Thread-safety and shared geometry data

J. Apostolakis

Overview

- Issues
  - RW data in the ‘live’ geometry tree
  - Causes
  - Implications
- Improving Geometry for MT !?
  - Reducing / eliminating RW data
  - Impact on kernel and user code
  - Expected and potential changes

Issues

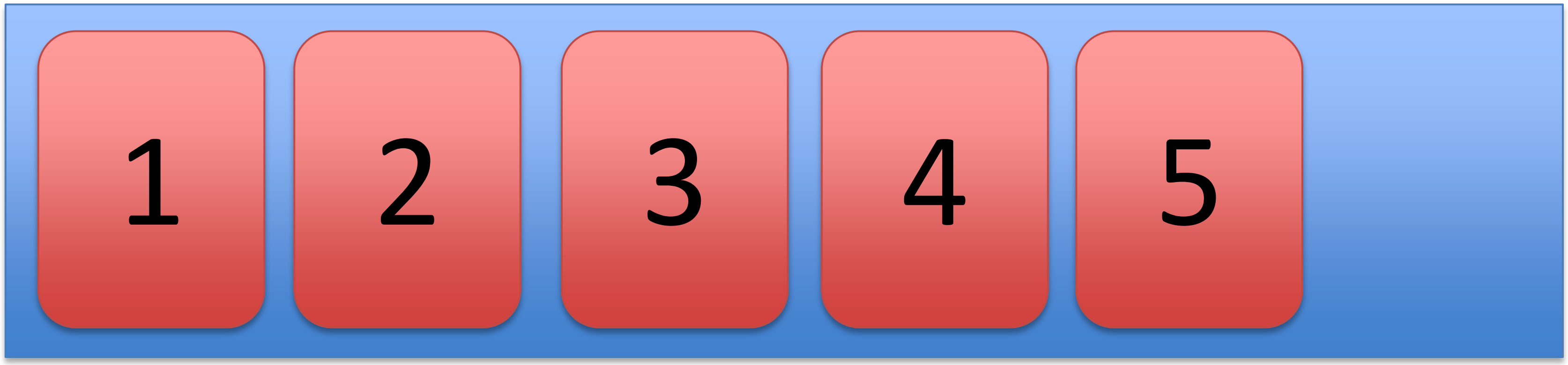
- Live geometry tree has ‘read write’ members
  - Physical Volumes – identify the copy number
    - PVReplica: int CopyId
    - PVParameterisedVolume: VSolid, Transformation, ..
  - This means that today the RW fields must be thread local in G4-MT
  - Can we make G4Navigator + Geometry ‘independent’ and fully thread-safe ?

What is RW in geometry?

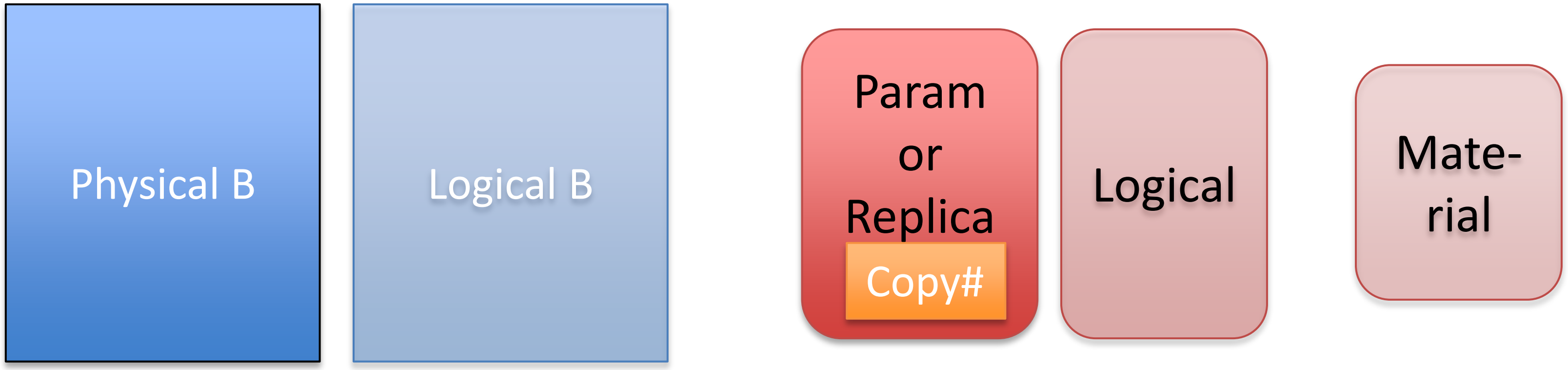
- In the LIVE geometry tree most Physical Volumes types are RW (except Placement)
  - Why? To address need to identify the copy number, and to have the correct properties
- Specifically
  - PVReplica: int CopyId
  - PVParameterisedVolume: VSolid, Transformation, Material, ..

Explanation

Logical View – Replica or Parameterised Volume



View in Memory – Geant4 objects



Who uses the RW information?

- Every client of the geometry
  - G4VPhysicalVolume GetLogicalVolume()
  - G4LogicalVolume GetSolid()
  - G4LogicalVolume GetMaterial()
- G4Processes
  - Any process that needs
- User code – in Sensitive Detector, Actions, …

Implications

- This means that today all the RW fields must be thread local in G4-MT
  - A mini-class is required to hold CopyId for Replica
  - Each Replica instance gets an InstanceId, and references a location in Local Array
    - Extra indirection + needs to use thread Id (worker?)
- Parameterised volumes have special way
  - A copy of each PV instance is created in master
  - Threads create a copy of each instance
  - Result: less memory savings; Unclear whether RegularNavigation will work.

Impact

- Each solid must have a Clone() method
  - A worker (thread) must create a clone of each solid of the Parameterised Volume
    - Can be thousands or millions
- This solution greatly reduces reuse of memory in MT
  - Clear impact on applications with large Param. Vol.
- Unclear whether NestedParameterisation and Regular Navigation work or not
- Alternative solution could / should be sought

Tentative Plans

- Goal: Reduce or eliminate RW information in ‘live’ geometry tree
  - Keep all information about current volume only in G4TouchableHistory
- Can it be done without breaking interfaces?
- Challenge: What about existing uses of key methods?
  - in PhysicalVolume->GetLogicalVolume
  - logicaVolume->Get(Solid,Material, ..)

Change: internal in Geometry

- Keep all information about current volume only in G4TouchableHistory
  - Requires changes in G4Navigator and its dependent classes
  - Additional information to be kept in Touchable History (tbc)

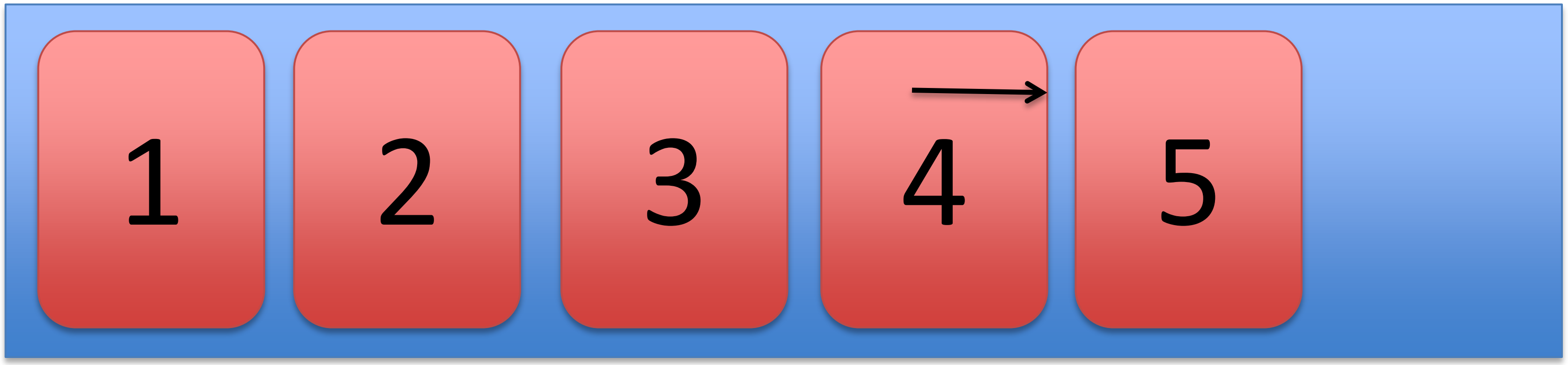
– If we need to maintain interface(s) of PV, LV,
  - Change interface of Parameterised Volume and Parameterisation ?
  - Create a clone of logical volume (?)

Changes: users’ considerations

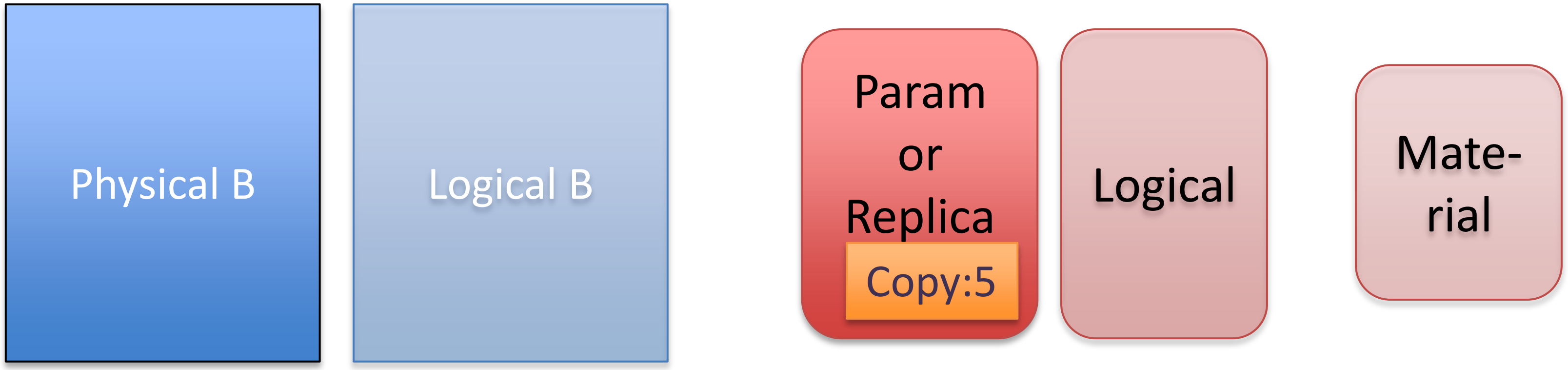
- If we do not keep compatibility all calls to GetLogicalVolume->Get(Solid, …)
  - Must migrate much kernel and user code
  - Some of this code is already wrong (must use TouchHist to get correct answer).
- How to solve uses of Physical/Logical Volume Get methods ?
  - Can ‘local’ clones of LogicalVolume solve this ?

Use in Tracking – after relocation

Logical View – Replica or Parameterised Volume



View in Memory – Geant4 objects



Additional issues

- Must associate correct Thread-local Field to the relevant LogicalVolume (or Region).
- The locations of these fields are:
  - Global (all space) – belongs to World log-vol;
  - Per Region – belong to region;
  - Per Logical Volume – to LV.
- Must ensure correct assignments
  - Note that LV has a FieldManager = field + ‘Solver’