GeantV scheduler, concurrency

Andrei Gheata

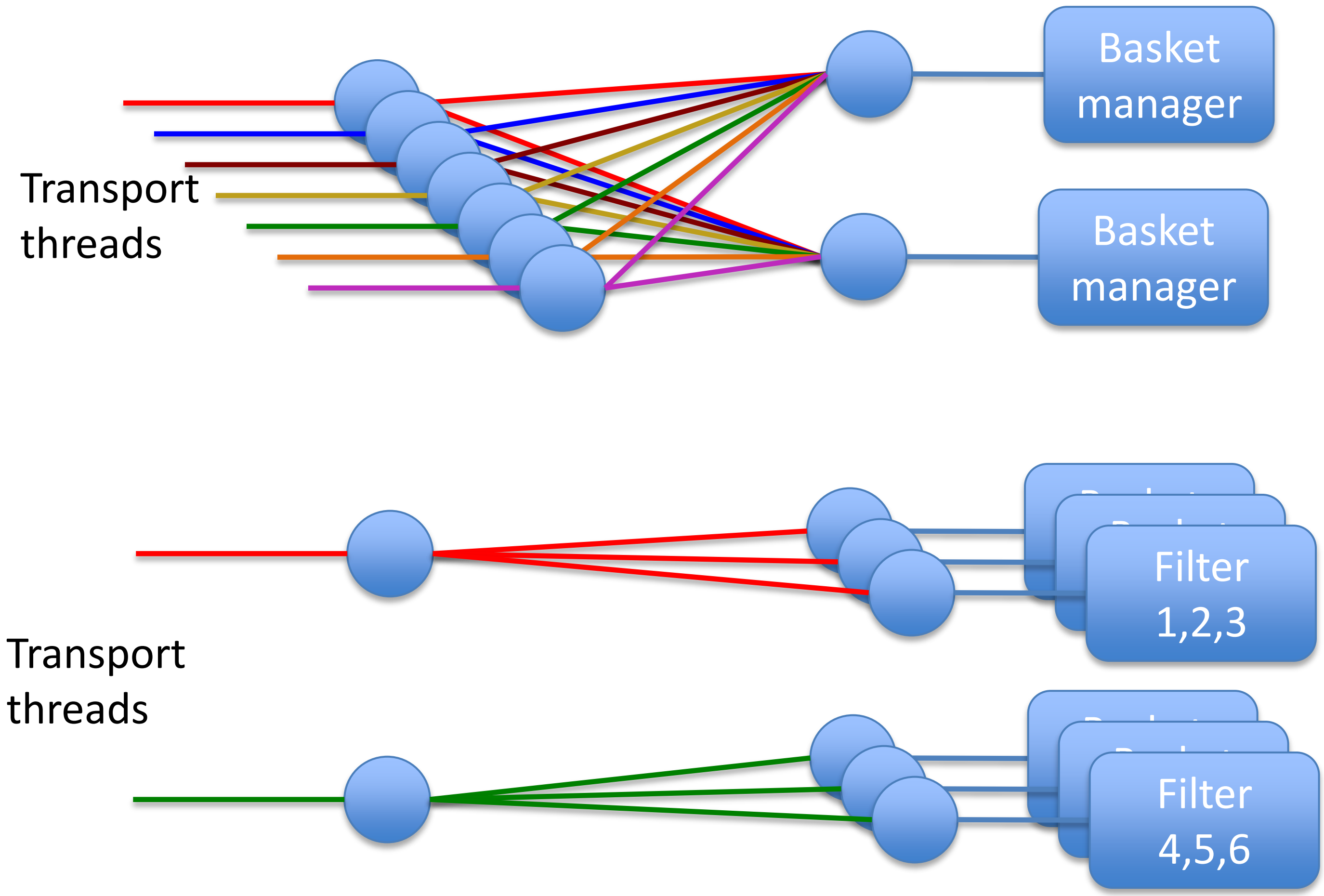
GeantV FNAL meeting

Fermilab, October 20, 2014

Parallelism

- Threads
  - 1 master: initialization
    - idle during transport
  - N workers: transport, re-basketizing
    - No cross-talk during transport
    - Concurrent write to pending baskets (N to 1)
  - 1 scheduler: trigger priority mode, garbage collections
    - Concurrent write with workers to pending baskets (N+1 to 1)
- Queues – a lot of work done to improve here
  - 1 work queue, 1 output queue (removed)
    - ~1E5 transactions/sec
  - N volumes basketizer queues
    - Pre-allocate baskets, activate current one (take from queue)
    - Lower throughput than main work queue
- Amdahl in both thread communication and queues
  - First can be alleviated by changing basketizing policy
    - E.g. thread “ownership” for basketizers

Basketizing -> filtering



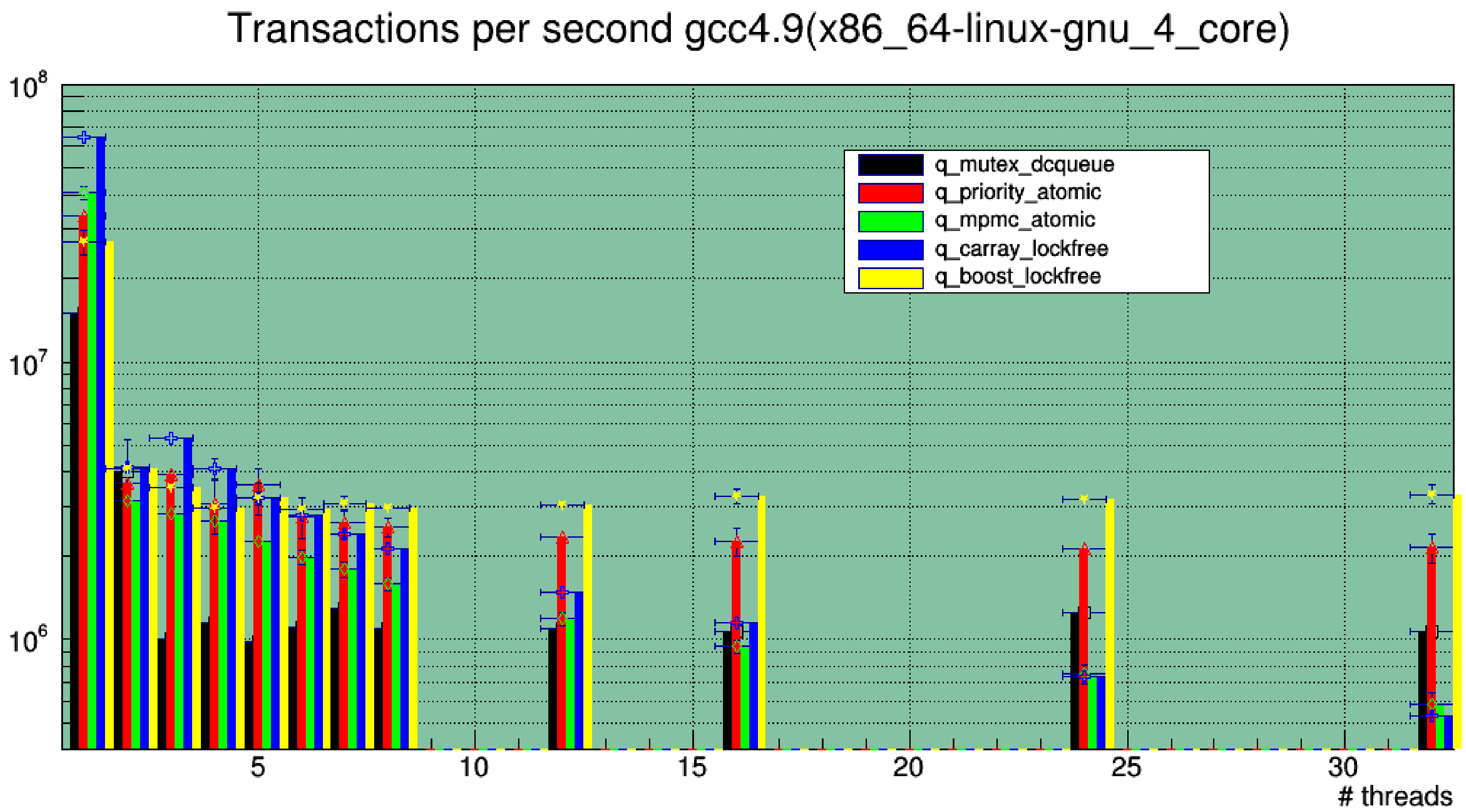
MPMC queues

- MPMC = multi producer multi consumer concurrent queues
- We exchange baskets and individual tracks between threads using concurrent queues
  - Major component for optimizing GeantV performance
- Many flavors of MPMC
  - Sync type: atomic (aka lockfree) versus mutex based
  - Memory management: bounded versus infinite
  - Functionality: provide priority mechanisms or not
  - Implementation: ring buffer versus linked list, …
- Performance measured in transactions/second
  - Queues give generally best throughput in single thread mode: one does not measure scalability, but only throughput degradation with NTHREADS

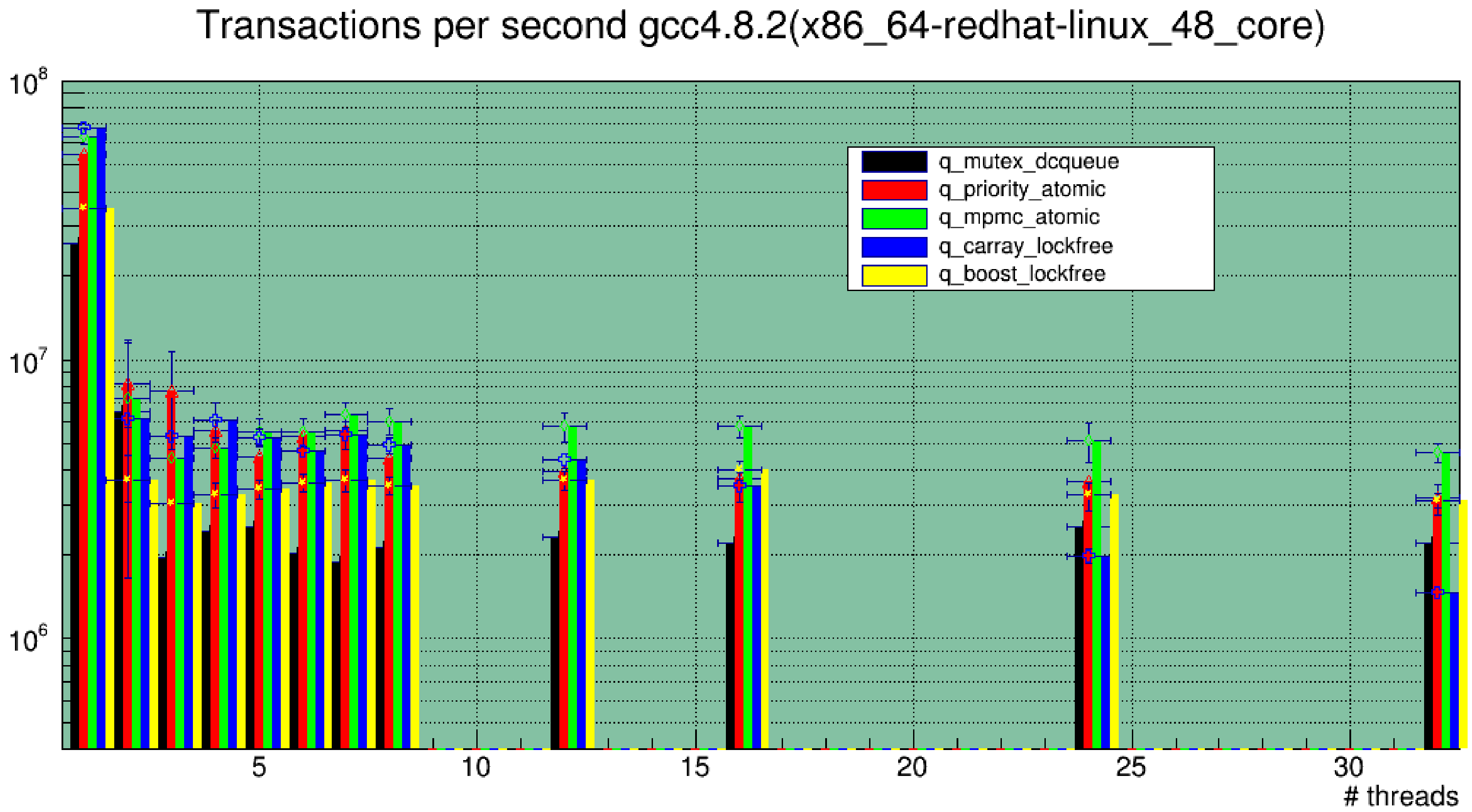
Queues in GeantV

- Mutex based dcqueue
  - In production as work queue, provides priority
- Mutex/atomic hybrid priority_atomic
  - Mutexed only in high concurrency regime, provides priority
- Atomic CAS (compare and swap) mpmc_atomic
  - In production for basketiser queues, replacing dcqueue
  - Circular buffer, no priority
- Array lockfree carray_lockfree (ported by Omar)
  - Another implementation of circular buffer queue
- Boost lock free queue boost_lockfree (ported by Omar)
  - Boost implementation of lcck free queue

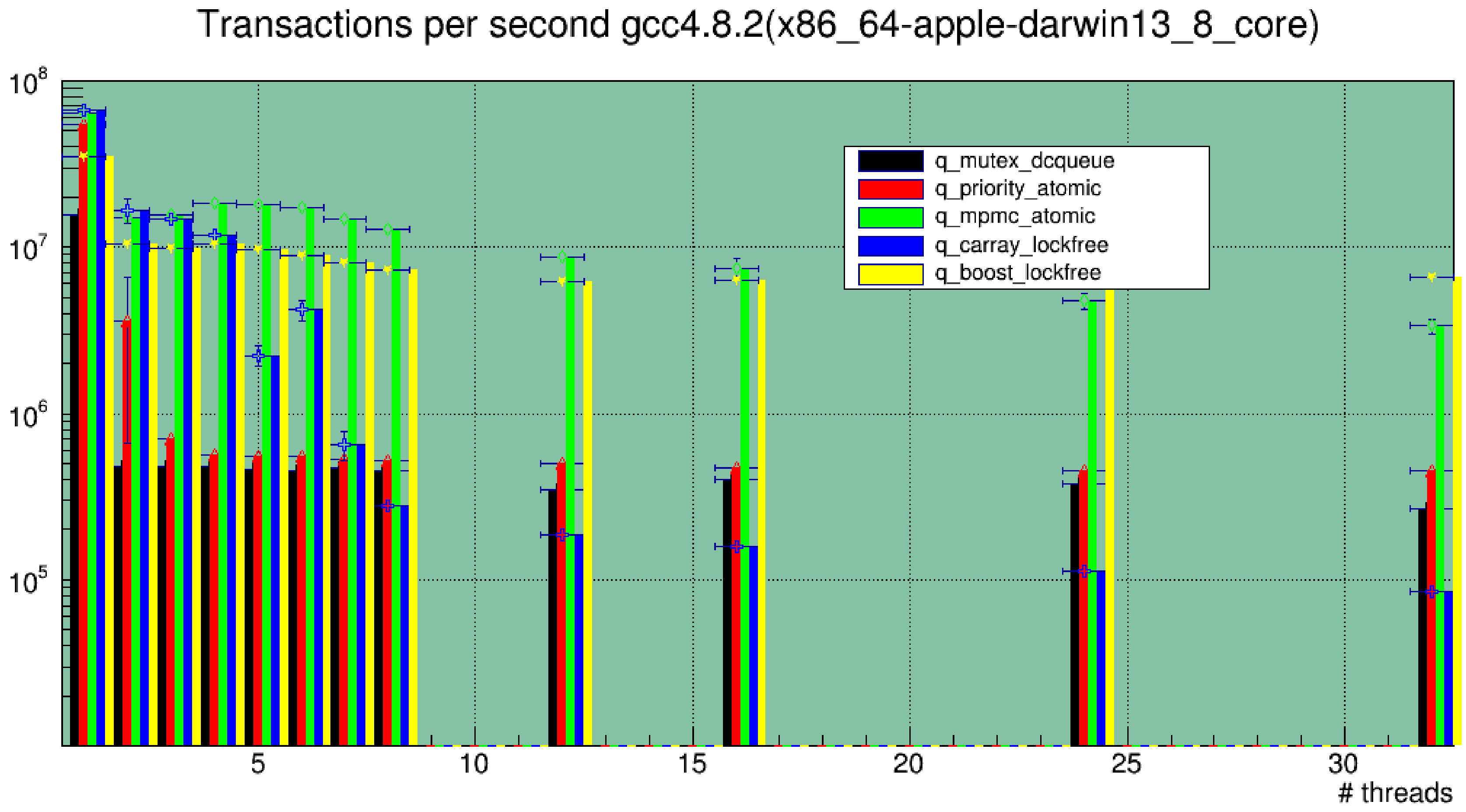
Performance x86_64_4_core



Performance x86_64_48_core



Performance x86_64_apple_8_core



Queues summary

- Our current dcqueue is outperformed by all the others on all platforms
  - We currently work at ~10^5 transactions/sec
- Lockfree queues are doing great on Mac compared to mutex-based ones (50x factor!)
- priority_atomic is the only current replacement for dcqueue (must provide priority)
  - We can expect a factor of 2 queueing improvement on x86_64 linux
- Reducing Amdahl requires revisiting the basketizing model

To do - scheduling

- Generic basketizers
  - Move from a volume oriented basket manager to a “filter” concept
  - Filters have to be complementary (F1+F2+…+Fn=ALL)
  - Parallelism model upgrade: no concurrency per filter
    - A thread can only invoke a subset of filters (owns the associated baskets)
- Integrate a working GPU scheduler
  - What gets filled in GeantTrackV?
  - What are the blockers?
  - GPU basketizer?
  - I hope to clarify the issues related to GPU scheduling during this workshop
- Integrate I/O with the scheduler
  - First step: User data structures (hits/digits)
  - Second step: kinematics
    - Requires event model (e.g HepMC, custom, …)