Extracting Scalable Parallelism by Relaxing the Contracts across the System Stack

Mahmut Taylan Kandemir, Anand Sivasubramaniam, Chita Das
Computer Science and Engineering, The Pennsylvania State University
{kandemir, anand, das}@cse.psu.edu

Abstract
Most of the existing software-level approaches to data-intensive computing have focused on either algorithmic innovations or system-level optimizations. In contrast to these prior efforts, our project employs Approximate Computing as a means to address the Big Data problem. The key components of our approach include: (i) a detailed application and architectures characterization; (ii) language/compiler support for approximate computing; and (iii) runtime support for approximate computing. Our ultimate goal is to develop a cross-layer framework based on approximate computing to study accuracy-performance-energy tradeoffs.

Current Status
• Exploiting staleness for approximating loads on multicores: using stale values of data for reducing coherence overheads in multithreaded execution.
• Approximate clustering: reducing the number of checks in k-means.
• Rearchitecting decision tree learning via approximate computing: eliminating redundant checks in decision tree traversals.
• Optimizing sparse access patterns: approximating sparse data access patterns using affine expressions.

Motivational Result: Potential of Eliminating Coherence Overheads

Exploiting Staleness for Approximating Loads on Multicores
• We propose to use coherence invalidated data for answering load misses in multithreaded applications.
• Three techniques: Reading Invalidated Lines (RIL), Stale Victim Cache (SVC), Stale Victim Cache with Time Bound (SVC+TB).
• Scientific applications pose inherent resilience towards staleness of data.
• Exploiting resilience of SPLASH2 applications shows a maximum of 0.08% error.
• Execution time speedup of up to 28.6% in SPLASH2 applications with an average of 10-15% across the entire suite.

Mechanisms for Exploiting Staleness
• On a load miss, two possible scenarios can take place: (i) The line is not present in the cache at all (none of the tags match), and (ii) The line is present in invalid state.
• For exploiting staleness, we answer the load using the invalid line in the second case (RIL).
• Since invalid lines are potential candidates for replacements, to maximize the benefits of using stale data, we use an 8 line Stale Victim Cache (SVC) to store coherence invalidated lines.
• To prevent applications from using “too stale” values, we also have a time bound on the life of invalid lines residing in SVC (SVC+TB).

Salient Features
• No change required from the programmer.
• Checkers for our applications to help us measure the impact of our approach on accuracy.

Checkers to Quantify Error (%)
• For Barnes, we compare the resultant position, velocity, and potential of each particle and report the % deviation.
• For Radiosity, we compare the final equilibria of RGB values and report the Euclidean deviation.
• For Raytrace, we report the average Euclidean deviation between the pixels in the output images.
• For Water, we check the resultant force between all pairs of water molecules in XYZ planes and report deviations.
• The other benchmarks (Cholesky, FFT, LU, Ocean and Radix) have their own checkers.

Side Effects
• Speedup: Since we answer load misses faster, we reduce parallel execution time.
• Error: When an execution uses older values for computation, it may deviate from the control flow of the accurate execution.
• Instruction Count: This deviation could in turn lead to a change in the total number of instructions (prolonged execution times), or a change in the output of the execution (erroneous).

Ongoing Work
• Currently using GEM5 simulator to measure and study performance and correctness.
• We are working on providing new tools/metrics for measuring approximations.
• New emphasis on approximate computing in the offering of CS parallel computing class.

Figure 1: Potential benefits

Figure 2: Modification to the coherence protocol

Figure 3: Speedup improvement

Figure 4: Loss in accuracy