On the Potential of Significance-Driven Execution for Energy-Aware HPC

Philipp Gschwandtner, Charalampos Chalios, Dimitrios S. Nikolopoulos, Hans Vandierendonck, Thomas Fahringer

University of Innsbruck, Austria
(Queen's University Belfast, United Kingdom)
philipp@dps.uibk.ac.at

September 1, 2014

Outline

- Motivation
- Code significance
  - Example: Jacobi
- Error model and experiment methodology
- Results
- Conclusion

Motivation

- 20 MW power limit (exascale extrapolation of Tianhe-2: 1 GW)
- Conventional saving approaches potentially insufficient (DVFS, power/clock gating, ...)
- Unconventional methods on the rise, e.g. approximate computing, Near-Threshold Voltage computation (NTC)
- But many yield high(er) computational error rates
- Need to examine code susceptibility to errors (and potential energy gains) when using NTC

Near-Threshold Voltage Computation

- Run hardware below specification (closer to threshold voltage than normal —i.e. super-threshold— operation)
- Power saving potential of $10 - 50\times$
- Decreases performance by $5 - 10\times$
- Overall energy reductions between 2 and $5\times$
- Increases probability of errors

NTC Implications

- Need to deal with higher error rates
  - many codes require exact computation
  - some are tolerant, but not to all kinds of errors
    - iterative solvers
    - signal processing codes
- Need to deal with performance degradation
  - parallelism

Fault Classes

- Bit flips in one form or another (e.g. functional units, registers/caches/memory; data/program)
- Software and hardware affected in different ways
  - no impact
  - data corruption
    - looping
    - non-silent: detectable without application knowledge
    - silent: not detectable without application knowledge
  - other (segmentation faults, illegal instructions, ...)

Code Significance

- Measure of susceptibility of code to errors and effect on end result
- Also data can have significance
- Ideas:
  - Is code significance variable?
  - Is there a need for selectively protecting portions of data or code?
  - Can we run code portions on high-power but reliable, and low-power but unreliable hardware to save energy?

Example: Jacobi

- Iterative solver for linear equation systems
- Well-studied
- Computes

$$x_i^{(k+1)} = \omega \left(\frac{1}{a_{ii}} b_i - \sum_{j \neq i} a_{ij} x_j^k \right) + (1 - \omega) x_i^k. \quad (1)$$

if

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|. \quad (2)$$

- Shows varying significance depending on affected data component, time and input data

Significance Depending on Iteration

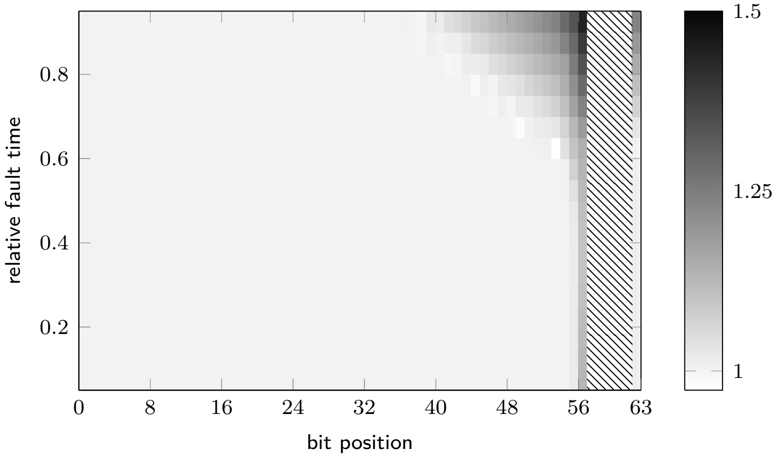

Figure: Relative run time compared to correct Jacobi run for various error times.

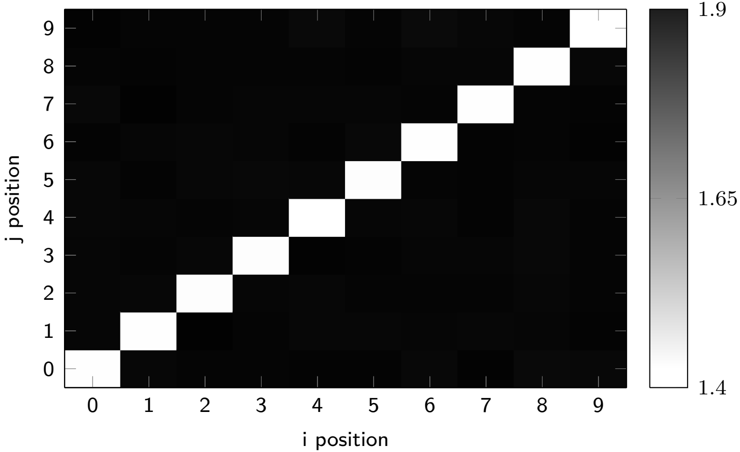

Figure: Relative run time compared to correct Jacobi run for various error locations in A.

Experiment Methodology

- Hardware: quad-socket Intel Xeon E5-4650 Sandy Bridge
- Simulate LLC-resident Jacobi running with NTC
  - simulate errors
    - single bit flips in various bit positions, elements, Jacobi iterations
  - Simulate power/performance effects
    - compare 1 reliable to 16 unreliable cores, same power footprint
    - examine both extremes of performance degradation: $5 - 10\times$
    - obtain Intel RAPL data and correct it with regard to NTC
- Analyze effect on run time and energy consumption
- Evaluate significance of Jacobi with regard to error properties

Sandy Bridge Power Consumption

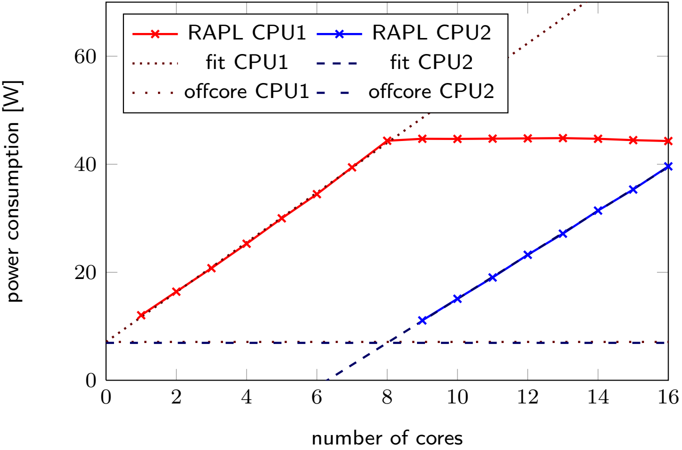

Figure: Power consumption per number of cores for weakly scaling parallel Jacobi runs.

16 Unreliable vs. 1 Reliable Core

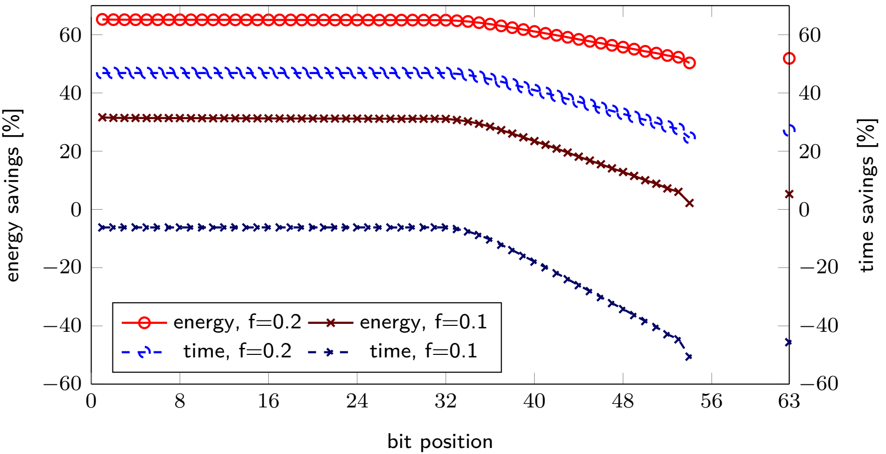

Figure: Energy and time savings over correct, sequential Jacobi for 16 unreliable cores.

16 Unreliable vs. 16 Reliable Cores

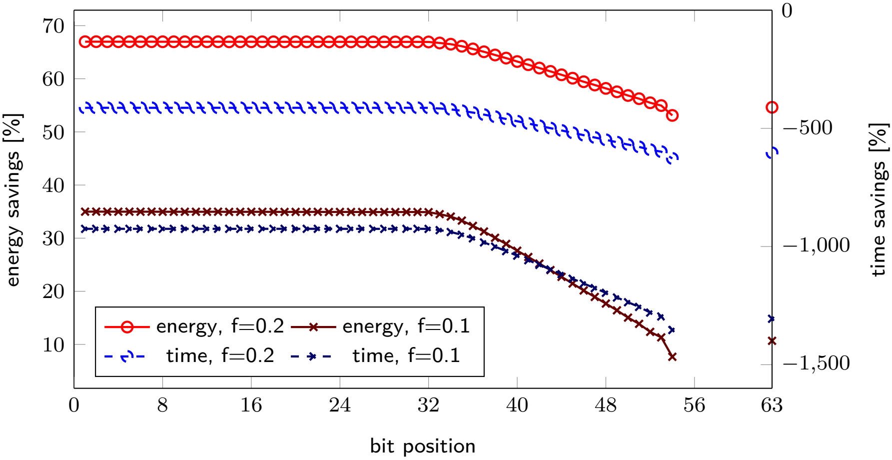

Figure: Energy and time savings over correct, parallel Jacobi for 16 unreliable cores.

Significance Depending on Iteration

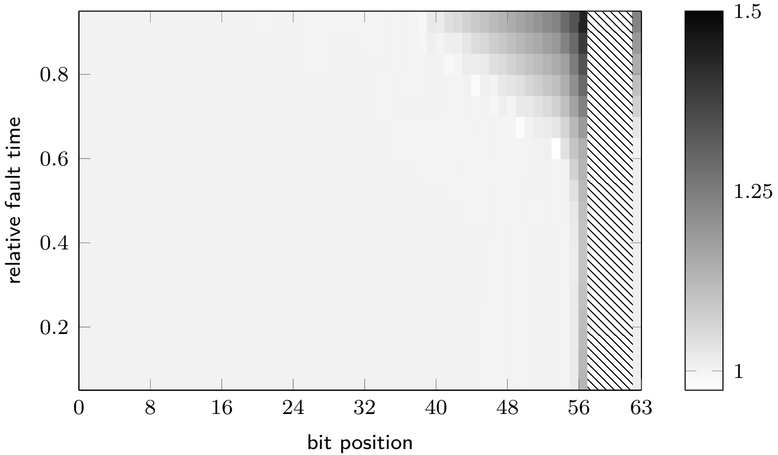

Figure: Relative run time compared to correct Jacobi run for various error times.

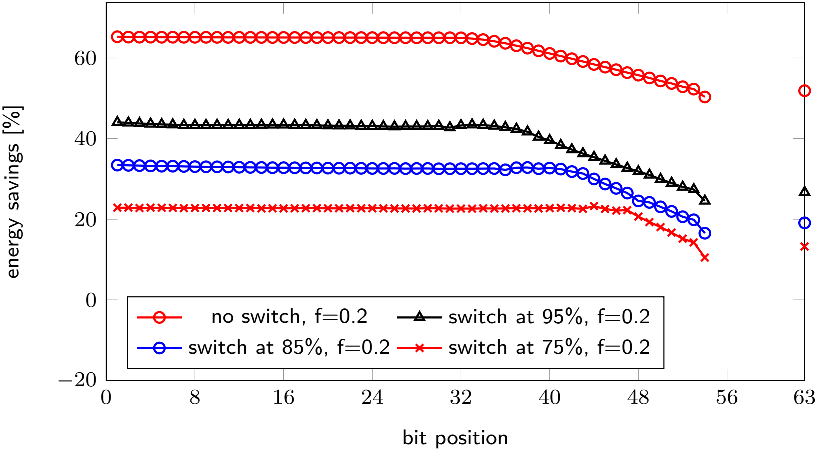

Figure: Energy savings when switching from NTC to reliable hardware during execution.

Significance-driven Execution: Time

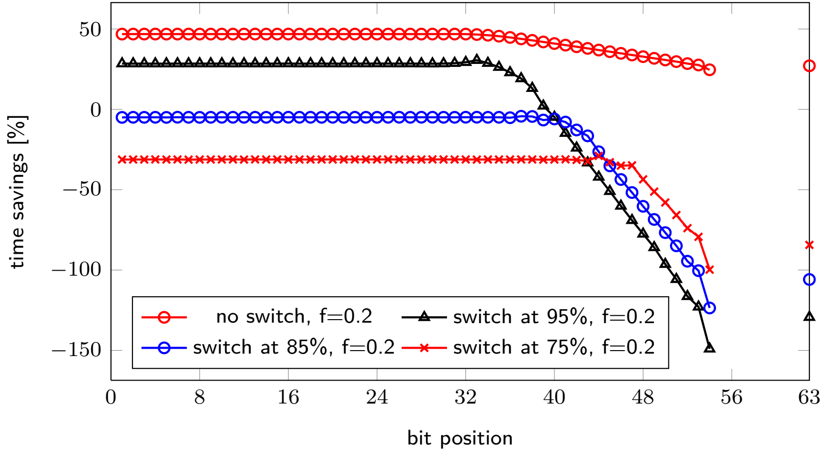

Figure: Time savings when switching from NTC to reliable hardware during execution.

Conclusion and Future Work

- Significance of code and data can be established
- Proof-of-concept: Jacobi
  - Categorization of effects of bit flips
    - none loss in energy or time, no protection necessary
    - observable loss in energy or time, protection optional
    - divergence, protection mandatory
  - significance variation too small to justify running late iterations on reliable hardware
- Future work:
  - analytic/automatic evaluation of code significance
  - examine more codes
  - explore potential protection mechanisms

Thank you!