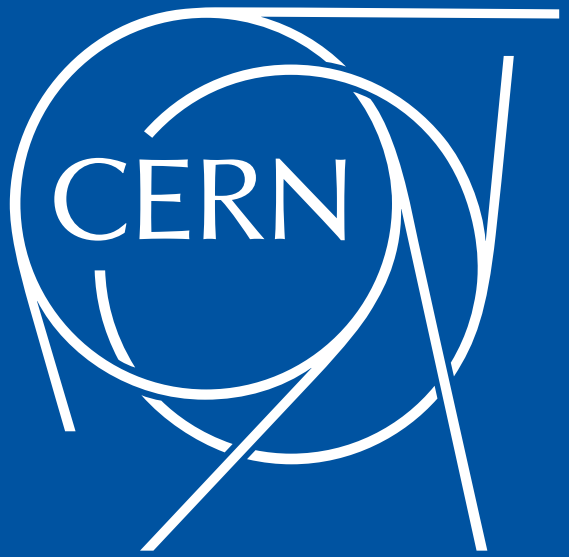
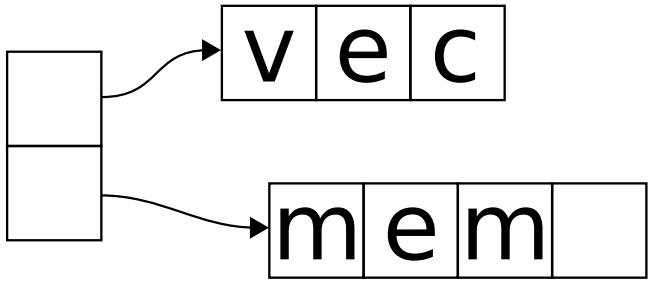

CERN

vecmem: Recent Developments

Stephen Nicholas Swatman
Attila Krasznahorkay
Paul Gessinger-Befurt

Introduction

- vecmem remains under active development
  - 82 pull requests since our last talk in this meeting (March?!)
- Purpose of this talk is to elucidate 'recent' developments in:
  - Compatibility
  - Memory management
  - Testing and continuous integration
  - And more!



Talk schedule

- Since our last talk here, vecmem has been presented in various other places:
  - Heterogeneous Computing and Accelerator Forum (March)
  - KKIO 2021 keynote (September)
  - Parallel Computing Systems group (September)
  - ATLAS Software and Computing HL-LHC Roadmap (October)
- We're hoping to present at the following venues:
  - ACAT 2021 (November)
  - Compute Accelerator Forum (December)

Recap: what is vecmem

- vecmem is a tool to bring the ergonomics of C++ programming to device memory
- Access and modify device memory through standard C++ containers
- Everything you need to create efficient memory allocation schemes

```
1  int main(void) {
2      vecmem::cuda::
       managed_memory mem;
3      std::vector<int> vec(&
       mem);
4
5      // This vector is
       accessible on the
6      // GPU without any
       explicit transfer!
7
8      vec.push_back(5);
9      vec.push_back(10);
10     vec.push_back(2);
11 }
```

Windows compatibility

- vecmem now supports MSVC, and works on Microsoft Windows!
- Buildable as a static library, or as a DLL
- Explicit symbol visibility can positively impact LTO on Linux
- Several documented instances where MSVC catches additional warnings and errors

Testing

- We now have an extensive test suite, with 147 tests
- Based on Google Test
- Rapid stress testing for new resources using ‘data’-driven testing
- These tests have caught more bugs than I am willing to admit...
- Passing the full test suite is a requirement for CI jobs to pass

Arena memory resource

- Gabriel's arena memory resource has been merged (#99)!
  - Fantastic work on incorporating this code into the vecmem model!
- This adds to our toolbox of caching memory allocators to improve performance
- Minor performance problems to hammer out, but available for use right now
- We love to see contributions to the project!

Instrumenting memory resource

- No longer is vecmem only about managing memory, it is also about monitoring that process
- Instrumenting memory resource works as any downstream resource, but captures useful information:
  - Did an allocation succeed?
  - What were the allocation parameters?
  - How long did the allocation take?
- Support arbitrary higher-order functions, so the sky is the limit
- Useful for benchmarking, profiling, debugging, and testing!

Conditional memory resources

- Is the memory management part of vecmem a library? Or is it a declarative EDSL for memory?
- Vastly increased expressive power and design space by adding control flow to memory management!
- Three new memory resources:
  - Conditional memory resource $((\mathbb{N} \to \mathbb{N} \to \mathbb{B}) \to M \to M)$: allocates only if a predicate function is true, fails otherwise
  - Coalescing memory resource $(\forall n \in \mathbb{N} : M^n \to M)$: attempt to allocate using multiple upstreams, and return the first successful one
  - Choice memory resource $((\mathbb{N} \to \mathbb{N} \to M) \to M)$: use a user-provided function to pick the right upstream resource

Minor features

- vecmem now has a strictly enforced style guide. No mess allowed!
- Compatibility with `libc++` has been expanded by providing polyfills for missing functions
- Support has been added for non-container allocations using memory resources
- Support has been added for smart pointers
- We now have a `std::array`-like class for statically sized data on GPUs
- Some additional memory resources which are only there to satisfy my own personal fascination with categories and abstractions

Miscellaneous improvements

- CI has been vastly expanded, supporting a wide range of platforms, and testing extensively on each of them
- Support for jagged vectors, atomics, and other primitives has been improved
- Performance of memory movement code has been greatly increased
- Very robust build system capable of supporting many different models of compilation, and support for recent versions of CMake

Bug fixes

- Of course we have also had to fix many bugs
- Too many too enumerate in this talk...
- Thanks to many of you here for submitting bug reports!

Future developments

- vecmem will continue to be developed to support more heterogeneous platforms, be more robust, and provide a feature set
- Currently, working towards improved HIP support, both in code and in the build system
- Also in talks with the LLAMA team to provide mutual compatibility!

Conclusions

- vecmem is under active development and maintenance
- Constantly adding features to support new use cases, improve existing ones, and to push the envelope of memory management
- Very exciting to see vecmem being used by so many different people in different projects!


CERN