PNNL NCE

DynAX

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

Award Number: DE-SC0008716
Dates of NCE Performance: 9/1/2015 to 12/31/2015
Report Date : 12/23/15

Principal Investigator

Guang Gao
ET International Inc
100 White Clay Center Dr, Newark DE 19711

Co-PI

Andres Marquez, Pacific Northwest National Laboratories

Executive Summary

Expected completion times for these tasks were impacted by personnel changes for this project at ETI (PI) and PNNL (Co-PI), causing realignment issues that are now fully resolved but initially introduced some additional delay.

For Task 9.1, this includes (1) a comparative study of the Architected Composite Data Types (ACDT) framework developed under the program on SWARM vs. an OCR implementation as part of measuring the overall impact of this technique across these runtimes; (2) the completion of the documentation with these new findings. For 9.2, this includes the completion of the documentation of the Group Locality (GL) compiler and the results of the applications that exercise it.

Task 9.1

The implementation of ACDT in SWARM already showed substantial potential of performance and power efficiency improvements for selected kernels like sparse Cholesky, as presented in the ICPADS'14 paper end of last calendar year.

Figure 1

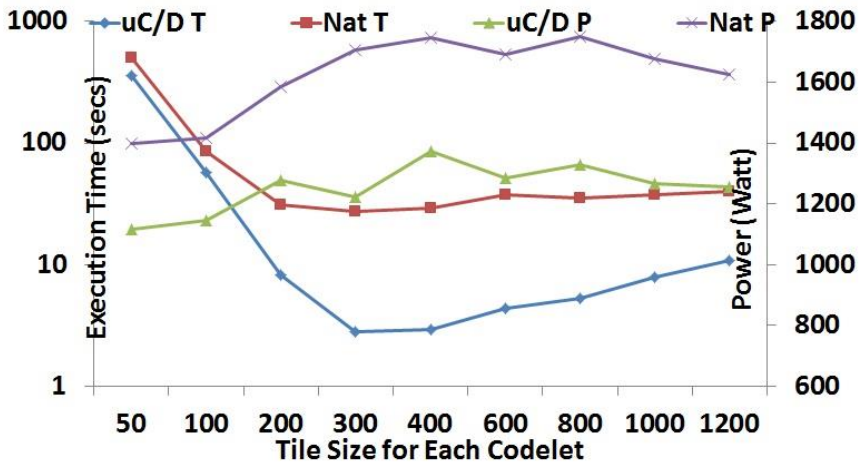


Figure 1 shows a sweep over various Cholesky tile sizes where SWARM ACDT execution time (uC/D T) can outperform standard SWARM Cholesky (Nat T) by two orders of magnitude. In addition, power savings for SWARM ACDT Cholesky (uC/D P) can yield ~30% compared to the standard SWARM Cholesky (Nat P).

It stands to reason that similar benefits can be had if we were to apply similar principles to OCR.

The Pacific Northwest National Lab Open Community Runtime (P-OCR) is a framework that implements the OCR standard across current distributed systems, exploring how fine-grained event driven tasks, movement of data, and dynamic resource adaption provide scalability for regular and irregular applications while managing those constraints.

With P-OCR on a general purpose cluster, we have found substantial speed-ups for various kernels of interest to DOE. E.g., a parallel tiled Smith-Waterman kernel using 4096 cores with the help of ACDT – showcased as a SC'15 poster at the DOE booth. The findings show that this speedup is even more substantial than for

Cholesky, as studied in the past. Figure 2 and Figure 3 demonstrate the relative speedups for various kernels within a node and across nodes for P-OCR.

Figure 2

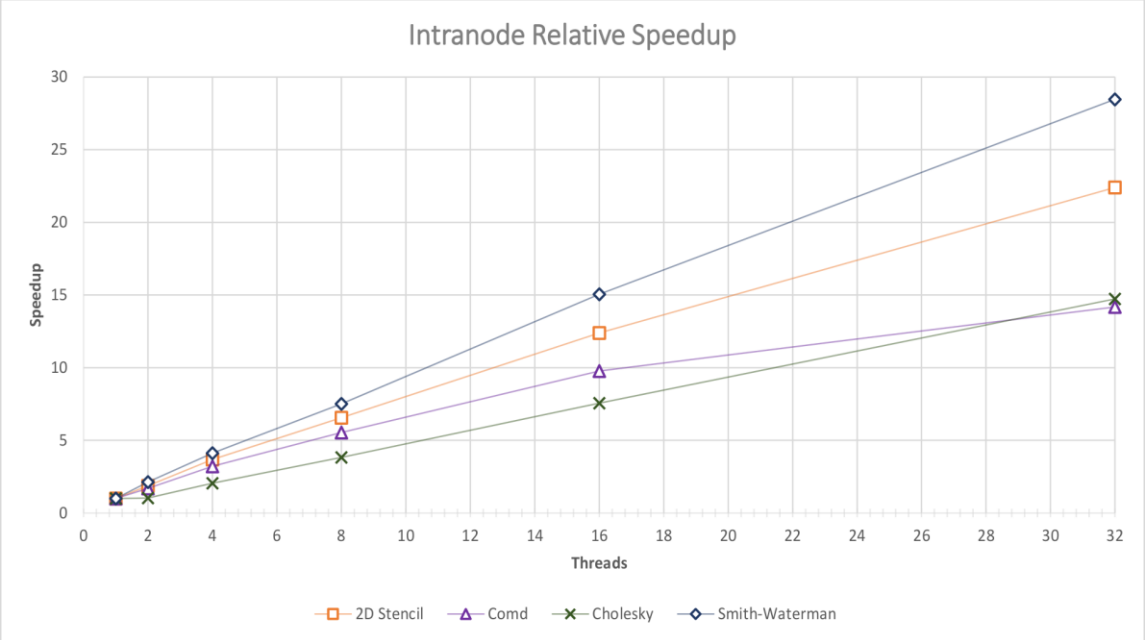
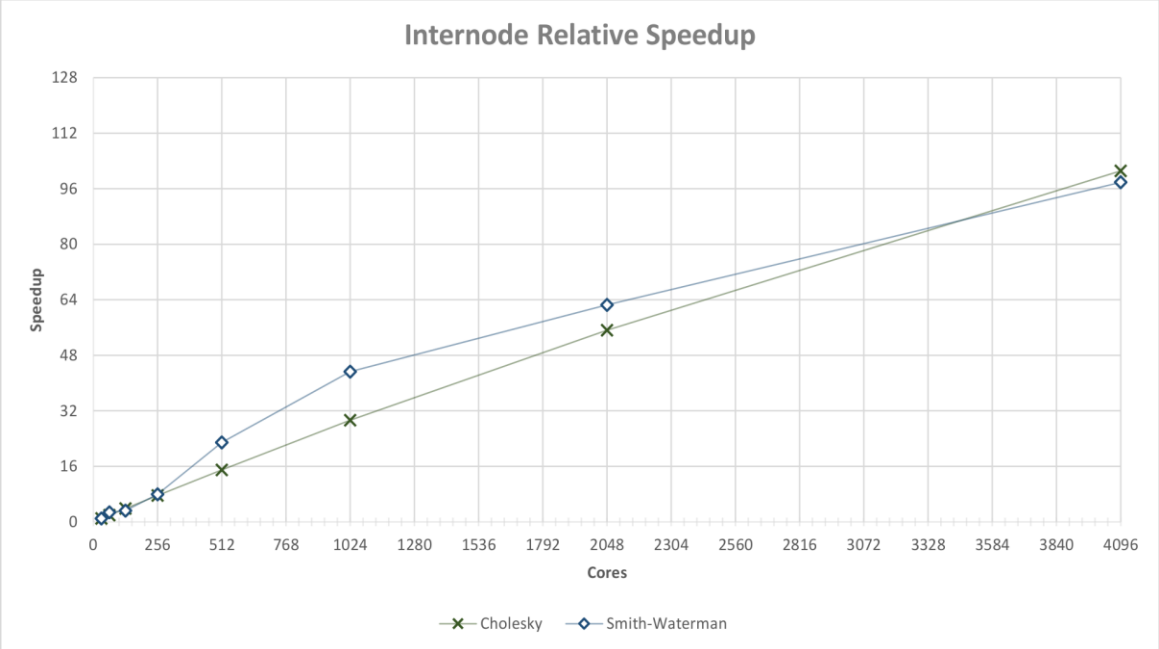


Figure 3



In summary, initial findings indicate that the ACDT methodology is applicable across different runtime systems with the potential of orders of magnitude improvements for selected kernels.

Task 9.2

Introduction

The concept of Group Locality (GL) has is origins around 2012. The basic ideas are simple yet the realization of these ideas is rather complex: In GL, multiple thread groups work together in close proximity in time and space as scheduling and allocation units, taking advantage of a group's data movements, hereby collaborating at a very fine-grain level within a group and across groups.
The task presented here examines GL orchestration with a-priori knowledge gained through static analysis at compile time for stencil kernels, a class of kernels important to DOE. The task was successfully completed, yielded numerous papers in renowned conferences, and resulted in a very interesting Ph.D. thesis as added bonus.

The rest of this section will highlight salient excerpts of Dr.'s Shrestha Ph.D. thesis. A complete manuscript of the thesis can be found under reference: "A Framework for Group Locality Aware Multithreading", Ph.D. Thesis by Sunil Shrestha, University of Delaware, 2015.

Jagged Tiling

Jagged Tiling is a new concept developed under GL that addresses the apparent contradiction of enabling efficient parallelism at multiple, hierarchical levels of granularity – in our lingo: parallelism within a tile group and across tile groups. Classical approaches concentrate on facilitating parallelism at only one level, usually the coarsest level. Jagged Tiling reshapes these tiles under valid transformations in such a way that parallelism is enabled at all levels of the hierarchy.

Figure 4 (a) two level classical hierarchical tiling (b) jagged tiling

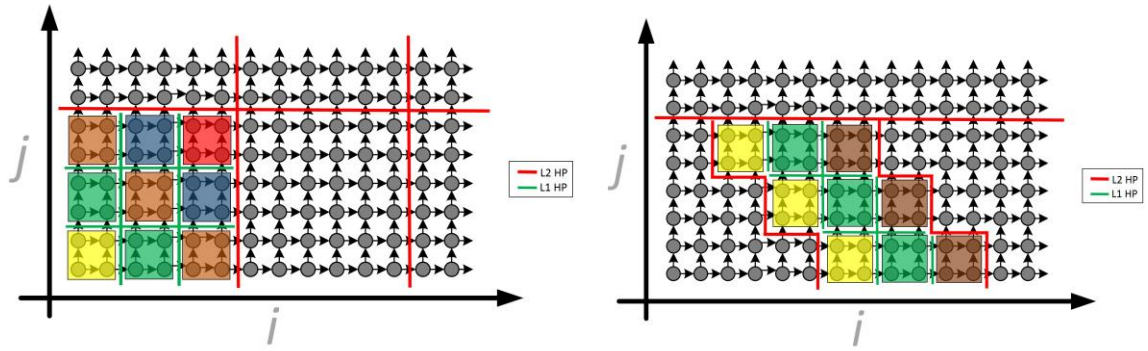


Figure 4 exemplifies jagged tiling reshaping from (a) to (b) spanned by a two level iteration space (i,j), that manifests itself as a uniform color front along the chosen execution front (L2 HP), hereby "rippling" the parallelism within a group across goups.

Gregarious Data Restructuring

It only appears natural that once scheduling units orchestrate threads in unison there should be opportunities to share data movements as well. We know that GPUs do this, so it is a valid question to examine data movement sharing in the context of GL. Gregarious Data Restructuring can go a step farther by reshaping data units *once* by a thread for improved ingestion to the benefit of all threads in a group.

Figure 5: Strided access (shown by downward arrows) and row arrangement of parallel tiles (A-E) transformed to have contiguous access using calculated offset (shown in red).

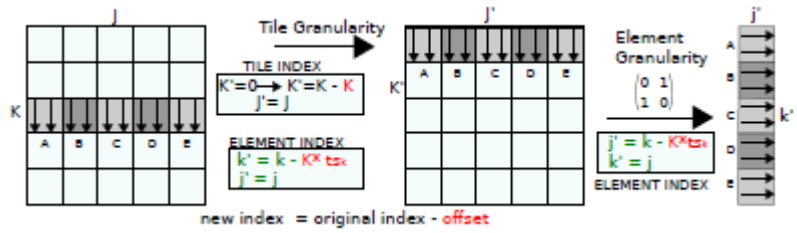


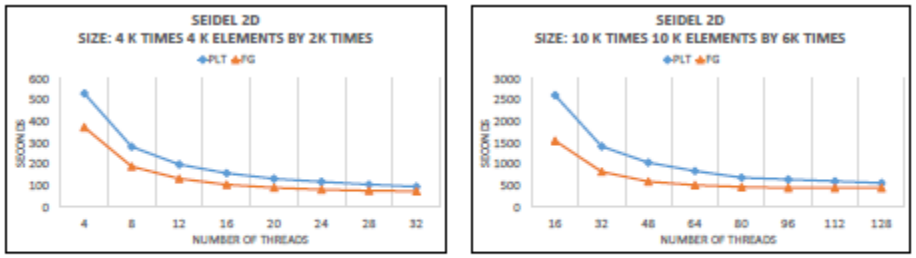
Figure 5 shows and example of a data transformation under GL that would be beneficial to LU decomposition.

Result Excerpts

(for a more detailed, in-depth analysis consult thesis)

Results presented in this section compare a reference PLUTO code compilation and optimization (PLT) vs. the GL framework (FG) running on an Intel Xeon Phi.

Figure 6

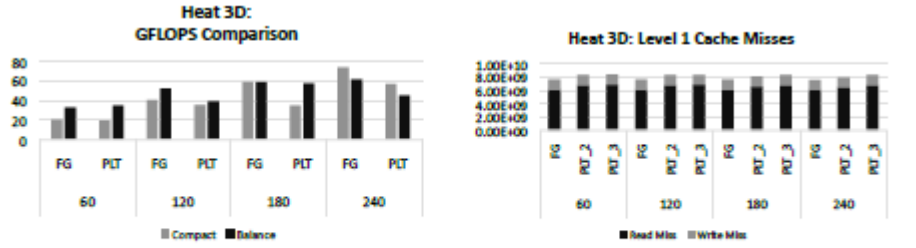


(a) Seidel 2D 4 thousand by 4 thousand elements Execution time

(b) Seidel 2D 10 thousand by 10 thousand elements Execution time

Figure 6 shows thread sweeps over the execution time of a 2D Seidel over two different data sets. The GL framework consistently outperforms the reference implementation.

Figure 7

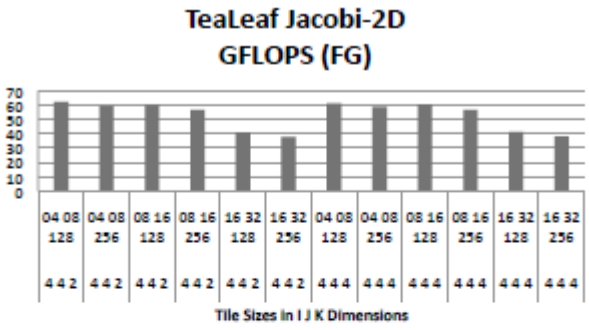


(a) Heat 3D GFLOPS comparison

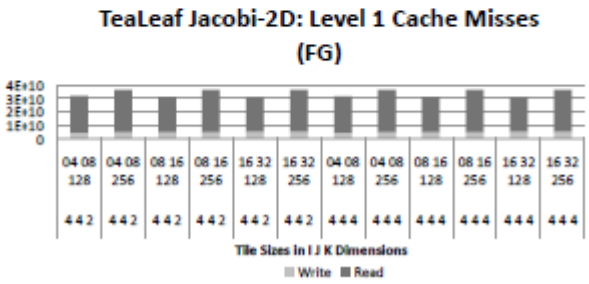
(b) Heat 3D Level 1 Cache Miss Behavior for PLUTO (PLT) and Jagged Tiling (FG)

Figure 7 shows advantageous comparisons in favor of GL for the Heat 3D kernels.

Figure 8: DOE Mini-App TeaLeaf Jacobi-2D



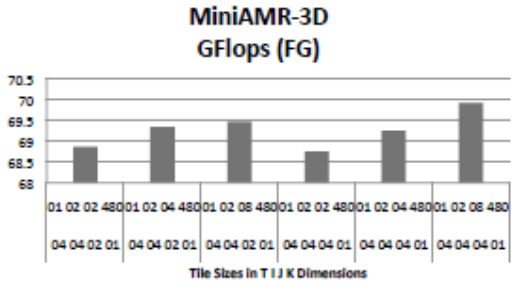
(a) TeaLeaf GFlops Performance



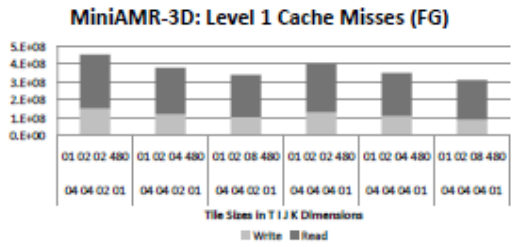
(b) TeaLeaf L1 Cache Misses

Figure 8 shows the performance of the Jacobi-2D stencil kernel embedded in the DOE Mini-App TeaLeaf. The three numbers on the x-axis represent the tile sizes in (I,j,k). GL outperforms by ~30% the reference implementation in PLUTO.

Figure 9: DOE MiniAMR-3D



(a) miniAMR GFlops Performance



(b) miniAMR L1 Cache Misses

Figure 9 shows the performance of the 3D stencil kernel embedded in the DOE Mini-App MiniAMR. GL and the reference implementation in PLUTO both reach 70GF, albeit at different tiling configurations.