Programming Systems for Emerging Architectures

Organization	POC	email	Y1	Y2	Y3	Y4
Oak Ridge National Laboratory	Jeffrey S. Vetter	vetter@ornl.gov				
Los Alamos	Kei Davis	kei@lanl.gov				
University of Oregon	A. Malony, S. Shende	malony@cs.uoregon.edu				
Argonne National Laboratory	Hal Finkel	hfinkel@anl.gov				

1 Overview Description and Impact

Application developers targeting evolving ECP architectures will find it increasingly difficult to address the dual challenges of programming productivity and performance portability without help from integrated programming and performance technology that can address the growing complexity of parallel software generation and tuning for heterogeneous parallel computing systems. In response to the ECP Software RFI, our team proposes to investigate and deliver specific research and development results that would help ECP applications obtain new levels of performance portability across a diverse range of architectures, and ease the transition of existing applications onto these evolving ECP architectures. In four years, our objectives would be

1. Develop programming solutions for emerging architectures (e.g., non-volatile memory, tightly coupled heterogeneous cores) in collaboration with ECP stakeholders;
2. Deploy these solutions along multiple outlets: a) release open source software (e.g., LLVM), b) deploy tools to ECP users, and c) champion their acceptance standards like OpenMP, OpenACC, and MPI, so that they will become part of future procurements (in ECP and elsewhere);
3. Provide integrated tools for comprehensive performance evaluation of programming systems and new technologies on ECP-targeted applications;
4. Promote outreach and education for our solution approaches and production software systems.

Our team has extensive experience and a demonstrated track record of accomplishment in all aspects of this proposed work. We will develop an integrated system (i.e. compilers, runtime systems, debuggers and performance-analysis tools) suitable for deployment in the 2019 timeframe. The experience gained from this development will feed in into vendor collaborations and proposals to standards committees to make key elements of our developed technology commercially available for ECP deployment in the 2023 timeframe.

Developing programming solutions for emerging architectures. We propose to develop the capability to rapidly develop programming solutions using our infrastructure in collaboration with ECP stakeholders (applications, hardware technologies, vendors). This strategy has multiple benefits. By working collaboratively within ECP, we will develop solutions for ECP applications that meet ECP requirements. ECP stakeholders know their applications, their priorities and concerns for performance portability, and scientific goals best. We have a strong track record of developing such solutions, and deploying these solutions within DOE. To achieve this goal of rapid prototyping, we will start immediately with our existing assets: the LLVM infrastructure, OpenARC heterogeneous compiler framework, TAU performance analysis tools, and

We have designed several preliminary methods and tools to help users program these systems. For instance, our OpenARC compiler to generate executable applications for NVIDIA and AMD GPUs, Xeon Phis, CPUs via LLVM, and Altera FPGAs directly from the same source code, has shown that a high-level programming model can mask diverse architectural complexity and provide performance portability. Complementing these efforts are our leading-edge tools for parallel performance measurement, analysis, tuning, and data mining. Together, our research and developments objective is to create performance-intelligent compilers and runtime systems that can be used productively by ECP application developers.

LLVM Compiler Infrastructure. LLVM, winner of the 2012 ACM Software System Award, has become an integral part of the software-development ecosystem for optimizing compilers, dynamic-language execution engines, source-code analysis and transformation tools, debuggers and linkers, and a whole host of programming-language and toolchain-related components. Now heavily used in both academia and industry, where it allows for rapid development of production-quality tools, LLVM is increasingly used in work targeted at high-performance computing. Research in and implementation of programming-language analysis, compilation, execution and profiling has clearly benefited from the availability of a high-quality, freely-available infrastructure on which to build. LLVM is becoming a cornerstone of DOE's HPC ecosystem.

Starting with bgclang on the IBM BG/Q supercomputers, and extending directly to the IBM/NVIDIA CORAL systems. For the latter, all three vendor-provided compilers rest of LLVM components (and for which we're directly investing in IBM's OpenMP 4 LLVM development work). Looking to the future, LLVM will be a critical part of our toolchain infrastructure regardless of which vendors architect our systems.

OpenARC heterogeneous compiler framework. To maintain the functional portability of directive-based high-level accelerator programming models such as OpenACC and OpenMP4, we have developed a high-level intermediate representation called HeteroIR, which encapsulates the common accelerator operations into high-level function calls. During application execution, these function calls are orchestrated on the target architecture by the runtime system. HeteroIR allowed OpenARC to generate nearly identical host code regardless of target architectures, while architecture-specific kernel codes are generated for each target device. To achieve performance portability of heterogeneous computing, we developed 1) OpenACC extensions to support hybrid programming of both the unified memory and the separate memory, and 2) device-aware directive extensions to support architecture-specific features while maintaining functional portability. Preliminary porting of some applications using the proposed OpenACC extensions shows that the extended OpenACC program can achieve nearly identical performance to the manual CUDA version, demonstrating that the proposed directive extensions effectively exploit architecture-specific features. OpenARC will be used for prototyping new functionality for OpenMP/OpenACC and initial application porting to diverse architectures.

Integrated Performance Analysis. Our programming innovations for heterogeneous systems will require a robust measurement and analysis infrastructure for performance characterization and optimization. We will build on and expand the TAU Performance System's rich features for major parallel programming models and machine platforms to integrate performance evaluation capabilities in the OpenARC, bgclang LLVM, IMPACC, and NVL-C frameworks. TAU's broad support for leading programming languages (C/C++, Fortran, python), shared and distributed memory parallelism (OpenMP, MPI), and accelerator/coprocessor devices (GPU, Xeon Phi) and development (CUDA, OpenACC, OpenCL) will enable deep integration of TAU technology. In particular, LLVM is an excellent infrastructure for implementing TAU's source analysis and automatic instrumentation mechanisms. The TAU Performance System, developed at the University of Oregon, has been been delivering robust software for performance evaluation and our team's deep knowledge and expertise in parallel tools would contribute to ECP technologies.

1.1 Tentative Milestones

(reorganize by emerging technology; uuuuuuuud)

Y1-4: Add the ability for LLVM's Clang frontend to combine information from runtime profiling, from the backend optimizer and from static analysis to suggest to users changes in their use of OpenMP and/or OpenACC directives, and data-structure reorganizations, to enhance performance. In later years, this will include integration with the developed NVRAM support.

Y1: Add specific optimizations to LLVM's backend which, using a built-in understanding of the semantics of certain OpenMP and MPI runtime calls, remove redundant barriers, and other expensive, but redundant, operations. In later years, this will also enable parallel-region fusion and other higher-level parallelism-overhead-reducing optimizations.

Y2: Add support to LLVM to represent the concept of "slow" memory (appropriate for modeling NVRAM on upcoming CORAL systems), and add the ability to generate memory profiles at runtime (either through instrumentation, sampling, or both), so that compiler transformations can intelligently allocate frequently-used memory in on-package memory and other memory in NVRAM.

Y3: Continue developing support for the "slow" memory concept in LLVM, and use it to model unified memory on accelerator systems. Teach LLVM's OpenMP 4 accelerator runtime to make use of unified memory, avoiding preemptive inter-device copies (based on profiling data) and making use of asynchronous copy hints for other data as appropriate. Experiment with extensions to OpenMP 4 exposing these more-convenient semantics to users.

Y1-4: Develop LLVM-based representations for higher-level semantic concepts such as various forms of parallelism and communication/synchronization to support parallel programming constructs such as parallel for and tasking, augment LLVM such that the semantics is respected through various transformations, implement analysis and optimizing transformations in terms of these semantics, and lower these representations to specific architectures and runtimes.

Y1-4: Identify and port DOE proxy applications, and other applications of interest, using a variety of LLVM-supported programming models, to LLVM's test suite to allow easy and continuous tracking of correctness and performance.

Y1-4: Using concepts from OpenARC's IR, and in collaboration with Intel, IBM, and other LLVM contributors, work on integrating higher-level parallelism semantics in LLVM's backend. Use this new capability to enable post-inlining and profiling-data-aware code-generation decisions for parallelized and accelerated code.

Y1: 1) Add/harden support for OpenMP4, and 2) develop more directive extensions and corresponding compiler/runtime passes to exploit architecture-specific features

Y2: 1) Add support for the latest OpenACC standard (V.2.5), and 2) develop compile-time and run-time optimizations to improve performance of NVL-C and IMPACC for CORAL, Summit, and expected exascale system designs.

Y3: 1) Update interfaces to the backend LLVM to the latest version, and 2) add Fortran front-end.

Y4: 1) Contribute LLVM components to LLVM projects, and 2) evaluate and improve OpenARC by porting key DOE applications to diverse architectures.

Y1: 1) Add support for OpenACC data transfer API in TAU, and 2) Provide initial support for NVLINK based transfers using CUPTI API in TAU, and 3) Improve support for C++, C, and Fortran parsers in PDT.

Y2: 1) Update support for OpenMP 4 and fine-grained compiler-based instrumentation in TAU to support LLVM on ECP early systems, and 2) Evaluate performance of ECP applications and create a TAUdb performance database, and 3) Update OpenARC OpenACC compiler support in TAU.

Y3: 1) Add support for LLVM Fortran compiler, and 2) Update support for evaluating effects of NUMA aware hierarchical memory management (including MCDRAM based systems) in TAU, and 3) Profile ECP applications and store data in TAUdb, and 4) Update TAU's ParaProf and PerfExplorer tools for improved scalability.

Y4: 1) Provide performance evaluation tools integrated with LLVM and OpenARC for comprehensive performance evaluation of ECP applications, and 2) Demonstrate support for profiling of unified memory, pre-emptive inter-device copies, and asynchronous copy hints for data, and 3) Evaluate performance of ECP applications using LLVM and OpenARC compilers using TAU.

2 System Requirements

Testbeds. To develop and evaluate our prototype software, we will need access to smaller testbeds hosting emerging technologies. We will typically need to install drivers, reboot, and tightly administer these systems. Once our software has been proven in this development stage on these testbeds, we will migrate our testing to the pre-Exascale DOE systems like CORAL and APEX, and to the ECP early systems.

ECP early delivery systems

operating systems and drivers

3 Technology Maturity

OpenARC. Maturity: early beta version (multiple external users including industry and research teams). Known gaps: limited support for languages other than C.

LLVM. Maturity: production quality infrastructure used as the basis for numerous commercial products (Apple's XCode, ARM's Compiler v6, Qualcomm's Hexagon SDK, Azul's JVM, etc.) and collectively maintained by many companies, other organizations, and individual contributors. Clang, LLVM's C/C++ frontend, is arguably the industry's best, and LLVM's OpenMP runtime library is based on Intel's production runtime.

TAU: Maturity: used in production across all DOE HPC systems, including the leadership class facilities and supported by commercial ISV and hardware vendors. TAU is being widely used on DOE applications for performance analysis and

optimization. TAU technologies are also being integrated with programming environments, runtime systems, and system software targeting large-scale computing.

3.1 Strategy for Technology Transfer

As mentioned above our strategy for technology transfer is multifaceted. We will deploy our solutions as 1/ open source software, 2/ tools directly to ECP users, and 3/ champion their acceptance in standard forums like OpenMP, OpenACC, and MPI, so that they will become part of future procurements (in ECP and elsewhere).

Open source. Our team has a strong commitment to release-quality, open source software development as demonstrated in the earlier deployments (SHOC, Aspen, OpenARC, TAU, LLVM, Scout, etc.) Source codes for software deliverables have been, and will be, openly available for use and modification throughout the scientific computing community. The open source license used may vary across deliverables because our work builds on existing software (with existing licenses), but it will typically be one of the licenses approved and certified by the Open Source Initiative (http://www.opensource.org/), such as the classic LGPL, BSD, and MIT licenses.

ECP user interaction.We will work collaboratively with the ECP community in order to deploy, apply, and refine our tools.

Championing standards.

4 Cross-team Collaboration/Integration

Our team has an extensive record of working with DOE applications teams including ExMatEx, CESAR, ExaCT, XPRESS, Vancouver, ARGO, ASCR PRIMA-X, SDMAV MONA, and SUPER projects.

5 Related Research

6 Other Considerations/Issues