Milestone 2 Status Report

Traleika Glacier X-Stack (DE-SC0008717)

March 1, 2013

Acknowledgment: This material is based upon work supported by the Department of Energy [Office of Science] under Award Number DE-SC0008717.

Table of Contents

Executive Summary

Essential infrastructure work, much needed to conduct research, is making good progress with ETI's release of the functional simulator with rudimentary timing model, and an open-source port of LLVM to generate code for the straw-man architecture by Reservoir. Encouraging result from the R-Stream compiler demonstrates 40% speedup for a 3-D heat equation kernel. The open community runtime is gaining acceptance. Delaware has studied the scope of self-awareness in the execution model, and now underway to outline the research plan. Illinois has preliminary plans for the API for HTA (hierarchical tiled arrays) and an API for software managed (incoherent) caches. PNNL has released the SCF code for the community, and UCSD has instrumented Nekbone application as a warm up exercise.

#	Due	Milestone	Lead
1	11/30/12	Architecture V2 spec & preliminary apps kernal identified for evaluation	Intel
2	3/1/13	Simulators V2 functional, tools (C + binutils) in place, IRR V1 identified	ETI, Reservoir
3	5/31/13	Selected kernels evaluated for 0(compute)	Intel
4	8/30/13	Basic timing in simulator, intelligent scheduling in Exec model, tools (LLVM, etc)	ETI, Rice, Reservoir
5	11/27/13	Selected kernels evaluated for 0(com), select apps coded with PGM system for IRR	UCSD
6	2/28/14	Architecture V2.5 spec, system evaluation of V2.0	Intel, UIUC
7	5/30/14	Simulators V2.5 functional, tools for V2.5 released	ETI, Reservoir
8	8/29/14	System evaluation of V2.5	UIUC
9	11/26/14	Arch V3.0 spec, selected apps evaluation with Exec model & PGM system for V2.5	Intel, UCSD, Rice, Reservoir
10	2/27/15	Simulators V3.0 functional, tools for V3.0 released	ETI, Reservoir
11	5/29/15	Release OCR (Open Collaboration Runtime) V1.0	Rice
12	8/28/15	Evaluation of all X-Stack technologies and report	Intel

Intel (Shekhar Borkar)

Accomplishments

Much of Intel's focus has been on finalizing the straw-man architecture such as instruction set, salient features needed in the hardware, and then capture them in the functional simulator. The Intel team worked closely with our partners on building the infrastructure and tools, such as the simulator, LLVM, and Binutils, creating the infrastructure for software evaluation. Most notably, the team has proposed a simple timing model for the simulator which is good enough, and is being implemented by ETI. The team also worked closely with Rice University to further investigate open community runtime (OCR), and determined role of Intel's research runtime.

Status

We helped ETI and Reservoir meet the second milestone, and are on track for the next milestone of evaluation of kernels.

Issues

None.

Plans

We plan to work with Rice on porting a version of OCR on the functional simulator which will help evaluate the software stack in the future, and will help us evaluate select kernels for the next milestone.

ET International (Rishi Khan)

Accomplishments

ETI's milestone was to create an initial version of a timing model for FSim. We have implemented a Lamport-clock-like model to track time in the distributed system. Since all our times are calculated at a uniform frequency, our system can accurately track time between agents with different clock speeds, and allows for dynamic clock speeds within an agent. Time within an XE is calculated per-instruction, taking into account the static cost for an instruction, along with the cost of any network messages sent. We calculate the cost of sending network messages based on the hops between the sender and destination. This calculation can be overridden, allowing a user of the simulator to define their own method for determining this cost.

Besides the timing model, we have made several other improvements to the simulator, such as support for multiple configuration files and improvements to the client allocation algorithm, in addition to general code cleanup.

Status

The timing model we implemented gives a rough estimate of time passing in the simulator. This is sufficient for tracking causality between network messages, as well as determining relative performance of simulated code. We have a few ideas on future enhancements to this model, including clock drift limiting and better network modeling, but we are waiting on feedback from initial users as to whether this functionality would be useful or desired.

Issues

Our largest issue was the inability to ensure the correctness of our implementation. Due to the lack of regression tests for the simulator, our only means of verification was to run a few small programs and confirm that the output was as expected. We did not have enough computing power to simulate real programs on a decent-sized system, or assess the scalability of the current implementation. In addition, differences between ETI's machines and Intel's machines exposed bugs that were present on one system but not the other.

Plans for next milestone

Now that we are making large modifications to the simulator, we feel it is necessary to implement a regression test suite. This will allow us to spend less time on manual testing when making future changes, and will help ensure that we are not breaking any current functionality.

We will also be taking into account feedback from users of the simulator on what tracing information would be interesting to them. In its current state, the simulator outputs an immense amount of tracing

information, and this information can only be enabled or disabled as a whole for a particular unit. We would like to work with researchers to identify which of this information is useful, and allow these portions to be enabled on their own. This will lessen the time spent by researchers sorting through data that is unimportant to them.

Reservoir Labs (Richard Lethin) – Proprietary Information

Introduction

This research memo describes the contributions of the Reservoir Labs X-Stack team during the period of performance December 15th 2012 - March 15 th 2013. The summary of contributions during the aforementioned period of performance is as follows:

- an open-source, non-proprietary port of the Runnemede LLVM codebase that was based on LLVM-2.9 to the new Traleika LLVM codebase based on LLVM-3.2,
- a research memo detailing contributions to LLVM and future research ideas. This memo and the associated research ideas are open-source and non-proprietary,
- a research memo detailing contributions to compilation for deep hierarchy and future research ideas. These contributions and forward looking ideas consist of extensions to the R-Stream compiler which will be delivered under SBIR Data Rights and proprietary,
- a research memo detailing contributions to compilation for optimization of stencil applications and intrusive uncertainty quantification as well as future research ideas. These contributions and forward looking ideas consist of extensions to the R-Stream compiler which will be delivered under SBIR Data Rights and proprietary. proprietary

A summary of each contribution is provided thereafter. Notably, we have developed results with the R-Stream compiler which provide a 40% speedup in execution for a 3-D heat equation kernel. These results will provide immediate benefits for block structured PDE solvers running on Petascale supercomputers consisting of Intel processors.

Contribution Period Dec. 2012 - Mar. 2013

This section describes at a high level our contributions during this reporting period.

Updating the LLVM Codebase to LLVM-3.2

During this reporting period we updated Intel's older Runnemede LLVM-2.9 compiler to the most recent release of LLVM, version 3.2, to support the Traleika Glacier architecture. This updated LLVM-3.2 source code was integrated into the main Traleika Glacier GIT repository.

The first step in this porting effort involved a straightforward merge of the older compiler source code files with the LLVM-3.2 source code files. Since the LLVM distribution contains over 35,000 source files, we used the Mercurial source code revision management system's merge functionality to automatically

compare and merge the source code files. This approach automated the updates needed to the majority of the files in the LLVM distribution, leaving a small number LLVM source files and Runnemede specific extensions that required manual inspection and modification.

After addressing build issues and creating a baseline LLVM-3.2 compiler, we began functionality testing. Although the LLVM community distribution contains a number of tests, they are not oriented toward exercising the code generator, nor are they intended for a simulation environment. Accordingly, we have focused our testing effort on building components of the X-Stack distribution (e.g. newlib). Our testing efforts are ongoing and the next step will be to run benchmarks using the FSim simulation environment.

LLVM Research Memo

The Reservoir research memo on LLVM support for the Traleika Glacier architecture [BV13] discusses the LLVM compiler Application Binary Interface (ABI) for Traleika Glacier, ideas and support for address space extensions, support for the Open Community Runtime (OCR) Event Driven Task (EDT) based runtime, ideas for future Intel ISA extensions and support for MemISA.

At the level of the ABI, we describe the current data model, alignment, partitions, calling convention (intra and inter-partition), assisted calling convention and function pointers. We discuss address-space ideas borrowed from OpenCL and our implementation for language support of Traleika Glacier address space extensions. For runtime support we discuss implications of EDT execution models on support for stack frames, local variables, register spills and outgoing arguments that do not fit in registers.

At the level of the ISA, we discuss ideas based on sliding register windows. In addition, we discuss the particular case of stencil applications and implications of register level tiling. We advocate for the support of addressable registers to greatly reduce the number of load-store instructions and enable further benefits from sliding register windows.

This research memo is still largely work in progress and will be updated next quarter when we receive updates to the ISA and MemISA specifications.

Deep Hierarchy Research Memo

The Reservoir research memo on automated mapping techniques for deep hierarchies [VM13] discusses the automation of support for deep hierarchies using the R-Stream compiler. We discusses stencil applications at Exascale levels of parallelism. In particular, we focus on the automatic generation of a parametrized mix of oblivious and conscious tasks suitable for execution by a runtime for Event Driven Tasks (EDTs). We examine and discuss current limitations and extension of polyhedral-based intermediate representations (IRs) based on the Chernikova decomposition. We provide ideas and simple experiments to motivate new approaches and extensions to such IRs to support the scalable generation of optimized EDTs for arbitrarily many levels of a hardware and runtime hierarchy. The research direction we propose is based on parametric EDTs. Such parametric EDTs will expose knobs to the LLVM compiler and runtime interfaces to allow auto-tuning and dynamic adaptation to peculiarities of the execution at any point

in time (e.g. voltage throttling, communication and computation imbalance, resilience, task relocation, etc ...).

We also propose a solution based on C++ expression templates to generate dependences between EDTs at runtime. This solution starts from a compact representation based on loop types which is a natural fit for the R-Stream compiler. This solution will be extended to support both oblivious and conscious parametric EDTs. We present experimental evidence of this technique using CnC and state of the art automatic techniques for stencil computations. We demonstrate over 40% performance improvement over the state of the art automatically generated solution for stencils [TCK+11, BPB12, WS13]. The preliminary results we present are on a 2-socket, 6-core per socket Intel Xeon E5-2620 @2.00 GHz.

We conclude with a discussion on data transfer management in the context of parametric and oblivious EDT generation which will guide our future research and experiments.

Stencils and Uncertainty Quantification Memo

The Reservoir research memo [VL13] on stencils and intrusive uncertainty quantification provides a detailed introduction to numerical solutions for solving linear and nonlinear partial differential equations.

We have identified different opportunities to apply the R-Stream compiler technology in the context of stencils extracted from LBL's BoxLib and in the context of aggressive fusion of stencil kernels with intrusive uncertainty quantification methods. Multiple opportunities arise to tradeoff locality, parallelism and memory consumption. Automated transformations that reduce storage requirements by a significant fraction (resp. an order of magnitude) are possible for BoxLib (resp. intrusive uncertainty quantification). Transformations that will enable incremental checkpointing are also within reasonable expectation. Future work will focus on studying opportunities to apply the R-Stream compiler technology to polynomial chaos expansion formulations for which we do not yet have a representative kernel.

These research concepts will converge with the ideas for deep hierarchy to enable the R-Stream compiler to transform sequential input programs into EDT-ready tasks.

Conclusion

During this reporting period, we furthered our project research objectives and delivered an updated LLVM codebase. The updated LLVM-3.2 codebase delivered supports Intel's current ISA for Traleika Glacier, and is integrated into the main Traleika Glacier GIT repository. Additionally, our LLVM work to date, and a discussion on future research directions is documented in a research memo. In support of our project research objectives, we completed research memos on utilization of the R-Stream compiler in research on both automatic mapping of adaptable code to deep hierarchies and on compilation for optimization of stencil applications and intrusive uncertainty quantification.

References

[BPB12] Vinayak Bandishti, Irshad Pananilath, and Uday Bondhugula. Tiling stencil computations to maximize parallelism. In *ACM/IEEE Su- percomputing (SC '12)*, Salt lake city, Utah, USA, November 2012. ACM.

[BV13] Preston Briggs and Nicolas Vasilache. Research memo: LLVM-3.2 support for the traleika glacier architecture. Technical report, 2013.

[TCK+11] Yuan Tang, Rezaul Alam Chowdhury, Bradley C. Kuszmaul, Chi- Keung Luk, and Charles E. Leiserson. The pochoir stencil compiler. In *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*, SPAA '11, pages 117–128, 2011.

[VL13] Nicolas Vasilache and Harper Langston. Research memo: Intrusive uncertainty quantification and fusion with stencil computations using the R-Stream compiler. Technical report, 2013.

[VM13] Nicolas Vasilache and Benoit Meister. Research memo: Support for deep hierarchy with the R-Stream compiler. Technical report, 2013.

[WS13] Dave G. Wonnacott and Michelle Mills Strout. On the scalability of loop tiling techniques. In *Proceedings of the 3rd International Workshop on Polyhedral Compilation Techniques (IMPACT)*, January 2013.

Rice University (Vivek Sarkar)

Accomplishments:

- A paper on distributed Habanero-C was accepted for IPDPS 2013. The ideas in that paper will contribute to distributed implementations of OCR
- Fixed bugs and improved documentation for OCR release. Special thanks to Benoit Meister and Nicholas Vasilache for pointing out some issues. Smith-Waterman, Cholesky and Parallel Merge Sort all work on OCR.

Issues:

- David Padua and Benoit Meister pointed out the difficulties of implementing libraries in OCR without some sort of finish{} support as in Habanero-C and without rewriting the API's in legacy C code that calls those libraries. We already know how to do this in Habanero-C using our implementations on current hardware, but we want to make sure that OCR does not make restrictive assumptions about future hardware. We are studying the design and implementation of startFinish() and endFinish() API's that would address those issues for a "flat finish" model using EDT local storage.

- As mentioned in the Y1Q1 report, a key open issue for the Rice team right now is guidance on which proxy applications should be used for research demonstrations in this project, especially in Year 1.

Status:

- Porting another application to OCR (which one?). Adding flat-finish support to OCR. Adding EDT Local Storage API's to OCR. Implementing dbCopy() in OCR.
- Starting work on CnC implementation on OCR. Discussions have started with the OpenCnC community on what would be the best approach on implementing CnC on OCR.

Plans for next two quarters:

- Y1Q3: Implement CnC on OCR Release 1 for SMPs w/ example programs and documentation
- Y1Q4: Prototype 1 of CnC tuning annotations using CnC-HC

UCSD (Laura Carrington)

Proposed milestone: Characterize proxy and develop characterization methodology

- Locality measure
- Flops/mem
- Hit rates on range of caches
- Instruction mix (+vectorization)

Progress:

We instrumented Nekbone, the proxy app for the Nek5000 application (CESAR), to capture cache hit rates, memory operations to flops ratio, and instruction mix. In addition, we implemented an extension to our MPI tracer tool to monitor and represent communication by data moved and distance.

To capture hit rates, data movement (to/from memory) and characterize the instruction mix we use binary instrumentation. The binary instrumentation counts basic blocks execution and simulates cache structures on the fly. As an example, for Nekbone we estimated 99.20% hit rate in L1, 3.92% hit rate in L2, and 10.25% hit rate in L3 on a Sandy Bridge-like architecture, and 7:1 memory ops to flops ratio. Our instrumentation tools can simulate alternative cache configurations, including fully associative caches as a surrogate of a scratchpad for comparison.

The communication profile helps quantifying data movement between MPI ranks, and also the distance traveled. This measure of locality will be helpful in understanding the impact on and the energy requirement of the interconnect. As an example, the communication matrix of Nekbone is presented in Figure 1, showing data moved between ranks. Figure 2 shows the distribution of data movement over

distance defined as the difference between sending and receiving rank.

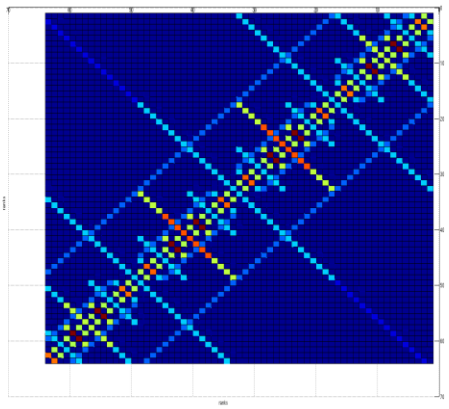


Figure 1 - Data communication matrix

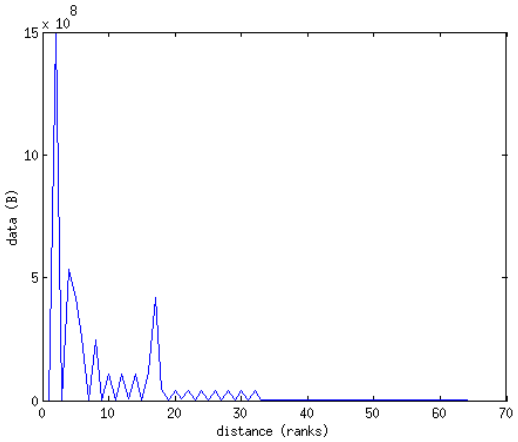
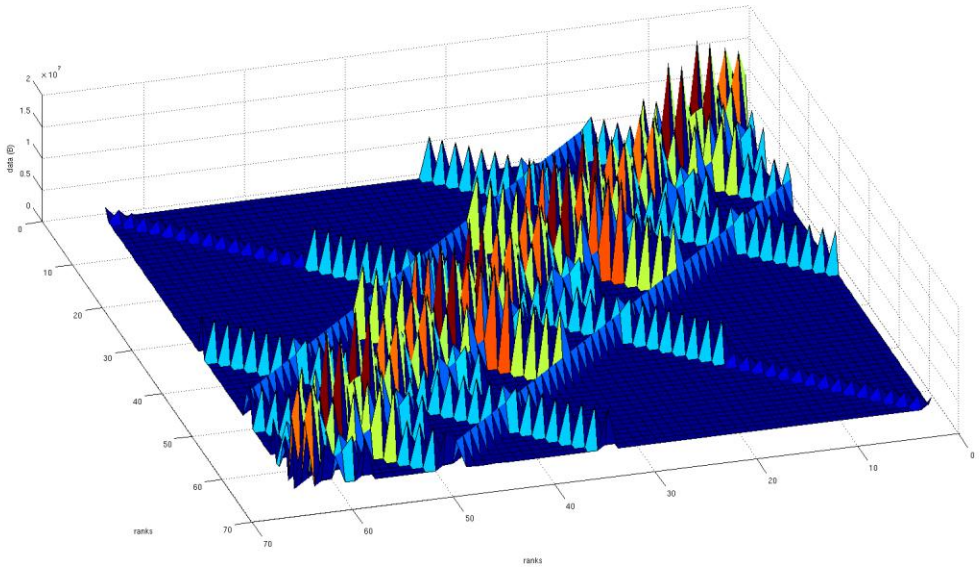


Figure 2 - Distance distribution



Finally, we initiated a discussion to organize our collaborative efforts and interaction with co-design centers, and to establish an organized code base for the team. We started listing all the applications, their proxies (and kernels etc...), their availability and location, programming language and model, lines of code, and a one-sentence description. This list was presented and used to lead the discussion on the needs of proxy apps and kernels to test xstack tool chain.

University of Delaware (Guang Gao)

Synergies with the larger project

UD has started studying OCR both from the "user" point of view (i.e. how to write applications for OCR) and from an internal point of view (to understand the architecture of OCR). Further studying of OCR implies deeper interactions with Rice and Intel.

Stéphane Zuckerman, Aaron Landwehr, and Kelly Livingston have presented their thoughts and findings about resource management, self-aware and energy-aware execution during one of the technical talks organized weekly by the Traleika Glacier team.

Accomplishments

UD's milestone was to perform the identification of the necessary mechanisms for the program execution model to achieve X-Stack's goals – in particular self-awareness in the codelet model. To this end, various tasks were performed. First, a study of current OS and runtime systems was made to evaluate current trends in system software resource management. This research also leveraged knowledge we gathered while attending the DOE-funded Exa-OS/R workshop organized on October 4-5, 2012. Another task was to study existing self-aware/self-adaptive systems. We performed an in-depth study of SEEC, a framework developed at MIT by Hank Hoffman. We have maintained contact with him to interact and deepen our understanding of his system.

The last task to perform which directly addresses the current milestone is to provide a high-level view of the mechanisms required to implement self-aware behavior on an exascale system. We have produced a first draft to define the basis of our reasoning with respect to hardware and software mechanisms required to implement adaptive and self-aware behaviors. In particular, there are two aspects to self-awareness which need to be studied: (1) at the lowest level (e.g. at the island level in FSim) there must be a way to hold very precise information about the "health" (Note: the term health means a combination of a number of things: the level of energy consumption vs. the goals and constraints, how many computing units are currently faulty, etc.) and of the subsystem, and (2) at the higher levels some aggregated data about global health must be propagated. It is important to note that the information available at the higher levels will necessarily be a digest of the precise information available at the lower levels in order to conserve energy and minimize the performance impact of communication. Obtaining this information is important for high level decisions because while a "local optimum" may have been found with respect to performance, power, precision, and failure rate; it is highly possible that at higher levels of the system's hierarchy the global health may be endangered (e.g. because too much energy is required to sustain the computation as it is). This is why we have a written draft of the various mechanisms the system software would require to implement self-awareness. This includes hardware requirements and preferences as well as software-only mechanisms.

Status

As our current document about self-aware exascale system is still a very high-level picture, it needs to be refined, and a design must be derived and implemented in OCR as a proof-of-concept. The design itself is the object of the next two milestones. As explained before, this implies studying the internals of

OCR in order to know what kind of meta-data, data structures, APIs, etc., would need to be added or modified to integrate self-awareness in the runtime.

Issues

As we explained in the first milestone report, the research on self-aware systems in the context of high-performance computing is almost inexistent. However, there are a few attempts: LibraOS, and the whole "OS as library" research effort which goes with it – embodied by the MultiLibOS and EbbRT projects; and SEEC, as we described earlier. However, the former is still very much in its infancy (isolated prototypes were produced, but the framework itself still remains to be built), and the latter seems to be usable only in a very "centralized" context. The platform we are targeting through the straw-man architecture simulated in FSim is intrinsically hierarchical and distributed within the chip, which makes the use of SEEC as a standalone self-aware component impractical. However, it is perfectly possible that SEEC could bring value as a building block for a group of cores (either at the island or cluster level).

Another issue is the connection of self-awareness requirement and the Traleika Glacier's architecture, specifically the memory model and organization. One issue raised by potential users of our technology is specifically the memory hierarchy, scratchpad memory and caches on our T.G. Chip. Having fast scratchpad memory on chip is critical to processor architecture evolution, and a self-aware runtime should explore such architecture features to meet the user goals.

Plans for next milestone

Now that a basis has been defined in our document, it needs to be fleshed out a bit more to be integrated in an event-driven environment. The design of a self-awareness API is the natural next step we have to accomplish. This in turn will require us to go back to our current draft to complete missing details and revise incorrect assumptions. The document itself is still very much in flux, and we hope to be able to provide a stable version of it by the end of next milestone. We do not plan to have preliminary results at that point, but a first draft of a self-aware API should be defined. To address the challenge of this API we plan to leverage the memory model work of our past and on-going work such as causal (acyclic) consistency model based on the foundation of location consistency.

To be able to test our design, we will need to rely on the applications already written for OCR and/or FSim. This implies a dependency on PNNL, and other co-design centers benchmarks.

University of Illinois Urbana Campus

David Padua

Accomplishments

Our work to develop a high-end interface advanced as planned during the last quarter. We now have a first draft of the document describing the Hierarchically Tiled Array (HTA) API, and a first implementation of this API on top of PIL, our intermediate language. Currently, we have implementations of PIL built on top of OpenMP and an earlier runtime system, IRR. We have not yet an implementation of PIL based on the Open Community Runtime (OCR). This is why we have had

discussions with Rice University and Intel members of the team implementing OCR in order to resolve the current obstacles to the implementation of PIL.

Issues

We are working on determining if the current version of OCR enables the implementation of APIs which, like our HTA, are in the form of a library. If it does, we should be able to port PIL within a few weeks to run on top of OCR. If OCR cannot be used in its current form, it will be necessary to wait for a solution or we will need to implement a compiler for the HTA API.

Plans for next milestone

We will continue improving the performance of the HTA implementation, addressing memory management features, and expanding the API to include other data structures. Our next goal is to develop a version capable of handling sparse arrays. Also, we have initiated conversations with Reservoir labs personnel to enable the use of automatic transformations by the R-Stream compiler to improve the performance of HTA operations. We will work during the next quarter to enable the integration between our implementation of HTA and the target of R-Stream.

Josep Torrellas

In this quarter, we have developed an API for software managed (or incoherent) caches. It is composed of cache invalidation and write back instructions, and more advanced constructs.

The programmer or compiler is expected to use this API to manage a set of incoherent caches.

We are working with the compiler folks in the team to see what the compiler can actually generate. Currently, the compiler infrastructure can analyze numeric codes with loops with affine functions in loop indices and generate calls to our API. Currently, we are able to transform a few kernels.

Our next step will be to try the API, both manually and with the compiler, for 1) many kernels and 2) some of the co-design center codes. An expected challenge will be whether the compiler can transform codes that use non-affine loop indices by itself or, instead, needs user help.

Pacific Northwest National Labs (John Feo, Andres Marquez)

Accomplishments

SCF code was made available to the community.

Status

We are building the underlying infrastructure that will guide us in the design of the Power Efficient Data Abstraction Layer (PEDAL) for Rescinded Primitive Data Type Access (RPDTA). This infrastructure is being leveraged in this project as well as the Brandywine project. The infrastructure consists mainly of three parts in different stages of development:

1) In order to observe and control the association between data structures and sets of threads, we are developing a fundamental theoretical framework called “Group Locality” (GL). The

framework will help us optimize data movement and data lifetime, taking into consideration the interaction of multiple threads. These two criteria are necessary (but not sufficient) to determine suitability of data manipulation (e.g., compression), as would be required for PEDAL.

2) Parallel to 1), we are performing a survey of state-of-the-art compression algorithms amenable to scientific computing. We will perform a selection of suitable initial compression candidates following metrics such a compression ratio, overhead and ease of implementation (either HW or SW). We expect the report to be out in a month.

3) As of today, we are adapting two measurement frameworks to probe data life time and data content on a node. Both parameters are required to assess data compression suitability. The first framework is an adaptation of Intel's Pin dynamic binary instrumentation tool set. The second framework is a modified Valgrind version: Both are designed to capture relevant data information associated with the EXM threads and the HW threads. Valgrind's status is currently more advanced and is capable of measuring lifetime in instruction counts, data address location and access pattern.

A new PhD intern will join us Mid-March to accelerate the development for Traleika.

Issues

None, but we expect issues to arise once we bolt PEDAL on top of OCR.

Plans for next milestone

The next kernel under consideration is a Coupled Cluster quantum chemistry code.

The Rice team is currently implementing a Cholesky version. We will use their Cholesky and instrument it similarly to the instrumentation undergoing in SWARM. We expect to analyze lifetimes, data content and access pattern.

We will explore positive and negative PEDAL interactions with the self-awareness API being developed by UDEL.

The compression survey will be available as a report.

We will start playing with Intel's FSIM (maintained by ETI) to explore what-if-scenarios of lightweight HW compression engines.