D-TEC – DSL Technology for Exascale Computing

Progress Report: March 2013
DOE Office of Science Program: Office of Advanced Scientific Computing Research
ASCR Program Manager: Dr. Sonia Sachs

1 Introduction

Work done supporting D-TEC goals is addressing the full breadth of the X-Stack program. Our project is separated into seven coupled areas each with their own staffing that pulls from the personnel within the D-TEC project. Most work is focused on the support for construction of DSLs and related to the many aspects of this work within the Exascale program. Some work is related to features that we expect to embed into DSLs constructed using the approaches being developed.

2 Project Wide Activities

Work with Exact Co-Design Center Work with the Exact Combustion Co-design center has developed a runtime system for many-core processors that permits us to explore the lack of cache coherency that we expect in future exascale processors. This work defines radical transformation requirements for typical code, recent work has automated some of the transformations as part of work with the University of Colorado at Colorado Springs (Qing Yi).

Work with simulators on power management As part of compiler support to optimize power usage in HPC applications we are using the Thrifty Simulator in collaboration with Josep Torrellas (Thrifty Exascale Architecture project PI and also a part of the Intel X-Stack project) and have developed resource detection capabilities on the compiled binary executables to drive the static analysis required to control hardware resources. This work is part of work on a new API developed to support management of power via compiler analysis and power transformations. This is a Thrifty simulator specific translator which can translate directives to call Power API functions provided by UIUC's simulator. The translator also has a resource analysis capability to statically figure out floating point and integer function units used by a portion of source code. This is required because the binary executable is what the hardware (or simulator) sees, resource usage on the binary is the only precise approach, and significantly more suitable than the source code or compiler intermediate representation (IR), since the binary executable is the only piece that accounts for all possible optimizations done.

Work on Compositional Dataflow Framework Work on program analysis has focused on a new compositional dataflow framework. This framework supports various types of symbolic dataflow and abstract interpretation on top of both dense control-flow graphs as well as sparse SSA graphs. Further, it enables various separately-implemented analyses to take advantage of each others' results without accessing each others' internal abstractions by formalizing a common interface that transparently encodes the results of most analyses. Work on this framework was begun by Greg Bronevetsky as part of the Compiled MPI X-Stack 1 project and has broadened to now include others at LLNL and Rice. This work will be continued within D-TEC and represents a collaboration with Rice and LLNL. This work is released in ROSE and is also under extended development. Program analysis will be an essential aspect of compiler support for DSLs in later years.

Work with network simulator on scaling As part of work with the CoDEX Exascale Architecture project, we have been supporting the development of automated skeleton generation; extracting smaller codes that isolate specific large scale characteristics (e.g. MPI message passing). This work is released in ROSE, and constantly improved as part of iterative software commit policy. This skeleton generation tool generates simulator inputs to drive how research groups can leverage hardware simlators within the co-design process. This work is a collaboration with the Galois Inc, in Portland, the ROSE project at LLNL, Sandia, and LBL which leads the CoDEX Exascale architecture project.

3 Recent Publications

Bamboo: Translating MPI applications to a latency-tolerant, data-driven form T. Nguyen, P. Cicotti, E. Bylaska, D. Quinlan and S. B. Baden http://cseweb.ucsd.edu/groups/hpcl/scg/papers/2012/Nguyen12-Bamboo-SC12.pdf

Portable Performance on Heterogeneous Architectures. Phitchaya Phothilimthana, Jason Ansel, Jonathan Ragan-Kelley, Saman Amarasinghe, In the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems, Houston, Texas, March 16-20 2013.

4 Advisory Board Meeting

March 15th will be our first advisory board meeting. We can add some comments about this after it is over.

5 Progress Within Different Sub-teams

The management of our project is divided into seven areas each working together by a combination of defining API, sharing team members, or defining specific couplings. The overview document on the D-TEC web site (www.dtec-xstack.org) explains each of the areas in greater detail, this document describes only specific progress in these areas.

5.1 Rosebud DSL Framework

Updated plan. We will develop Rosebud as a sequence of fully operational versions, focusing in Year 1 on the core technologies of mixed-language parsing and rewrite-based code generation. Generator and Translator applications will implement the complete Rosebud front end but use a simple deterministic pure-Stratego rewriting system, deferring cost based search and integration with ROSE analysis and transformation. Year 1 Rosebud will support C++, Fortran, and X10 as host languages. Our first exemplar will be a stencil DSL based on previous work by OSU. IBM will investigate X10-based DSLs and we'll build a set of toy DSLs to study key technical issues in isolation at low cost.

Design work. We've started defining Rosebud's input language, chosen open-source components to serve as Rosebud's foundation, prepared a detailed design of the Rosebud front end, and defined requirements for adding extensible AST support to ROSE, and are tackling the difficult issue of mixed language type checking. Stratego rewrite rules will operate on a ROSE-provided view of AST nodes as ATerms, the various DSLs' rewrite systems will be run in round robin, and rules will access ROSE semantic analysis as annotations on ATerms.

Implementation status. We've built the skeleton of an entire Rosebud system and are implementing two-phase parsing within it. A Fortran 2008 grammar in SDF is almost done and work on C++ and X10 grammars will begin next month. Based on the stencil language definition and prototype implementation contributed by OSU, we are preparing an SDF grammar and designing a Rosebud plug-in.

5.2 Refinement

We will be leveraging the Sketch Synthesis infrastructure to support manual and semi-automated refinements from high-level to low-level code. The plan is to develop a bridge between the Sketch synthesis infrastructure and Rose so that synthesis functionality can be leveraged directly from Rose in lowering high-level code down to an efficient implementation. In order to support this effort, we have been refactoring our code-base in order to make the interaction between the two infrastructures cleaner and simpler.

On the technical side, we have developed a new algorithm to help support modular synthesis by allowing complex functionality to be abstracted away behind simple interfaces. This new algorithm allows large complex synthesis problems to be broken down into simpler problems, enabling us to tackle problems that were too big for the synthesizer to solve. The new algorithm has been implemented under the Sketch infrastructure, and we are currently in the process of benchmarking it against some of our benchmarks. The ability to reason modularly about large problems will be instrumental in supporting refinement of most interesting solvers and kernels.

Finally, we have implemented a new encoding strategy in the Sketch solver to reduce the memory footprint of the solver, which will improve our ability to scale to complex algorithms.

5.3 Compiler Extensions

Recent work has extended the C and C++ work within the ROSE compiler framework to support C11 and C++11, this work has also made the C and C++ support more robust. CUDA support within ROSE is supported using extensions to the EDG front-end, OpenCL support has been supported using a connection of the LLVM/Clang project to ROSE; allowing alternative use of LLVM/Clang instead of EDG for C and OpenCl language handling. Additional work has extended ROSE's OpenMP implementation to support the latest OpenMP Accelerator Model directives for NVIDA GPUs. ROSE now can translate code with OpenMP Accelerator Model directives (latest draft specification) into CUDA code, with support for GPU data handling, kernel extraction, execution configuration, reduction operation, and shared data regions. The implementation involves extensions to parsing, IR, translation (especially out-liner to generate CUDA kernels), and the XOMP runtime library. This work complements the UCSD Mint Compiler released in ROSE by Scott Baden and Didem Unat which translates C to CUDA, released in ROSE.

We have added preliminary support for X10, enough to allow the work to add X10 language support to ROSE to get started with IBM. We expect this work may take advantage of the Java language support in ROSE, since the X10 grammar is similar to the Java language grammar.

Resiliency research work has included recent work to define now we will leverage binary analysis support in ROSE to define formal models of resilience for application kernels. This work is ongoing and we will be evaluating its ability to scale to different sizes of applications. This work defines a set of a few dozen primitives upon which the resilience of applications can be described. This work leverages vendor compilers to generate the binary executable from whole programs, and then operates directly on the binary using binary analysis capabilities in ROSE. This formal model will help us evaluate metrics for resilience defined on application programs and mapped to processor hardware specifics. This work is a collaboration with the ROSE project, Jacob Lidman (student), and Sally McKee (professor) at Chalmers University, Sweden.

Work with the HPCompose project (an X-Stack 1 project) has allowed Mini-termite, a ROSE connection to Stratego, to be reworked and released in ROSE. This work connects ROSE to a formal rewrite system

(Stratego) to support the evaluation of this technology for use in DSL compilation by the D-TEC RoseBud framework. Within ROSE, we have implemented a similar AST matching language as is used internally within Stratego. This represents an important piece of a formal rewrite system research, enabling us to better understanding formal rewrite systems.

Recent update of LLVM support to LLVM version 3.2. This work supports generation of LLVM directly from ROSE IR for the C language. This work was first developed at Rice as part of the DARPA PACE project and is now supported and released in ROSE as part of D-TEC supported research work to provide connections to LLVM-based tool chains within the Exascale program.

Work on a high-level DSL for stencil computations (SDSL) has developed the syntax and semantics of the language, together with compiler transformation strategies for multi-target code generation. SDSL provides an exemplar of a language to be implemented in the Rosebud framework, raising issues such as the choice of abstractions, interoperability with host languages, and code generation that takes advantage of domain knowledge and architecture characteristics. At the back end, an automated code generator has been developed to create high-performance code for GPUs like Nvidia Fermi and Kepler from DSL stencil specifications. Work is in progress to integrate data layout transformation along with "split-tiles" based time-tiling in an automated code generator for short-vector SIMD architectures such as the Intel Xeon Phi. A third target for code generation from SDSL is FPGAs. Optimizations have been implemented in the PolyOpt polyhedral transformation system in Rose to generate suitably transformed output code from SDSL input that synthesizes high-performance FPGA implementations by coupling with the AutoESL high-level synthesis tool from Xilinx.

5.4 Abstract Machine Model

In preparation for beginning the design of a machine model for D-TEC, we have begun to study the capabilities of existing low-level machine models used for code generation, including the machine model used by Reservoir Laboratories R-STREAM compiler and machine models used for GPU code generation, as well as higher-level semantic models previously used to guide DSL rewriting in the Tensor Contraction Engine.

5.5 Runtime

The initial focus of the Runtime team is to evolve the X10 runtime system into the APGAS (Asynchronous Partitioned Global Address Space) Runtime that will support the APGAS programming model for a range of languages of interest to DOE (not just X10).

The X10 runtime is mostly implemented in X10. Therefore the early focus of the team has been on reorganizing the X10 runtime to reduce the its usage of the X10 standard library, thus simplifying the task of ensuring that the core X10 runtime can be run integrated in a C++ application with manual memory management. The work has been underway since mid-January and is expected to complete by the end of April (it will appear in the X10 2.4 release targeted for late Spring 2013).

Progress has also been made in porting the X10 runtime (and thus X10) to run on BlueGene/Q systems, with a functional port completed and running at small scale.

5.6 Tools

The D-TEC project plans to provide tool support for four purposes: analysis of legacy codes to identify opportunities for leveraging DSLs, migration support for rewriting aspects of legacy codes to leverage DSL technologies, mechanisms for tracking and reporting analysis results and rewriting transformations to provide DSL developers insight into the process of DSL compilation, and end-to-end performance analysis that will relate performance characteristics of generated code back to source code that includes fragments specified using DSLs.

For modernization of legacy code, we have devised transformations for MPI collectives that enable us to interleave computation and computation at a finer granularity than would be possible if we treated collectives as monolithic calls. Evaluation of this approach is ongoing.

To support performance analysis of DSL-generated code, ROSE imports performance data collected by Rice University's HPCToolkit performance tools and uses line map information present in an executable to guide annotation of ROSE's intermediate representation with performance information. Today's mapping capabilities suffice for relating performance information to code that is the final result of DSL compilation. We have begun to contemplate tools requirements for tracking analysis, code transformations, and mappings between code fragments as they are produced and consumed by transformations during DSL compilation. This process will continue as the project team begins to flesh out the design of the analysis and transformation capabilities for DSL compilation.

5.7 Applications

Still need input from Phil/Brian.