April 2019

Application-Specific Memory Subsystem Benchmarking

Mahesh Lakshminarasimhan

Boise State University
Application-Specific Memory Subsystem Benchmarking

Abstract
Achieving high application performance depends on the combination of memory footprint, instruction mix and order, and memory access patterns. Most memory benchmarks which provide information on the achieved memory performance are confined to simple access patterns that are not representative of patterns found in real applications.

We present AdaptMemBench, a configurable benchmark framework designed to explore the performance capabilities of compute kernels extracted from applications. AdaptMemBench provides a framework to emulate application-specific memory access patterns. The build system accommodates the polyhedral model, which provides a convenient testbed for code optimizations. AdaptMemBench supports reproducibility in experimental results and facilitates sharing results.

This scholarly poster is available at ScholarWorks: https://scholarworks.boisestate.edu/gss_2019/39
PROBLEM STATEMENT
Optimizing scientific application performance is a continual challenge due to:
- The lack of understanding of performance capabilities of the target application.
- The complexity and diversity in contemporary multicore computer architecture.
- The effect of memory access patterns on achieved memory performance.

\textbf{GOAL} Generate energy-efficient optimized code

\textbf{AdaptMemBench: A Configurable Memory Benchmarking Framework}

- Benchmark any access pattern
- Flexible for evaluating optimizations
- Supports any degree of parallelism
- No limitations on working set size
- Enhances research reproducibility
- Easy to port experimental results

\textbf{THE POLYHEDRAL MODEL}

- Source Code
- Iteration Space
- Domain Variables
- Synthetic Variables
- Space Variables
- Set Variables
- Problem Formulas
- \( D = [m, n] = [i/j]; 1 \leq i < m \) and \( 0 \leq j < n \)

\textbf{INTERLEAVED SCHEDULING FOR TRIAD}

\begin{align*}
\text{For} \ (\text{int} \ i = 0; \ i < n; \ i++) \{ \\
A[i] = B[i] + \text{scalar} \times C[i]; \\
\}
\end{align*}

\begin{align*}
\text{For} \ (\text{int} \ i = 0; \ i < n/2; \ i++) \{ \\
A[i] = B[i] + \text{scalar} \times C[i]; \\
A[i+n/2] = B[i+n/2] + \text{scalar} \times C[i+n/2]; \\
\}
\end{align*}

\textbf{TILING OPTIMIZATION FOR STENCILS}

- A stencil operation updates each point in a structured multidimensional grid with weighted contributions from a subset of its neighborhood elements.
- Rectangular Tiling blocks along the spatial dimension only. It improves data locality.
- Overlapped Tiling blocks along the spatial as well as the temporal dimensions. It enhances parallelism.

\textbf{HARDWARE COUNTERS AND TIMERS}

PAPI provides low overhead access to hardware performance counters. It helps in understanding the interaction of software and hardware stack. It aids with performance analysis, tuning, and monitoring.

\textbf{Performance Exploration and Optimization Using AdaptMemBench}

- Extract expensive kernels
- Identify performance bottlenecks
- Manipulate data layout & execution order
- Explore potential optimizations
- Visualize raw performance metadata

- Design of the Framework
- POLYHEDRAL CODE GENERATION
- PATTERN SPECIFICATIONS
- CONFIGURE BENCHMARK TEMPLATE
- CUSTOM BENCHMARK DRIVER
- BENCHMARK DRIVER TEMPLATE
- LATE ANALYSIS