From Detection to Optimization: Impact of Silent Errors on Scientific Applications

SC'16 Doctoral Showcase

Jon Calhoun, Luke Olson (Advisor), and Marc Snir (Advisor)

University of Illinois at Urbana-Champaign

15 November 2016

Why look at errors in HPC?

Computing systems are vulnerable to failure

- Component cost and power budget dictate how reliably components are
- Rates of errors may increase as design trade-offs become more mainstream, e.g. low power, cheaper commodity parts

Need efficient error detection and recovery schemes

Slowing of Moore's Law demands we need more out of current hardware

- **Approximate computation**
- **Approximate data movement**

Silent Error Detection

Error Detection

Resilient applications are critical when operating in fault prone environments

Solving linear systems is central to many HPC applications

Algebraic Multigrid (AMG) is and efficient solver than can be used as a preconditioner for Conjugate Gradient (CG) and Generalized Minimal Residual (GMRES)

What happens to AMG when it encounters SDC?

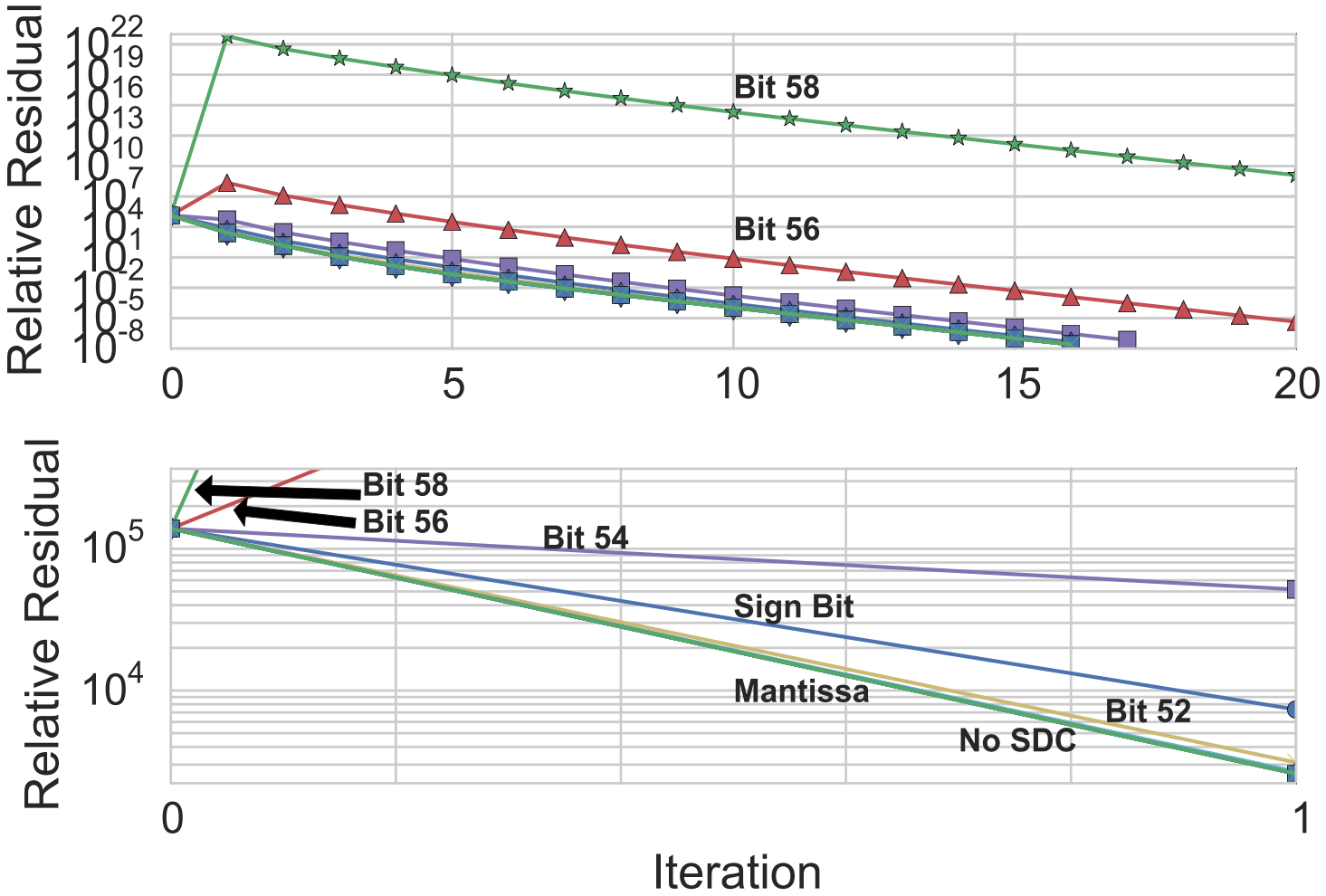
Possible Bad Outcomes

- Slows down convergence
- Converge to wrong solution
- Program crash

Possible Good Outcomes

- Nothing
- Converges more quickly

Why should we care about SDC in AMG?



Impact of a selective single bit-flip injection into a residual calculation of a 2D Poisson problem.

AMG Resiliency

Detectors:

- Residual Check - heuristic bound on range of residual
- Energy Check - monitor energetic stability of problem

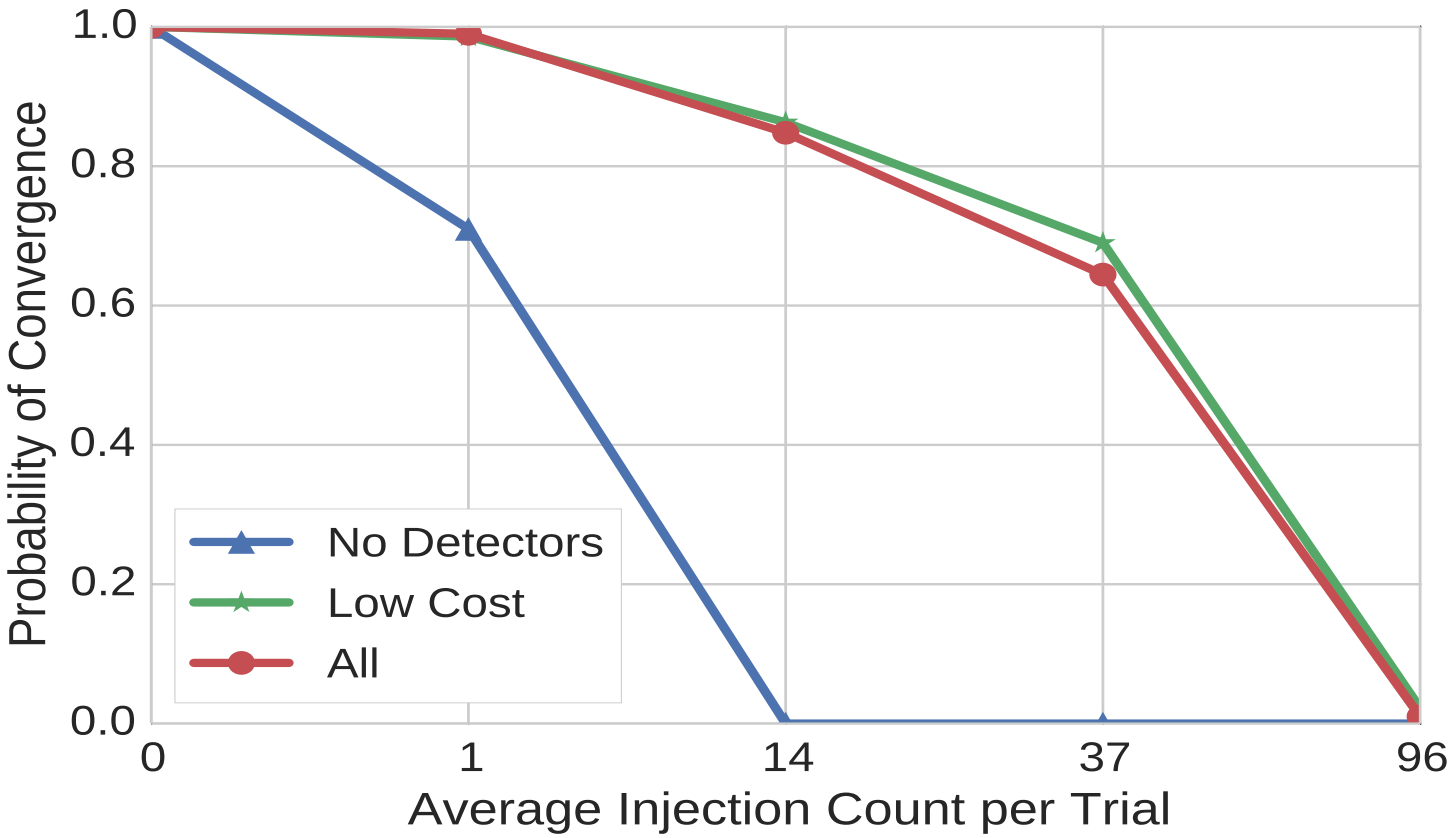
Recovery:

- Local - idempotent linear algebra operations
- Level - restart from previous level in AMG hierarchy
- Cycle - restart iteration

Configurations:

- **Low cost** - Residual check + Full recovery (1% at 1K cores)
- **All** - Full detection + Full recovery (18% at 1K cores)

Convergence Results Solving 2D Laplacian



Normal AMG (blue) not resilient to multiple injections

All suffers more segfaults due to frequent level restarts

Error Propagation

Why look at error propagation?

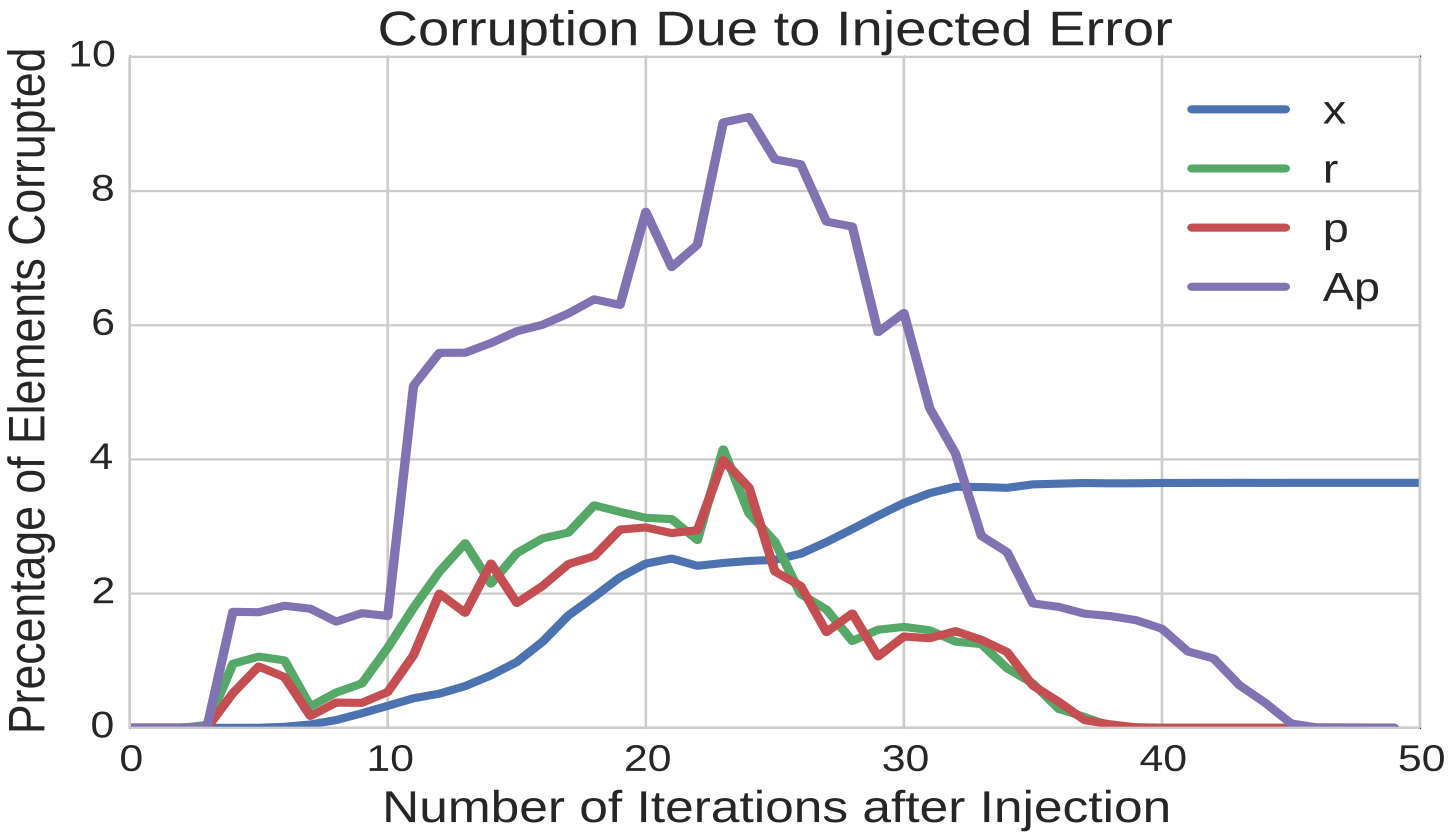
Large changes in a value or a crash are easy to detect

Latency of detection from a silent error can be high

Understanding how silent errors propagation will allow:

- More resilient code
- Creation of better detectors
- Helps define containment boundaries for local node/process level recovery

HPCCG - Propagation of All Injection Types

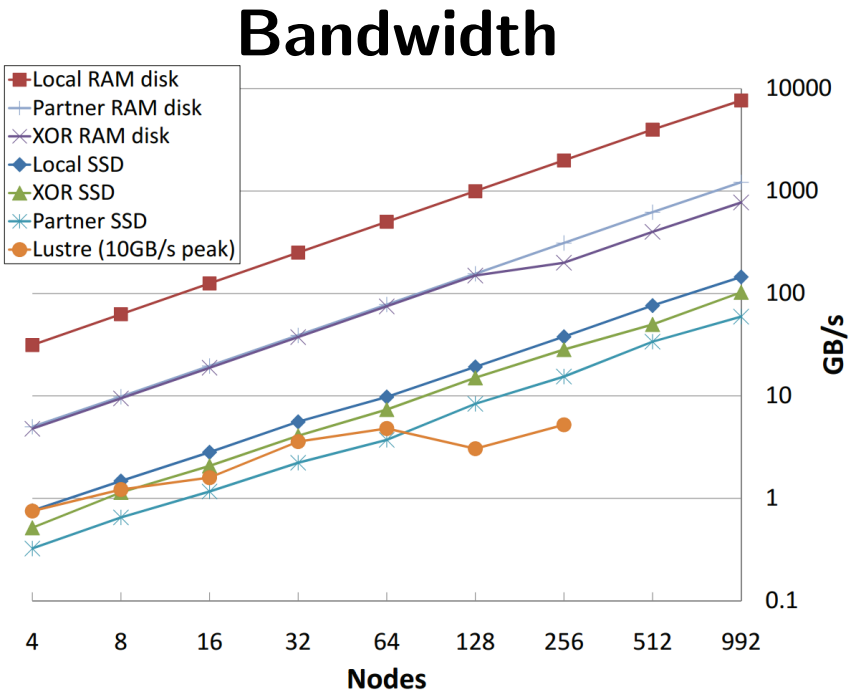


Average number of elements corrupted on injected rank
Propagation of error due to SpMVs

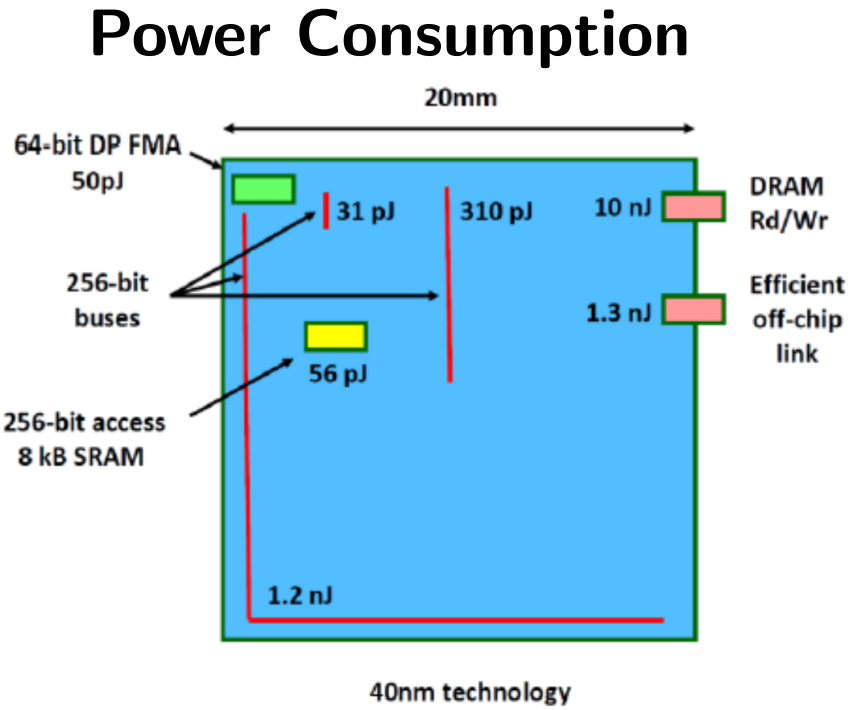
Lossy Compression

Data Movement Problem

Data movement is becoming a limiter to HPC performance due to:



Aggregate write bandwidth [Moody et al. 2010]



Power consumption for CPU operations. [Keckler 2011]

Compression

Data compression techniques fall into two categories:

Lossless

- No data loss
- Small compression factors
- Can be costly (time)
- Research on integration in hardware memories [Pekhimenko et al. 2012] [Sardashti, Seznec, and Wood 2014]

Lossy

- Some data perturbation
- Larger compression factors
- Low/moderate cost
- Research on using in HPC checkpointing [Laney et al. 2013] [Ni et al. 2014] [Sasaki et al. 2015]

Solutions to many HPC applications are approximations

Let's investigate how lossy compression can be effectively used

Lossy Compression

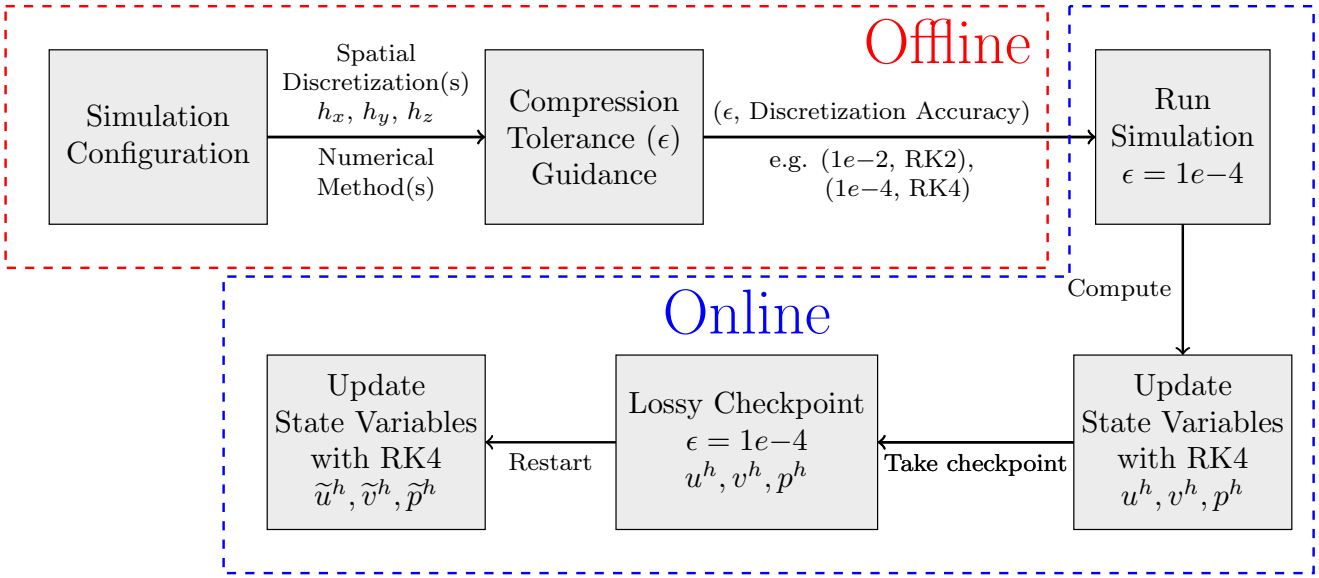
Majority of my remaining PhD research will be devoted to better understanding lossy compression, and answer the following questions:

1. How to select a lossy compression error tolerance?
2. Can lossy compression be used for inter-node communication?
3. Can we leverage problem specific information when selecting a compression tolerance?

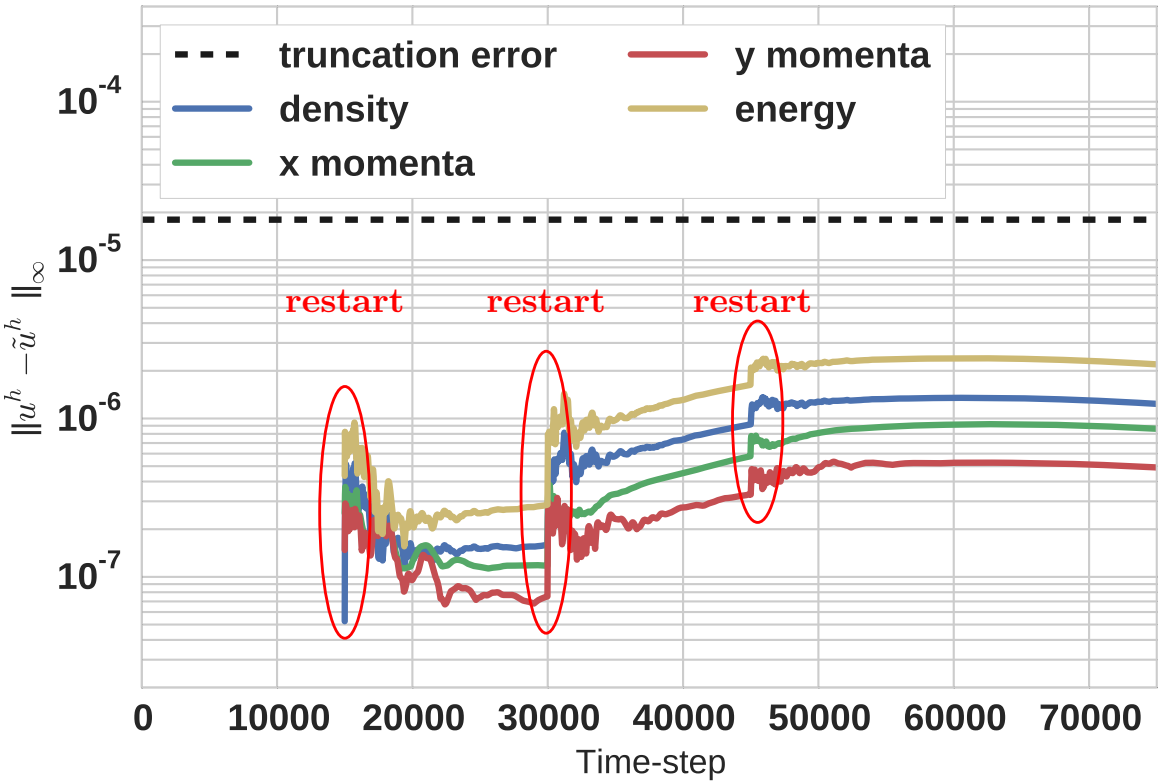
Selecting Lossy Compression Error Tolerance

Select compression tolerance less than truncation error of numerical method:

- Use RK4 to solve 1-D PDE with $h_x = 0.1$ (accurate to $1e{-}4$)
- $\widetilde{u}^h = u^h + \epsilon$ is equivalent to u^h if $\epsilon < 1e{-}4$.
- $\widetilde{u}^h$ with $\epsilon > 1e{-}4$ can be mapped to related method – e.g., $\epsilon = 1e{-}2$ is RK2.



PlasComCM - Max Error



- Average compression factor of 7x with SZ-1.3 [Di and Cappello 2016]
- Error accumulates with each checkpoint, but is slowly removed
- Accumulation is still less than truncation error

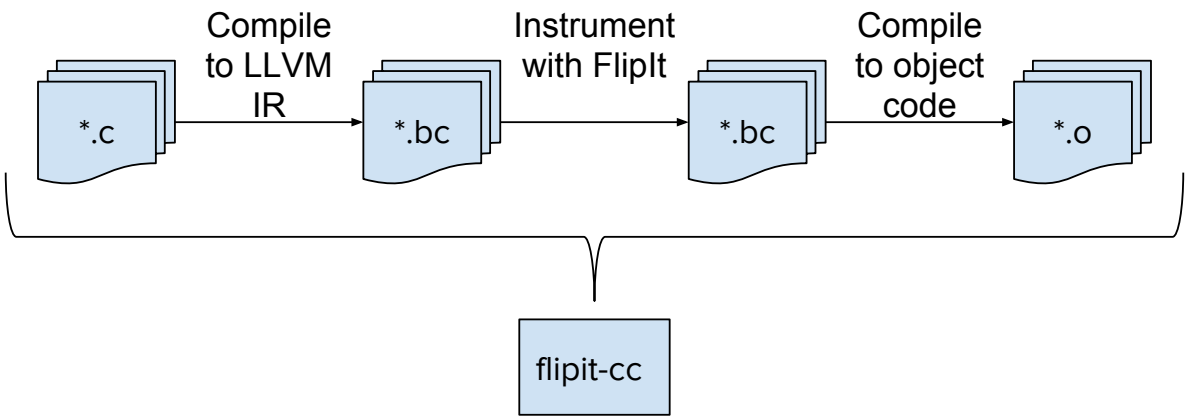
Tools and Software

Fault Injection - FlipIt

Fault Model:

- Memories protected by error correcting codes (ECC)
- Faults arise in processor computations and manifest as register perturbations (single bit-flip)

FlipIt is an LLVM compiler pass that instruments compiled code to support fault injection at runtime



FlipIt allows for user defined injection probabilities to customize fault injection for a given system

Fault Visualization - FaultSight

Many fault injection frameworks report similar data

No common tool for fault injection visualization

FaultSight Goals:

- Easily generate high level graphs for fault injection campaign
- Quickly select subsets of fault injection data
- Ability to conduct hypothesis testing

Summary

Future HPC systems will experience more errors due to hardware selection and algorithm choice

To prepare for this future, my thesis focuses on:

- SDC detection and recovery for AMG
- Error propagation study
- How lossy compression can be more effectively used

Tools:

- FlipIt - Fault Injector (https://github.com/aperson40/flipit)
- FaultSight - Fault Visualizer (https://github.com/einarhorn/faultsight)

Acknowledgments

- This work was sponsored by the Air Force Office of Scientific Research under grant FA9550-12-1-0478
- This work is supported by the Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract DEAC0206CH11307
- This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications