Fermilab

Managed by Fermi Research Alliance, LLC for the U.S. Department of Energy Office of Science

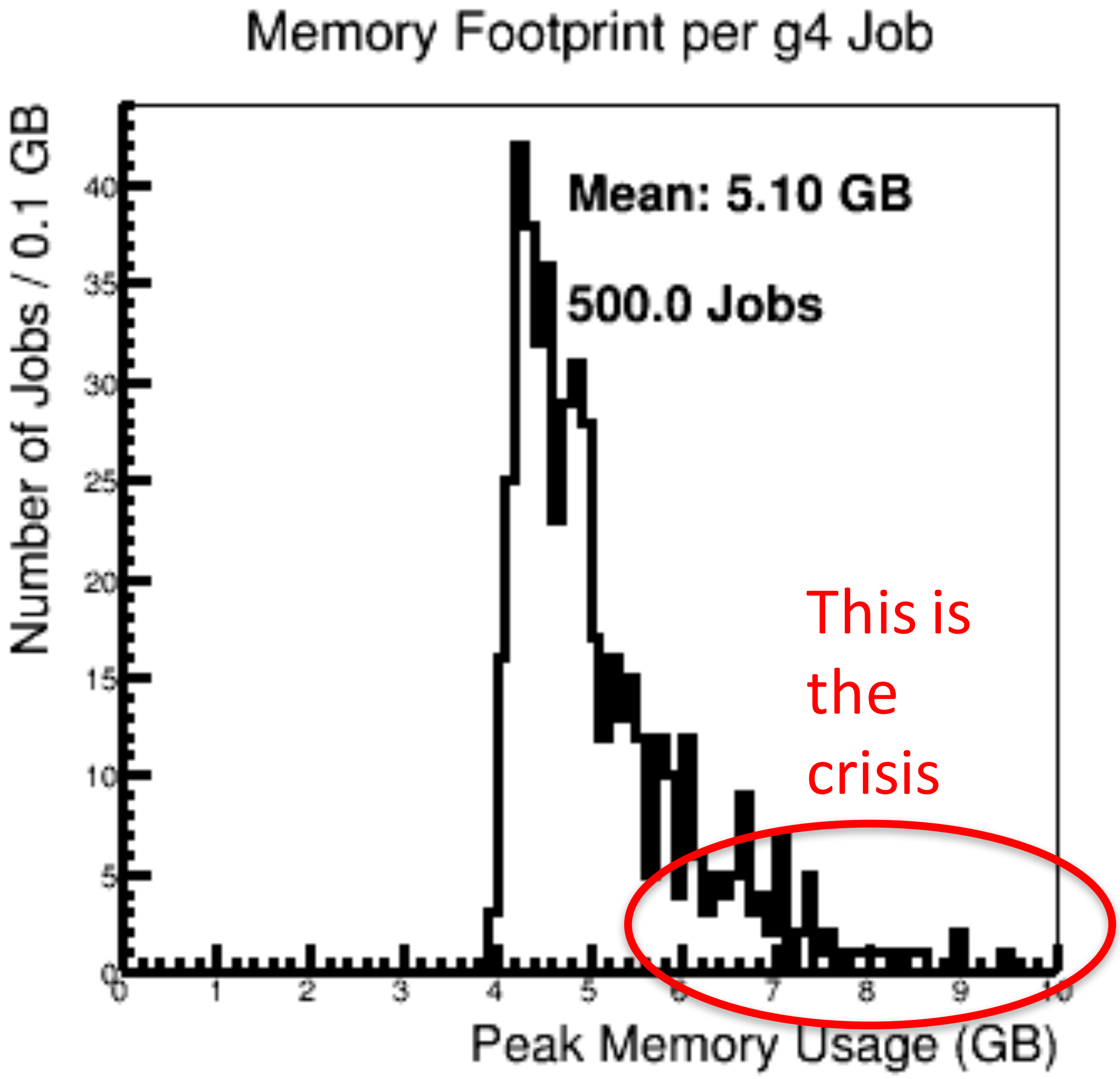
Memory usage crisis in MicroBooNE

Vito Di Benedetto, Chris Jones, Jim Kowalkowski, Marc Paterno,
Gianluca Petrillo, Saba Sehrish

LArSoft Steering Group Meeting

4/8/2016

Problem



- Memory in the MC workflow jumped significantly.
- Seems to have appeared when shower generation with CORSIKA was turned on
- uboonecode version: v04_36_00_01
- LArSoft version: v04_36_00

Addressing the problem - procedure

- Locate workflow
- Perform Initial tests
- Produce memory profile
- Analyze results
- Implement recommendations
- Produce report

Locate workflow

- Obtain workflow: MC Production workflow is the culprit
- Establish test environment
  - FNAL GPVM machines worked for this
  - Access to ancillary files made easy here
  - Easy to get up and running quickly
- Configure and run
  - gen: prodgenie_bnb_nu_cosmic_uboone.fcl
  - g4: standard_g4_uboone.fcl
  - detsim: standard_detsim_uboone.fcl
- Establish scope of exploration
  - Data structures within code
  - Experiment can handle run-time configuration changes

Perform initial tests

- Test software infrastructure configuration (art) under this workflow
  - Rule in or out the ROOT I/O module and its configuration
  - Option combinations investigated
    - splitLevel : 0 # no splitting (no column-wise n-tuple)
    - splitLevel : default # full
    - saveMemoryObjectThreshold : <reduced_value>
    - treeMaxVirtualSize : <reduced value>
    - physics.end_paths:[] # No output module with this configuration
  - Utilize memory tracking service (built into art)
  - **Result**: output module not the key area to explore
- Isolate problem stage in workflow using memory tracker
  - G4 stage found
  - Geant4 (LArG4), creation of MC tracks and showers (MCReco)

Produce profile

- Vito Di Benedetto drafted to help with all this
- Obtain compute resources for evaluation and testing
  - larsoftdev6 commandeered
- Generate test data sample
  - Turned off auto-seeding procedure!
- Select, install, and configure tools
  - massif (valgrind),
  - igprof
- Perform profiling run
  - Really long to do 20 events under massif (>16 hours!)

Analyze results

- Gather team for profile review
  - Developers: Saba Sehrish, Gianluca Petrillo, Vito Di Benedetto
  - Analysts: Chris Jones, Marc Paterno, me
- Produce list of candidate functions and data structures from profile (see next slide)
- Produce testing plans
- **Analysis sessions required all of these people to be present**
  - Required expertise with the tools, the application and domain, the language
  - A lot of dedicated hours working together
  - Experts on the team have been doing this for >20 years!
- Need access to appropriate resources: experiment people, computing facilities

Massif – high power tool

- Valgrind tool that tracks memory use across the application
  - Emulates x86 instruction sequences so it is very very slow! (>10x)
  - Require big machine run – memory footprint large
- Points directly to functions with stack trace:
  - 21.14% (528,938,544B) 0x25A200EA: **sim**::**MCRecoEdep::MakeMCEdep**(std::vector<sim::SimChannel, std::allocator<sim::SimChannel> > const&)
  - 38.26% (522,945,536B) 0x1ADDBB7A: **larg4**::**ParticleListAction::SteppingAction**(G4Step const*) (vector.tcc:101)|| ->38.26% (522,945,536B) 0x1B4A603F: g4b::UserActionManager::UserSteppingAction(G4Step const*)
- *Does not imply that the problem is easy to fix!*
- Requires good test sample

Implement recommendations

- Gianluca and Saba implemented all of the code improvements
- From Saba's MicroBooNE report: We looked at the design and use of data structures
  - Some data structures in MCReco are merged and compacted
  - MCReco no longer holds memory between events
  - Particles are dropped ASAP when filtering by volume
- Testing
  - Gianluca Petrillo provided data product dumping modules for detailed results comparisons after code changes
    - DumpMCParticles, DumpMCTracks and DumpMCShowers
  - No observed change in physics results
  - Results available in the MicroBooNE 4.36 production branch.

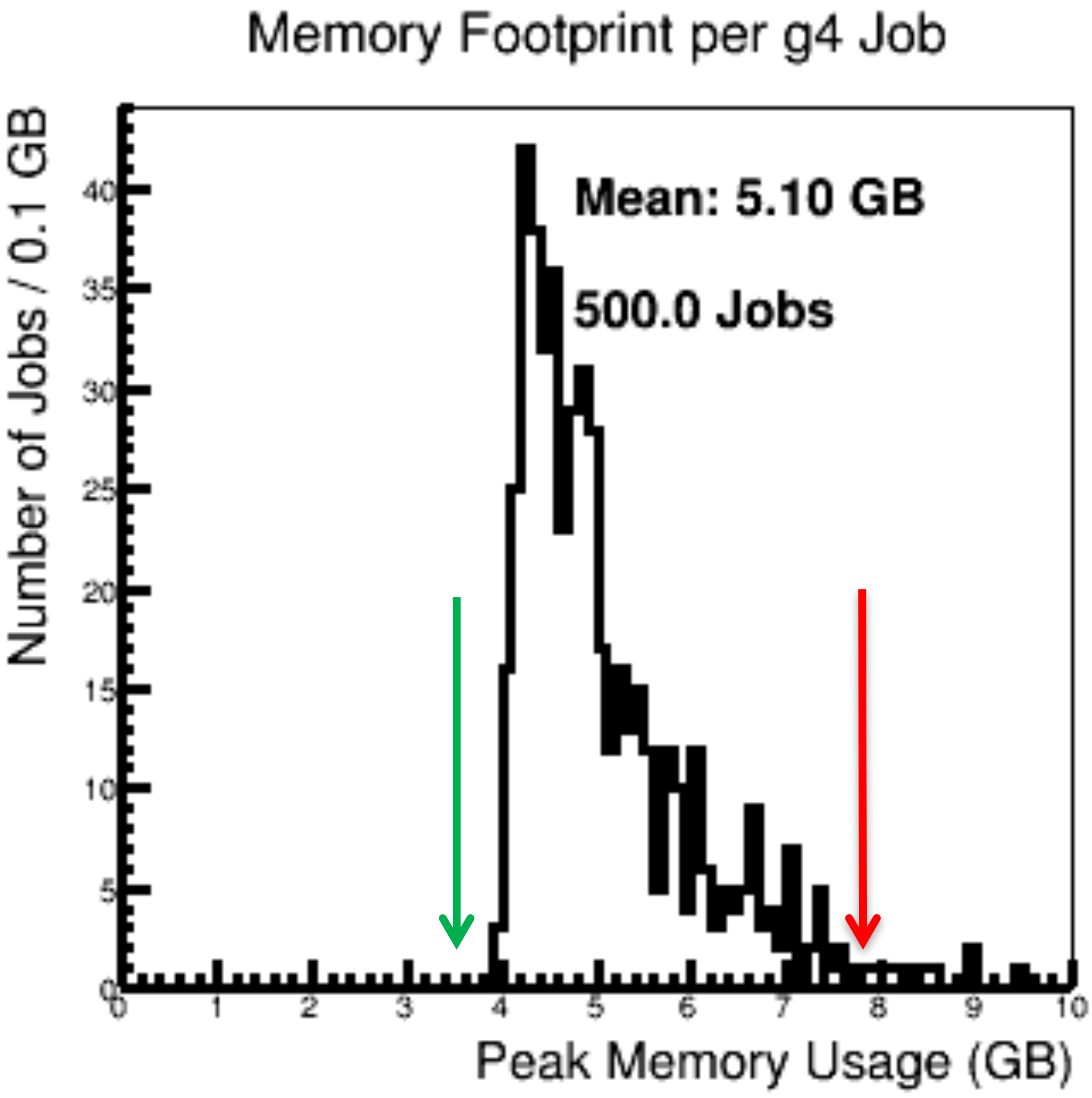
Summary - remarks

- A written report has been produced and is being finalized
- Recommended next steps
  - Continue with other memory hot spots at a normal pace (quite slow) instead of in crisis mode
  - Continue performance work in this mode
    - demonstrate fixes and document (my opinion).
    - *Reviews are not enough*. Learning through example is more effective.
- Outcome
  - MicroBooNE discussing integration and final testing already
    - Code also being moved to the development branch
  - Sample events added to CI infrastructure
    - to be used in test suite to report changes in memory use across releases.
    - Memory tracker service is enough to measure changes

Lessons learned

- Don't clear memory at the front of calls into member functions that cache data.
  - Clear out the cache as soon as the data is not needed
  - Clearing at the front causes the previous event's data to be retained
  - Recommended practice: don't cache data in algorithmic objects
- Complex data structures used for small look-ups cost memory
  - Maps of numbers to vectors
  - Vectors of maps
  - Vectors or vectors
  - Maps when hash tables or simple linear structures will do
  - Maps and vectors as data members of simple structs
- All of these hurt when the number of live instances goes way up, and this is what happened with the additional particle flux (latent data structure problems)
- Request for recommended practices and exercises
  - To be added to any future art / LArSoft course
  - To be given at collaboration meetings
  - This way of making performance improvements motivates people

Summary - performance



- Sample events originally had ~8 GB memory footprint
- After changes, footprint reduced to ~3.6 GB
- Larger scale run needed to reproduce this distribution
- Expected to shift and compress, unclear where that new peak will be
- Would be nice to catch the data structure design issues earlier to avoid a crisis: better communications during design