Parallel 7B Report

Hadronic Issues related to MT.

A. Dotti, W. Pokorsky

U.S. DEPARTMENT OF ENERGY
Office of Science

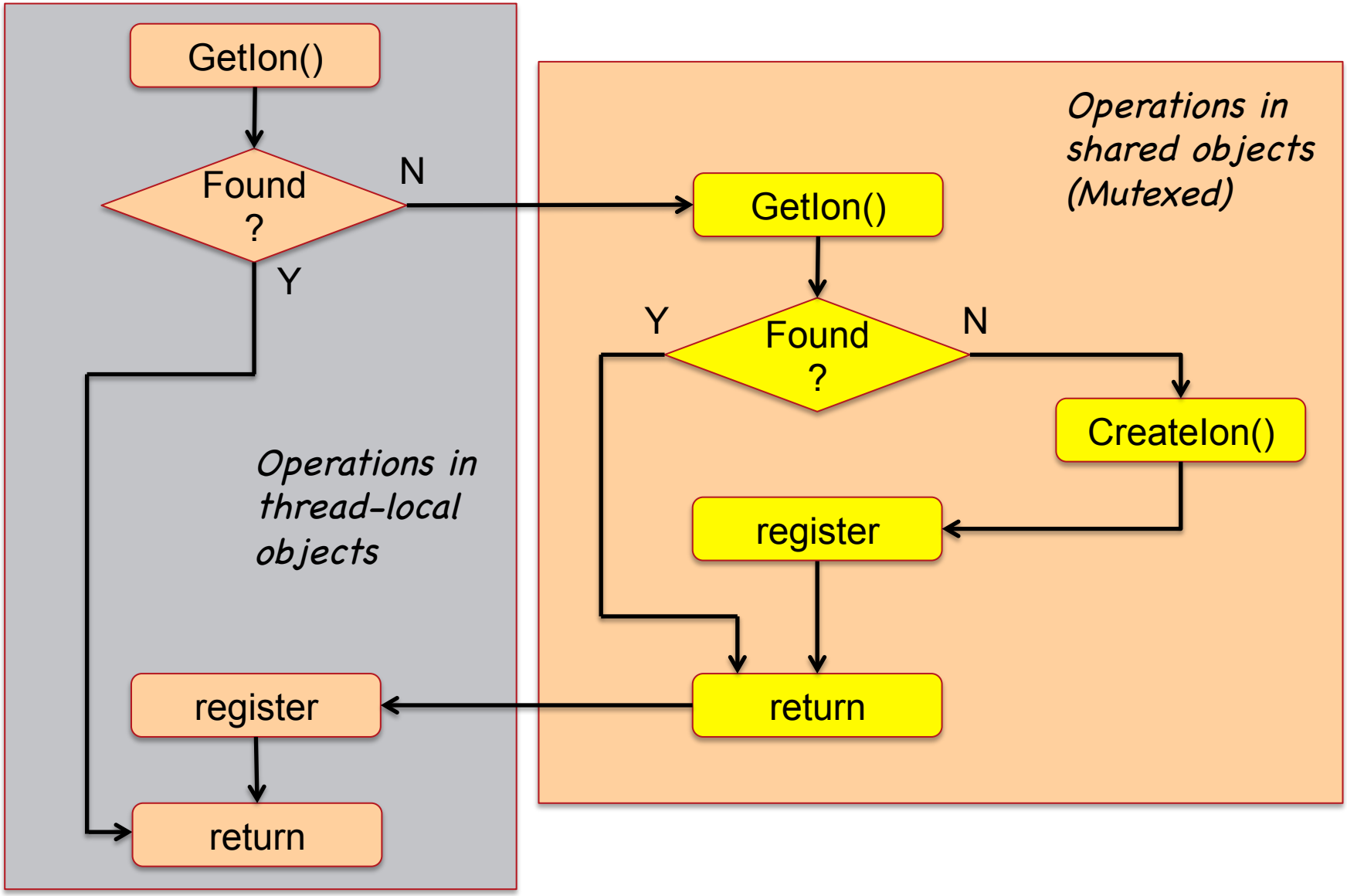
SLAC NATIONAL ACCELERATOR LABORATORY

Reports

SLAC

- Discussion Session on implications of Multi Threading for Hadronics framework
  - "hot-topics", first time we look at MT and HAD
- Contributions:
  - Makoto: "Use of ions for MT"
  - Witek: "Hadronics Cross Sections re-implementation and thoughts on MT"
  - Mike: "General issues for MT: receipts and todos"
  - Andrea: "Memory profiling of hadronics and extension of hadrnoics framework"

How G4IonTable::GetIon() works in worker thread



Ions Conclusions

SLAC

- Final design allows for creation "on the fly" of ions that are G4GenericIon
  - Same functionality ss in 9.6 (and before)
  - Particles that have dedicated process-manager cannot use this mechanism
- These are shared among threads
  - Final design minimize use of locks: absolutely mandatory minimize ion creation during event loop
  - Relies on table of ions to be created before event loop (to be prepared by HAD WG)
    - Size of table is a concern, will need iterations, smaller <1000 is MUCH better
- New CPU-optimized interfaces in G4IonTable
  - In particular G4IonTable::GetMass(…)

Cross-Section Redesign for MT

SLAC

Cache in CHIPS XS

- caching has been greatly simplified by moving to per-element cross sections
- we need to validate it in MT environment
  - probably need a lock for writing in the cache
    - writing in cache happens only at the beginning (when going through new materials), so lock should not be a problem

Memory Trade-offs

Balance between footprint ("memory used by thread") and churn ("memory needed during run")

Memory churn happens as small objects are created, used, then destroyed asyncronously

Small blocks of free memory are left unusable by later, larger allocation needs

- Eliminate temporary buffers (function-local objects/arrays/vectors)
- If class has thread-lifetime, use data member buffers
- Pass output objects into functions as non-const references

Reducing Memory Footprint for MT

SLAC

- First analysis of what we can share in MT
- Top priorities:
  - The hadronics most memory hungry (5MB) component is **BIC** model. Some rework needed
  - The second components using more memory are **cross-sections** (2.2MB) stored in G4CrossSectionDataStore
  - 1 MB all processes together in first event. Comment during discussion: need higher statistic can be under-estimate
- It is realistic to reduce memory footprint for Hadronics of a factor 2
- Step-by-step receipts to achieve this have been discussed (involving removing G4ThreadLocal and making objects "const")

Hadronics Framework and MT

SLAC

- Need to implement possibility to initialize models differently per worker and master
- Cross-sections are easier: factory mechanism guarantees simple sharing design
- Memory profile guided: start from top offender, improve, repeat
- Evolutionary approach: framework is complex and we need to get it right, we'll take the time we need

Conclusions

SLAC

- Now: migrate to new G4IonTable interfaces as appropriate
- For Geant4 Version 10:
  - Implement changes in API for hadronic models base classes
  - Verify sharing of cross-sections between thread
  - Need action on BIC memory usage (Note: could be impossible to fix in time, meta-programming implementation of few classes)
- For 2014:
  - Start migrate models to MT with sharing of memory consumption items
  - Proposal (to be further discussed): develop a generic MT-safe "cache" to be used generally in G4