Implementation of GlobalModuleIndex in ROOT and Cling

Arpitha Raghunandan
National Institute of Technology Karnataka, India
Google Summer of Code 2019 with CERN-HSF
May 27th - August 26th
Mentors: Yuka Takahashi, Vassil Vassilev, Oksana Shadura

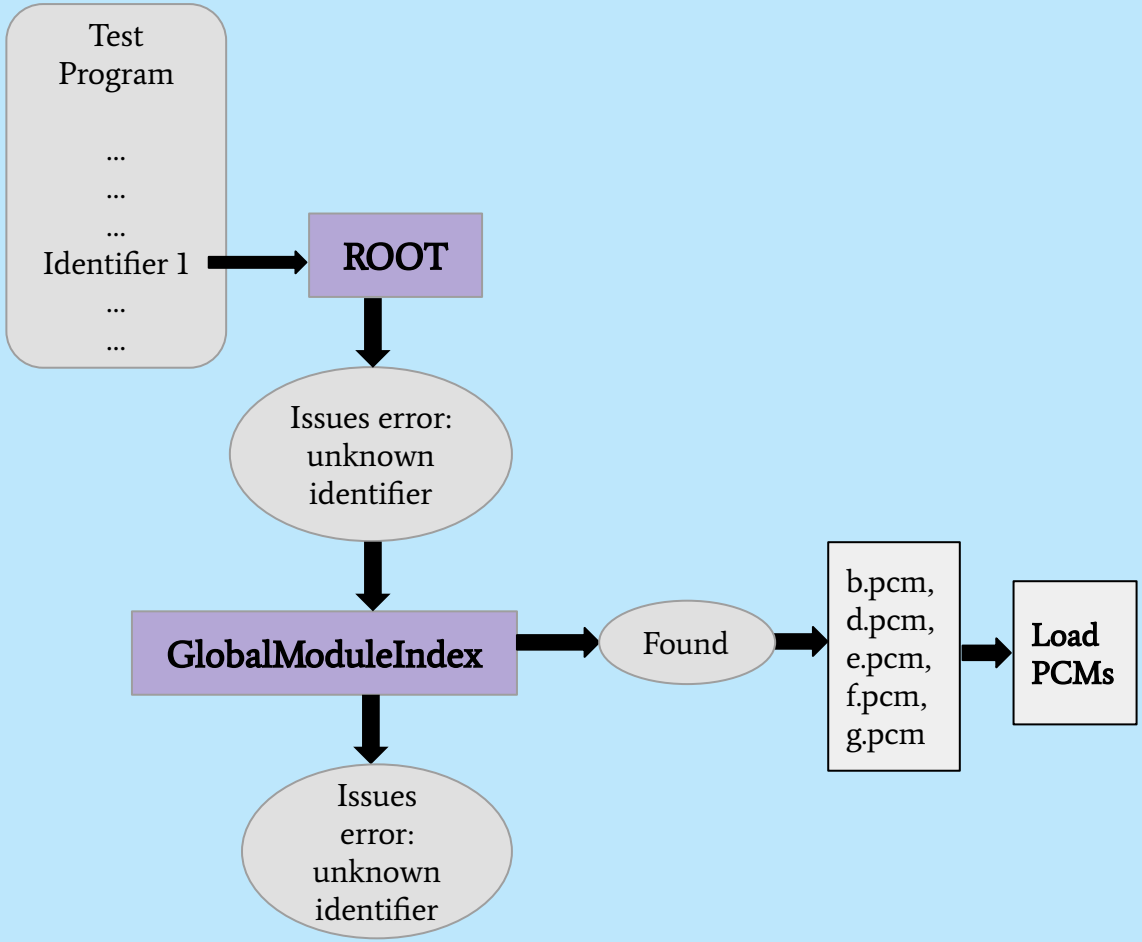
Motivation of using C++ Modules

- *Textual includes*
  - Expensive and fragile
- *PCH or Precompiled Headers*
  - Monolithic
- *PCMs or Precompiled Modules*
  - Solves the above problems
  - Envision the better performance for experiments

Problem Statement: Implementation of GlobalModuleIndex

Current implementation of GMI returns a superset of required modules, and hence is underperforming

GlobalModuleIndex		
1	Identifier 4	a.pcm, b.pcm, c.pcm, d.pcm, h.pcm, k.pcm
2	Identifier 1	b.pcm, d.pcm, e.pcm, f.pcm, g.pcm
3	Identifier 2	b.pcm, c.pcm, g.pcm, h.pcm, i.pcm



Problem Statement: Implementation of GlobalModuleIndex

- **GlobalModuleIndex**: A mechanism to create the table of symbols and PCM names so that ROOT can load a corresponding library when a symbol lookup fails.
- On-disk hash table containing all identifiers present in all PCMs.
- Helps improve ROOT's performance by speeding up its start-up time.

Phase 1: May 27th - June 24th

Comparison of Modules and Identifiers

- Found and compared the (number of) modules loaded both with and without the GlobalModuleIndex implementation for:
  - ROOT start-up time
  - tutorials/hsimple.C test
  - tutorials/geom/geometry.C test
- Found the identifiers which cause the PCMs to be loaded in the above 3 cases, which can be seen here, here and here.
- Found the methods which cause the above PCMs to be loaded in the 3 cases.
- Found a few ROOT identifiers (not in stl and libc) which required the loading of modules

Tests and Performance

- Initially, around 450 tests were failing, which reduced to around 50 by cleaning the code.
- Compared the time and memory footprints of the module implementation with and without GlobalModuleIndex
- Found that without rdict PCMs fewer modules are loaded at ROOT start-up time

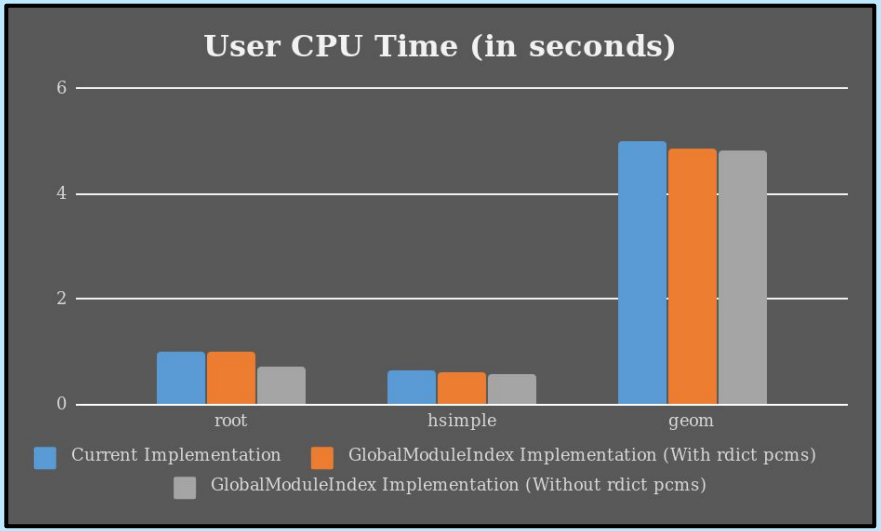
Phase 2: June 28th - July 22nd

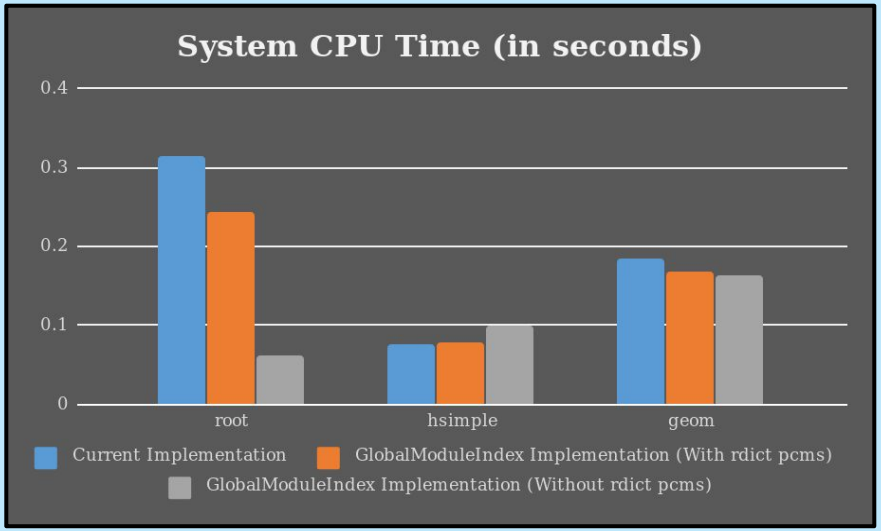
Tests and Performance

- Rdict PCMs were generated by Rootcling to efficiently store information needed for serialization. The GlobalModuleIndex showed better performance without these Rdict PCMs.
- Found the *rdict.pcms that get loaded for:
  - root start-up
  - tutorials/hsimple.C
  - tutorials/geom/geometry.C
- Developed a script to find the modules/libraries loaded by each test/tutorial run by ctest
- Fixed some failing tests with a change, when run with modules on
- Fixed a number of failing tests by adding some modules to FIXMEModules (Temporary solution)

CPU Time Performance Evaluation

There is an increase in System CPU time in the case of hsimple because GMI loads 30 redundant modules, which we know how to fix.

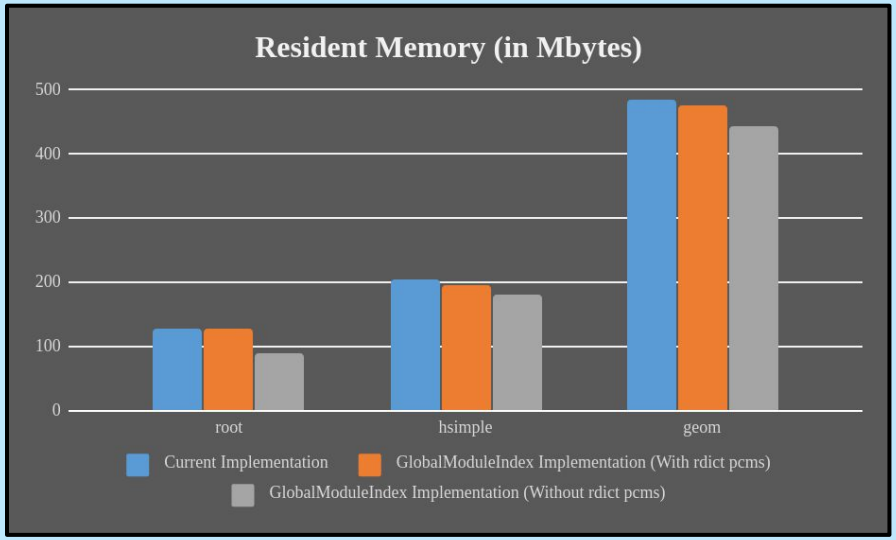


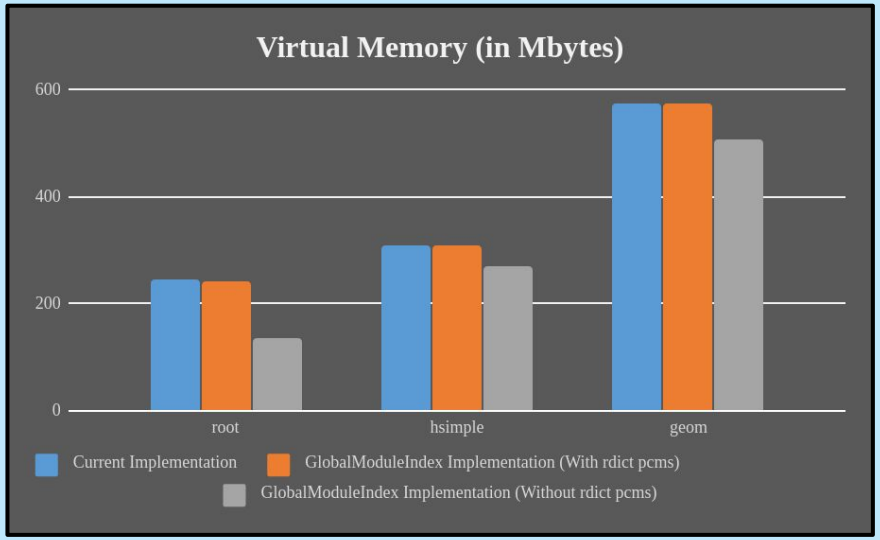


Root :	28% decrease
hsimple.C :	8% decrease
geometry.C :	4% decrease

Root :	80% decrease
hsimple.C :	32% increase
geometry.C :	11% decrease

Memory Performance Evaluation





Root :	31% decrease
hsimple.C :	12% decrease
geometry.C :	9% decrease

Root :	44% decrease
hsimple.C :	13% decrease
geometry.C :	12% decrease

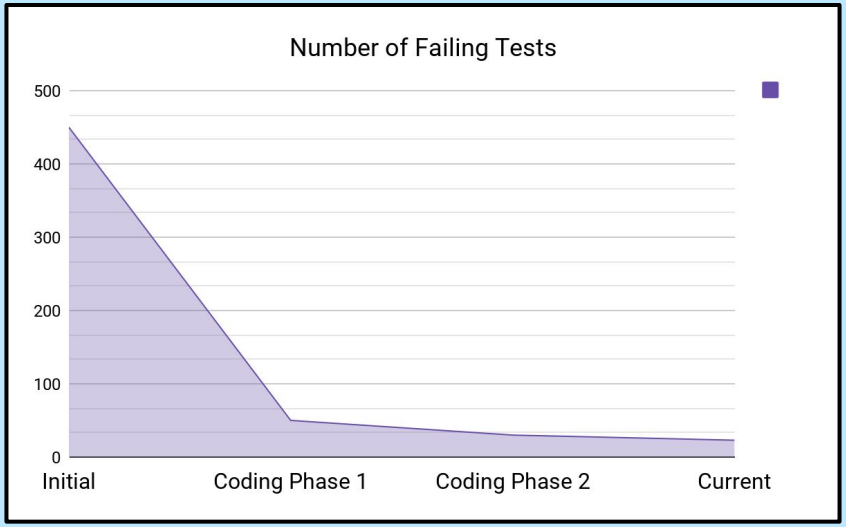
Phase 3: July 26th - August 19th

Fixing Failing Tests

- Fixed a number of failing tests by adding some more modules to FIXMEModules
- Bug in Clang:
  - The issue occurs in HeaderSearch.cpp when both PrebuiltModulePath and ModuleCachePath point to the same folder
  - Working on coming up with a reproducer test for Clang

Performance Evaluation and Tests

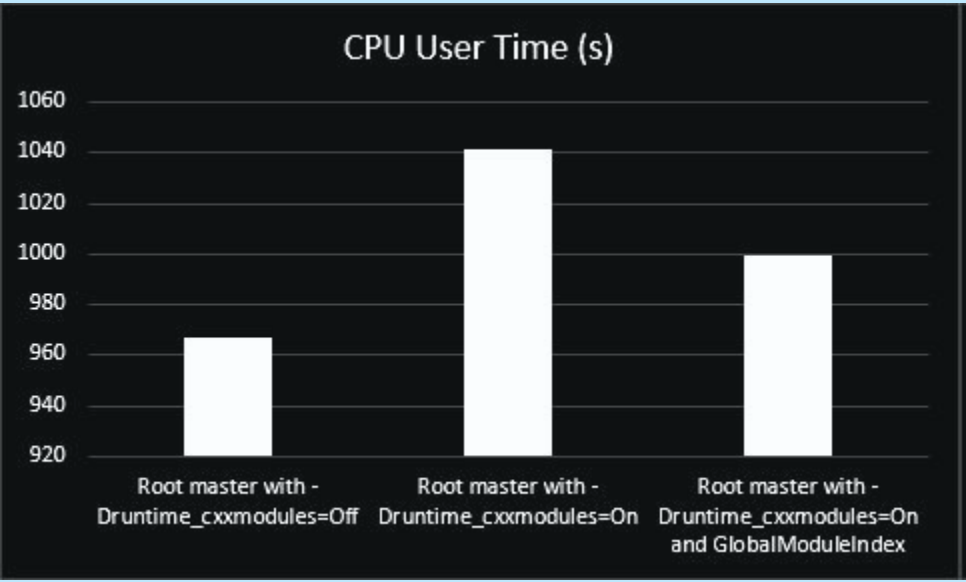
- Difference in number of failing tests over the coding period

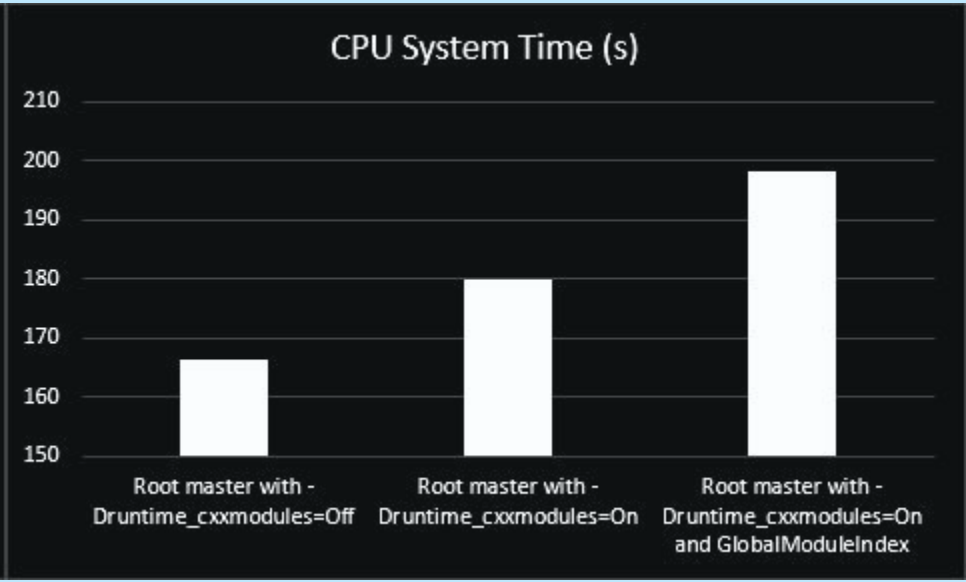


- Developed a script to extract time and memory measurements for each test/tutorial run by ctest
- Latest performance results can be found here and here.

CPU Time Performance Evaluation

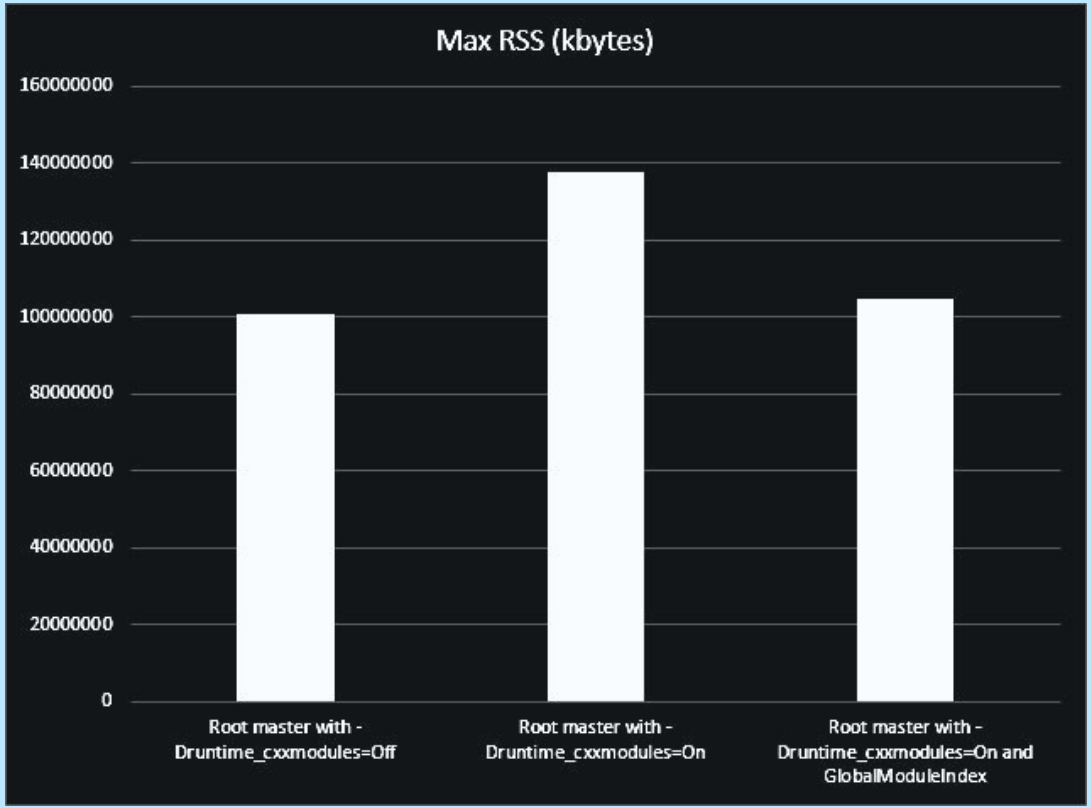
ROOT master uses PCH, which is highly optimized.





The increase in System CPU Time is due to the fact that we are making too many calls to the virtual method which is supposed to resolve a missing symbol

Memory Performance Evaluation



ROOT master with modules and GlobalModuleIndex use almost the same memory as ROOT master with PCH !!

The memory usage can be further decreased as currently a superset of the required modules is getting loaded

Conclusion

- The current GlobalModuleIndex implementation shows promising results in improving performance
- However, the current implementation is still not fine tuned as a lot more modules are being loaded unnecessarily.
- If the identifier → module mapping can be made such that the definition is stored first, the number of modules loaded can be further reduced, further improving performance

Acknowledgement

I would like to thank my mentors, Yuka, Vassil and Oksana for all their help over the summer.

Thank you