Final Project Report

DynAX

Innovations in Programming Models, Compilers and Runtime Systems for Dynamic Adaptive Event Driven Execution Models

Award Number: DE-SC0008716
Dates of Performance: 9/1/2012 to 8/31/2015
NCE Dates of Performance: 9/1/2015 to 12/31/2015
Final Report Date: 12/31/2015

Principal Investigator

Guang Gao, ET International, Inc.

Co-PIs

Benoit Meister, Reservoir Labs, Inc.
David Padua, University of Illinois Urbana Champaign
Andres Marquez, Pacific Northwest National Laboratories

Table of Contents

Executive Summary

Under this DOE award (award number DE-SC0008716; dates of performance 9/1/2012 - 8/31/2015), the Dynax Project conducted collaborative and multidisciplinary research on Innovations in Programming Models, Compilers and, Runtime Systems for Dynamic Adaptive Event Driven Execution Models. Much of the material presented here has already been submitted separately to the DOE in three annual reports and twelve quarterly reports. Therefore, this report is a summary of R&D accomplishments based on these reports. The reports of the NCE period of each participating Institutions are provided as separate reports, and this report includes links to these reports accordingly.

As we stated in our original Dynax proposal 3+ years ago: although present-day software can largely rely on minimally invasive operating systems and system libraries to provide its execution environment, these underlying layers of software – as well recognized by now – are inefficient and insufficient for exascale computing. Instead, the Dynax proposal identifies and addresses a critical technology gap in the system software stack of current high-end computing platforms that must be filled to meet the DOE's exascale system challenges. That critical technology is a dynamic adaptive runtime coupled with underlying parallel architecture features, which is vital for achieving the cost-effectiveness, power efficiency, desirable level of performance and scalability, and resiliency needed to fully exploit the heterogeneous exascale systems of the next decade.

To this end, the Dynax proposal is based on a novel codelet execution model, first described by the PI and his associates in a CAPSL Technical Memo dated April, 2011[1]. It also leverages a commercial runtime, called SWift Adaptive Runtime Machine (SWARM), and associated programming technology enhancement as well as compiler tools[2]. SWARM allows the programmer to express concurrency, synchronization, data locality, and resource needs; and at runtime uses this information to dynamically map and execute computational tasks to available resources. Activities under the DynAX's proposal are focused on large scale parallelism and addresses the issues of high-level programming models, compilers, and runtime systems in the

[1] Toward an Execution Model for Extreme-Scale Systems-Runnemede and Beyond , Guang R. Gao, Joshua Suetterlein and Stephane Zuckerman, April, 2011.

[2] http://www.etinternational.com/downloads/swarm_docs/swarm/

context of the codelet based execution model. The key features of these components include scalability, programmability, portability, resilience, and energy efficiency. The research in this proposal is to be verified and tested on benchmark programs leveraging DOE PNNL work on NWChem application and programming environment as they become available.

The Brandywine team contains a number of institutions that have decades of research and development experience in HPC. The industry partners (ETI, Reservoir) utilize product knowledge to extend their base products with research and development in revolutionary execution models as well as the software stack to support them. The research partners (UIUC and PNNL) bring experience in scientific applications and domain specific languages to bear on the research. The result of this work has brought significant technology advancements that will move research and development towards feasible solutions for DOE at exascale. The industry partners have used the results of this work to build and extend commercial products for DOE and the industry.

Exascale Challenges and the Dynax Project

Future generation HPC systems comprising many-core sockets and GPU accelerators will impose increasingly difficult challenges in programming, efficiency, heterogeneity, and scalability for exascale computing. Emerging execution models using event-driven task-based parallelism, dynamic dependency and constraint analysis, locality-aware computation, and resource-aware scheduling show promise in addressing these challenges and meeting the needs of the future of supercomputing. Applications employing these innovative runtime systems have shown significant gains in performance and utilization of computational resources in comparison with conventional methodologies[3].

The Dynax project has described and implemented solutions to the challenges exascale systems pose, including new programming models that allow developers to express concurrency, locality, and synchronization in a concise manner. There are fundamental challenges in developing applications that successfully utilize modern petascale systems, and these difficulties will only be exacerbated on future exascale systems. First, current MPI programming models are based on the communicating sequential processes (CSP) model

[3] Swarm Graph500. http://www.graph500.org/nov2011.html

which, although have been a dominant parallel programming model for parallel supercomputing applications, the research community has increasingly recognized the difficulty in expressing dynamic parallelism under MPI as it does not take advantage of the dynamic availability of resources. This in turn makes it extremely difficult for programs with irregular or global data accesses, or with non-uniform computational needs, to fully exploit available hardware resources with high and smooth scalability. Second, use of heterogeneous systems requires multiple languages and programming models. For example, MPI is commonly used to distribute work across a computing cluster, while OpenMP distributes work across threads within a node, and OpenCL distributes work to GPUs within a node. This incongruous mixture impedes development of new applications and adaptation of existing applications to new platforms or hardware configurations. Synchronization provides other challenges for application developers. CSP programs typically resort to frequent use of barrier and collective constructs, which require centralization of communications and control, thereby increasing contention and delaying program advancement. These constructs may not always be necessary, however, and are often used to broadly oversimplify what could otherwise be much more fine-grained parallelism. This constrains flexibility of resource management by forcing long idle periods during synchronization when portions of the program could otherwise be completing useful work. Finally, the debugging and performance tuning of parallel programs written in traditional parallel programming models (such as the CSP based model or beyond) can be exceedingly difficult because synchronization and data locality are implicit in the message-passing semantics and not always intentional.

Accomplishment

Under the DOE XStack award for the DynAX project, we have accomplished the objectives presented in our proposal that have addressed the above challenges. The accomplishment of this project are summarized in the rest of this report.

Accomplishments

Overview

Overall, we have reached our objectives as stated in our Dynax proposal.

The accomplishments are summarized below for each of the three years (Y1, Y2, and Y3). Further details are summarized in the 12 quarterly reports Q1-Q12.

As mentioned during the negotiations with the program manager, each institution provided a separate report for the no-cost extension period. These are available as hyperlinks on the modelado wiki under the DynAX project (see the NCE Reports section of this document). Additionally, ETI's report is attached to this document in Appendix C. Reservoir provided their final report directly to the program manager on October 31st for the end date approved for their NCE. PNNL provided their report directly to the program manager on December 23rd for the end date approved for their NCE. UIUC provided their report on August 31st. UIUC did not additional work after this date and will not be using the remaining funds.

Appendix A includes the list of the hyperlinks to the full year reports of all three years, as well as links to the 12 Quarterly Reports.

Appendix B includes tasks that were described in the proposal. Note that due to the budget constrains, the original tasks have been adjusted and there has been an agreement reached for the final statement of work (SOW). These adjustments are highlighted in the appendix.

Appendix C contains a verbatim copy of the NCE report submitted by ETI to the DOE program manager.

Year 1

ETI Accomplishments

In year 1, ETI focused on studying Cholesky Decomposition and Self-Consistent Field) codes. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y1 report as well as the Q1, Q2, Q3, and Q4 reports.

We used Cholesky to study issues of tiling, scheduling, and memory management at scale. We started with a basic implementation within SWARM and made significant scalability improvements. We note that simple round-robin assignment of work leads to significant workload imbalance and found methods to produce a more even distribution of work. We additionally found that a priority scheme for task prioritization that encourages compute nodes to traverse the matrix in a particular order causes parallelism to be much better exposed. We implemented a form of prefetching to reduce data starvation for nodes needing remote data at scale. The results are discussed in detail in the Q1 report. For SCF, we optimized the code provided by PNNL. Specifically, we improved the twoel kernel of the application by implementing multi-node versions for twoel using MPI and SWARM and providing several serial optimizations to the kernel. The results can be found in the Q2 and Q3 reports.

PNNL Accomplishments

In year 1, PNNL focused on application and performance studies. Specifically, we identified two NWChem modules: Self-Consistent Field (SCF) and Coupled Cluster Method (CCM) as representative benchmarks for Exascale chemistry applications. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y1 report as well as the Q1, Q2, Q3, and Q4 reports.

The SCF method is generally the central and most time consuming computation in ab initio quantum chemistry methods. Of the SCF method, the twoel module is the most computationally intensive piece of code. The CCM module is interesting in that the code is generated automatically from tensor equations supplied by the user. Our Y1 report discusses these two methods in detail.

Reservoir Labs Accomplishments

In year 1, Reservoir Labs looked at some of the more impactful optimizations performed by ETI on the SCF code and defined how to implement these optimizations within a compiler supporting both polyhedral loop optimization and more traditional SSA analysis. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y1 report as well as the Q1, Q2, Q3, and Q4 reports.

To this end, we demonstrated how known symmetries in input data can used to automate optimizations to reduce the number of evaluations in input data by up to 2^n. Additionally, we showed how to automate pre-computation of heavily-reused sub-expressions in loop codes by combining polyhedral scope analysis with traditional subexpression analysis. We defined an algorithm to detect large sub-expressions and operates at the intersection in trading off subexpression size for number of the number of loop indices subexpressions dependent on. Finally, we implemented generic support for parallelization of codelet-based codes into R-Stream. The details of this can be found in the Y1 report.

UIUC Accomplishments

In year 1, UIUC implemented various enhancements to the Parallel Intermediate Language (PIL) from which SWARM code can be generated. Enhancements include tiled array data representation, structured PIL (SPIL) extensions, and SPMD PIL. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y1 report as well as the Q1, Q2, Q3, and Q4 reports.

To evaluate the overhead of PIL programs and our new extensions we began a study of Cholesky decomposition using PIL and compared the execution time to a hand-coded SWARM implementation. PIL codes targeting both SWARM and OpenMP showed scalability. One of the primary usability enhancements to PIL is the ability to write library functions as a collection of PIL nodes promoting reuse of code and allowing kernels to be self-contained within larger PIL programs. We implemented Hierarchically Tiled Arrays (HTAs) using this facility. Support for a data structure for creating and manipulating tiled arrays using parallel operations was incorporated into the design. Additionally, we completed the design of SPIL which provides syntactic sugar for common operations in the PIL programming language. Finally, for distributed systems, a we designed SPMD PIL targeting distributed SCALE. This allows PIL code be implemented in a SPMD fashion (i.e. all processing elements execute the same program concurrently with different data).

Publications

- [SuetterleinEtAl13] J. Suetterlein, S. Zuckerman, G. Gao. **"An Implementation of the Codelet Model,"** In Proceedings of 19th International European Conference on Parallel and Distributed Computing (Euro-Par 2013), Aachen, Germany. August 26th, 2013.

- [LandwehrEtAl13] A. Landwehr, S. Zuckerman, G. Gao. **“Toward a Self-aware System for Exascale Architectures.”** In Proceedings of Euro-Par 2013: Parallel Processing Workshops; the 1st Workshop on Runtime and Operating Systems for the Many-core Era (ROME 2013), Aachen, Germany. August 26th, 2013.
- [ChenEtAl13A] C. Chen, Y. Wu, J. Suetterlein, L. Zheng, Minyi Guo, G. Gao. “**Automatic Locality Exploitation in the Codelet Model.”** In Proceedings of 11th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA-13), Melbourne, Australia, July 2013.
- [ChenEtAl13B] C. Chen, Y. Wu, S. Zuckerman, G. Gao. **“Towards Memory-Load Balanced Fast Fourier Transformations in Fine-grain Execution Models.”** In Proceedings of Workshop on Multithreaded Architectures and Applications (MTAAP 2013), May 24, 2013, Boston, Massachusetts USA.
- [GarciaGao13] E. Garcia, G. Gao. **“Strategies for improving Performance and Energy Efficiency on a Many-core.”** In Proceedings of 2013 ACM International Conference on Computer Frontiers (CF 2013), May 14-16, Ischia, Italy, ACM, 2013.

Additional Activities

Additional accomplishments including delivered technologies and presentations may be found the Other Accomplishments section of this document.

Year 2

ETI Accomplishments

In year 2, ETI studied LULESH and TCE, and extended SCALE. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y2 report as well as the Q5, Q6, Q7, and Q8 reports.

As written, LULESH 2.0 demonstrated workload imbalance in terms of loop counts for certain regions (i.e. regions with high loop factors have more work than other regions). We proposed a few serial performance improvements for the application and during study came to the conclusion that the performance characteristics of the applications would result in similar performance between the MPI+X version and a SWARM adaptation. As such, we suggest rebalancing the static workload instead. Furthermore, we produced a standalone version of the Tensor Contraction Engine (TCE) with reference input and output data. We additionally, incorporated the ability to produce OCR and SWARM code on a function-by-function and block-by-block basis and studied the performance characteristics of the application. We additionally extended the SCALE syntax to support basic object-oriented-programming techniques in order to ease programmer burden with developing with SWARM. This enables programmers to more easily interact with SWARM while avoiding some of complexities of the underlying runtime and language.

PNNL Accomplishments

In year 2, PNNL studied two application codes: the Coupled Cluster Method (CCM) in NWChem, and LULESH. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y2 report as well as the Q5, Q6, Q7, and Q8 reports.

For CCM, we provided sets of tensor equations, a driver program, as well as, an input deck used to to develop and evaluate an automatic code generator. We additionally, developed a CnC control and data flow diagram and ported the code for execution within the simulator. We then refactored the code into a CnC program and evaluated the algorithm within Intel's CnC framework and Rice's CnC-OCR framework. Finally, we created the Architected Composite Data Types (ACDT) conceptual framework to exploit opportunities w.r.t. access patterns, data composition, dynamic range, etc.

Reservoir Labs Accomplishments

In year 2, Reservoir Labs worked on a number of aspects of the DynAX project. Given that the efficiency of exascale systems is strongly influenced by efficient communications we focused on this. We collaborated with PNNL, ETI, and the University of Delaware on adaptive sparse data structure transfer optimization techniques. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y2 report as well as the Q5, Q6, Q7, and Q8 reports.

During this period, we identified scalability issues in the synchronization constructs within SWARM, CnC, and OCR. As a result, we proposed modifications to count dependencies within SWARM which would provide optimal scalability in terms of sequential overhead of task creation, in-flight tasks, and the space used by synchronization objects and their garbage collection. These extensions were implemented in the R-Stream runtime layer for SWARM and are used by the codes generated by R-Stream. Additionally, we developed a communication library for x86 platforms that provides virtual DMA operations by leveraging the x86 vector cache. We then modified R-Stream to produce SWARM code using the virtual DMA library. Finally, in order to devise an approach to support a class of unstructured codes, we looked at existing art. We studied structured adaptive mesh refinement and looked at an approach that uses Legion as a task graph programming API.

UIUC Accomplishments

In the prior year, UIUC made various enhancements to the Parallel Intermediate Language (PIL) in terms of incorporating tiled array data representations, providing Structured PIL (SPIL) extensions, and designing SPMD PIL. In year 2, we implemented and evaluated benchmark programs in the HTA notation, and extended HTA to support irregular tiles as well as distributed computing environments. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y2 report as well as the Q5, Q6, Q7, and Q8 reports.

In Q5, we implemented six NAS Parallel Benchmarks using HTA and the results can be executed using OpenMP and SCALE. A detailed discussion is provided in the Y2 report. In Q6, we designed a strategy for expanding HTA programs to support SPMD execution, thus, allowing an efficient mapping to distributed memory SCALE. In Q7, we extended HTA to support irregular tile partitioning, thus, allowing applications to create and manipulate irregular tiles.

Publications

- [WeiEtAl13] H. Wei, G. Gao, W. Zhang, J. Yu. **“COStream: A Dataflow Programming Language and Compiler for Multi-Core Architecture.”** In

Proceedings of Data-Flow Models (DFM) for extreme scale computing Workshop 2013 in conjunction with Parallel Architectures and Compilation Technologies (PACT 2013), Edinburgh, Scotland, September 8 of 2013.

- [SolinasEtAl13] M. Solinas, R. Badia, F. Bodin, A. Cohen, P. Evripidou, P. Faraboschi, B. Fechner, G. Gao, A. Garbade, S. Girbal, D. Goodman, B. Khan, S. Koliai, F. Li, M. Luján, L. Morin, A. Mendelson, N. Navarro, A. Pop, P. Trancoso, T. Ungerer, M. Valero, S. Weis, I. Watson, S. Zuckermann, R. Giorgi. **"The TERAFLUX Project: Exploiting the DataFlow Paradigm in Next Generation Teradevices."** In Proceedings of the 16th Euromicro Conference on Digital System Design, Santander, Spain, September 4-6, 2013.
- [ZuckermanEtAl14] S. Zuckerman, A. Landwehr, K. Livingston, G. Gao. **"Toward a Self-Aware Codelet Execution Model."** In proceedings of 4th Workshop Data-Flow Execution Models for Extreme Scale Computing (DFM'14), August 24, 2014, Edmonton, Alberta, Canada.
- [DennisGao14] J. Dennis, G. Gao. **"On the Feasibility of a Codelet Based Multi-core Operating System."** In the 4th Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM'14), August 24, 2014, Edmonton, Alberta, Canada.
- [ArteagaEtAl14] J. Arteaga, S. Zuckerman, E. Garcia, G. Gao. **"Position Paper: Locality-Driven Scheduling of Tasks for Data-Dependent Multithreading."** In Proceedings of Workshop on Multi-Threaded Architectures and Applications (MTAAP 2014), May 2014.
- [MeisterEtAl14] B. Meister, N. Vasilache, M. Baskaran, T. Henretty, R. Lethin. **"RStream experiments with hierarchical iteration tiling."** 16th SIAM Conference on Parallel Processing for Scientific Computing, February 2014, Portland, Oregon, USA.
- [VasilacheEtAl14] N. Vasilache, M. Baskaran, T. Henretty, B. Meister, R. Lethin, **"A tale of three runtimes."** Arxiv preprint # 1409.1914. http://arxiv.org/abs/1409.1914
- [StJohnEtAl14] T. St John, B. Meister, A. Marquez, J. Manzano, G. Gao, X. Li. **"ASAFESSS: A Scheduler driven Adaptive Framework for Extreme**

Scale Software Stacks." In proceedings of The 4th International Workshop on Adaptive Self-tuning Computing Systems, ADAPT '14, Vienna, Austria, January 2014. Recipient of the best paper award.

Additional Activities

Additional accomplishments including delivered technologies and presentations may be found the Other Accomplishments section of this document.

Year 3

ETI Accomplishments

In year 3, ETI finished study of the TCE application provided by PNNL. Additionally, throughout the year ETI incorporated and studied resilience techniques within SWARM and how to achieve MPI interoperability. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y3 report as well as the Q9, Q10, Q11, and Q12 reports.

In Q9 and Q10, we completed a block-parallel implementation of TCE within OCR and a task-parallel version within SWARM. Additionally, we began research into resilience and fault tolerance using containment domains. We implemented a prototype implementation within SWARM and began implementing Cholesky using it. We also began investigating MPI interoperability within SWARM and reached a number of conclusions highlighted in our Q9 and Q10 reports. In Q11, we produced a report summarizing our studies of MPI interoperability as well as another report detailing the feasibility of a containment domain based Cholesky within SWARM. In Q12, we continued design studies of the Cholesky and published a full paper to the Mini-Symposium on Energy and Resilience in Parallel Programming (ERPP2015). Additionally, we created a practical matrix multiplication example showcasing MPI interoperability with SWARM and conducted a comparative study of MPI+X within the field.

PNNL Accomplishments

In year 3, PNNL focused on improving the management of data movement and resource underutilization within our Group Locality framework. We also introduced a data restructuring framework within the Group Locality framework. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y3 report as well as the Q9, Q10, Q11, and Q12 reports.

Our group locality framework creates scattering functions to increase resource utilization and data locality at compile time. Additionally, we introduced two novel tiling techniques, named jagged and diamond jagged tiling that introduces additional tile shapes to expose inner parallelism at the lower levels of tiling hierarchies. Moreover, we developed a data restructuring framework that uses a Polyhedral formulation to restructure and move data while considering thread access patterns. The contribution of our methodology is a low overhead transformation technique that exploits locality and additionally incorporates a collaborative restructuring of data for group reuse.

Reservoir Labs Accomplishments

In year 3, Reservoir Labs identified block-sparse computations as a significant subset of unstructured codes. We incorporated support for block-sparse computations for distributing computing into R-Stream. We also worked with UIUC to integrate PIL generated code and R-Stream optimized code. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y3 report as well as the Q9, Q10, Q11, and Q12 reports.

We developed support for distributed computing and block-sparse computation in R-Stream as well as UIUC's HTA/PIL. We first began developing a backend runtime to use R-Stream for the parallelization of dense array computations on clusters based on a PGAS abstraction. We then moved to implementing an R-Stream mapping path (backend and runtime) to produce block-sparse computations on clusters. Additionally, we completed support for HTA and PIL as a front-end to R-stream.

UIUC Accomplishments

In year 3, UIUC worked on implementing SPMD execution of PIL and HTAs within the the SWARM runtime and implemented in the HTA notation a dense Cholesky factorization to analyze the execution behavior (in year 1 a version of cholesky was implemented in PIL. The HTA version accomplishes the same goal in an array notation that enables task parallelism). Additionally, as discussed above, we worked with Reservoir labs to integrate PIL generated code and R-Stream optimized code. The subsequent discussion provides a brief summary of our activities and more detailed discussion can be found in the Y3 report as well as the Q9, Q10, Q11, and Q12 reports.

In Y1 and Y2, We developed a strategy to map HTA program executions into SWARM in a fork-join fashion; however, this restricts parallelization by forcing global synchronization via barriers. To this end, we began experimenting with an SPMD mode implementation that supports point-to-point synchronization to reduce overheads and promote load-balancing. We chose a dense Cholesky factorization because the computation is an irregular task graph and allows simple modification of input parameters. The results of our study are discussed in the quarterly reports. Finally, we integrated PIL generated code and R-Stream optimized code allowing the user to write more intuitive parallel program code in PIL and use the R-Stream compiler to optimize the computation kernels.

Publications

- [LiEtAl15] X. Li, J. Dennis, G. Gao, W. Lim, H. Wei, C. Yang, R. Pavel. **"FreshBreeze: A Data Flow Approach for Meeting DDDAS Challenges."** In proceedings of the International Conference On Computational Science (ICCS2015). Reykjavík, Iceland, June 2, 2015.
- [KaplanEtAl15] S. Kaplan, S. Pino, A. Landwehr, G. Gao. **"Landing Containment Domains on SWARM: Toward a Robust Resiliency Solution on a Dynamic Adaptive Runtime Machine."** To Appear in the proceedings of the 2015 international Parallel Computing conference (ParCo'15). Edinburgh, Scotland, UK, September 1 – 4, 2015.

- [ShresthaEtAl14A] S. Shrestha, J. Manzano, A. Marquez, and G. R. Gao. **"A Framework for Resource Aware Multithreading."** Poster presented at the International Conference for High Performance Computing, Network, Storage and Analysis (SC 14). New Orleans, LA, USA, November 16 – 21, 2014. Best poster nominee.
- [ShresthaEtAl4B] S. Shrestha, J. Manzano, A. Marquez, J. Feo and G. R. Gao. **"Jagged Tiling for Intra-tile Parallelism and Fine-Grain Multithreading."** In the 27th International Workshop on Languages and Compilers for Parallel Computing, Hillsboro, OR, USA, September 15 – 17, 2014.
- [ShresthaEtAl15A] S. Shrestha, J. Manzano, A. Marquez, S. Zuckerman, S. L. Song and G. Gao. **"Gregarious Data Re-structuring in a Many Core Architecture."** Invited paper to the 17th international conference on High Performance Computing and Communication (HPCC 2015). New York, USA, August 24 – 26, 2015.
- [ShresthaEtAl15B] S. Shrestha, J. Manzano, A. Marquez, J. Feo and G. R. Gao. **"Locality Aware Concurrent Start for Stencil Applications."** In the 2015 International Symposium on Code Generation and Optimization, San Francisco, CA, USA, February 7-11, 2015.
- [Shrestha15] PhD Thesis Sunil Shrestha **"A framework for Group Locality Aware Multithreading."** Fall 2015.
- [TavarageriEtAl15] S. Tavarageri, B. Meister, M. Baskaran, B. Pradelle, T. Henretty, A. Konstantinidis, A. Johnson, R. Lethin. **"Automatic Cluster Parallelization and Minimizing Communication via Selective Data Replication."** In proceedings of the 2015 IEEE High Performance Extreme Computing conference (HPEC'15), 15-17 September 2015, Waltham, MA.
- [MarquezEtAl14] A. Marquez, J. Manzano, S. Song, B. Meister, S. Shrestha, T. St. John and G. R. Gao. **"ACDT: Architected Composite Data Types Trading-in Unfettered Data Access for Improved Execution."** In the 20th IEEE International Conference on Parallel and Distributed Systems, Hsinchu, Taiwan, December 16 – 19, 2014.

- [YangEtAl14] C. Yang, J. Pichel, A. Smith, D. Padua. **“Hierarchically Tiled Array as a High-Level Abstraction for Codelets.”** In the Fourth Workshop on Data-Flow Execution Models for Extreme Scale Computing, 2014.
- [Smith15] A. Smith. **“The Parallel Intermediate Language.”** Ph.D. dissertation, Computer Science Dept., University of Illinois at Urbana-Champaign, September 2015.

Additional Activities

Additional accomplishments including delivered technologies and presentations may be found the Other Accomplishments section of this document.

Other Accomplishments

Technologies Delivered

Additional information on deliverables can be found on the DynAX page of the XStack wiki:
https://xstackwiki.modelado.org/DynAX#Deliverables

Q1-Q3

- NWChem SCF module:
  - Serial C version **(PNNL Q2)**
  - Optimized C version **(ETI Q2)**
  - OpenMP Version **(ETI Q2)**
  - MPI Version **(ETI Q2)**
  - SWARM single and multi-node Version **(ETI Q2-Q3)**
- NWChem TCE module:
  - Serial C Version **(PNNL Q3)**
  - OpenMP Version **(PNNL Q3)**
  - CUDA Version **(PNNL Q3)**
  - Fortran99 Version **(PNNL Q3)**
- Cholesky Decomposition optimization for scheduling, memory management and self-awareness **(ETI Q1)**
- Stencil framework for CnC and SWARM **(Reservoir Q3)**
- Single-node untuned automatic mappable C → SWARM parallelization with R- Stream **(Reservoir Q3)**

- Parallel Intermediate Language (PIL) → SCALE translator
    - Single Node **(UIUC Q1-Q2)**
    - Multinode **(UIUC Q3)**
- PIL API Document **(UIUC Q2-Q3)**

Q4

- NWChem TCE Module:
    - SWARM single node version **(ETI)**
    - Basic single-node parallelization to SWARM using R-Stream
- R-Stream **(Reservoir)**:
    - Basic R-Stream tuning for single-node mappable C → SWARM parallelization
    - Simplified domain computation (Improvement of compiler engine tractability) in R-Stream
- Sparse data representation in PIL **(UIUC)**

Q5-Q6

- NWChem TCE module:
    - Standalone application **(ETI Q5-Q6)**
    - Function-level parallel OCR version **(ETI Q6)**
- Tuned SWARM R-Stream backend - autodecs **(Reservoir Q5-Q6)**
- Tuned SWARM R-Stream backend - virtual DMA, preliminary **(Reservoir Q6)**
- NAS Parallel Benchmark Implementation with PIL HTA (for Shared Memory SCALE) **(UIUC Q5)**
- HTA-to-PIL-to-SCALE interface design changes for supporting SPMD **(UIUC Q6)**
- Coupled Cluster equation, driver program, and input sets **(PNNL Q5)**
- C++ version of CnC based LULESH application code **(PNNL Q6)**

Q7-Q8

- NWChem TCE module:
    - Function-level parallel SWARM version **(ETI Q7)**
    - Block-level parallel OCR and SWARM versions **(ETI Q7-Q8)**
- Tuned SWARM backend - virtual DMA, advanced **(Reservoir Q7-Q8)**
- Research on leveraging communication layer and asynchronous task graph execution model to target mesh computations **(Reservoir Q7-Q8)**

- PIL HTA (Shared Memory) Performance Evaluation with NAS Parallel Benchmarks **(UIUC Q7-Q8)**
- Support for irregular tiles **(UIUC Q7)**
- NAS Parallel Benchmark support for Distributed Memory SCALE **(UIUC Q8)**

Q9-12

- Prototype of nested containment domains within SWARM **(ETI Q9-Q11)**
- Case study of CD based cholesky decomposition within SWARM **(ETI Q12)**
- Case study of MPI+X **(ETI Q12)**
- NWChem TCE module:
    - Block-level parallel OCR **(ETI Q9-Q10)**
- R-Stream cluster backend based on PNNL's Global Arrays **(Reservoir Q9-Q10)**
- R-Stream block sparse cluster runtime **(Reservoir Q11-Q12)**
- PIL compiler and HTA SPMD execution mode implementation **(UIUC Q9-Q10)**
- HTA SPMD performance evaluation using NAS Parallel Benchmark and task parallel Block Cholesky Factorization **(UIUC Q11-Q12)**
- PIL SCALE backend integration with R-Stream **(UIUC & Reservoir Q12)**
- New loop tiling technique for parallel start applications [CGO'15] **(PNNL Q11)**
- Data restructuring framework for affine applications codes [HPCC'15] **(PNNL Q12)**

Presentations

- Project overview presentation: PI Kickoff meeting in September 2012.
- Project overview presentation: DOE ASCR meeting in October 2012.
- Presentation on Cholesky Decomposition enhancements in December 2012.
- Presentation to TACC on SCF optimizations in February 2013.
- Progress-to-date presentation: 6-month PI meeting in March 2013.
- X-StackProject overview and results: EXaCT all-hands meeting in May 2013.
- Extreme scale technical review on TCE as a proxy app in December 2013 (audience includes Traleika Glacier project teams and a broader audience).

- Presentation on an implementation of the codelet model at Europar’13 [SuetterleinEtAl13].
- Presentation discussing a path forward to a self-aware system for exascale architectures at ROME’13 [LandwehrEtAl13].
- Presentation on automatic locality exploitation within the codelet model at ISPA’13 [ChenEtAl13A].
- Presentation on a memory-load balanced FFT for fine-grained execution models at MTAAP’13 [ChenEtAl13B].
- Presentation on strategies to improve performance and energy efficiency in many-core architectures at CF’13 [GarciaGao13].
- Presentation on COStream at DFM’13 [WeiEtAl13].
- Presentation on TERAFLUX at DSD’13 [SolinasEtAl13].
- Extreme scale technical review on the adaptive task-centric runtime framework in February 2014.
- Presentation on distributed scheduling challenges in Cholesky to OCR core team in March 2014.
- Presentation on Virtual DMA optimization to Extreme scale technical review audience in April 2014.
- Presentation on a Self-Aware Codelet Model at DFM’14 [ZuckermanEtAl14].
- Presentation on the feasibility of codelet based operating systems at DFM’14 [DennisGao14].
- Presentation on locality-driven scheduling of tasks for data-dependent multithreading at MTAAP’14 [ArteagaEtAl14].
- Presentation on R-Stream experiments with hierarchical iteration tiling at SIAM’14 [MeisterEtAl14].
- Presentation on a scheduler driven adaptive framework for extreme-scale software stacks at ADAPT’14 [StJohnEtAl14].
- Presentation on a framework for resource aware multithreading at SC’14 [ShresthaEtAl14A].
- Presentation on jagged tiling for intra-tile parallelism and fine-grained multithreading at LCPC’14 [ShresthaEtAl14B].
- Presentation on ACDT: architectured composite data types at ICPADS’14 [MarquezEtAl14].
- Presentation on Resilience and containment domains presentation given by Sam Kaplan in May 2015.

- Deepdive on group locality and gregarious data restructuring given to projects collaborators in July 2015.
- Deepdive on containment domains within SWARM given to project collaborators in August 2015.
- Presentation on FreshBreeze at ICCS'15 [LiEtAl15].
- Presentation containment domains within SWARM at ParCo'15 [KaplanEtAl15].
- Presentation on Gregarious data restructuring within many-core architectures at HPCC'15 [ShresthaEtAl15A].
- Presentation on locality aware concurrent start of stencil applications at CGO'15 [ShresthaEtAl15B].
- Presentation on automatic cluster parallelization and minimizing communication via selective data replication at HPEC'15 [TavarageriEtAl15].
- Presentation on hierarchically tiled arrays as a high level abstraction for codelets at DFM'14 [YangEtAl14].
- Presentation on hierarchically tiled arrays for exascale computing at PADAL'15.
- Ph.D dissertation defense on 'A Framework for Group Locality Aware Multithreading' [Shrestha15].

Note: The participants in this award have made various other presentations not listed here. A list can be provided on demand.

Websites

The URLs listed below contain STI delivered during the course of the DynAX project or software or methods relevant to delivered STI.

- Deliverables of DynAX project:
  https://xstackwiki.modelado.org/DynAX#Deliverables
- Scalapack's two dimensional block-cyclic distribution:
  http://netlib.org/scalapack/slug/node75.html
- NWChem:

http://www.nwchem-sw.org/index.php/Main_Page

- TCE correlation models:
  http://www.nwchem-sw.org/index.php/TCE#CCSD.2CCCSDT.2CCCSDTQ.2CCISD.2CCISDT.2CCISDTQ.2C_MBPT2.2CMBPT3.2CMBPT4.2C_etc._--_the_correlation_models

Appendix A: Reports

Quarterly Reports

- Brandywine XStack Report Q1
- Brandywine XStack Report Q2
- Brandywine XStack Report Q3
- Brandywine XStack Report Q4
- Brandywine XStack Report Q5
- Brandywine XStack Report Q6
- Brandywine XStack Report Q7
- Brandywine XStack Report Q8
- Brandywine XStack Report Q9
- Brandywine XStack Report Q10
- Brandywine XStack Report Q11
- Brandywine XStack Report Q12

Annual Reports

- Brandywine XStack Report Y1
- Brandywine XStack Report Y2
- Brandywine XStack Report Y3

NCE Reports

- ETI NCE Report
- PNNL NCE Report
- Reservoir Labs NCE Report

Appendix B: List of Tasks

Number	Information
1.1	Research scheduling policies
1.2	Research scheduler linkages such as tree hierarchies or other graphs
2.1	Research compiler hints/directives for runtime scheduling
2.2	Research data placement topologies
2.3	Research memory access semantics
2.4	Research compiler code generation for data placement and movement
3.1	Target codelet execution model
3.2	Generate SWARM codelet code
3.3	Improve scalability of polyhedral mapping
3.4	Optimize unstructured mesh computations
3.5	~~Optimize structure based codes~~
4.1	Representation of sparse arrays.
4.2	Irregular tiles
5.1	Design of the PIL API
5.2	Implementation of the PIL API
5.3	Evaluation of the PIL implementation and API
5.4	~~Locality of enhancement and task parallelization~~
5.5	~~Removal of barriers~~
5.6	~~Aggressive inlining and simplification of data parallel operations~~
5.7	SCALE and R-Stream code generation
5.8	Optimize dense PIL loop codes
5.9	~~Optimize dense Chapel loop codes~~
6.1	Develop list of compact application kernels
6.2	Evaluate baseline performance and energy requirements
6.3	Design codelet-based version of compact applications
6.4	Research algorithm and solution stack improvements
6.5	Further study of performance and energy requirements
6.6	~~Iterative optimization of codelet based version of compact applications~~
7.1	Define intermediate representation
8.1	Research integration of containment domain execution and recovery with codelet scheduling
8.2	~~Research integration of CD persistence infrastructure API with SWARM~~
8.3	~~Research language and compiler integration of coarse-grained and fine-grained checkpointing operations~~
9.1	Research multi layered power optimized data representations
9.2	Investigate initial and static data placement at each memory level
10.1	Study MPI interoperability
10.2	~~Study OpenMP Interoperability~~

Appendix C: ETI NCE Report

Note: The following pages contain the attached no-cost extension report from ETI. PNNL and Reservoir Labs reports are available on the DynAX wiki.

NCE Project Progress Report for ET International Inc.

DynAX: Innovations in Programming Models, Compilers and Runtime Systems for Dynamic Adaptive Event Driven Execution Models

Award Number: DESC0008716
Dates of Performance: 9/1/2015 to 12/31/2015
Report Date: 12/31/2015

Principal Investigator
Guang Gao, ET International. Inc.

CoPIs:
Benoit Meister, Reservoir Labs, Inc.
David Padua, University of Illinois Urbana Champaign
Andres Marquez, Pacific Northwest National Laboratories

Introduction

ETI sent an NCE request letter on August 14th. This report outlines the work performed by ET International Inc. during the no-cost extension period granted. During the period we worked on two types of tasks: (1) tasks related to the dissemination of STI, and (2) tasks related to coordination and organization.

Summary

To summarize our STI dissemination efforts, (1) ETI presented our work on containment domain based resiliency within SWARM at the 2015 International Parallel Computing Conference (ParCo'15) in Edinburgh, Scotland[4], and (2) based on feedback from the community, we revised and extended our paper which is to be included in the primary ParCo'15 proceedings.

To summarize our coordination and organization efforts, (1) ETI as the lead institution coordinated with other PI institutions for the completion of their NCE, (2) organized completion of the Y3 report, and (3) put together the final report.

Dissemination of STI

The following sections discuss CD based resilience and contain improvements based off of feedback from the community following our presentation at ParCo'15. These changes are also reflected in our extended paper.

Containment Domain Based Resilience

For completions sake we will repeat that at a high-level, a containment domain contains four components: data preservation, to save any necessary input data; a body function which performs algorithmic work; a detection function to identify hardware and software errors; and a recovery method, to restore preserved data and re-execute the body function. The detection function is a user defined function that will be run after the body. It may check for hardware faults by reading error counters, or for software errors by examining output data (e.g. using a checksum function). Since containment domains can be nested, the recovery function may also escalate the error to its parent. Since no coordination is needed, any number of

[4] S. Kaplan, S. Pino, A. Landwehr, G. Gao. **"Landing Containment Domains on SWARM: Toward a Robust Resiliency Solution on a Dynamic Adaptive Runtime Machine."** To Appear in the proceedings of the 2015 international Parallel Computing conference (ParCo'15). Edinburgh, Scotland, UK, September 1 – 4, 2015.

containment domains may be in existence, with multiple preserves and recoveries taking place simultaneously.

Containment Domains API within SWARM

The containment domain API has been modified from what we initially reported to allow for additional features. Specially, the preservation calls have an additional parameter called *type* allowing the user to specify into which domain to preserve data. Additionally, we have clarified the language used in the API descriptions. The API's are now as follows:

swarm_Containment_Domain create(parent): Create a new containment domain as a child of the specified parent domain.

swarm_Containment_Domain begin(THIS, body, body ctxt, check, check cxt, done, done ctxt): Begin execution of the current containment domain denoted by THIS by scheduling the codelet denoted by body ctxt. When the codelet finishes execution, the codelet denoted by check ctxt is scheduled to verify results. If the result of the execution is TRUE then the codelet denoted by done cxt is scheduled.

swarm_Containment_Domain preserve(THIS, data, length, id, type): In the containment domain denoted by THIS, do a memory copy of length bytes from data into a temporary location inside the CD. We support multiple preservations per CD (e.g. to allow preservation of tiles within a larger array, such that the individual tiles are non-contiguous in memory), by adding a user-selected id field. For each containment domain in SWARM, a boolean value is set based on its execution status.On the first execution, data is preserved normally. On subsequent executions, data is copied in reverse (i.e. from the internal preservation into the data pointer). The CD in which the data is preserved is denoted by type. This can either be the currently activated CD or the parent CD.

swarm_Containment_Domain finish(THIS): Close the current containment domain denoted by THIS, discard any preserved data, and make the parent domain active.

Experimental Results

We've expanded upon and clarified the results we initially obtained and provided a more thorough discussion from that of the quarterly reports. We evaluate our CD based approach through three primary means: feasibility, efficiency, and resilience and make a number of key observations. To show that our prototype implementation has sufficient functionality, we instrument a Cholesky decomposition program in SWARM to use containment domains. For the experiments, the program was run on a dual-processor Intel Xeon system, using 12 threads. The workload sizes were confirmed to not exhaust the physical memory of the machine.Though we found insignificant variance between runs, we have averaged all times over 5 program runs due to the natural variation in run time due to extraneous system factors (such as scheduling differences).

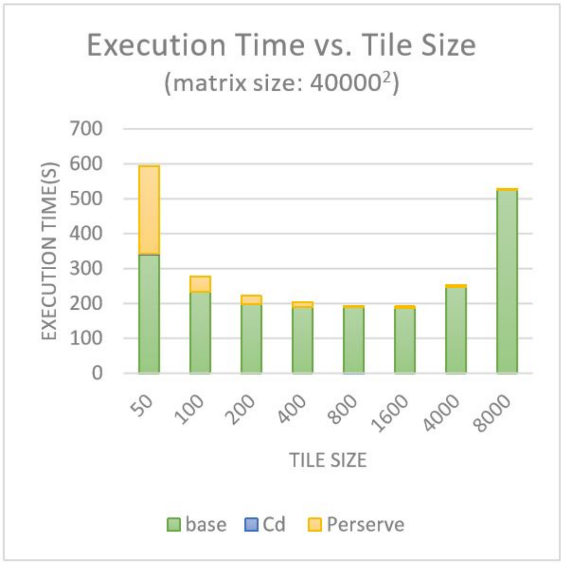

Figure 1: Execution Time of Cholesky

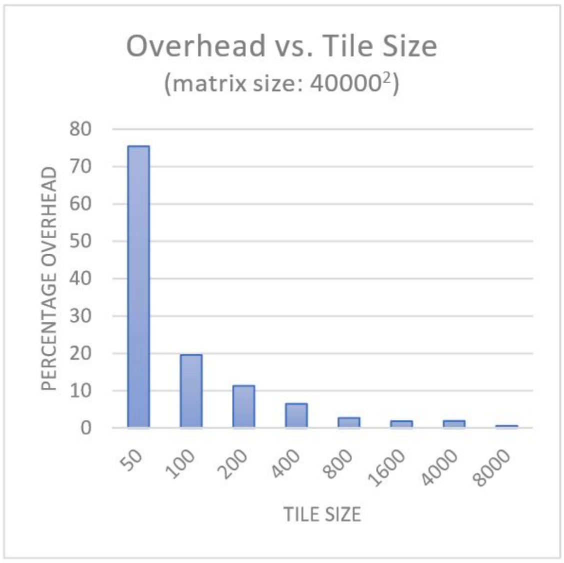

Figure 2: Percentage overhead

The Cholesky program has three main codelets, one for each linear algebra routine run on a tile (POTRF, TRSM, and GEMM/SYRK), and each of these is called a number of times for each tile. For our purposes, each of these is considered a containment domain. In our model, we only considered faults similar to arithmetic errors; that is, incorrectly calculated results. For this reason, we did not need to preserve input data unless it would be overwritten by an operation (e.g. the input/output tile for a POTRF operation). In order to simulate arithmetic errors, rather than relying on error counters from actual faulty hardware, a probabilistic random number generator is used. If a random number is below the specified configurable threshold, a fault is deemed to have occurred. Fault generation occurs within the check codelets causing checks to fail at random and the subsequent re-execution of the entire failed containment domain.

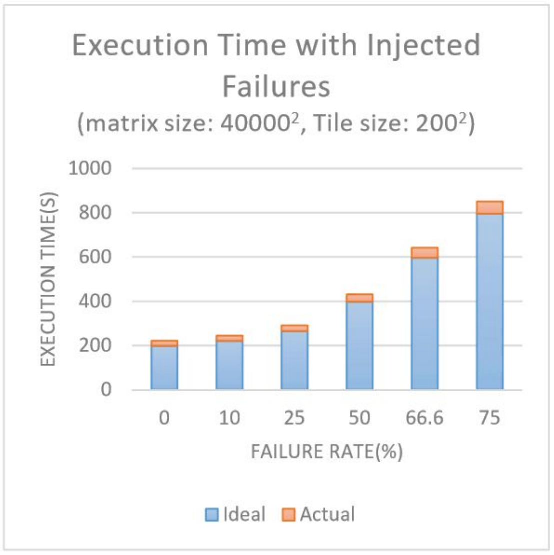

Figure 3: Execution time with simulated errors

Firstly, through the implementation of our framework in SWARM, and the working Cholesky application using said framework, we observe that it is feasible to adapt a codelet-based application to use containment domains. Secondly, Our implementation shows very low overhead. Figure 1 shows the execution time for various tile sizes executing Cholesky of size 40000x40000. Base denotes the Cholesky kernel runtime without containment domains or data preservation. CD denotes the time spent within CD related API calls without preservation. Preserve denotes the time spent preserving data. One can see that the API itself adds negligible overhead and that the only significant overhead comes from actual preservation of data. As with many tile based approaches to computation, choosing an inappropriate tile size will cause performance degradation due to inefficient use of caches or by limiting parallelism. In this particular case, tile sizes above 1600 limit the core utilization to 10 or 5 threads respectively, yielding the U-shape shown in the graph. Figure 2 shows the total overhead (preservation+API calls)relative to the base cholesky code without containment domains. The trends indicate that as tile sizes (workload per codelet) increase the overhead is mitigated and eventually becomes negligible.This trend is unsurprising given that the runtime overhead per API call is relatively constant and as the tile size increases less and increasingly larger data sized preservation calls are made to the runtime. Additionally, the cost of preservation (i.e. for data movement) increases at a much slower rate than the cost of the Cholesky computation as tile sizes are increased. Overall, this trend shows that there is a sweet spot in terms of granularity and that proper decomposition is key to mitigate preservation overheads and maximize performance.

Figure 3 shows simulated injected failures that result in codelet re-execution within the SWARM framework.The idealized case is computed by taking the average execution time without faults for a Cholesky of size 40000x40000 and tile size of 200x200, and computing the expected execution time for various fault rates using the geometric distribution. In order to accurately access the overhead of our implementation, this projected base execution time includes neither CD or preservation overhead.The actual case shows the execution times actually obtained from running SWARM with injected failures. We note that there is around 11% overhead without failures and that this overhead decreases to 6% at a failure rate of 75%. This is because some allocation and API overheads are not present upon re-execution. Trend wise, we note that maximal overhead occurs when faults are not present in the system. We additionally note that the execution time follows reasonably well to that of the idealized case and that it is possible to project with reasonable accuracy the execution time in the event of failures using data from actual runs without failure.

Coordination and Organization Efforts

During the NCE period, we prepared the Y3 and Final reports to be submitted to the DOE. Activities included coordinating and communicating with each collaborating institution for the content and publications to submit as well as interfacing with the other institutions to make sure that the final report contained a smooth interface referencing the NCE reports by each institution which were submitted to the program manager as well as ETI. With respect to this, we received Reservoir Lab's NCE report dated 10/31/2015; PNNL's NCE report dated 12/23/2015. UIUC provided their report directly to the DOE on 8/31/2015. UIUC did not additional work after this date and will not be using the remaining funds. Additionally, we posted the NCE reports to the XStack Project Wiki. Direct Links to reports are included at the end of this document.

Status

We have completed the Y3 and Final reports and are in the process of submitting all documentation to the DOE and placing all documentation on the XStack Modelado wiki.

Report Links

The following URLs lead directly to other reports that have been submitted directly to the DOE for the DynAX project:

- PNNL NCE Report
- Reservoir Labs NCE Report
- Year 3 Report