TBB integration on GeantV

Andrei Gheata

Joel Fuentes

TBB Components

Generic Parallel Algorithms
parallel_for
parallel_while
parallel_reduce
pipeline
parallel_sort
parallel_scan

Concurrent Containers
concurrent_hash_map
concurrent_queue
concurrent_vector

Task scheduler

Synchronization Primitives
atomic, spin_mutex, spin_rw_mutex,
queuing_mutex, queuing_rw_mutex, mutex

Memory Allocation
cache_aligned_allocator
scalable_allocator

Task Scheduler

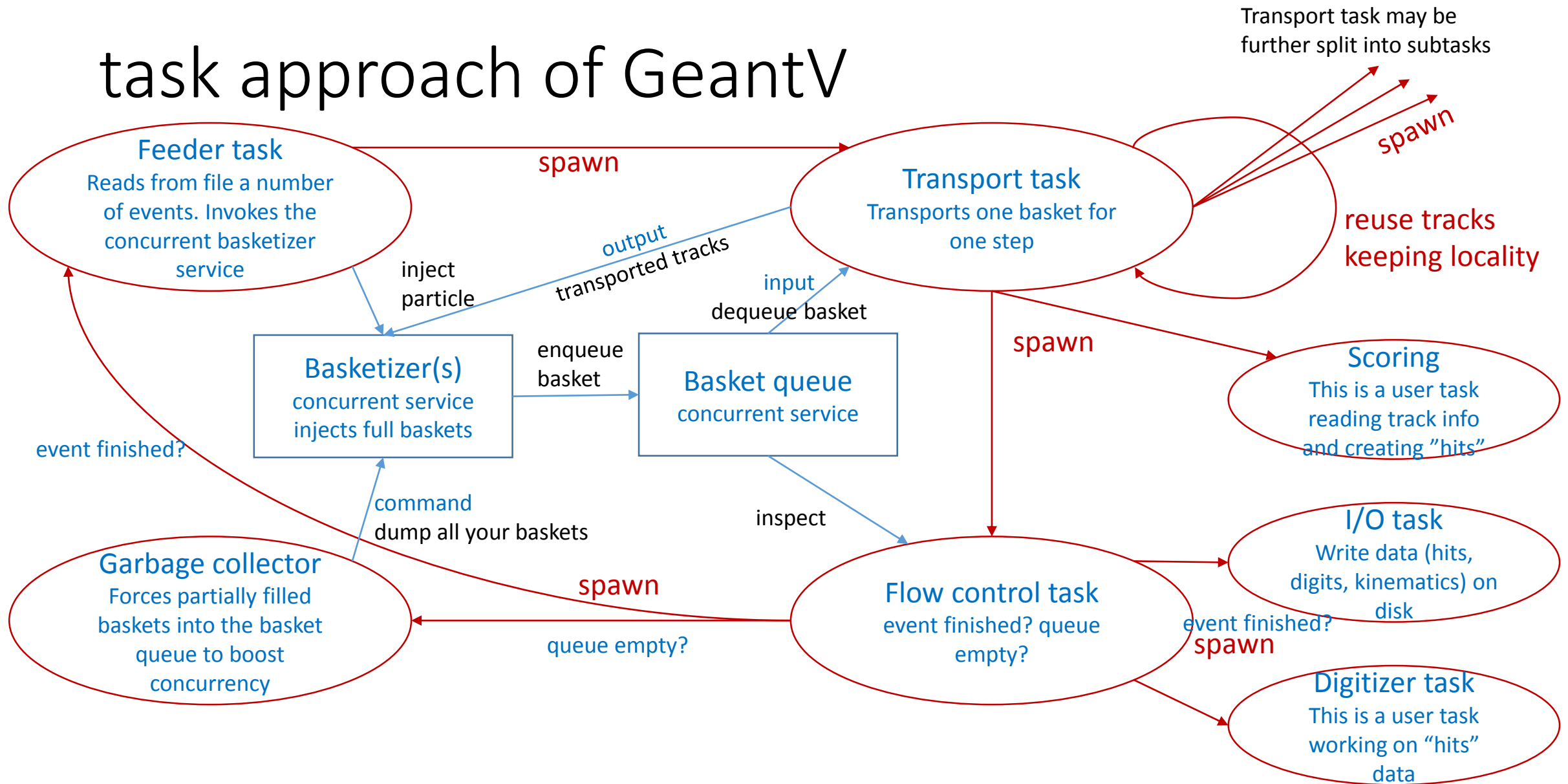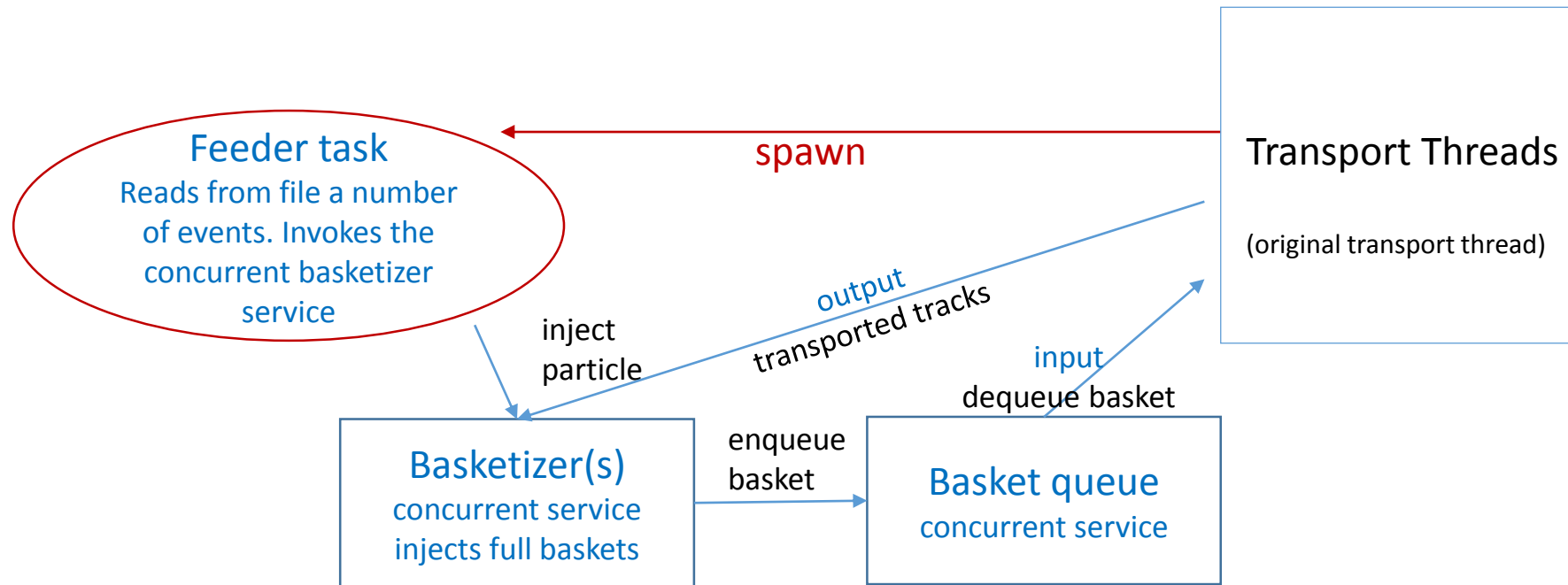| Problem	Solution
Oversubscription	One TBB thread per hardware thread
Fair scheduling	Non-preemptive unfair scheduling
High overhead	Programmer specifies tasks, not threads
Load imbalance	Work-stealing balances load
Scalability	Specify tasks and how to create them, rather than threads
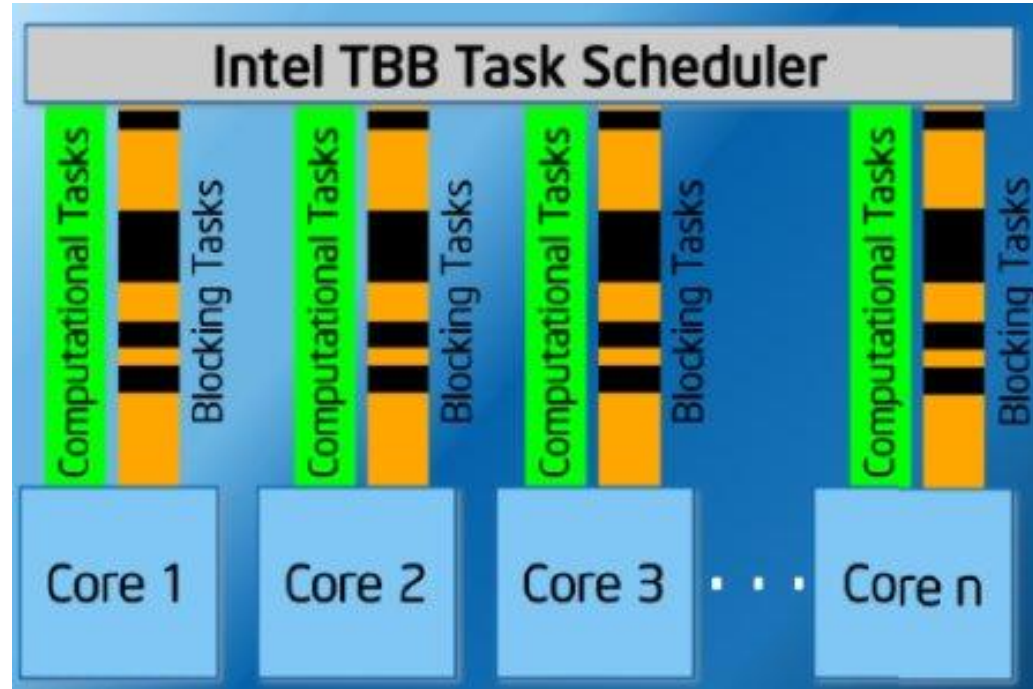
- Each thread has its own ready pool, which is a lists of tasks.
- A task goes into each pool when it is allocated.
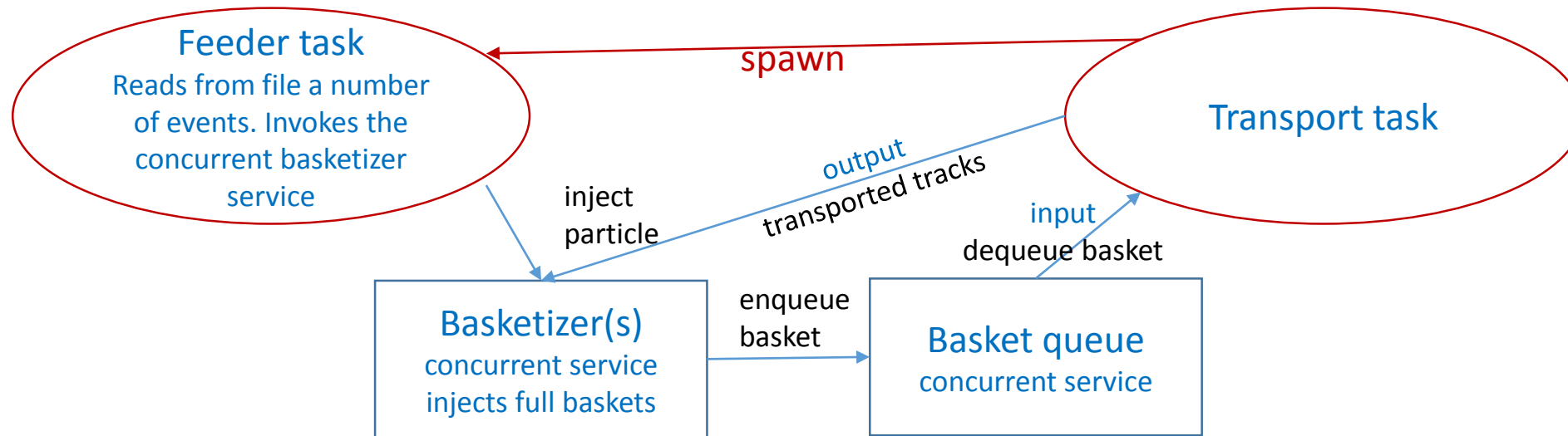- Each thread steals tasks from other pools when necessary.
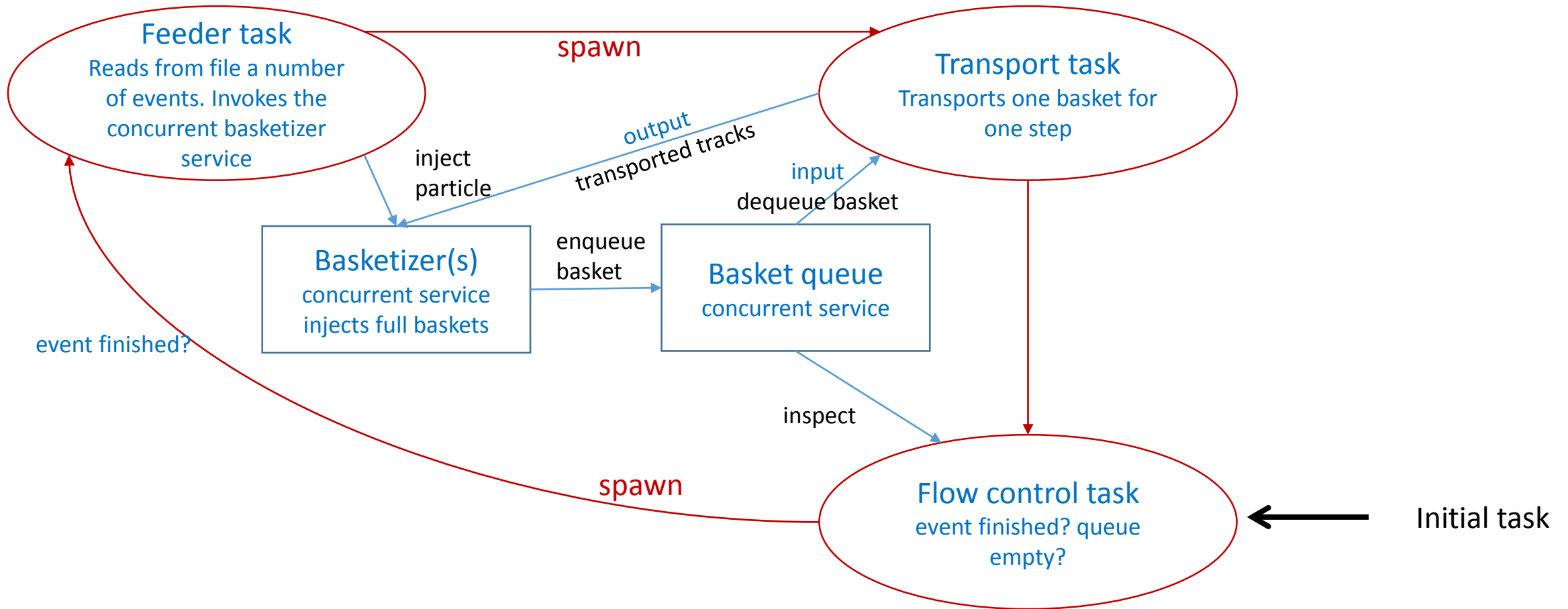
task approach of GeantV

Transport task may be further split into subtasks

Feeder task
Reads from file a number of events. Invokes the concurrent basketizer service

spawn

Transport task
Transports one basket for one step

spawn

reuse tracks keeping locality

output
transported tracks

inject particle

input
dequeue basket

spawn

Basketizer(s)
concurrent service injects full baskets

enqueue basket

Basket queue
concurrent service

Scoring
This is a user task reading track info and creating "hits"

event finished?

command
dump all your baskets

inspect

spawn

I/O task
Write data (hits, digits, kinematics) on disk

Garbage collector
Forces partially filled baskets into the basket queue to boost concurrency

queue empty?

Flow control task
event finished? queue empty?

event finished?
spawn

Digitizer task
This is a user task working on "hits" data

First step: Feeder Task

Second step: Transport Task

Feeder task
Reads from file a number of events. Invokes the concurrent basketizer service

spawn

Transport task

inject
particle

output
transported tracks

input
dequeue basket

Basketizer(s)
concurrent service
injects full baskets

enqueue
basket

Basket queue
concurrent service

Third step: Flow Control Task

Some issues for migrating to tasks

- Thread ID integration
  - Now we have static threads with unique id's, how to deal with this in task mode
  - Use of Thread ID for tasks.
    - Advantage: locality.
    - Disadvantage: It may cause work-stealing to fail.
    - At least the Feeder and Transport Tasks use data accessed by Thread ID.

- Feeder task: only one task at a time
- Assignation of new tasks to be executed in same thread

Affinity methods on TBB

These methods enable optimizing for cache affinity. They enable you to hint that a later task should run on the same thread as another task that was executed earlier. To do this:

- In the earlier task, override note_affinity(id) with a definition that records id.
- Before spawning the later task, run set_affinity(id) using the id recorded in step 1,