Multi-threading and other parallelism options

J. Apostolakis

Summary of parallel session. Original title was “**Technical aspects of proposed multi-threading approach**”

Recent news

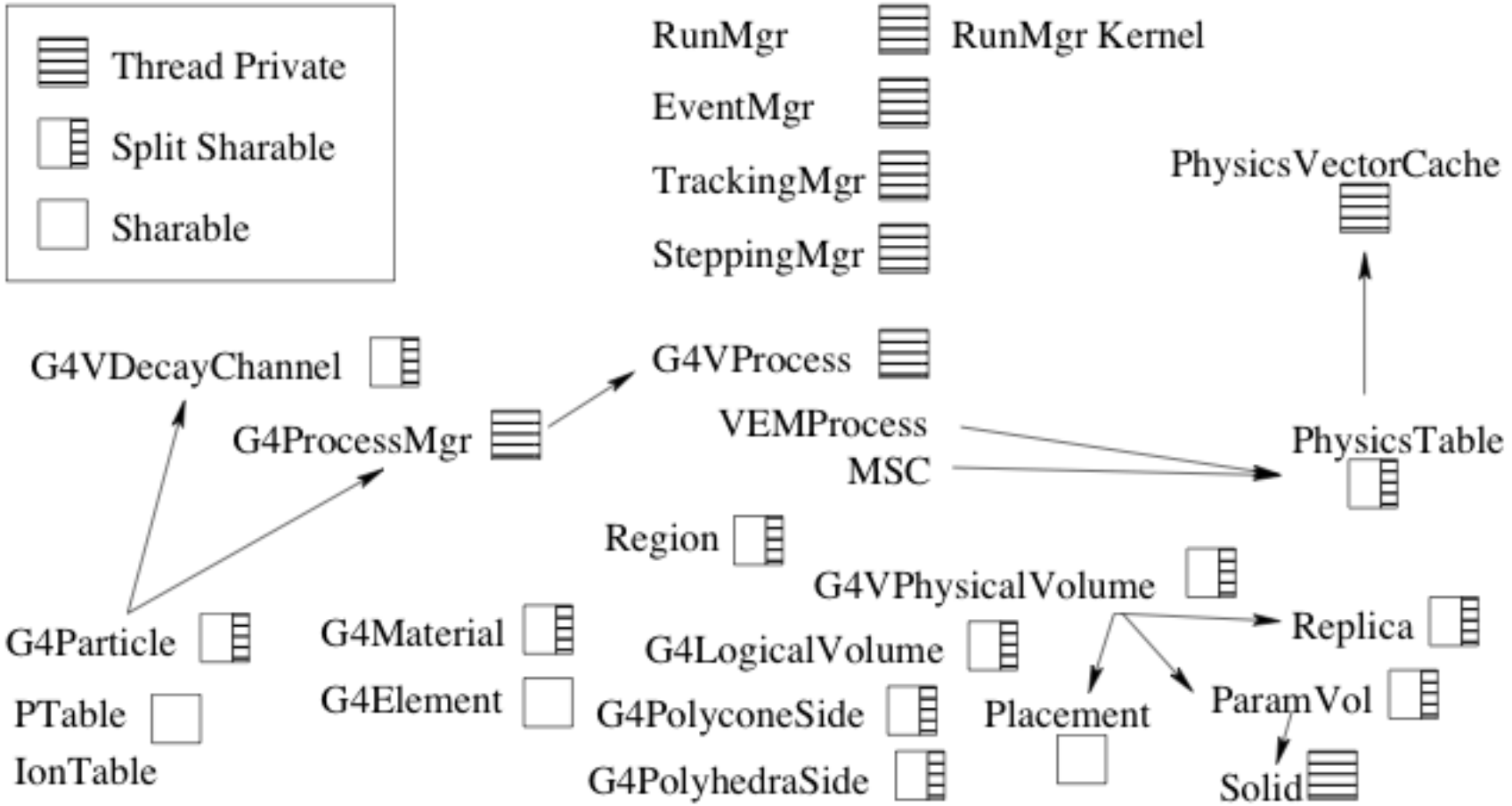
- Geant4 Multi-threaded
  - Demonstrated good scaling performance
    - With 24 cores, see speedup of about 24
  - Revised prototype with Geant4 9.4 beta
- Proposal to move UI-parallel ‘MPI’ into G4 source

Recipe for Geant4 Multi-Threaded

- Add __thread to globals, static data members
- Identify data members which need to be separate for each thread
- Create new classes to hold these data members for multi-threading
  - And methods to initialize worker data by copying master
- Make scheme's implementation more readable and simpler
  - So that it can be maintained by G4 developers
- Create tools to check code, run-time integrity

The Design of Geant4 Multi-Threading

Worker threads share read-only fields for some object instances



Thread safety and STL

- Is thread safety of STL an issue?
  - Nearly all classes using STL are thread-private
  - Important objects which are shared must be treated carefully, and are
    - IonTable now initialized with 3000 stable isotopes at the start
    - A previous implementation used locks to ensure that table insertion is thread-safe.
- Which parts of STL could have issue with thread-safety

Tools used by Geant4 MT

- Parser g++ to find/check static data
- Urdb
  - Tool to use checkpointing to ‘go backwards’ when running your debugger session.
- Protecting the master thread heap

Types of parallel Geant4

"Older style"

- Farming events using "TopC"
  - Can use native, MPI or distributed memory
  - Facility for Joint results files
- UI mediated parallelism
  - Splitting job using MPI
  - Separate result files must be arranged

Aiming to reduce memory footprint

- Multi-process G4
  - fork, to use copy-on-write
- Multi-threaded G4

Challenges

- Four potential solutions for one issue is a large number
  - Need to consider effort for documentation, maintenance
  - Identify which solutions are appropriate for which use cases
  - Search for existing and further commonalities

Trying out G4 multi-threaded

- You can find the latest version
  - /afs/cern.ch/user/x/xdong/public/phase2/source4.9.4.b01.forsequentialclhep.tar.gz
- And a recent version using ‘MT-revised’ CLHEP
  - /afs/cern.ch/user/x/xdong/public/phase2/geant4.9.4.b01mt.tar.gz

Ideas for improvement & concerns

- Web site for G4MT prototype
- Need to clarify use cases
    - including constraints, performance needs, current and future
  - Adopt solutions to address these.
  - And guide users to use corresponding solutions.
- My concern(s)
  - Diversity of solutions is a challenge
  - Consolidation must be considered
    - E.g. must have **only ONE way** to 'install' MPI version(s).