MetaMorph: A Modular Library for Democratizing the Acceleration of Parallel Computing across Heterogeneous Devices

Paul Sathre, Wu-chun Feng
Department of Computer Science
Virginia Tech
Blacksburg, VA 24060
{sath6220, feng}@cs.vt.edu

***Abstract*—Heterogeneity in computing has become ubiquitous. Computing systems ranging from smartphones to supercomputers now consist of multiple types of computing brains, typically at least one multi-core CPU and a many-core GPU. Such systems offer the promise of increased performance and energy efficiency over CPUs if the resources within such systems are used judiciously.**

Alas, extracting the full performance potential from accelerator devices requires architectural expertise, expertise that is in short supply. Thus, there exists a need for software tools and ecosystems to support the development, maintenance, and upgrading of accelerated codes by non-expert domain scientists. To address this need, we present METAMORPH, our initial prototype of a modular library of drop-in accelerated functions, which abstracts current and future accelerator backends behind a single, unified API. By incrementally replacing computational primitives with calls to the METAMORPH API, domain scientists can transparently migrate code to current and future accelerator devices.

I. INTRODUCTION

Heterogeneity of computational devices is rapidly becoming the norm across all computational scales, from cell phones with multicore CPUs and integrated graphics processing units (GPUs) all the way to distributed memory supercomputers with both intra- and inter-node heterogeneity. While this brings benefits of both performance and power efficiency, it also brings with it additional layers of complexity, due to the need to manage several devices of differing capability and internal behavior. Therefore, special expertise in optimizing codes for these rapidly-changing architectures is required to extract the peak performance from accelerator devices.

Unfortunately, the demand for accelerated performance far exceeds the supply of architectural experts. Further, accelerator programming models are often too low-level to be immediately accessible to domain scientists who wish to extend their codes to execute across heterogeneous resources. Therefore, a substantial need exists for software tools to simplify porting legacy codes to modern heterogeneous platforms and developing new accelerated codes from scratch. The METAMORPH library is an effort to abstract away the gritty details of optimizing computational primitives for accelerator devices behind a unified API to a highly modular library.

II. BACKGROUND

METAMORPH was born of the desire to provide a simple mechanism for domain scientists in computational fluid dynamics (CFD) to *incrementally* accelerate their existing codes, by replacing common computational primitives originally written for CPUs with drop-in calls to a portable library of highly-accelerated variants. We have anecdotally observed a key indicator that existing libraries, frameworks, and abstraction layers are not always sufficient for accelerating pre-existing large-scale scientific codes: despite many options, our domain science collaborators still opt to learn the comparatively low-level accelerator programming languages OpenACC, CUDA C/Fortran, or OpenCL rather than deal with the complicated installations and extensive application refactoring that existing solutions require!

We considered several high-profile existing solutions before creating our own library. OpenFOAM [1] and PARALUTION [2] are both powerful solver frameworks but require recasting an application to use complex cases and object types, respectively, which then present a barrier to incremental porting. Magma provides powerful multi-GPU and intelligently-scheduled BLAS and LAPACK algorithms [3], but due to the dependency on external libraries was difficult to install, configure, and tune, and does not yet provide unified or consistent capability across its CUDA, OpenCL, and Intel MIC implementations. Finally, Trilinos has a massive capability set within a multitude of subpackages with some accelerator support and a large user community [4], but it has grown so large that the up-front learning curve turns away new users. We found that another option was needed, one where ease of use, incremental adoption, portability, and forward-compatibility were first-order design constraints.

III. DESIGN PRINCIPLES

Based on our experiences collaborating with domain scientists to develop accelerated software in the CFD domain and beyond, METAMORPH has been designed from the ground up with a few guiding principles. These principles were selected based on the core desire to provide a more useful means for our domain science colleagues to access the massive performance afforded by accelerator devices, while addressing the limitations of existing solutions.

- Modularity and customizability:
  - *Uniform API with mix-and-match backends:* Write an application once, and only build in the backend support needed for your deployment.

  - *Runtime backend selection:* Support selecting the right platform on-the-fly.
  - *Separate optional plugins from core features:* MPI, profiling, and Fortran-compatible interfaces, are useful features for some, but associated overheads should not be forced upon users who do not need them.
- Reduced or eliminated barriers to entry:
  - *The pay wall:* METAMORPH itself will be open source and builds with free compilers.
  - *Cumbersome installation with hidden dependencies:* Bare-bones METAMORPH compiles and runs (serially or with OpenMP) with only a C compiler even if accelerated compute environments are not available.
  - *Massive code refactoring:* METAMORPH provides non-destructive data copies from C buffers just like CUDA or OpenCL, and it is implemented with their native device buffer types, providing more interoperability with hand-written accelerator code than proprietary object types would.
  - *Complex accelerator context and device management:* METAMORPH users need only declare a mode and device (as an integer or string identifier) to initialize METAMORPH. All device/platform, context, and stream/queue management is handled internally.
- Forward-compatibility:
  - *Community-driven tuning for new devices:* By using a "library of shared libraries" approach and open-sourcing METAMORPH's canonical backend, we promote the ability for users to tune backends for new devices and use them as drop-in replacements (without any recompilation of host applications).

IV. PERFORMANCE

As METAMORPH is extremely new, the focus thus far has been on ensuring the stability and validity of core functionality. However, the end goal is to produce a library that is both highly portable and highly performant. Therefore a preliminary investigation of the library's performance characteristics was performed — figures are provided on the poster, but omitted here, to preserve space. First and foremost, our first two hand-implemented dense matrix kernels, a 3D dot product and 2D transpose, were compared with common alternatives. We found that at sufficient data sizes, our solutions — with either a CUDA or OpenCL backend on Nvidia c2070 GPUs — are competitive with those of MAGMA [3] and clAmdBlas [5]. (A PARALUTION version of our benchmark was developed, but various runtime errors were encountered in all three backends: OpenMP, CUDA, OpenCL.) After measuring kernel performance, a brief effort was undertaken to understand *current* overheads associated with adding timing and Fortran compatibility plugins to the core C API, noting several non-intuitive cases where the addition of plugin code actually improved performance, warranting futher investigation. Finally, we sought to demonstrate a feature of our newest plugin — the ability to transparently pack, MPI exchange, and unpack device buffers between OpenCL and CUDA backends and compare the performance cost to our GPUDirect pack-exchange-unpack operation. Unfortunately a deadlock in an MPI_Send/MPI_Recv pair within the plugin prototype prevented testing of exchanges larger than 16KiB when using MVAPICH 2.0 [6], an issue we are currently investigating.

V. FUTURE WORK

In the immediate future, the main focus is to expand the feature set to support the needs of local CFD developers. Primarily, this will consist of providing a richer set of computational primitives underlying core components of CFD applications, i.e., solvers, preconditioners, stencil computations, and support for sparse computations and unstructured grids.

In the more distant future, METAMORPH will be expanded to simplify and further modularize the development of additional plugins and accelerator backends. As a first expansion backend, there is high demand for generic runtime-scheduling that is capable of transparently executing accelerated primitives intelligently across all devices within a node. To implement such a backend we plan to draw heavily upon the CoreTSAR runtime [7], which has already demonstrated such intelligent scheduling capability.

VI. CONCLUSION

METAMORPH provides a new approach to simplifying the development of accelerated codes by domain experts. By providing a uniform API to a modular, "build-your-own" library that abstracts the optimization and implementation details of various accelerator programming models, METAMORPH strives to realize the "write once, run anywhere" idea within a heterogeneous computing context. Further, due to its modularity, it provides a high degree of future-readiness by allowing the community to transparently update its compute backends with variants tuned for new classes of compute devices.

ACKNOWLEDGMENT

This work was funded in part by the Air Force Office of Scientific Research (AFOSR) Basic Research Initiative from the Computational Mathematics Program via Grant No. FA9550-12-1-0442.

REFERENCES

[1] H. Jasak, A. Jemcov, and Z. Tukovic, "OpenFOAM: A C++ library for complex physics simulations."

[2] D. Lukarski, "PARALUTION project v0.7.0," 2012.

[3] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, and I. Yamazaki, "Accelerating Numerical Dense Linear Algebra Calculations with GPUs," in *Numerical Computations with GPUs*. Springer, 2014, pp. 3–28.

[4] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, "An overview of the Trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.

[5] AMD. clMath (formerly APPML). Accessed 2014.10.17. [Online]. Available: http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-math-libraries/

[6] D. K. Panda. MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE. Network-Based Computing Laboratory, The Ohio State University. Accessed 2014.10.17. [Online]. Available: http://mvapich.cse.ohio-state.edu/

[7] T. Scogland, W.-c. Feng, B. Rountree, and B. de Supinski, "CoreTSAR: Adaptive Worksharing for Heterogeneous Systems," in *Supercomputing*, ser. Lecture Notes in Computer Science, J. Kunkel, T. Ludwig, and H. Meuer, Eds. Springer International Publishing, 2014, vol. 8488, pp. 172–186.