Software and Hardware Support for Programming Heterogeneous Memory

MCHPC'19

CASC
Center for Applied Scientific Computing

November 18th 2019

David Beckingsale

LLNL-PRES-797199

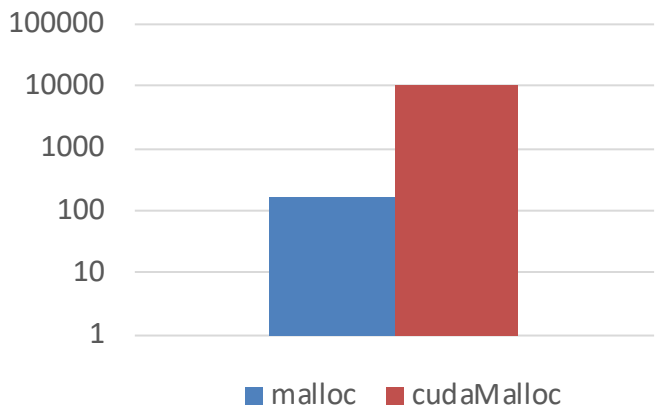
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Heterogeneous memory introduces a number of performance and portability challenges for application developers

Some of the challenges are fundamental issues with hardware/software:

- Limited capacity of high-bandwidth/low-latency memories
- Not all memory locations can be accessed from everywhere
- Increased cost of allocations (2 orders of magnitude slower than malloc)

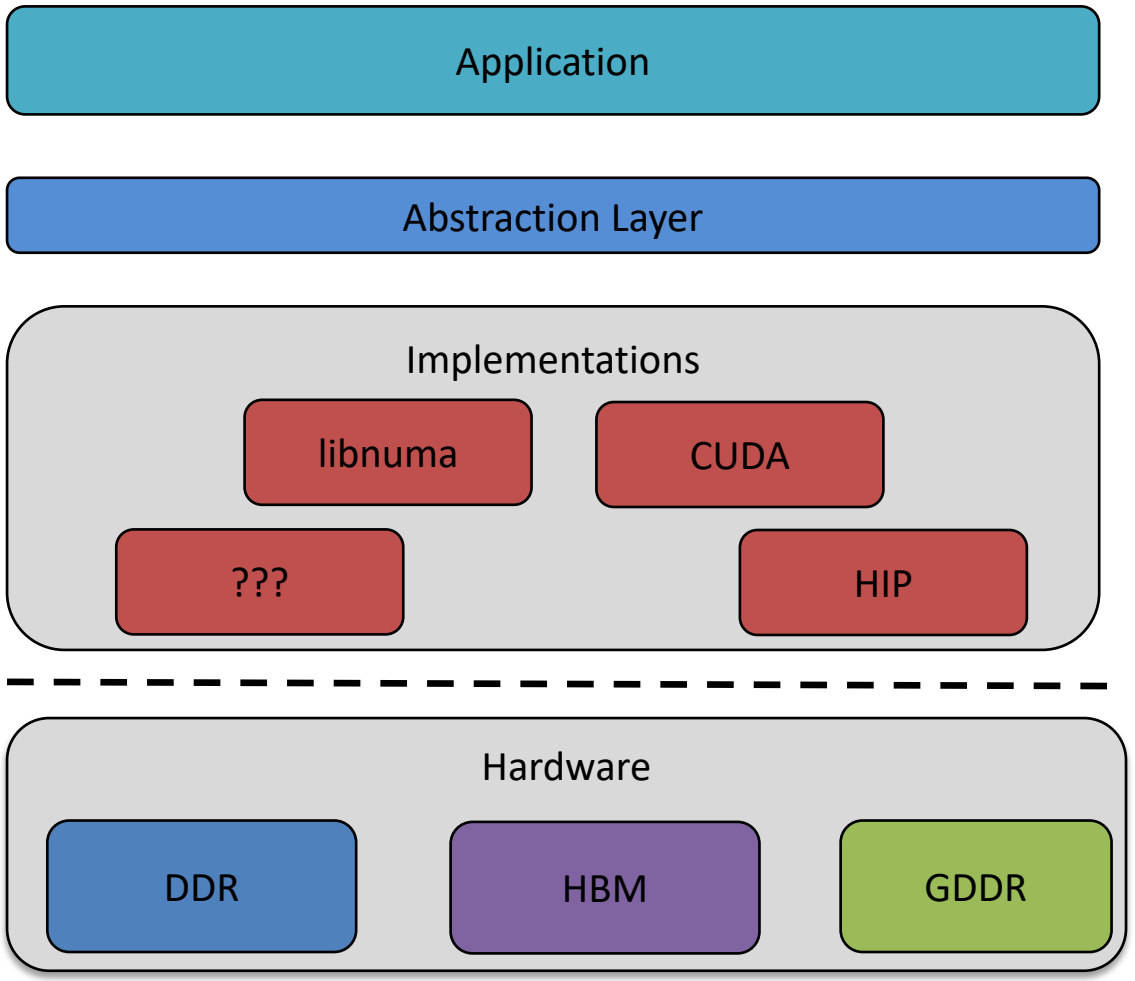


Other concerns are related to software engineering

- Code must be portable to different memory systems
  - Vendor-specific APIs required to access some hardware
- Applications need to leverage the underlying hardware without introducing too much complexity
  - Have to balance this against losing power from being restricted to the "lowest common denominator" feature set
- Memory usage between applications & libraries must be coordinated

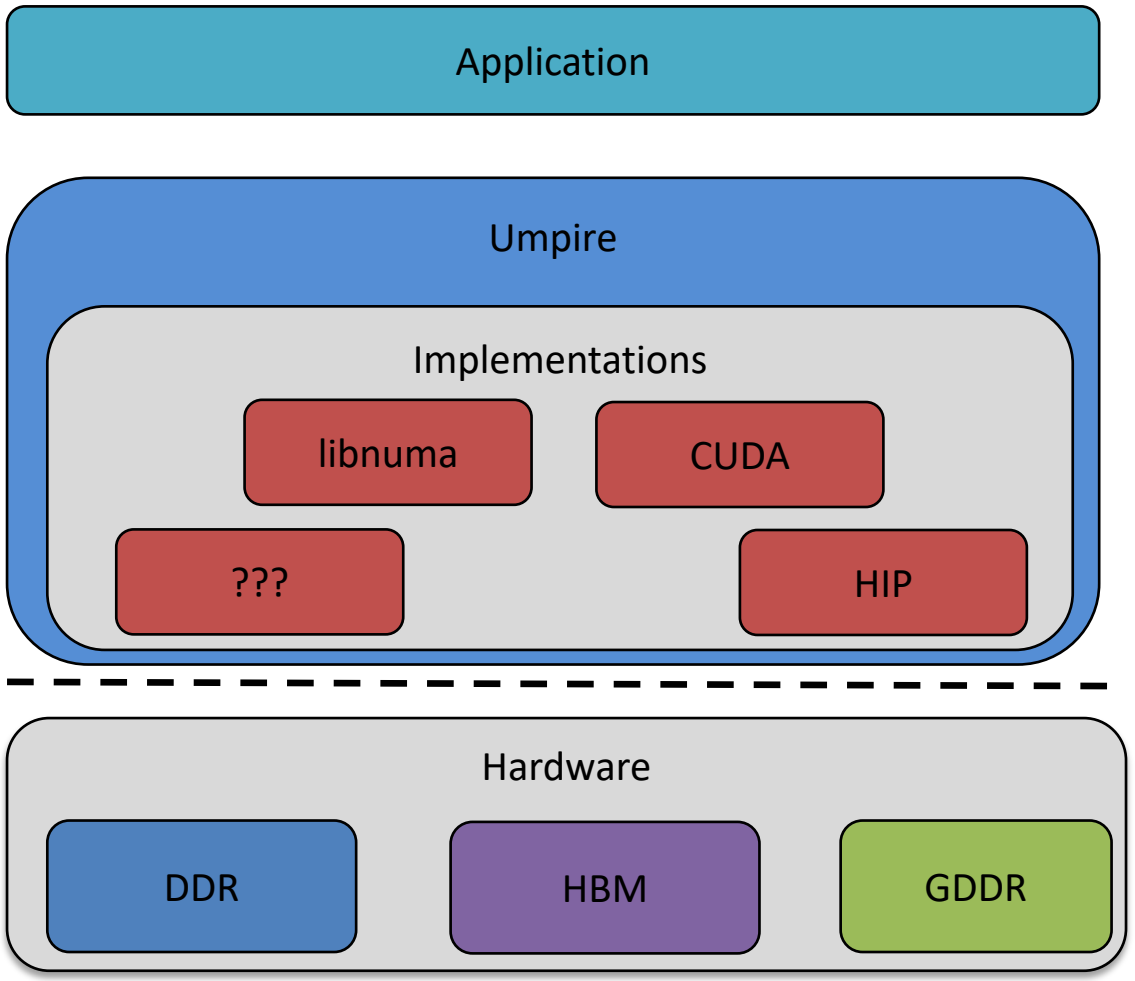
Addressing these challenges requires a powerful and portable layer between the application and hardware

- This layer must be simple, but expose the full features of the underlying hardware
- It should allow applications developers to reason about different memory areas
- Tools should be provided to help mitigate any performance impact
- It must balance simplicity and power



Umpire is a library that provides concepts that address both the fundamental limitations and software engineering challenges

- Provides software abstractions to enable **portability** and **co-ordination** across applications and libraries
- **Reduces cost** of memory allocations using memory pools
- Leverages power of underlying hardware via vendor-specific APIs
- Allows introspection into allocations and kinds of memory to deal with limited space



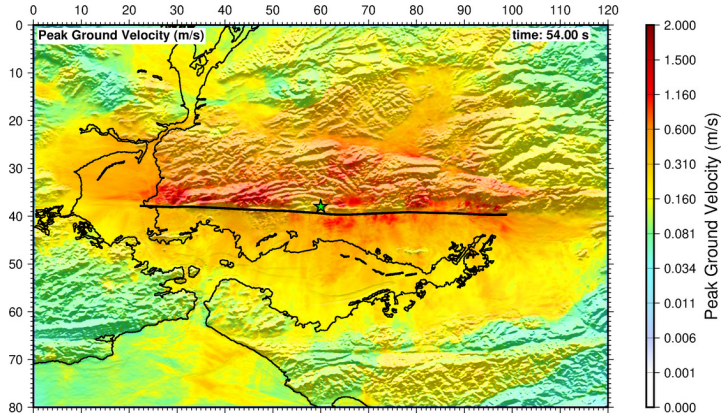
Umpire is being leveraged by production applications at LLNL, running on Sierra

- No single strategy works for all applications, and multiple paths to success have emerged

- Key points across all applications:
  1. Pools used to mitigate cost of allocations
  2. Data partitioned into different "kinds", allocated in different ways
  3. Memory motion is a first-class concern, avoiding memory motion is a key to performance

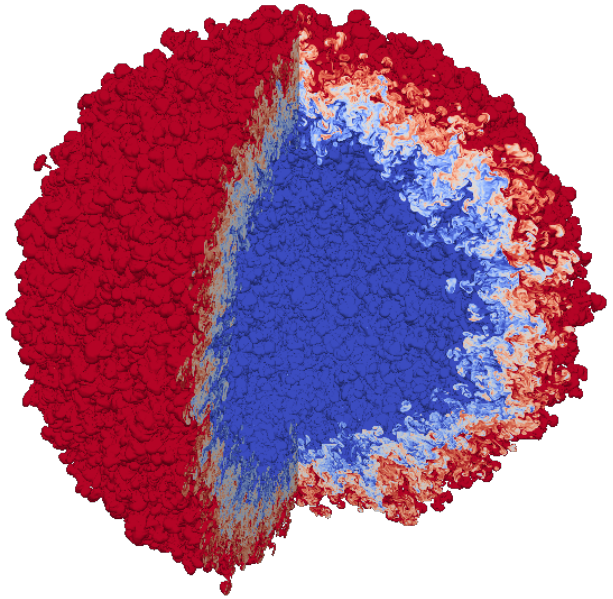
SW4 uses managed memory for transfers, data reuse on device amortizes cost of memory motion

- SW4 is a 3D seismic modeling code that solves the wave equation on Cartesian and curvilinear grid
- Managed memory is initialized on the host and then transferred to the device once
- All kinds of device memory allocated in pools to mitigate allocation costs
- 9% reduction in memory usage, 11% runtime improvement



ARES uses explicit data transfers for performance, with managed memory for libraries and code simplicity

- ARES is a massively parallel, multi-dimensional, multi-physics code
- Pools are used for different kinds of data:
  - Simulation state (stored in unified memory)
  - Temporary data (uses device memory)
  - Communication buffers (pinned memory)
- Managed memory is used for data that needs to move between CPU and device memory
  - But unnecessary transfers avoided at all costs!



Idealized ICF Simulation

ARDRA uses CHAI to automatically move data, but data only moves when used

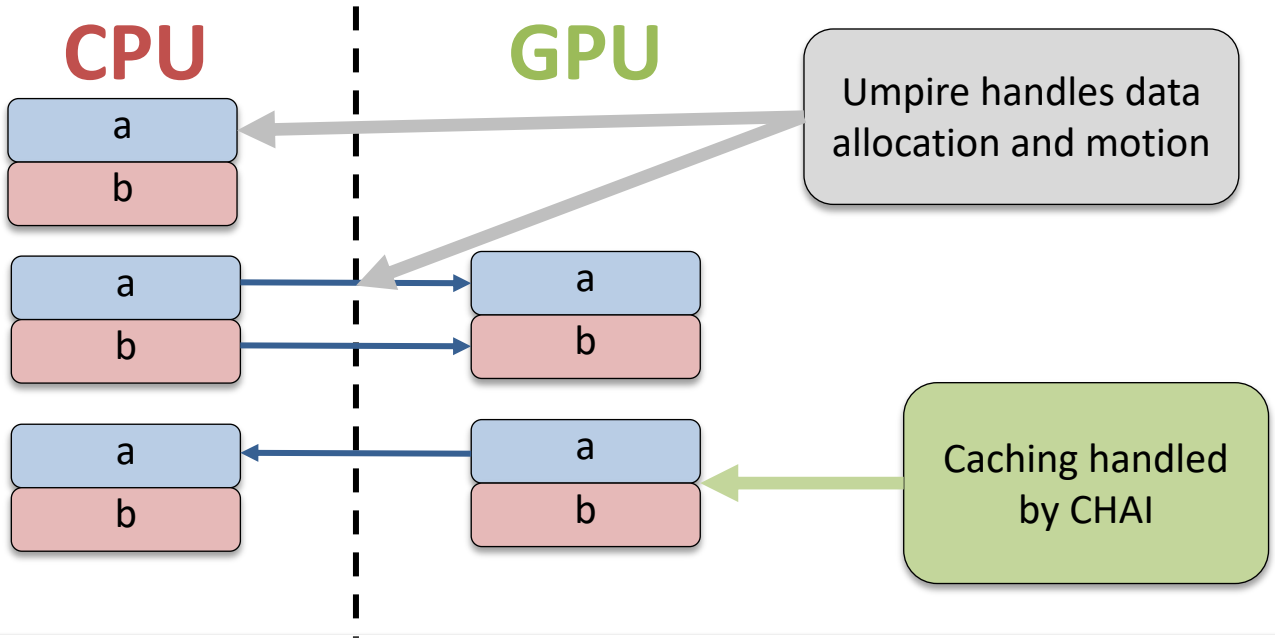
- ARDRA is a 3D Sn deterministic particle transport code
- CHAI provides smart arrays that migrate data between host and device automatically, but use explicit data transfers
- Different kinds of arrays will be allocated using different pools

```
chai::ManagedArray<float> a(100);
chai::ManagedArray<const float> b(100);

const float x = 1.0;

forall<cuda_exec>(0, 100, [=] (int i) {
  a[i] = a[i]*x + b[i];
});

forall<seq_exec>(0, 100, [=] (int i) {
  std::cout << "a[i] = " << a[i];
  std::cout << std::endl;
});
```



Developers need tools to reason about and control memory & data

- Applications can then apply these tools in the way that makes most sense
- “Kinds” of memory and transfers between them become first-class concerns in portable application
- Developers need to think about their data, so it's essential that they have the tools to do so
- Libraries like Umpire provide these tools, and have enabled performance and productivity gains on heterogenous HPC systems at LLNL

CASC

Center for Applied
Scientific Computing