consumption
  - Reduces number of required resources
  - Allows concurrent handling of multiple events
    - **Pre-requisite for offloading to heterogenous resources**
  - More challenging software wise !

Basic Principle

- Our work is already split into smaller tasks (a.k.a. algorithms)
- Task execution is in theory constrained by two concepts

- **Data Flow**
  - Algorithms depend on data products other algorithms can produce
  - E.g. electron reconstruction requires ecal clusters
- **Control Flow**
  - Conditional execution of algorithms or sequences thereof
  - Trigger as prime example

Resolve these dependencies automatically.

Run everything in parallel that isn't constrained by control flow or data flow.

Our Goal

Provide refurbished Gaudi framework which supports

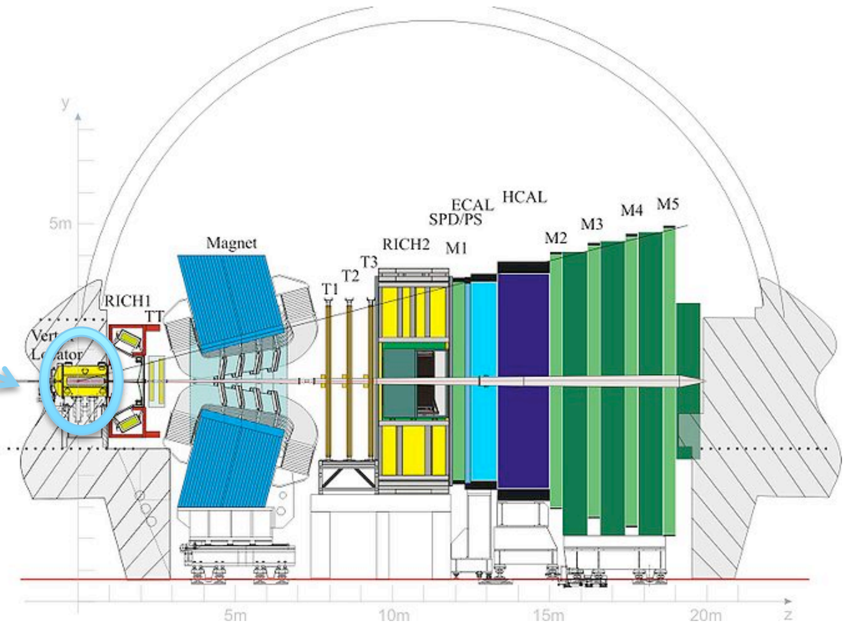
1) Concurrent execution of algorithms

2) Simultaneous processing of multiple events

Pragmatic approach: start from slice of real LHCb reconstruction workflow (called MiniBrunel in the following)

- ~20 algorithms and associated tools: raw decoding and Velo tracking

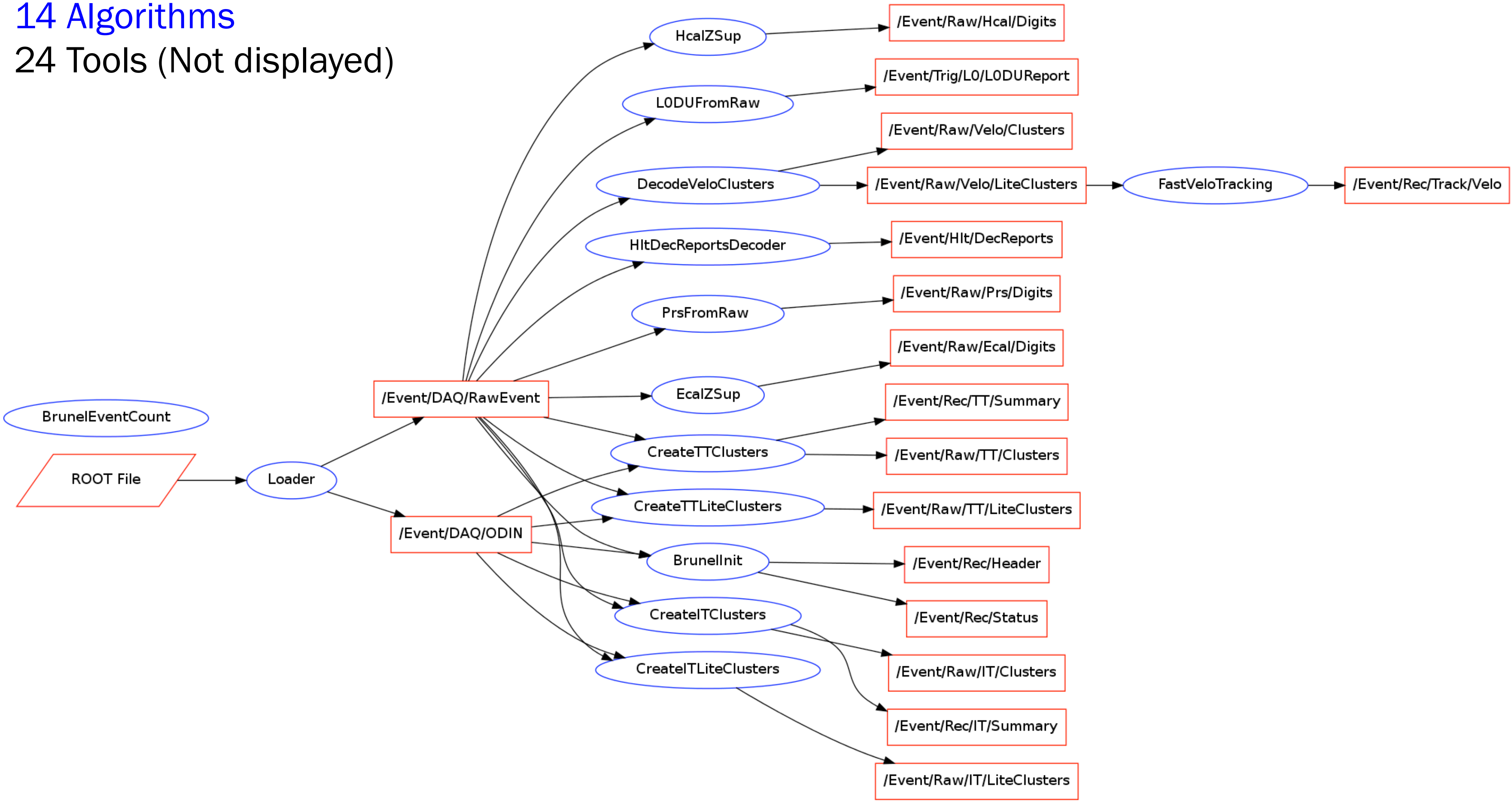
MiniBrunel span within the detector



See backup for useful links about the project.

MiniBrunel: Data Dependencies

14 Algorithms
24 Tools (Not displayed)



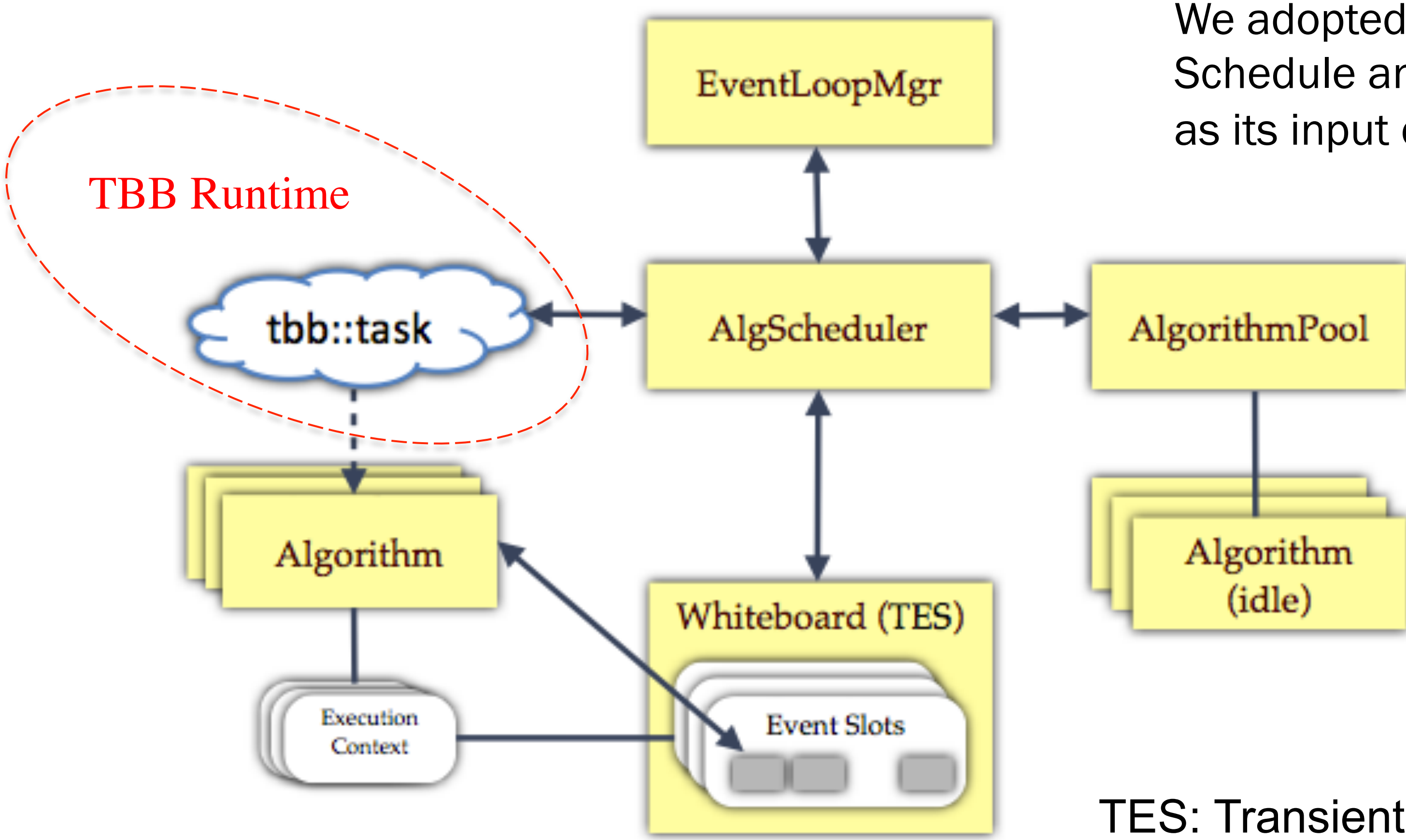
Control flow dependencies not displayed

General Strategy

- Classify and document issues encountered during this effort
  - Build a “matrix of costs” - assess the size of the effort that would be required to migrate the complete LHCb stack
- Identify solutions and migration strategies
  - Not only thread safety: assumptions valid in the serial case are broken
  - Get experience on existing large codebase
  - Aim for minimal changes of interfaces
  - Provide new components compatible with present design
- Timescale: end of June (a.k.a. internal “0.5 Release”)

Components Overview

- New components added to Gaudi to support concurrency
  - E.g. Scheduler, Whiteboard, AlgPool
- Existing components upgraded
  - E.g. ToolSvc, EventLoopMgr



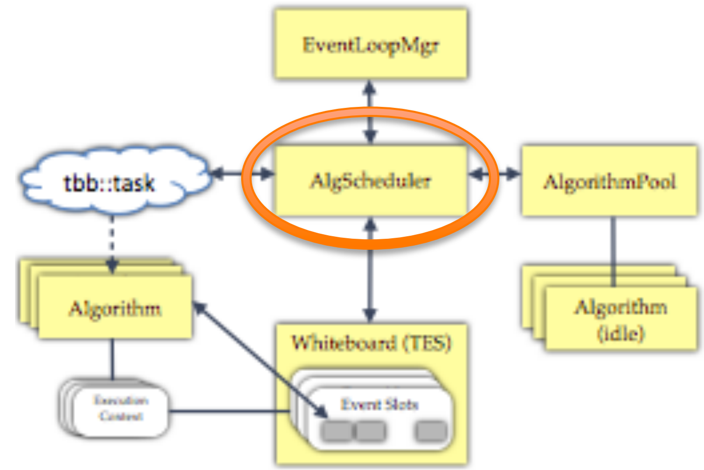
We adopted forward scheduling: Schedule an algorithm as soon as its input data are available

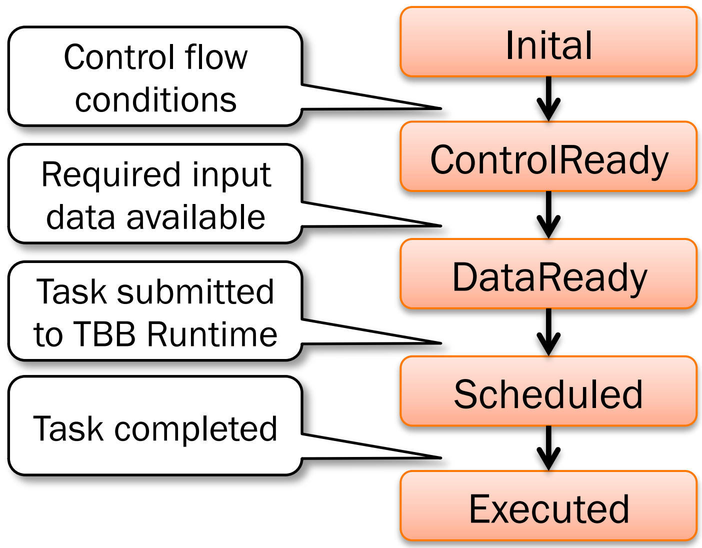
TES: Transient Event Store

The Forward Scheduler

Keeps the state of each algorithm for each event

- Simple finite state machine
- Receives new events from loop manager
- Interrogates whiteboard for new DataObjects
- Pulls algorithms from AlgorithmPool if they are available
- Encapsulate them in a tbb::task for execution
- Absorbs asynchronous events (e.g. arrival of finished tasks) with a thread safe queue of lambda closures (*actions*). Same pattern used for new message svc.





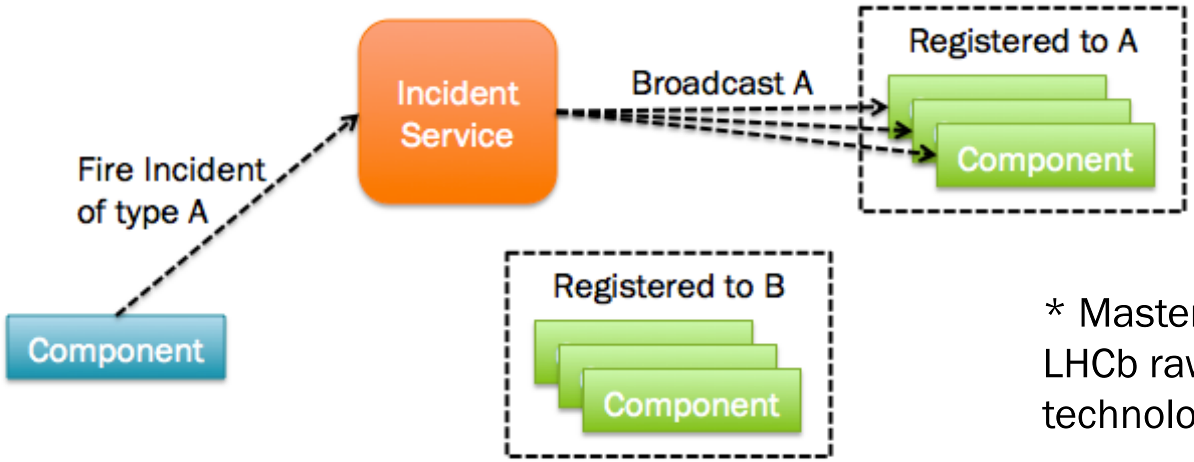
See Backup for more details!

Other Code Changes: Executive Summary

- **Algorithm dependencies**
  - Data dependencies: announced by the algorithms themselves
- **Tools**
  - A few tools served as back-door communication channels bypassing the official (event data) channel
- **Incidents**
  - Meaning of many global incidents radically changed (e.g. BeginEvent)
- **MDF* Conversion**
  - Support multiple events in flight

See Concurrency Forum meeting on April the 24th
https://indico.cern.ch/conferenceDisplay.py?confId=248560

An incident:



* Master Data Format
LHCb raw banks persistency technology

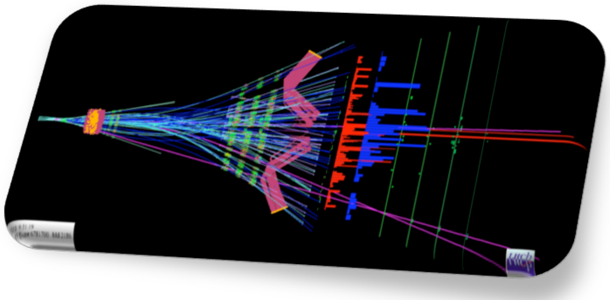
So where are we now?

“The Real Thing”

Concurrent execution of Minibrunel works!

- Real algorithms running on real data producing real plots
  - January 2013 software stack, 2011 collision raw data
- Tested with various scenarios
  - Different number of events in flight
  - Several algorithms in parallel
- Assumption for this prototype:

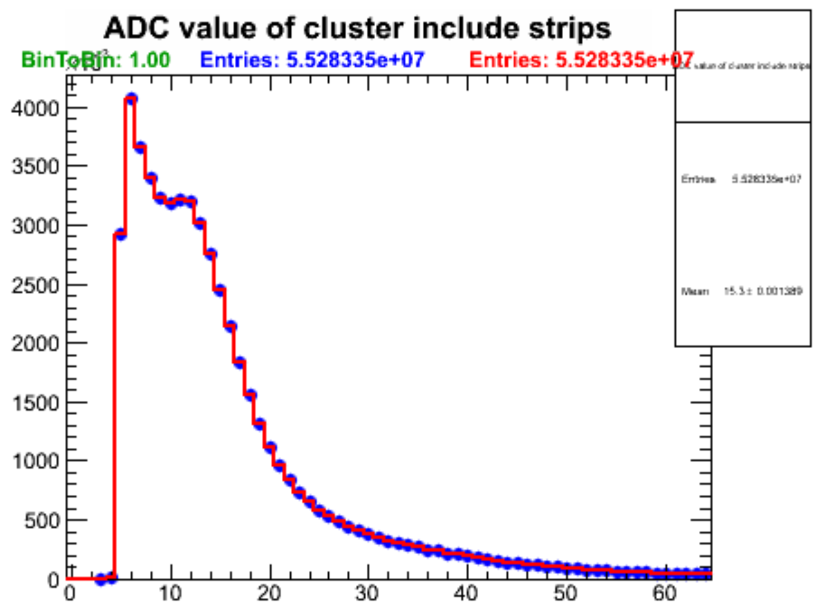
  no change of detector conditions during run



Output Validation

- Only successfully tested software is working software
- Our test case: LHCb standard set of data quality monitoring histograms
- Necessary but not sufficient to guarantee production quality results
- Check histograms for serial and concurrent version (high number of simultaneous events and algorithms)

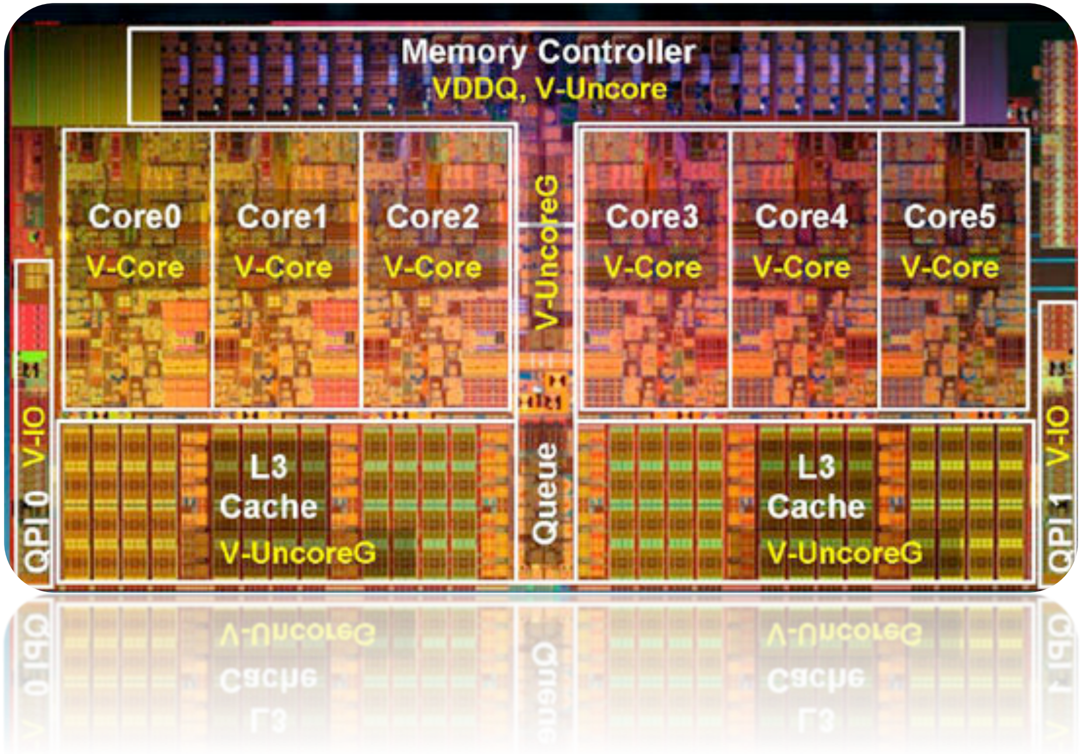
Example of data monitoring histogram: ADC counts.



All standard histograms identical bin by bin

Measuring Performance

- The Testbed
  - 10k events (60k for the physics performance estimation)
  - SLC6, gcc46
  - TCMalloc
  - Xeon L5640 @2.27 GHz
  - 2 sockets 6+6 HT Cores each (Westmere)



Runtime Overhead

There is an overhead when using new components designed for concurrency when limiting to one worker thread only (as ~expected)

Timing for the event loop only (no initialisation/finalisation):

Serial Gaudi (no new components) ….. 72.9 s

Concurrent Gaudi 1 evt in flight ……... 97.7 s

Concurrent Gaudi 2 evts in flight ……. 73.9 s

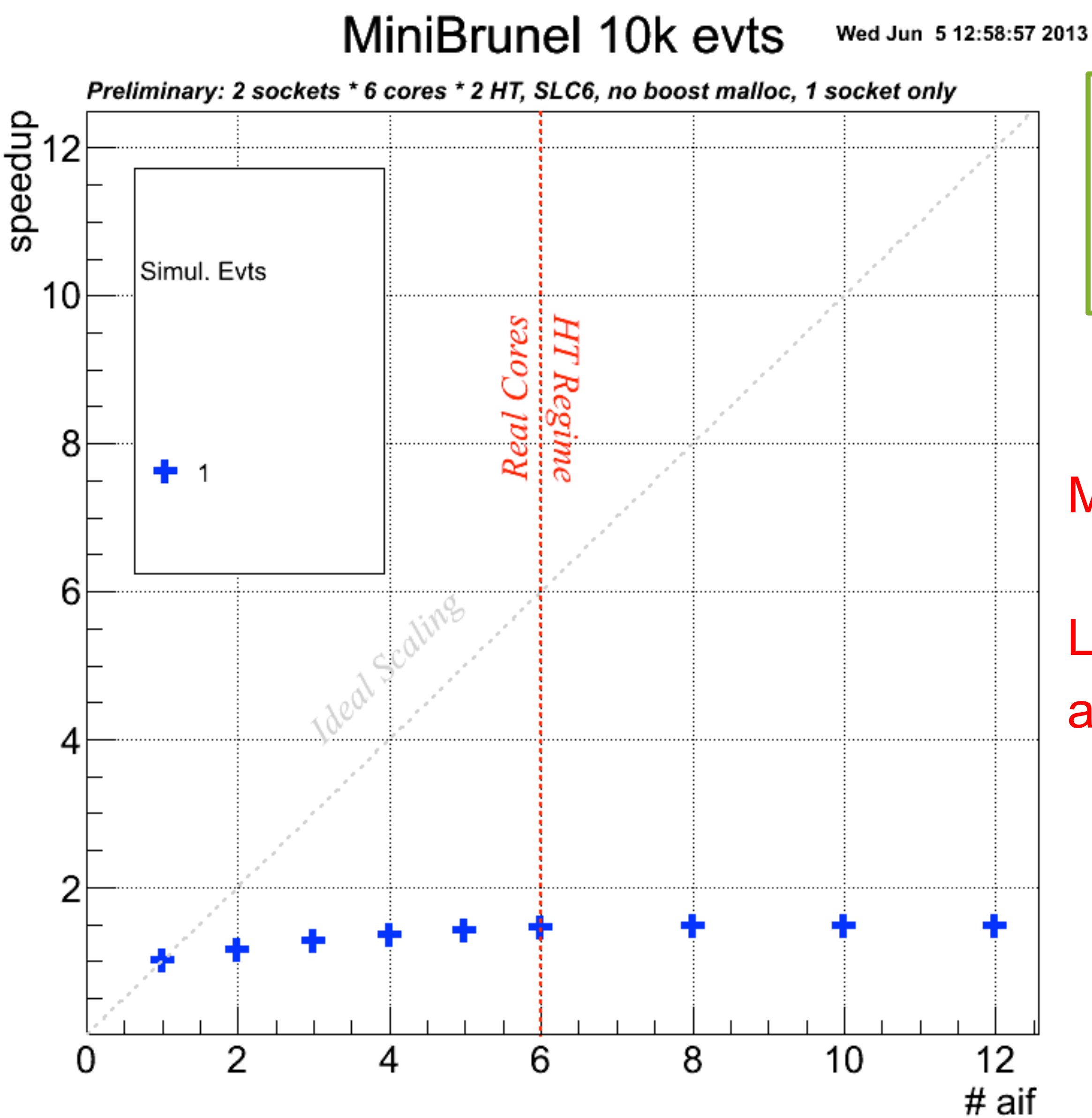
Concurrent Gaudi 10 evts in flight …... 72.3 s

1 algorithm running at the time

Frequency of task queue updates is too small to keep worker thread busy with only one event in flight

2 events in flight: enough to get rid of 'starvation'

Scaling on One Processor

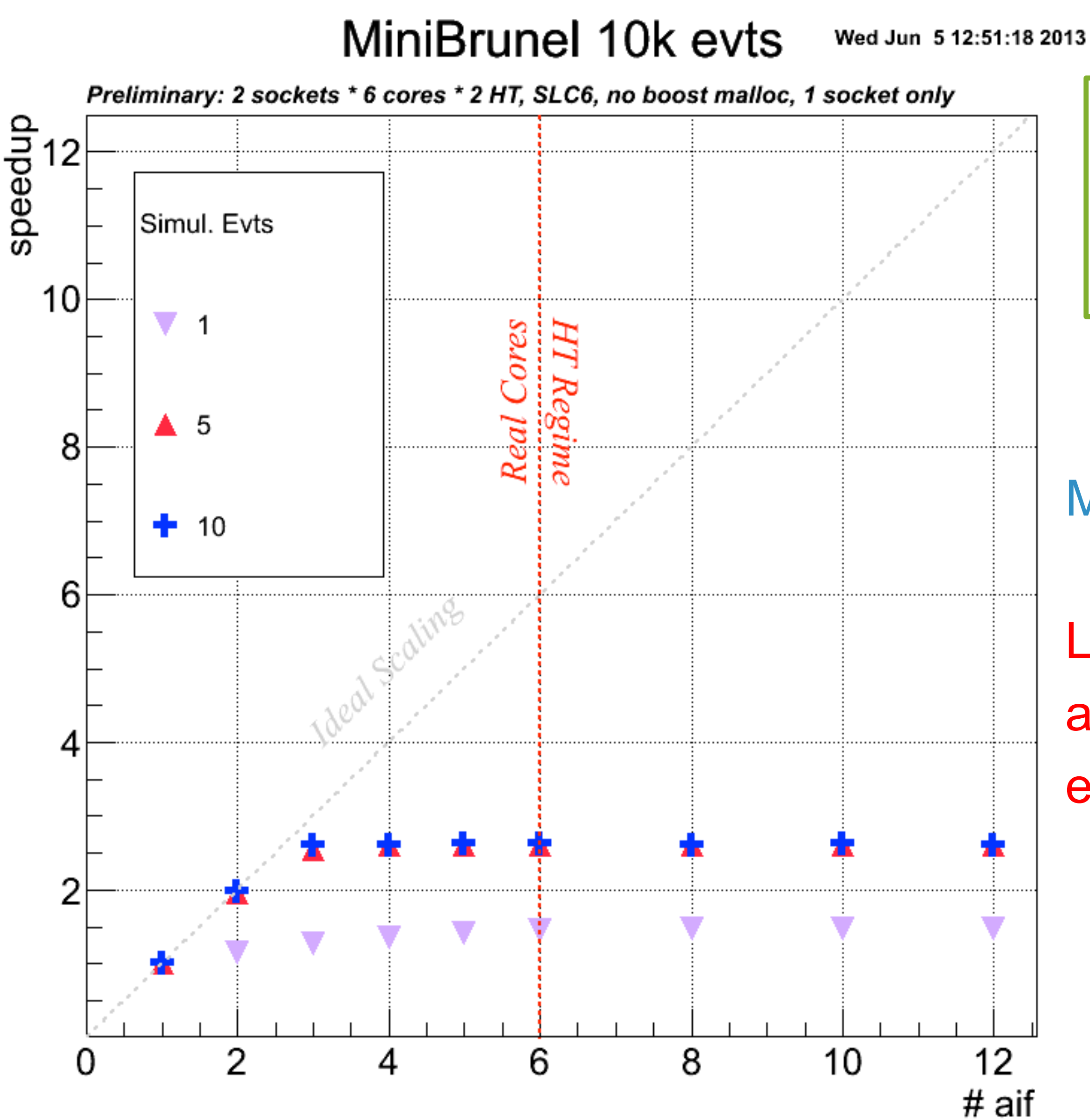


N algorithms simultaneously

Maximum Speedup: ~30%

Limited by **critical path** in algorithm dependencies

Scaling on One Processor

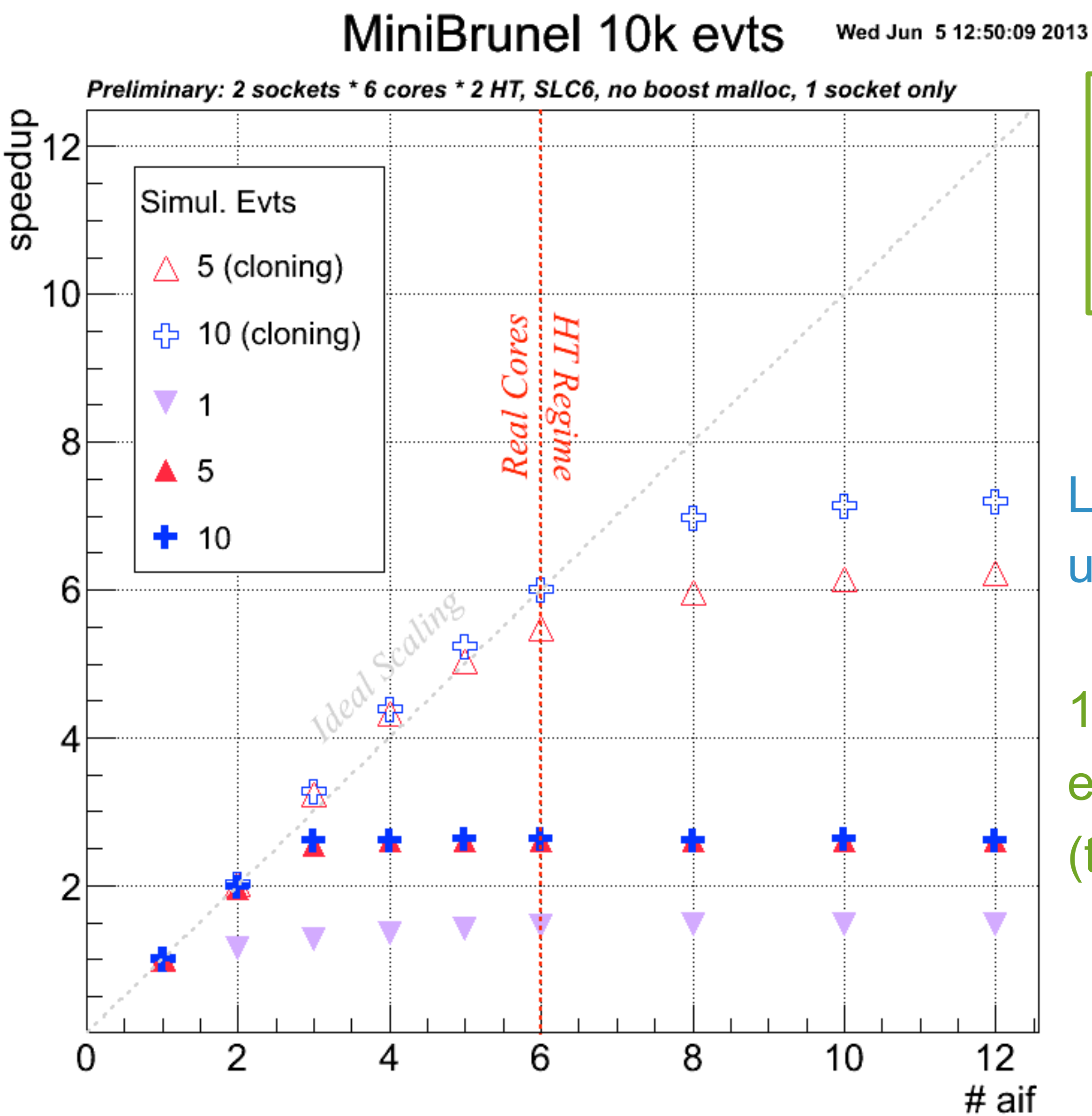


Multiple events in flight
N algorithms simultaneously

Maximum Speedup: 2.5x

Limited by **availability** of algorithm instances for execution

Scaling on One Processor



Multiple events in flight
Clone 3 most time consuming algs (1 copy per event in flight)

Linear scaling of speedup up to number of physical cores

10 events in flight already enough for peak performance* (thanks to HT)

* See backup for a complete study

Memory Overhead

Running mode:

- 1 clone per event in flight of 3 longest running algorithms
- Full TBB thread pool (24 threads)
- Limit algorithms in flight to 6

Resident Set Size at the end of the event loop (no finalisation):

Serial Gaudi (no new components) ….. 478 MB
Concurrent Gaudi 1 evt in flight ……... 480 MB
Concurrent Gaudi 2 evts in flight ……. 485 MB
Concurrent Gaudi 10 evts in flight …... 514 MB

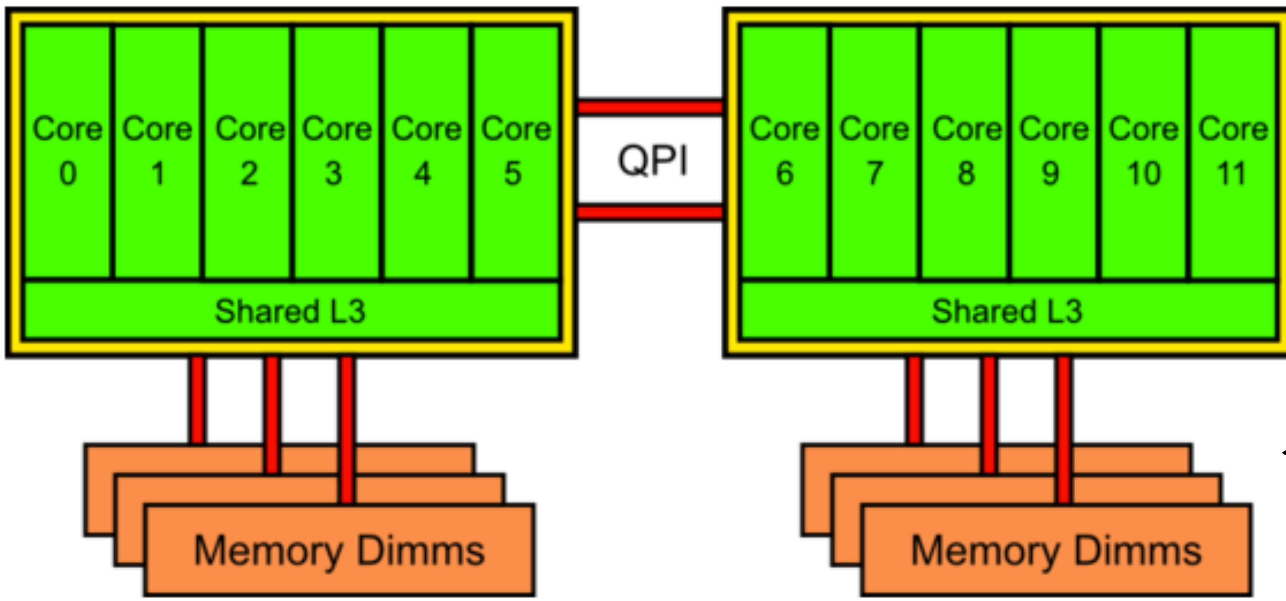
Note: Not full LHCb events but Minibrunel events.

6 algorithms running simultaneously

Memory: multithreaded solution is cheap!

Scaling on Two Sockets

- Behaviour of the application on a full NUMA* node is not trivial
  - E.g.: remote DRAM access, cross-socket caches synchronisation…



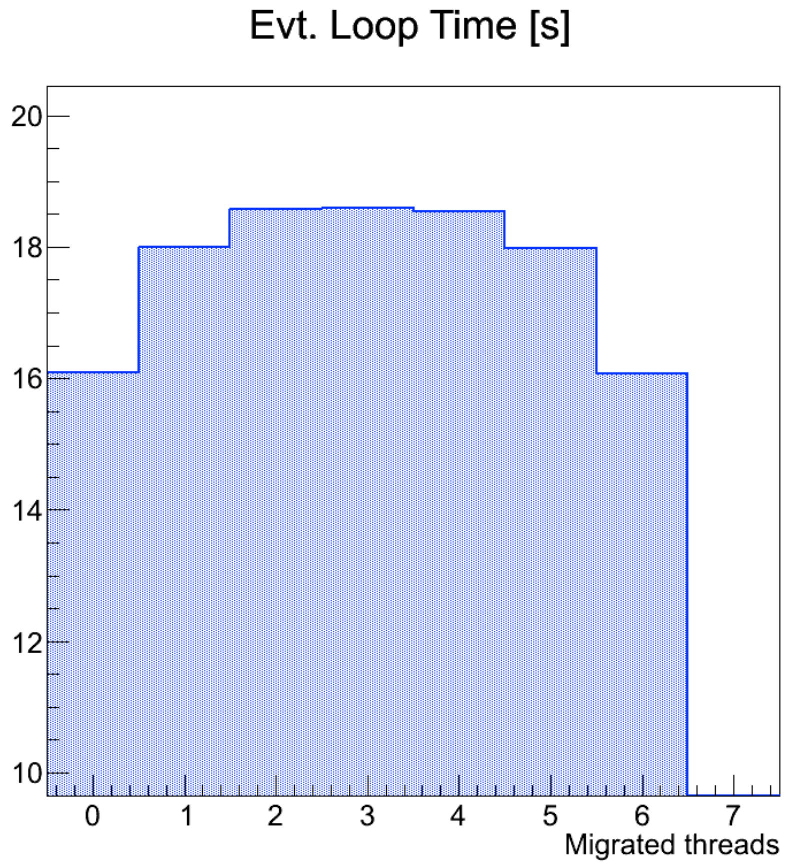
Very Simplified!

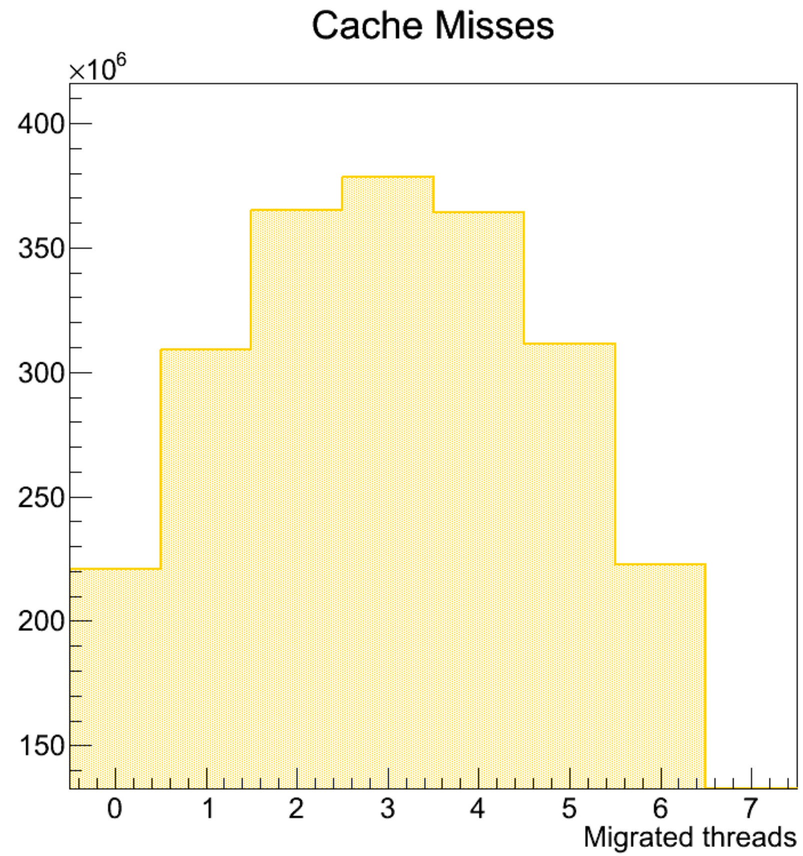
* NUMA = Non-Uniform Memory Access

Numa Effects

Thanks to Vincenzo I. for the fruitful discussions!!

- Run with 10 events in flight and 6 threads
- Use the "taskset" command to assign cpus to a process
- Start with 6 cpus on one socket, move them one by one to the other
- Measure event loop time and use perf to count cache misses





See backup for nice measuerements of uncore events!

Scaling on Two Sockets

- We don't understand completely the behaviour of the application (performance degradation) yet.

But using the full numa node with 2 sockets is not the only possible deployment scenario!

- Runtime of one full-socket job alone on the machine and two simultaneous one-socket jobs was verified to be identical.

Along the lines of the "one job per cpu" philosophy behind our data processing since years, but with *much* less memory (even HT cores usable!)

One job per socket deployment scenario: successful

Conclusions

A concurrent framework is possible and worthwhile

- Supporting concurrency at all levels

All developments necessary for the Minibrunel exercise finished

- Framework: components for MT execution (Scheduler, EventLoopManager) and integration with TBB runtime
- Usercode: input declaration, thread-safety fixes, compatibility with >1 event simultaneously processed

Outcome of real-world test very successful

- Serial and concurrent Minibrunel yield identical physics output
- Concurrent MiniBrunel scales linearly on a single die (on the test machines available)
- Negligible increase of memory consumption

Thanks to the LHCb core software team for the support!

NUMA to be tackled as one of the next items

- But one job per socket solution successfully tested!

What next?

We did a lot, but there is quite some work ahead!

- Consolidate code and documentation for **beta release at the end of June**
- Afterwards: supporting condition changes, more instrumentation, …

Support ATLAS in setting up a reconstruction slice with concurrent Gaudi

- Dedicated sprint next week

Collect requirements from experiments

- In close contact with LHCb experts
- We are participating in the *FFReq* work group (Future Framework Requirements)

To summarize:
We are preparing at full steam for the future

Backup

Useful Material

Project Page on the Concurrency Forum Site:

http://concurrency.web.cern.ch/GaudiHive

Main Twikipage:

https://twiki.cern.ch/twiki/bin/view/C4Hep

Git Repository Web Interface:
http://lcgapp.cern.ch/git/GaudiMT/

Jira:

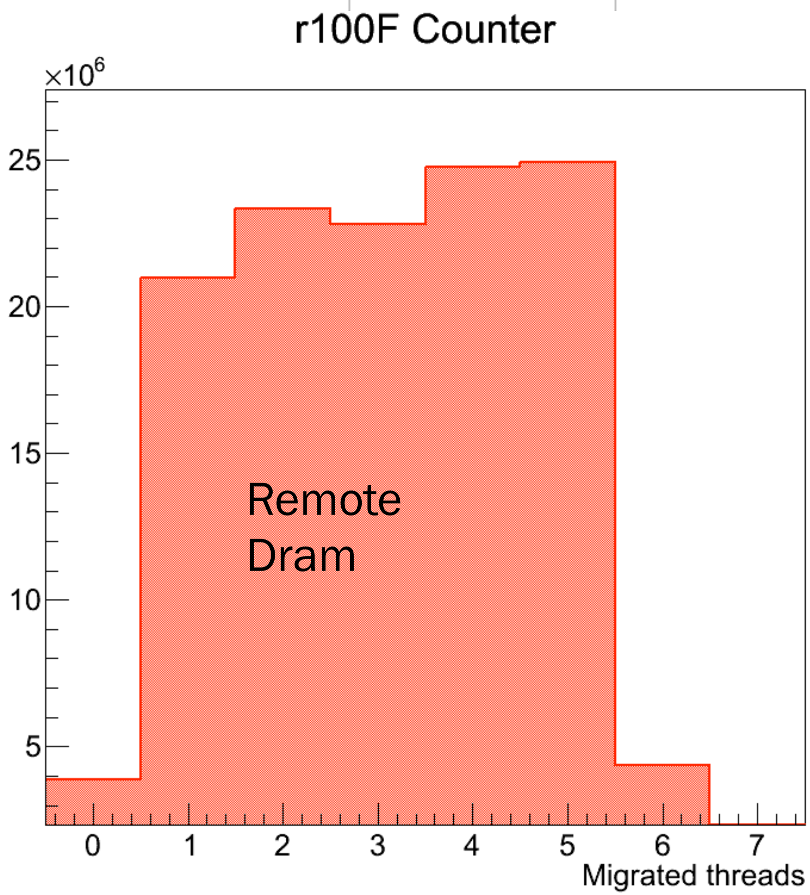
https://sft.its.cern.ch/jira/browse/CFHEP

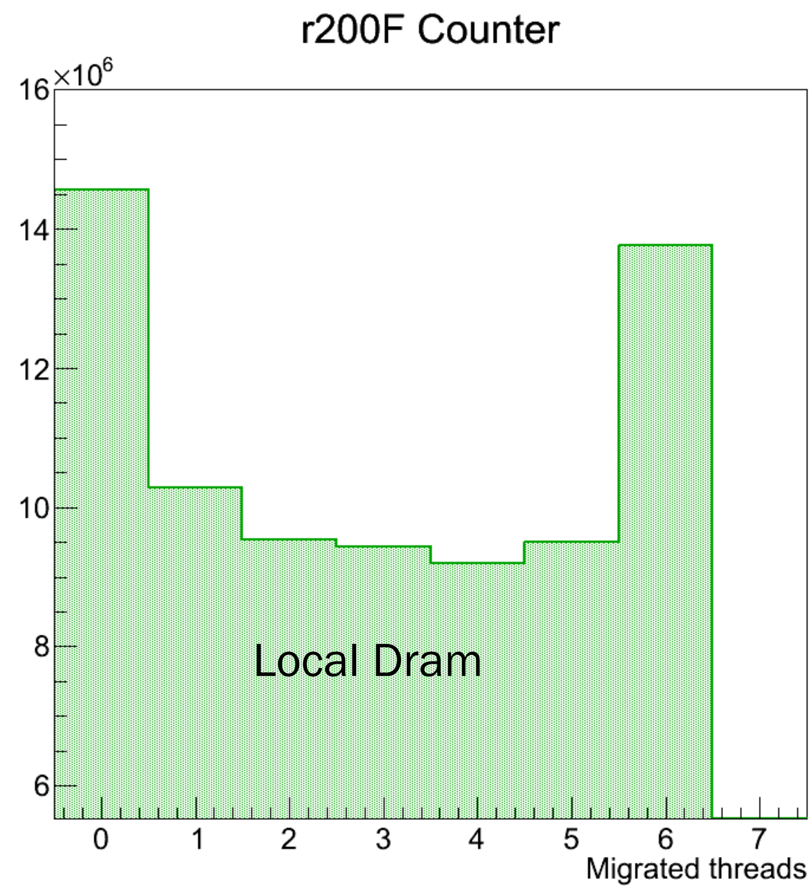
Weekly (Thursday 10:30 a.m., with phoneconf) Working Meeting Minutes:

http://sync.in/k5XvRql9y9

Performance Counter Analysis

REMOTE_DRAM	0x10	Load instructions retired remote DRAM and remote home-remote cache HITM (Precise Event)	Counts number of memory load instructions retired where the memory reference missed the L1, L2 and LLC caches and was remotely homed. This includes both DRAM access and HITM in a remote socket's cache for remotely homed lines.
LOCAL_DRAM	0x20	Load instructions retired with a data source of local DRAM or locally homed remote hitm (Precise Event)	Counts number of memory load instructions retired where the memory reference missed the L1, L2 and LLC caches and required a local socket memory reference. This includes locally homed cachelines that were in a modified state in another socket.





http://software.intel.com/sites/products/documentation/doclib/iss/2013/amplifier/lin/ug_docs/reference/pmbk/events/mem_uncore_retired.html

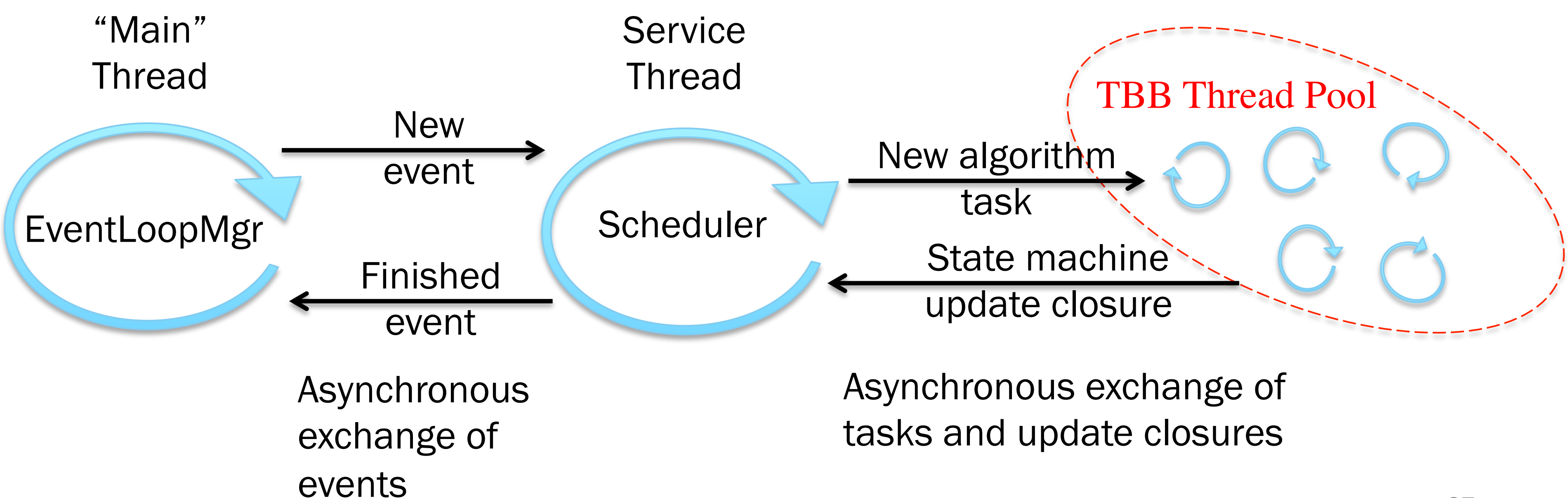
Service Threads

An additional "service" thread (outside the tbb pool, which contains "worker" threads) is spawned:

- Host the scheduler method to update the state machine when an algorithm has run. If no work is available, it sleeps.

The "main" thread manages the event loop ("little more than an event factory"). While the scheduler processes the events, it sleeps.

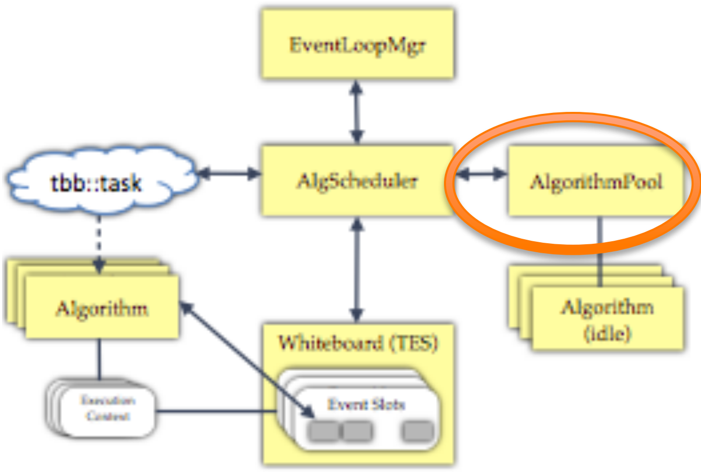
Other service threads existed and continue to exist (e.g. conditions watchdogs)



Algorithm Pool

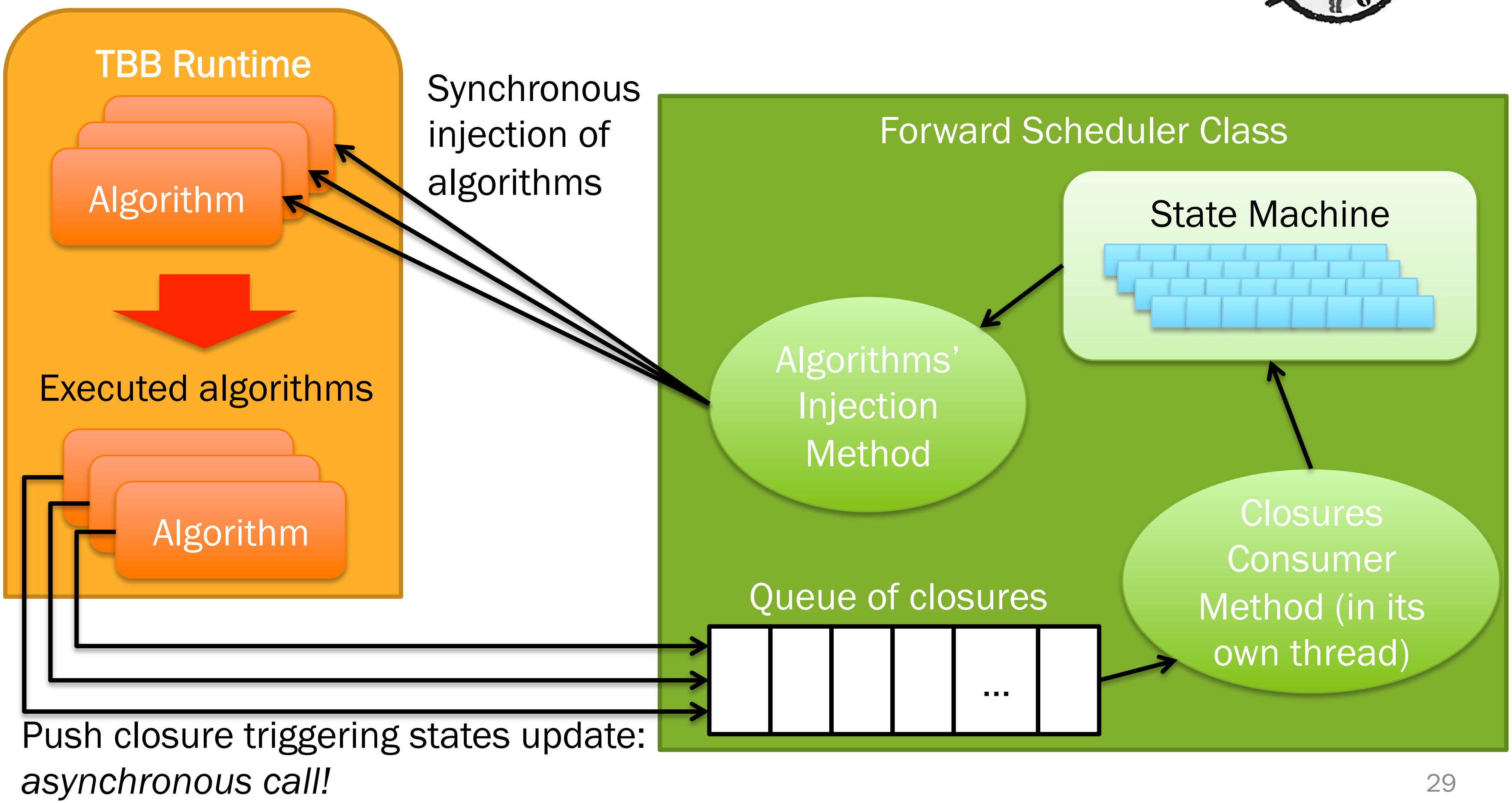
Contains algorithms and coordinate them

- Gives away instances to run, retrieves ran algorithms
- Clones algorithms (via AlgManager)
  - Number depends on code re-entrancy: non re-entrant (1 copy only), non re-entrant (use n copies), fully re-entrant (re-use same instance n times)
- “Flattens” sequencers
- Allow for exclusive resource checking: e.g. if 2 algos using a non re-entrant external library, only one at the time can run.



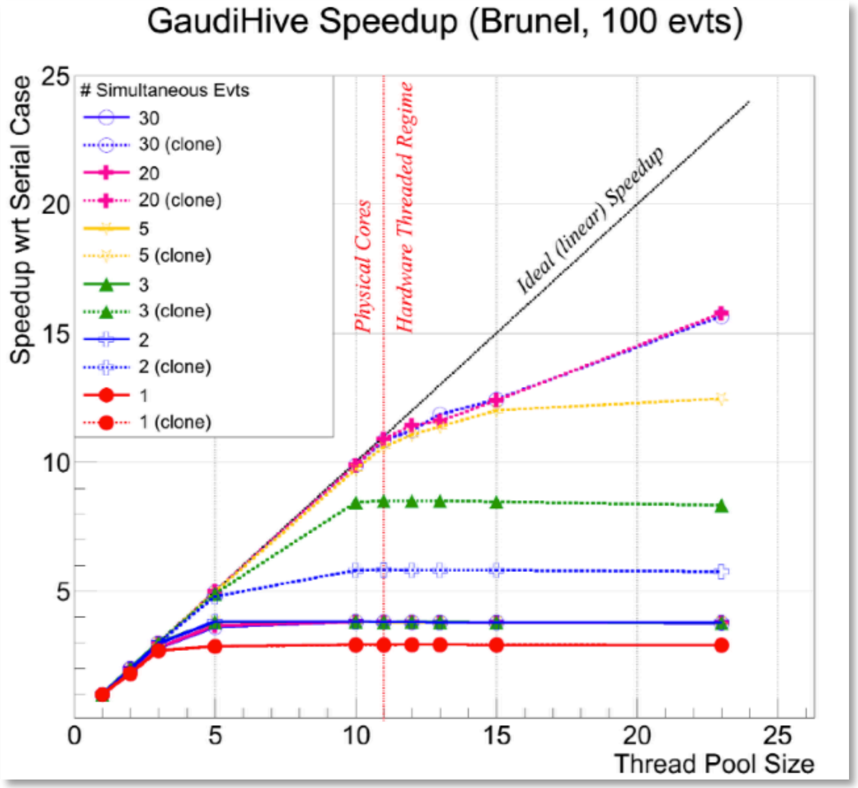
Forward Scheduler

- Component that submits to TBB runtime algorithms according to their data and control flow dependencies
- Absorb the asynchronous finishing of submitted tasks
- Update internal algorithms' state machine accordingly



The Past (2012): CPUCrunchers Demonstrator

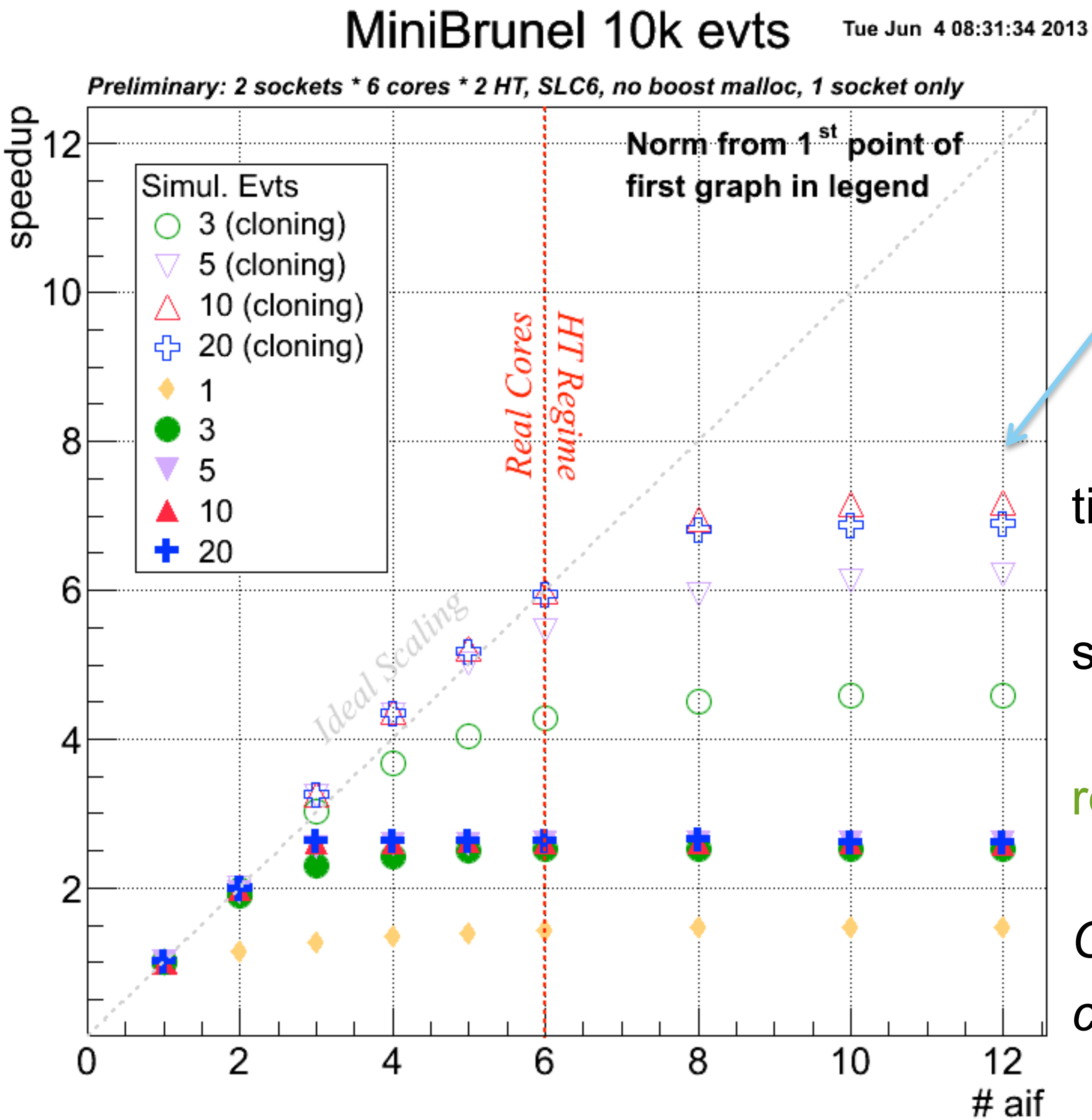
- Emulate an LHCb full reconstruction workflow with CPUCrunching algorithms (no real work done, just keep cpus busy)
- Explore expected behaviour
- Demonstrate potential of the multithreaded approach



~8 Months ago

Evolving LHC Data Processing Frameworks for Efficient Exploitation of New CPU Architectures
B. Hegner at al, IEEE-NSS 2012

Scaling on One Processor



12 simultaneous algorithms: run the application occupying the full socket

One event processed at the time: ~30% speedup

No cloning: saturate at a speedup of 2x

Cloning: ideal (linear) scaling reached with ~10 events in flight

Cloning of the 3 most time consuming algs only