WACAS March 2, 2014

Improving Coverage and Reliability in Approximate Computing Using Application-Specific, Light-Weight Checks

Beayna Grigorian, Glenn Reinman
UCLA Computer Science Department
{bgrigori, reinman}@cs.ucla.edu

Introduction

Existing Approaches: Application quality is often coupled with the accuracy of the unit of approximation (i.e. approximate accelerator)

+ Efficient quality analysis using ***offline, static techniques***
- Potential compromise of ***coverage*** and ***reliability***

Cases that potentially result in unacceptably inaccurate solutions are exempted from approximation

Cannot provide absolute guarantees for satisfying QoS constraints

Our Approach: Leverage high-level, application-specific metrics, or ***Light-Weight Checks***, for dynamic error analysis and recovery

Light-Weight Check (LWC)

Key Insight: While finding a solution may be complex, checking the quality of that solution could be simple

Characteristics

- ***Light-weight*** to evaluate (relative to application)
  - Usage at runtime: Test approximated output and initiate recovery if needed
- ***Application-specific***, yet ***algorithm-independent***
  - E.g. Scene analysis for physics-based simulation

Benefits

- Reliable, dynamic guarantees on user-specified QoS
- Better coverage for potentially good approximations
- Platform-agnostic with negligible overhead

Application Examples

Application	Sample Algorithm	Application Domains	LWC
Inverse Kinematics	Cycle Coordinate Descent	Robotics, Gaming, Graphics	Forward Kinematics
State Estimation	Kalman Filter	Navigation, Finance, Signal Processing	Measurement Comparison
Physics-Based Simulation	Gilbert-Johnson-Keerthi Distance Algorithm	Fluid Dynamics, Control Systems, Gaming	Energy Conservation
Image Denoising	Total Variation Minimization	Computer Vision, Medical Imaging	Universal Image Quality Index (UIQI)

Implementing LWCs

How do I *find* an LWC?

- **LWCs are user-defined**
- LWCs could be based on:
  - ***Internal values*** (i.e. inputs, approximated outputs, and intermediate values)
  - ***External values*** (e.g. mobile robot application with supplemental sensory feedback)
- Certain application categories may have easy-to-identify and/or reusable LWCs
  - Iterative refinement applications
  - Image processing applications

How do I *use* an LWC?

- LWC is integrated directly into the application
- Code is modified to execute the following:

(1) Call approximate accelerator

(2) Evaluate LWC; determine QoS

(3) **If QoS constraint is not met**:

➔ Initiate recovery

➔ Reprocess current input with exact computation

(4) Continue to next input

Implementing LWCs

How do I *find* an LWC?

- **LWCs are user-defined**
- LWCs could be based on:
  - ***Internal values*** (i.e. inputs, approximated outputs, and intermediate values)
  - ***External values*** (e.g. mobile robot application with supplemental sensory feedback)
- Certain application categories may have easy-to-identify and/or reusable LWCs
  - Iterative refinement applications
  - Image processing applications

How do I *use* an LWC?

- LWC is integrated directly into the application
- Code is modified to execute the following:

(1) Call approximate accelerator

(2) Evaluate LWC; determine QoS

(3) **If QoS constraint is not met**:

→ Initiate recovery

→ Reprocess current input with exact computation

(4) Continue to next input

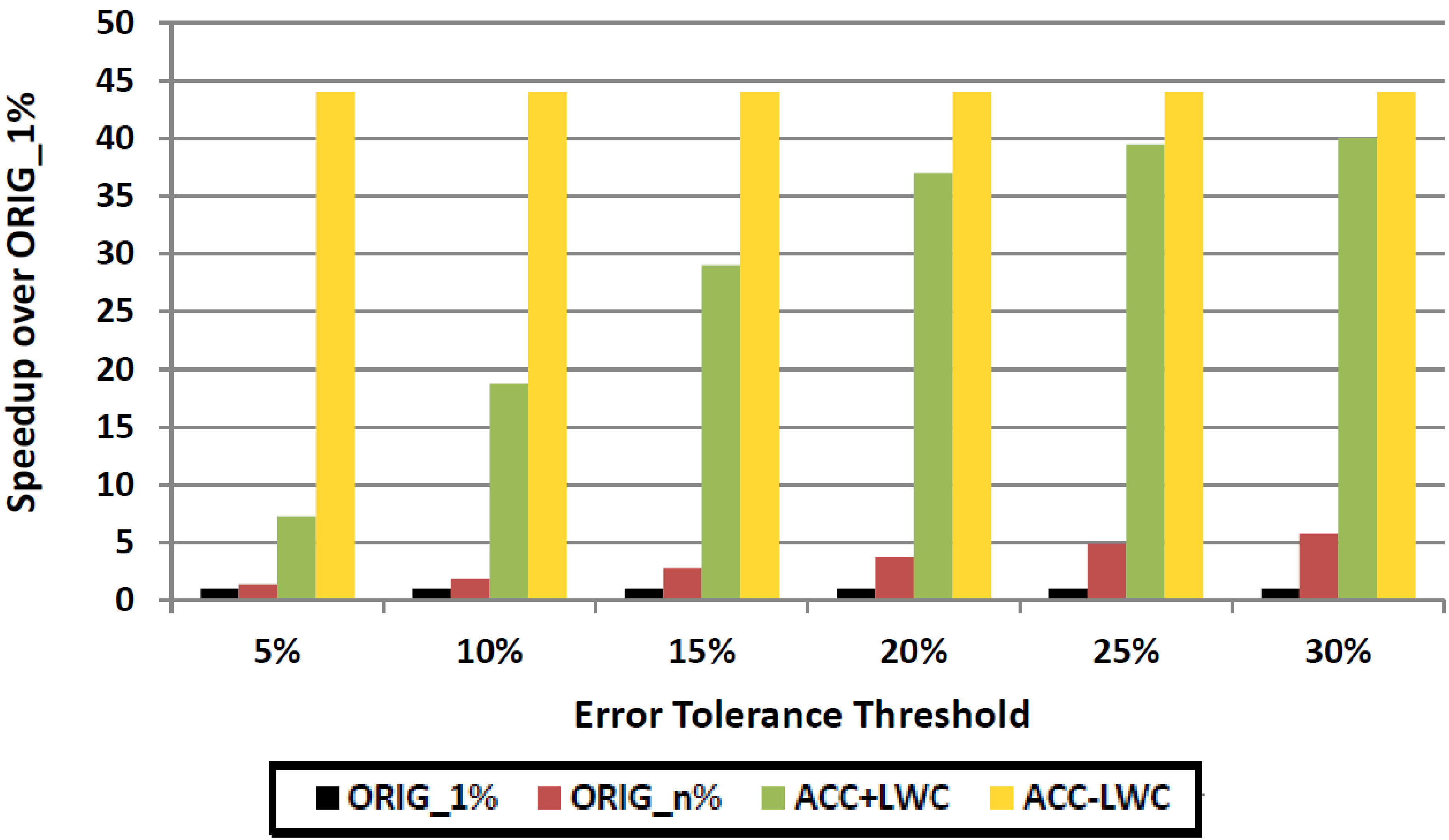
→ **Compiler support?**

Experimental Setup

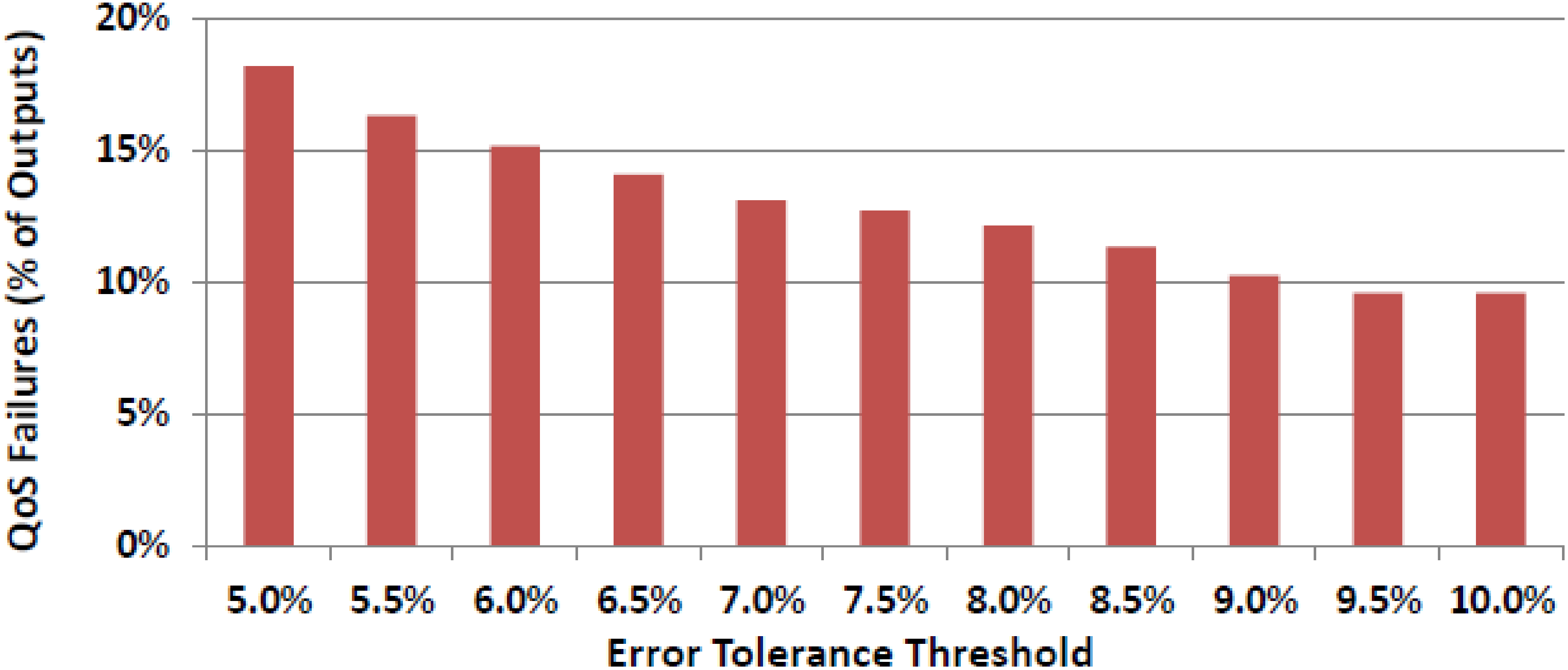
- **Benchmark:** *Inverse Kinematics* (3-joint arm)
- **Error**: Distance from end effector to target location
- **Error Tolerance Threshold**: maximum percentage of error the user is willing to accept for *any* application output
- **Approximation**: Software-based Neural Network (8x8)
- **Schemes**
  A. **ORIG_1% – orig. benchmark (1% set threshold)**
  B. **ORIG_n% – orig. benchmark (adjustable threshold)**
  C. **ACC+LWC – benchmark integrated w/ NN & LWC**
  D. **ACC-LWC – benchmark integrated w/ NN & no LWC**

■ ORIG_1% ■ ORIG_n% ■ ACC+LWC ■ ACC-LWC

Results: *Performance*

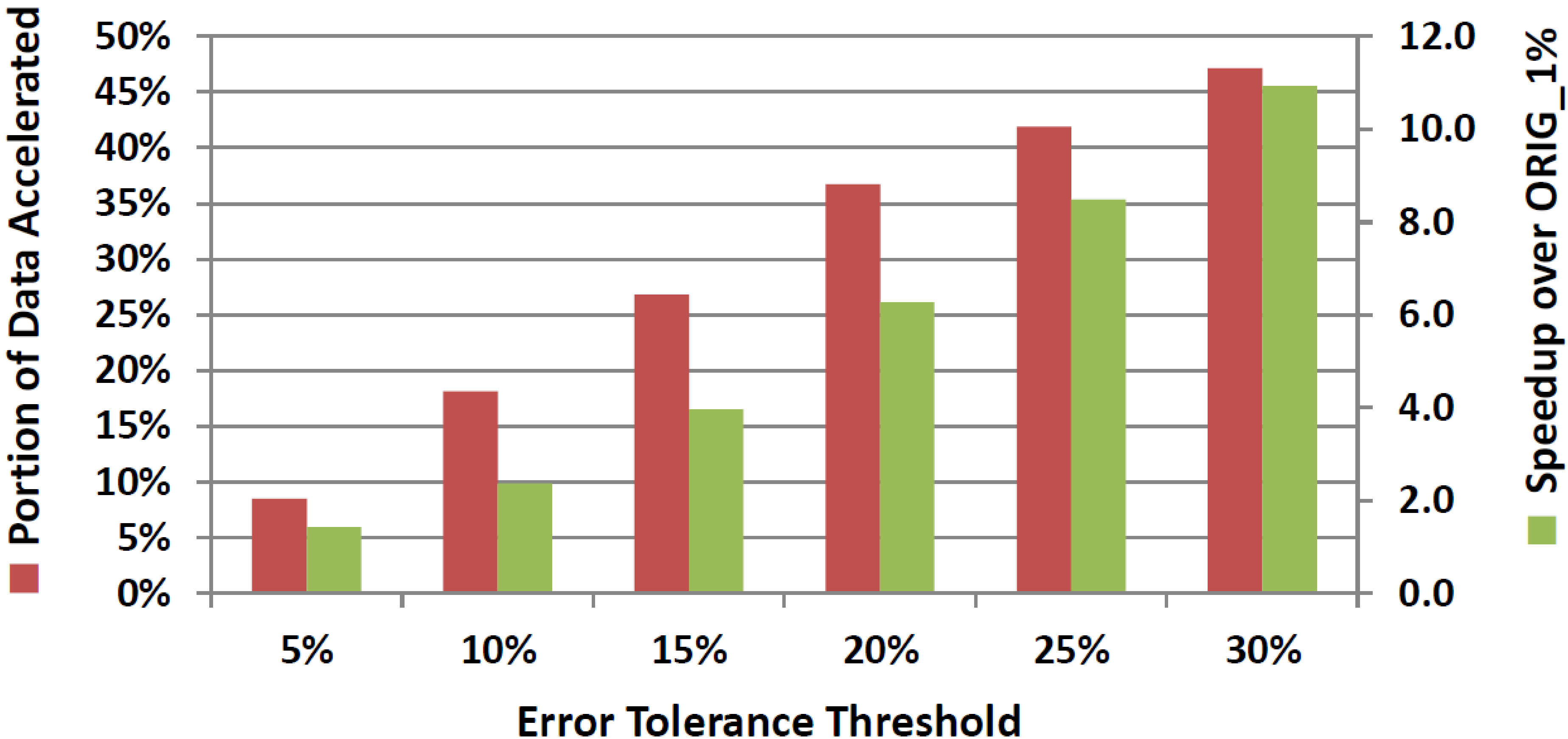


Results: *Reliability*

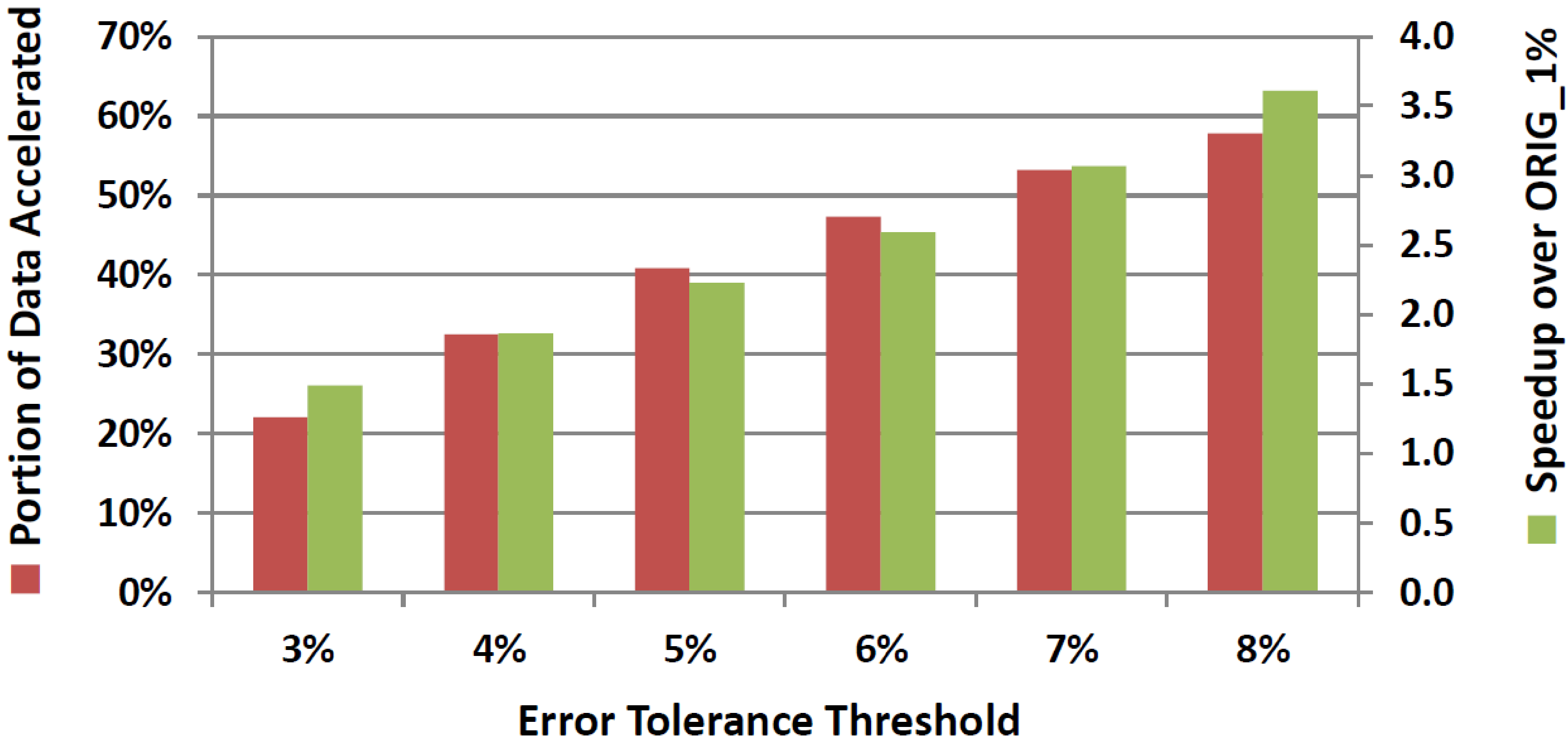


- **ACC-LWC: No LWC ➔ No dynamic reliability!**
- **Significant portions of data are subject to failed QoS**

Results: *Coverage for Out-of-Range Inputs*



Results: *Coverage w/ Less Accurate Approx.*



Conclusion

- <u>Main Idea</u>: Leverage application-level tolerance of imprecision to improve ***coverage*** and ***reliability***

- <u>Approach</u>: Perform online error analysis and recovery based on ***LWCs***

- Platform-agnostic in nature, LWCs allow for an elegant solution to dynamic error control

Questions?

Thank you!