

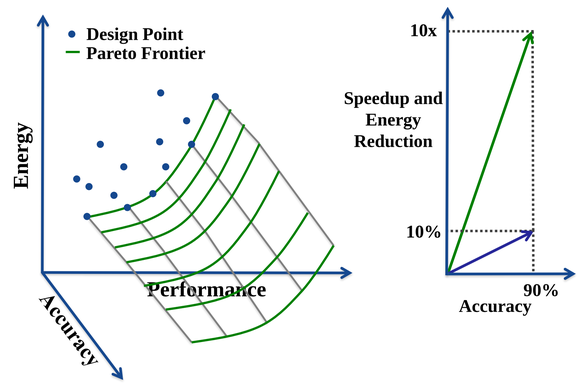

XPS: FULL: CCA: Scalable Approximate Computing for Data Parallel Applications

PIs: Scott Mahlke, Z. Morley Mao, Jason Mars, Lingjia Tang
EECS Dept., University of Michigan, CCF-1438996

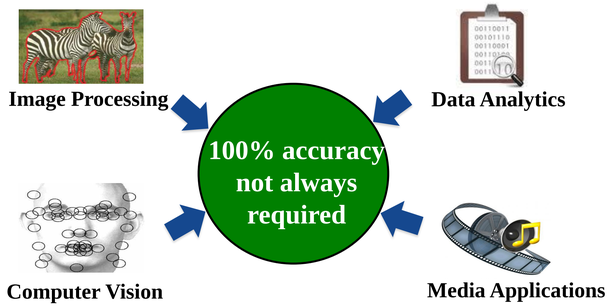
Project Objectives

- Attack key challenges to make approximate computing pervasive
  - What to approximate?
  - How to approximate?
  - How to detect approximation error?
  - How to manage execution and the user experience?
- Hardware vs. software approximation methods
- Demonstrate with real-world prototypes

Approximate Computing



Opportunities for Approximation

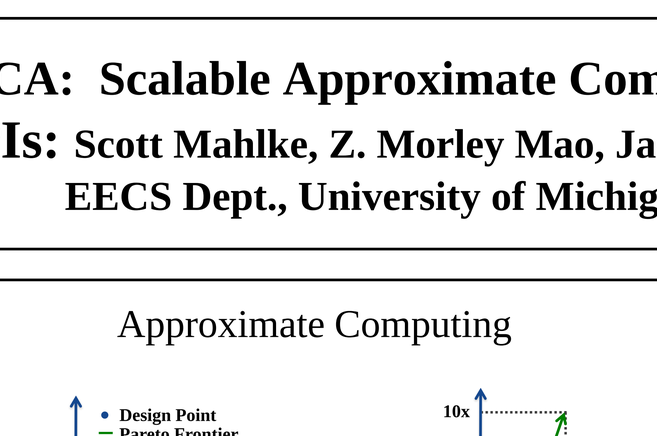


Online Quality Management

Provide Strong Guarantee on Quality

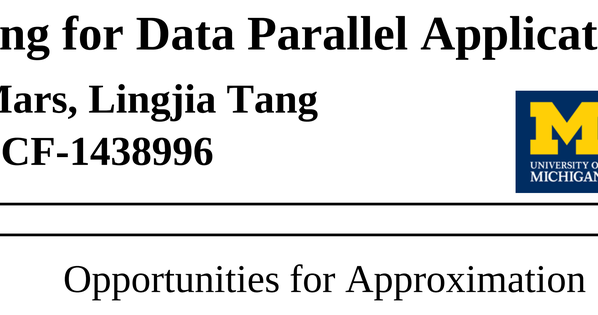
- Avoid large errors in output elements
- User tunable accuracy and efficiency

Lightweight Online Quality Control



Quality control made practical by:
- Predictive detection
- Continuous monitoring
- Tunable with quality feedback

Quality Vs. Recomputations (inversek2j)

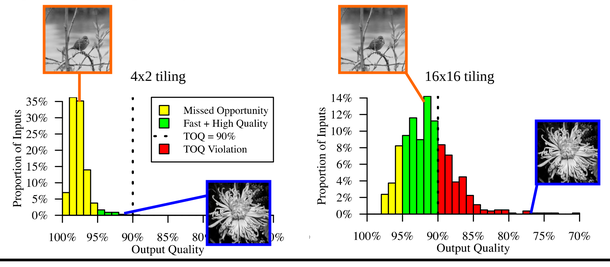


- On average, 2x reduction in mean relative error

Input Responsive Approximation

The Case for Input Responsiveness

- Key question in approximation systems – *how to approximate (and how aggressively)?*
- The answer depends on the input
- Example – gamma correction + tiling approx. on 800 images, target output quality = 90%



Taking Advantage of Differing Inputs

- Goal – make a decision about how aggressively to approximate for individual inputs
- Our approach
  1. Create canary input, a small version of full input that possesses key properties of full input

  2. Quickly test approximation options on canary, choose the most effective option
  3. Apply most effective approximation to the full input

Large Speedup, Small Quality Loss

- Software-based, can be used on commodity systems today
- Far larger speedups than state-of-art software systems (average – 10.8x IRA, 2.5x SAGE)