Experience at Parallelizing Geant4 (version 3.0)*

Gene Cooperman

Northeastern University
Boston, MA, USA

(Run and Events Working Group, CERN affiliate)

Collaborators:

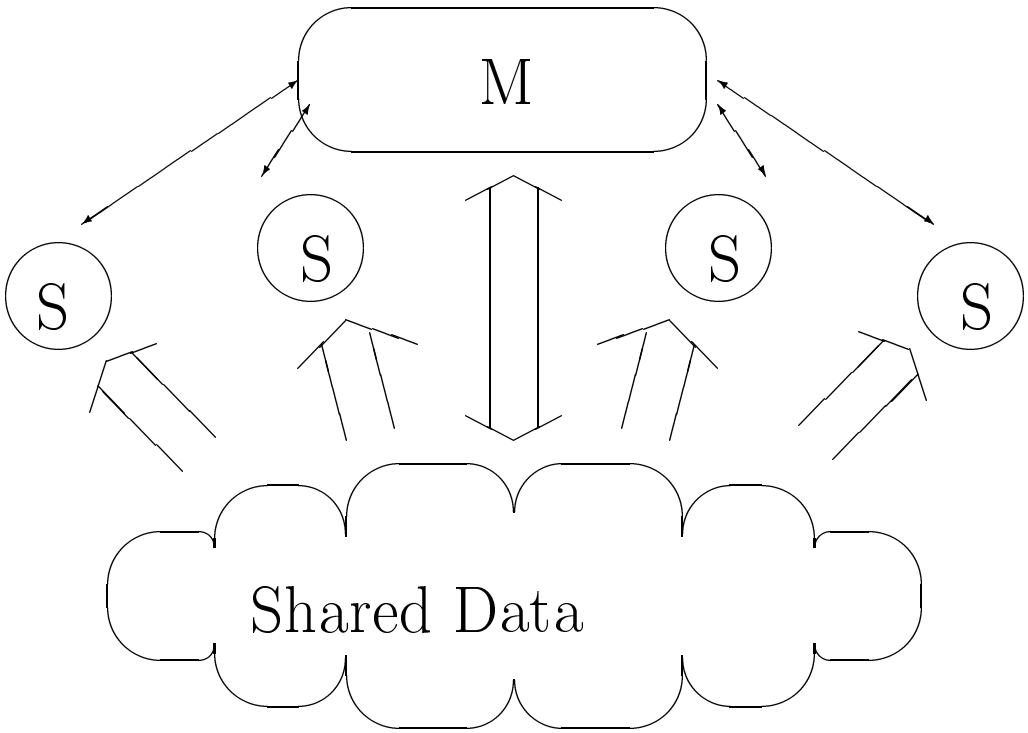
George Alverson, Luis Anchordoqui,
Victor Grinberg, Thomas McCauley, Steve Reucroft,
Edgar Salazar and John Swain

*Partially supported by NSF Grant ACR-9872114.

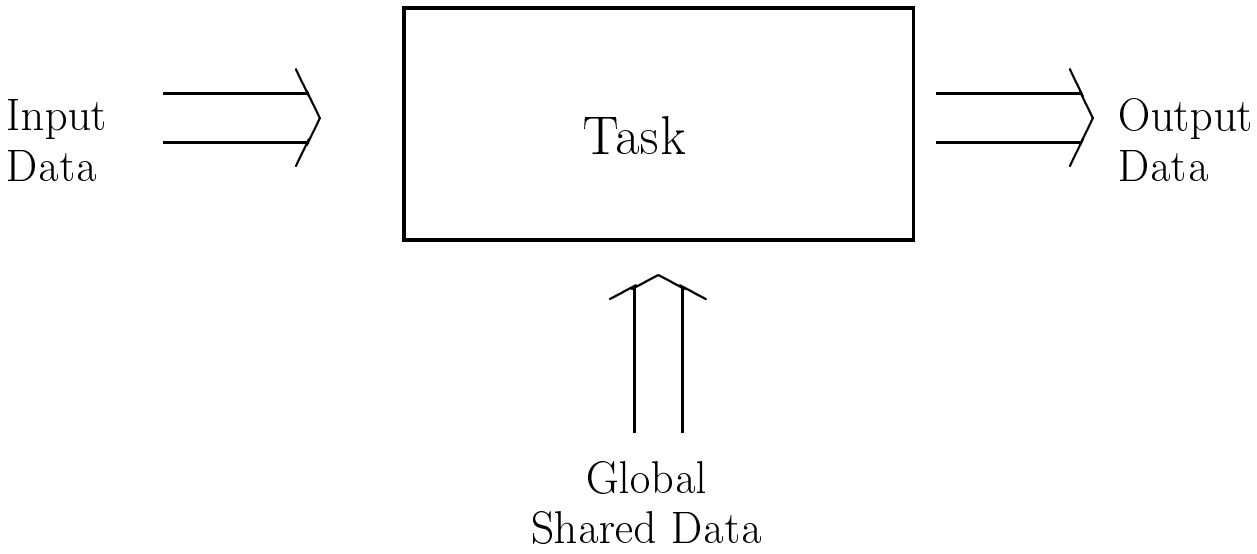
Approach Using TOP-C

TOP-C = Task Oriented Parallel C/C++
`http://www.ccs.neu.edu/home/gene/topc.html`

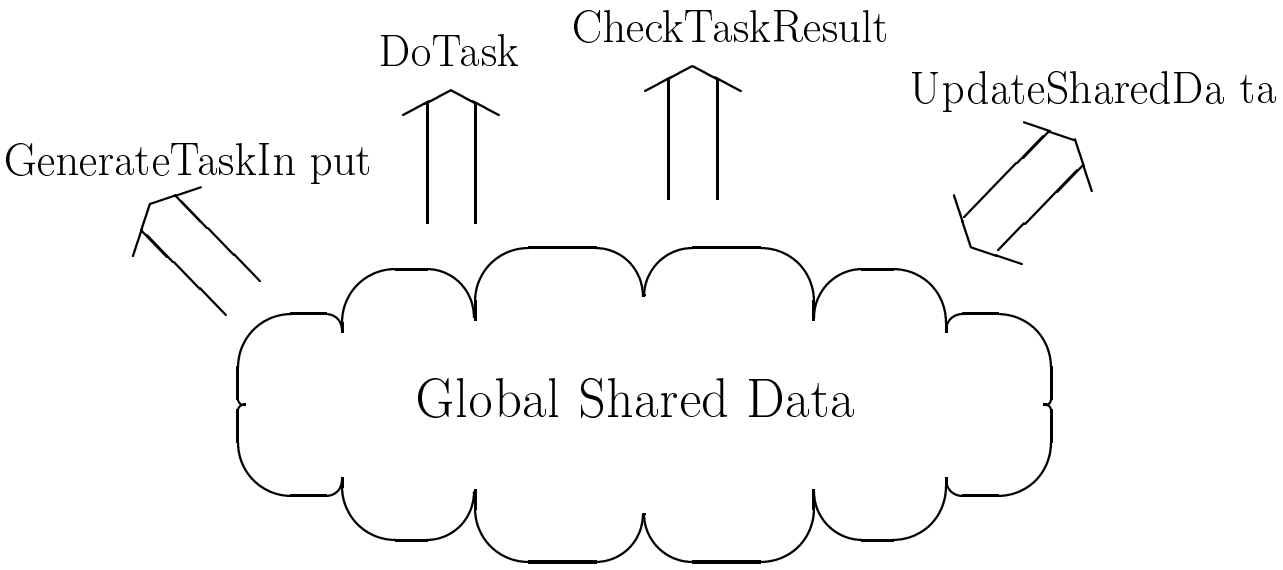
Free open source software (GNU LGPL license)
"Footprint" of approximately 40 KB as stripped binary



Concept 1: Task



Concept 2: Shared Data



Concept 3: Action (implements parallel strategy)

(For parallelization of Geant4, trivial parallelism sufficed.)

Track-Level Parallelism

- Each slave executes a task
- The input to the task is a primary track
- The output of the task is all secondary tracks (and all detector hits)
- The global shared data includes all ParticleDefinition's, PreAssignedDecayProduct's, Touchable's, CreatorProcess, UserInformation, etc.
  NOTE: TOP-C does not need to know where global shared data is stored

Lessons Learned:

- Geant4 can be parallelized primarily by modifying G4EventManager.cc
- The same principles could be applied to implement event-level parallelism
- By using the TOP-C library, relatively little effort is required to parallelize Geant4. A crude version (parallelizing tracks only, not hits) was implemented in about one week.
- The main difficulty (for an outsider) is to understand and take advantage of the Geant4 design decisions and coding style.

Features of TOP-C model

- Easy to Use (small number of primitives, primarily to define the task)
- Same Geant4/TOP-C source code works with three TOP-C communication layers:
  1. Sequential (emulation by single process, easy to debug)
  2. Distributed Memory (over sockets, or over MPI)
  3. Shared Memory (over POSIX sockets)
- Easy to parallelize sequential code
  1. No requirement to declare to TOP-C the global shared data
  2. about 250 new lines of TOP-C related Geant4 code
  3. about 550 lines for marshalling Geant4 classes:
     (a) G4ElectronOccupancy
     (b) G4DynamicParticle
     (c) G4Track
- Simple Calling Parameters

  (Assume `a.out` compiled and linked with TOP-C libraries)

  ```
  ./a.out --TOPC-help
  ./a.out --TOPC-num-slaves=5 --TOPC-verbose
          --TOPC-trace=0 --TOPC-safety=10 ...
  ```
- Easy To Install (./configure; make)
- 40-page manual (on-line and postscript; examples, advanced features, etc.)

Issues in Parallelizing Geant4

- **minor issue:** a few missing Set/Get functions for certain members
- **pointer members:** (difficult to marshal)
  1. **G4DynamicParticle:** theParticleDefinition, thePreAssignedDecayProducts
  2. **G4Track:** fpTouchable. fpNextTouchable. fpStep. fpLVAtVertex, fpCreatorProcess, fpUserInformation
- parallelization of user function `ProcessHits()`
  - **PROBLEM:**
    1. Two tracks are processed on two different slaves
    2. Each track generates a HitsCollection on that slave
    3. Each slave passes back to the master its HitsCollection
    4. PROBLEM: The master needs to combine the two HitsCollection,
  - **EXAMPLE from novice/N04:**
    1. detector is calorimeter
    2. if two tracks strike the same cell, they have the same HCID (Hits Collection ID),
    3. in sequential code, the user function ProcessHits() sums the two energies and store only a single G4VHit;
    4. But on the master, information from aStep is no longer available, and so ProcessHits() can't be used.