Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Exploring the Interplay between Energy Efficiency and Resilience for Future Exascale Systems

Shuaiwen Leon Song (PNNL), Tan Li (UCR), Zizhong Chen (UCR), Darren Kerbyson (PNNL)

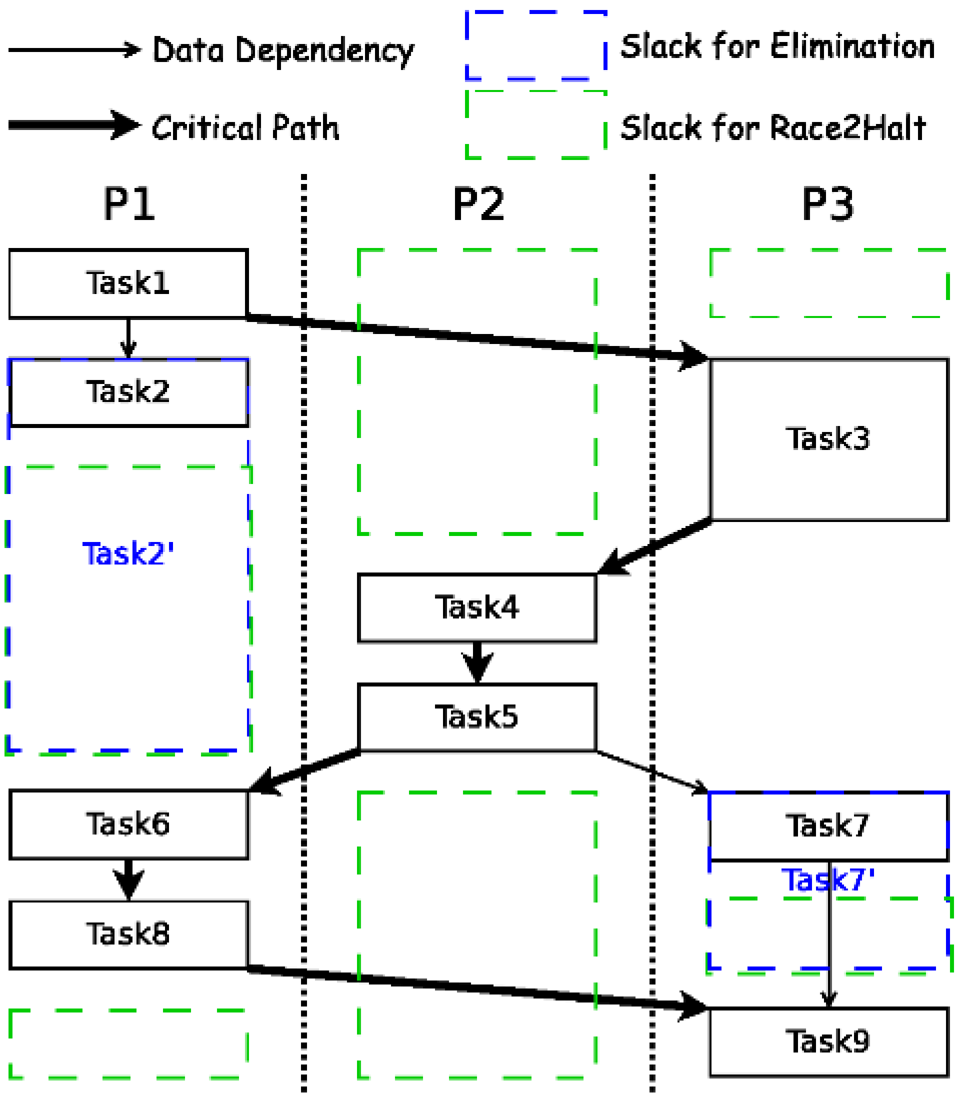
U.S. DEPARTMENT OF ENERGY

Power and Energy Concerns in HPC

- Power and energy costs of high performance computing systems are a growing severity nowadays → *operating costs* and *system reliability*
  - AvgPwr of top 5 supercomputers (TOP500)→10.1MW
  - 20MW *power-wall* by DOE for exascale (10^{18} FLOPS)
  - Overheat problems (aging/failures) and cooling costs

- Dynamic Voltage and Frequency Scaling (DVFS)
  - CMOS-based components(CPU/GPU/mem.) *dominant*
  - Strategically switch processors to low-power states when the *peak* processor performance is *unnecessary*
  - voltage/frequency ↓ → power ↓ → energy efficiency

Two Classic Energy Saving DVFS Solutions



- ❑ *Critical Path Aware Slack Reclamation*
- ❑ *Race-to-halt/idle*

Beyond DVFS: Undervolting With Fixed Frequency

Pacific Northwest NATIONAL LABORATORY

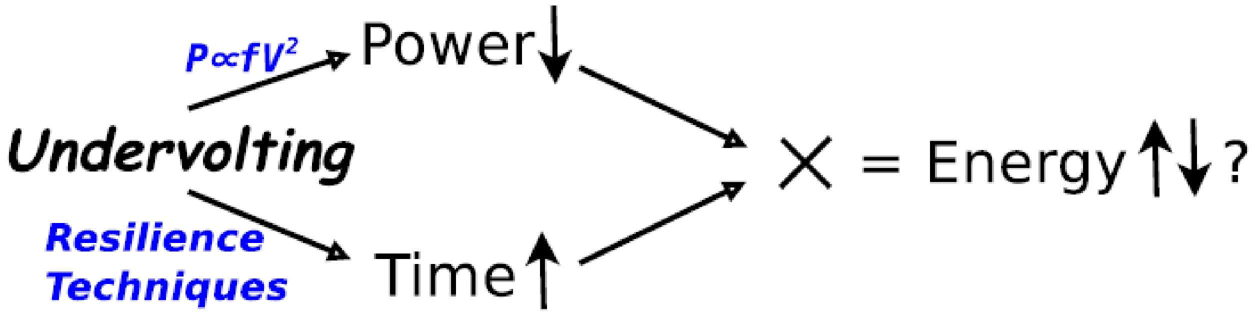
Proudly Operated by **Battelle** *Since 1965*

- Basics of the employed techniques
  - Dynamic power consumption of these components
  - Supply voltage has a positive correlation with (not $P \propto fV^2$ proportional/linear to) dynamic power

- Limitations of Existing Solutions
  - Most DVFS techniques are *frequency-directed*
  - *Undervolting*: For a given frequency, hardware can be supplied w/ a voltage lower than the original *paired* one
    - Original part of the throughput can be preserved due to *fixed* frequency
    - *Uniformly* applied to both *slack* and *non-slack* of HPC runs (using the same DVFS techniques to find appropriate frequencies for each time interval, but with further reduced voltage).
    - Drawbacks: may cause increasing error rates

- Can lowering Vdd with fixed nominal frequency be applied to HPC runtime? Are energy savings going to be offset by software-level fault-tolerance overhead? Any theoretical condition that needs to meet?

Contributions

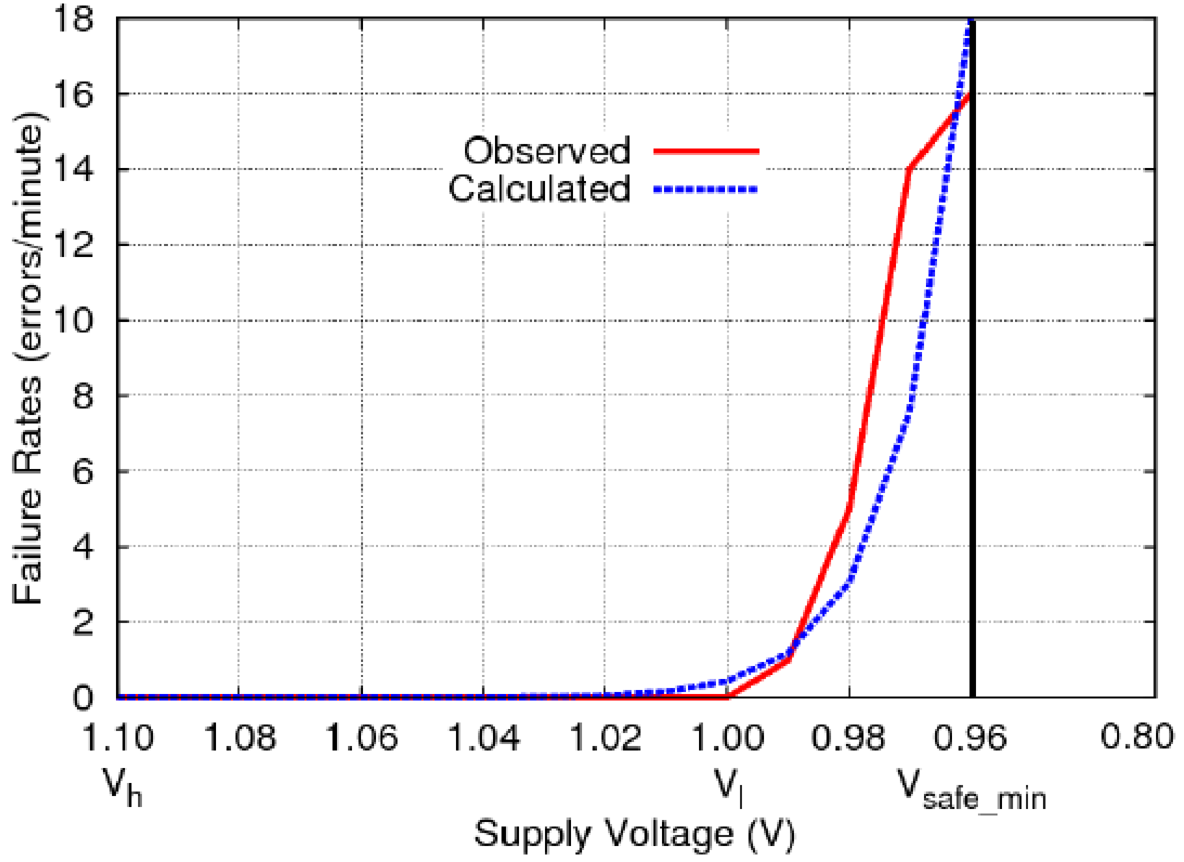


- Key Contributions
  - We observe that energy saving could be achieved using undervolting by leveraging appropriate mainstream resilience techniques
  - No requirements of pre-production machines and no modifications to the hardware
  - Modeling performance and energy under undervolting analytically
  - Up to 12.1% energy savings against baseline and 9.1% more energy saved than a state-of-the-art DVFS technique (Adagio).

Estimating Failure Rate According to V_{dd}

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by **Battelle** *Since 1965*



➢ Assumption based on [Alameldeen, ISCA'11][Zhu, ICCAD'04][Bacha, ISCA'13]
Failures of combinational logic circuits follow a Poisson distribution, determined by frequency and voltage:

$$\boxed{\lambda(f, V_{dd}) = \lambda(V_{dd}) = \lambda_0 \; e^{\frac{d(f_{max} - \beta(V_{dd} - 2V_{th} + \frac{V_{th}^2}{V_{dd}}))}{f_{max} - f_{min}}}}$$

Main-stream Software-level Fault Tolerance in HPC

- Resilience Techniques
  - Disk-Based Checkpoint/Restart (DBCR)
    - Checkpoints saved in disk, high I/O overhead
  - Diskless Checkpointing (DC)
    - Checkpoints saved in memory, trade-off (mem. + generality)
  - Triple Modular Redundancy (TMR)
    - Detect and correct one erroneous run within three runs
  - Algorithm-Based Fault Tolerance (ABFT)
    - Leverage algorithmic characteristics to correct errors online

Fault Tolerance in HPC (Cont.)

- Examples (CR and ABFT only)

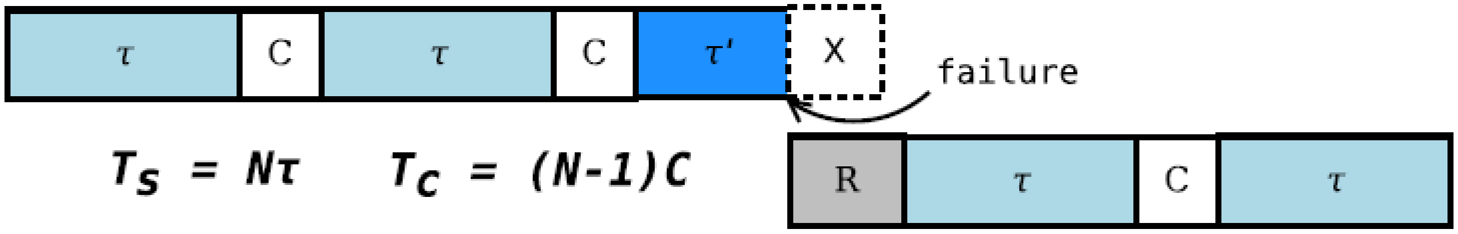


Figure 3. Checkpoint/Restart Execution Model for a Single Process.

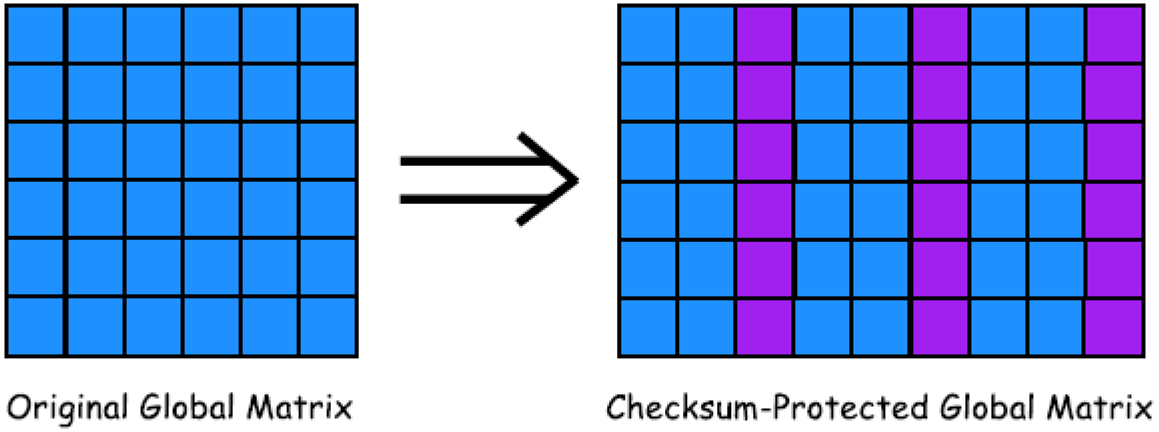


Figure 4. Algorithm-Based Fault Tolerance Model for Matrix Operations.

Performance Modeling

- Checkpoint/Restart (CR) for General Applications
  - Given a failure rate, there exists an *optimal* checkpoint interval that *minimizes* the total CR overhead
    - At *nominal* voltage, $\lambda(V_{dd})$ is small (close to zero)

$$\tau_{opt} = \sqrt{2C(\tfrac{1}{\lambda} + R)} \qquad \text{for } \tau + C \ll \tfrac{1}{\lambda}$$

    - At *further reduced* voltage, $\lambda(V_{dd})$ is raised significantly

$$\tau_{opt} = \begin{cases} \sqrt{\frac{2C}{\lambda}} - C & \text{for } C < \frac{1}{2\lambda} \\ \frac{1}{\lambda} & \text{for } C \geq \frac{1}{2\lambda} \end{cases}$$

  - Performance breakdown:

$$T_{cr} = T_s + (\frac{T_s}{\tau} - 1)C + \phi(\tau + C)n + Rn$$

Performance Modeling (Cont.)

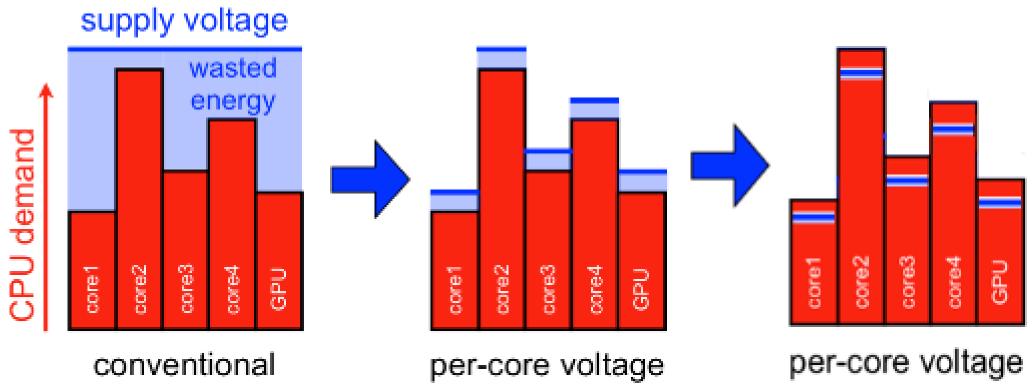
- Algorithm-Based Fault Tolerance (ABFT) for Matrix Operations (Cholesky/LU/QR factorization)
  - In CR, checkpoints are periodically *saved*
  - While in ABFT, checksums are periodically *updated*
    - Interval of updating checksums is *fixed* and not affected by the variation of failure rates → more cost-efficient
  - Performance breakdown (for example, ABFT-enabled dense matrix factorizations--Cholesky factorization):

$$T_{abft} = \frac{\mu C_f \mathbb{N}^3}{P} t_f + \frac{\mu C_v \mathbb{N}^2}{\sqrt{P}} t_v + \frac{C_m \mathbb{N}}{nb} t_m + T_d + T_l + T_c$$

- Performance modeling for other resilience techniques is conceptually similar

Energy Savings over State-of-the-art (Adagio)

- Frequency-directed DVFS Approaches
  - Processors equipped with a range of frequencies
  - Predict and apply *appropriate* freq./volt. during slack
    - Accurate workload prediction, frequency approximation, etc.
    - Major Related work include Adagio and CPU-miser
  - Can we further save energy beyond DVFS?
    - Employ a state-of-the-art DVFS technique Adagio
    - Continue undervolting further per selected appropriate F/V
    - Also leverage resilience solutions to guarantee correctness, which costs additional overhead



Energy Savings over Adagio (Cont.)

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by **Battelle** *Since 1965*

- Our Strategy
  - Use the frequency Adagio predicted for eliminating slack and further lower the voltage paired with it
  - Theoretical energy savings over baseline runs

$$\begin{aligned}\Delta E &= E_{base} - E_{uv} \\ &= (P_h - P_m)T_s \oplus (P_h - P_{uv}^{slack})T_{slack} - \\ &\quad \left(P_m \phi \tau n + P_l \left(\left(\frac{T_s - \tau}{\tau} + \phi n\right) C + Rn\right)\right)\end{aligned}$$

Example: Energy Saving Conditions over Baseline

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

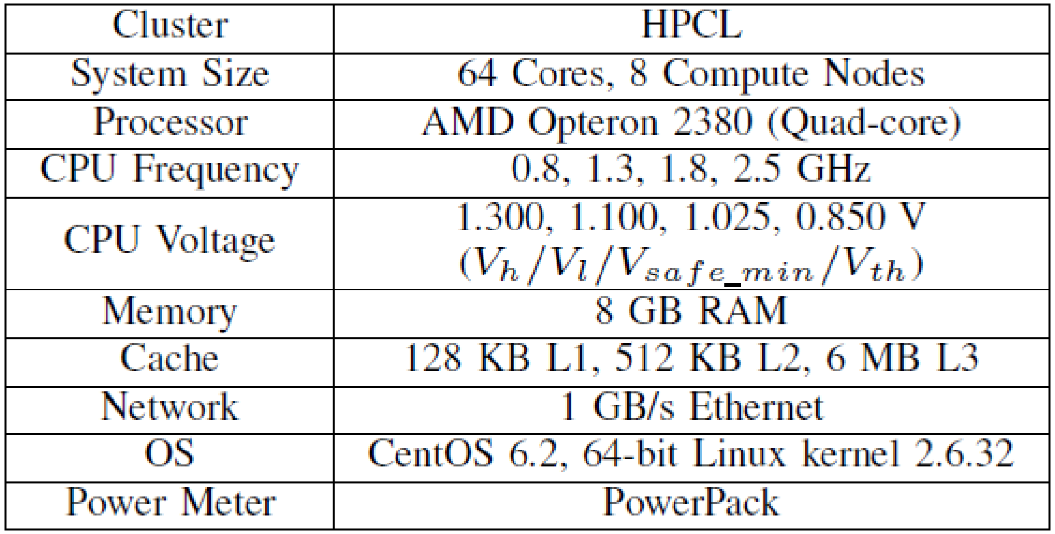
- Given Platform-dependent Parameters (c_1, c_2, c_3, AC', I_{sub}, f, V, P_C)
  - Before Model Relaxation

$$T_s > \frac{c_3\left(\left(\frac{\lambda C}{\sqrt{2\lambda C}-\lambda C}-C\right)\right)}{c_1-c_2\left(\sqrt{2\lambda C}-\lambda C\right)-c_3\left(\frac{1}{2}C+R\right)\lambda}$$

  - After Model Relaxation (N-$1 \approx N$)

$$R > \left(\frac{1}{\sqrt{2\lambda C}-\lambda C}+\frac{1}{2}\right)C+\frac{c_2}{c_3}\left(\sqrt{\frac{2C}{\lambda}}-C\right)-\frac{c_1}{c_3\lambda}$$

Experimental Setup



Cluster	HPCL
System Size	64 Cores, 8 Compute Nodes
Processor	AMD Opteron 2380 (Quad-core)
CPU Frequency	0.8, 1.3, 1.8, 2.5 GHz
CPU Voltage	1.300, 1.100, 1.025, 0.850 V $(V_h/V_l/V_{safe\_min}/V_{th})$
Memory	8 GB RAM
Cache	128 KB L1, 512 KB L2, 6 MB L3
Network	1 GB/s Ethernet
OS	CentOS 6.2, 64-bit Linux kernel 2.6.32
Power Meter	PowerPack

Implementation (Cont.)

- Undervolting Production Processors
  - Modify the northbridge/CPU FID and VID control reg.
    - Register values are altered using *Model Specific Register*
  - This approach needs careful detection of the upper and lower bounds of supply voltage of the processor
    - Hardware-damaging issues may arise
  - Different from the undervolting approach in [ISCA'13]
    - Software/firmware control
    - Pre-production processor is required (commonly not accessible)
    - Advanced ECC memory support is required

Implementation (Cont.)

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

- NB/CPU FID/VID control register format and formula

Bits **(64 bits in total)**	***Description***
63:32, 24:23, 21:19	Reserved
32:25	Northbridge Voltage ID
22	Northbridge Divisor ID
18:16	P-state ID, Read-Write
15:9	Core Voltage ID, Read-Write
8:6	Core Divisor ID, Read-Write
5:0	Core Frequency ID, Read-Write

- frequency = 100 MHz * (CPUFid + 10hex)/(2^CPUDid)

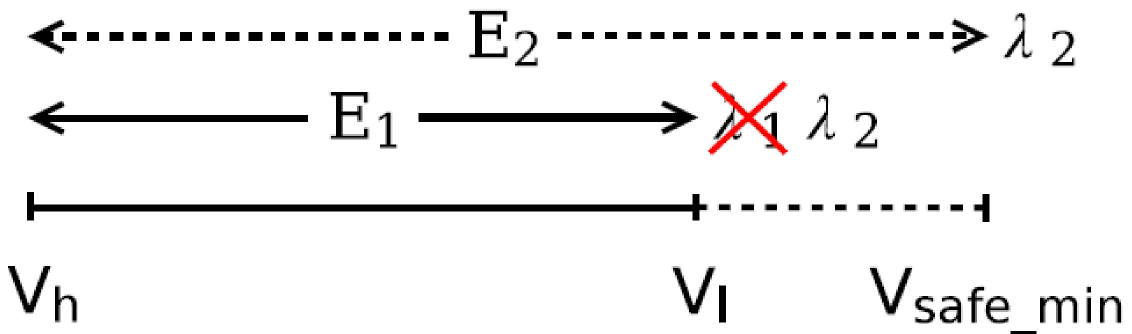
E.g.: 0x30002809 -> frequency = 100 * (9+16)/2^0 = 2.5 GHz

- voltage = 1.550 V – 0.0125 V * CPUVid

E.g.: 0x30002809 -> voltage = 1.550 - 0.0125 * 0010100h = 1.300 V

Implementation (Cont.)

- Error Injection
  - Minimum voltage we can undervolt to is V_l
    - No errors will be observed due to *close-to-zero* failure rates
  - Based on the failure rates between V_l and V_{safe_min}, we inject errors to emulate the erroneous scenarios
    - Crashes caused by soft errors: manually kill an arbitrary MPI process
    - Other soft errors (arithmetic errors and storage corruption): e.g., bit-flips of matrix elements randomly or modify assembly codes



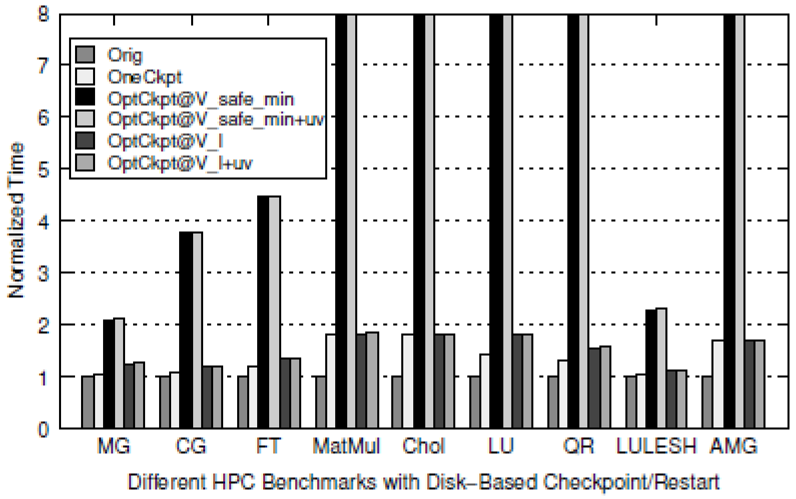
Benchmarks

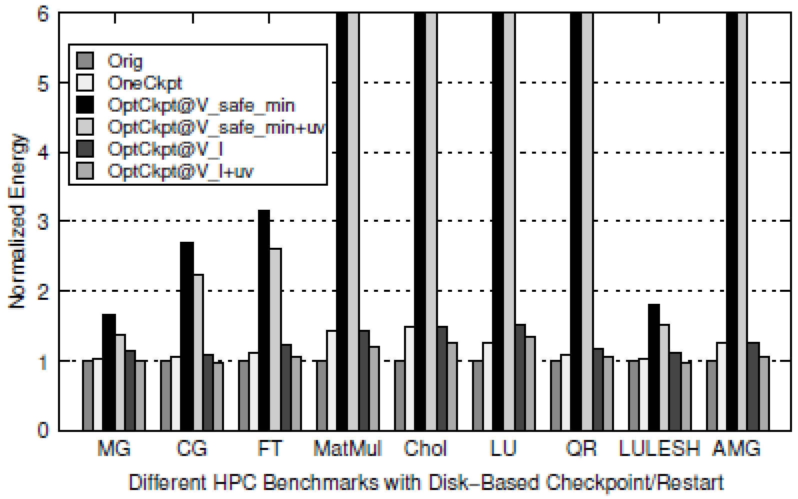
- NASA-concerned HPC Benchmarks
  - MG, CG, and FT from the NPB benchmark suite
- DOE-concerned HPC Benchmarks
  - LULESH
  - AMG
- Widely-used Numerical Linear Algebra Libraries
  - Matrix multiplication
  - Cholesky factorization
  - LU factorization
  - QR factorization

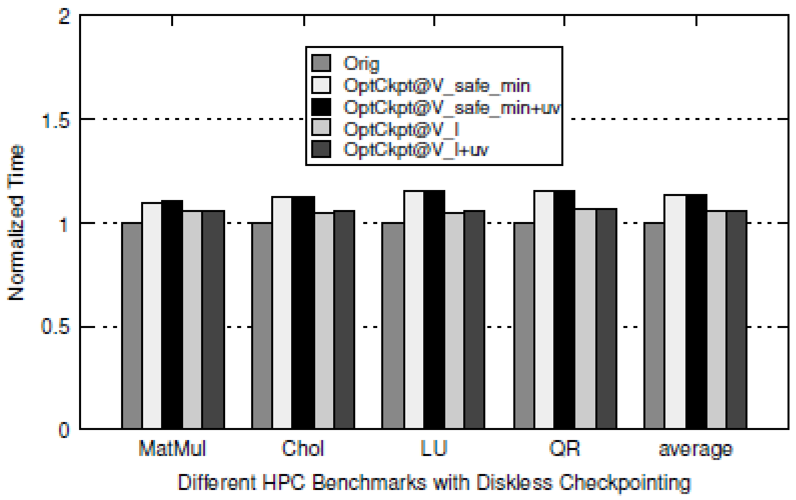
Experimental Results (DBCR vs. DC)

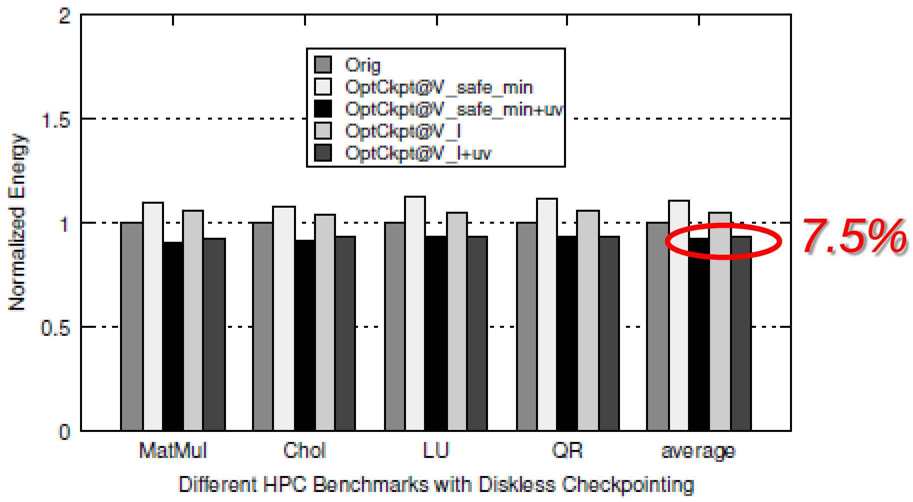
Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*



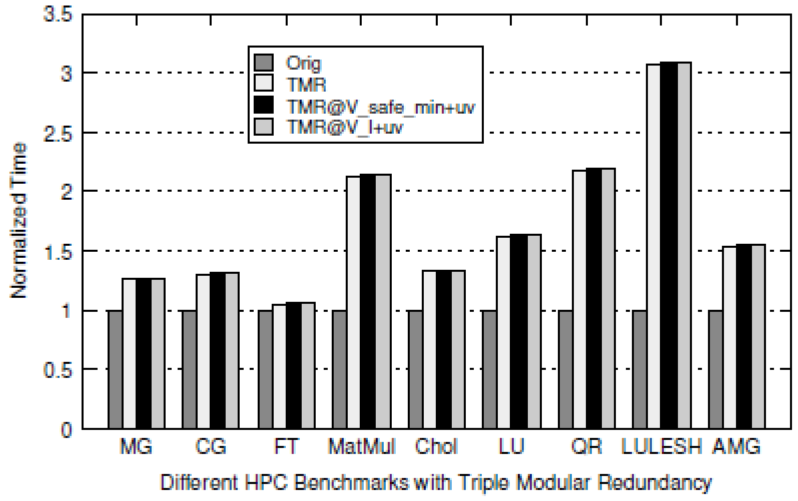


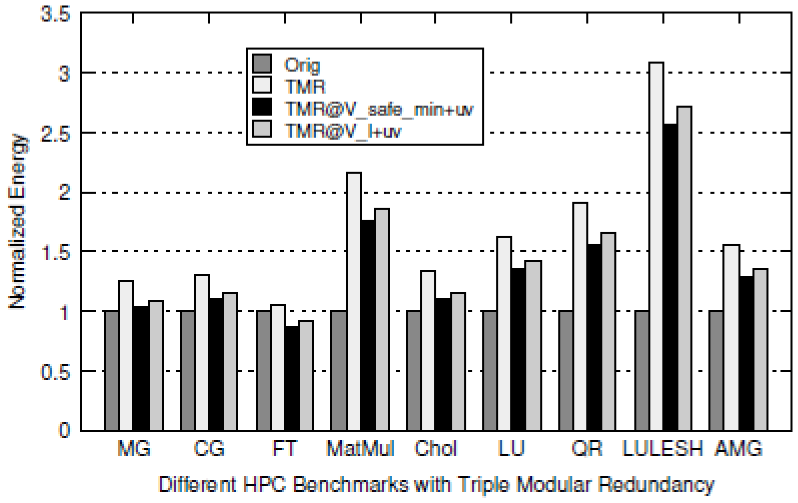


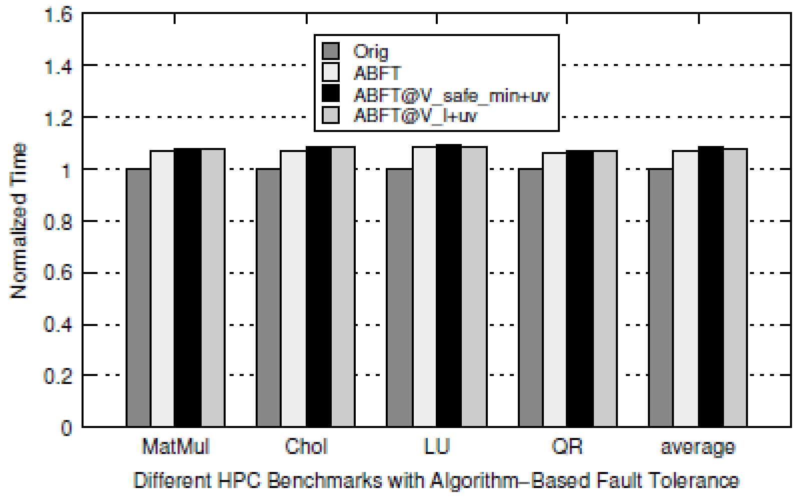


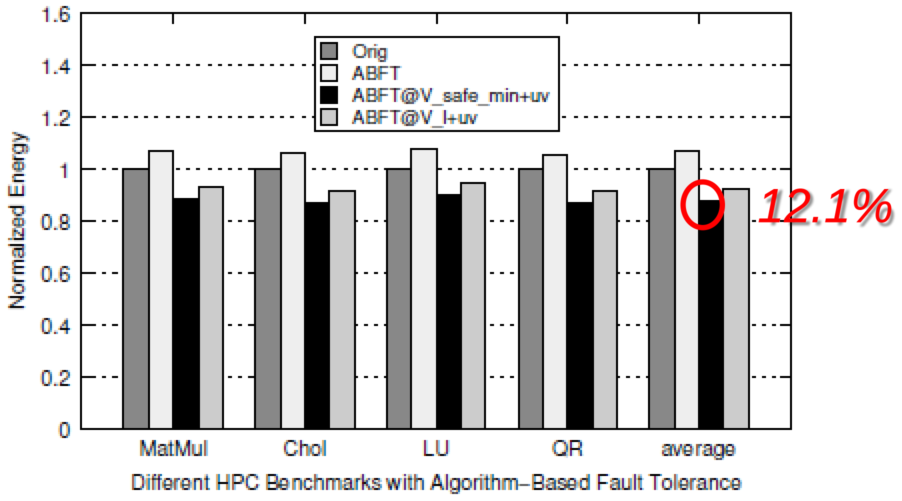
Experimental Results (TMR vs. ABFT)

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

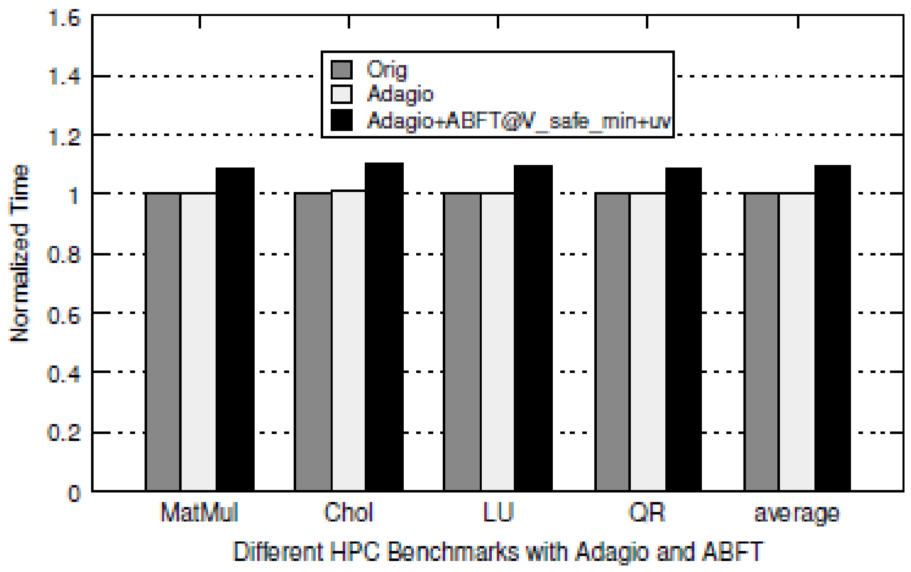


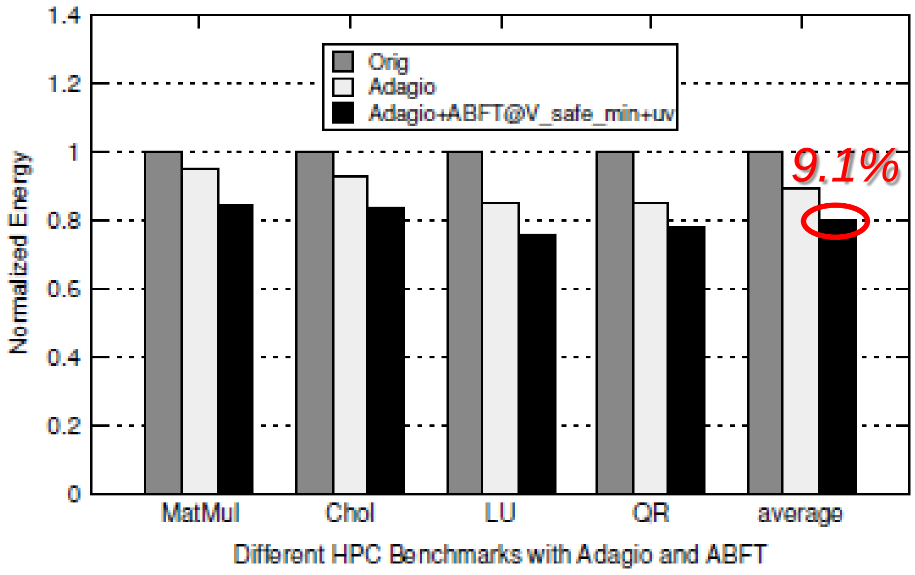






Experimental Results (Adagio + Undervolting)





If frequency switching is more frequent, cycle modulation (or duty cycle modulation) can be adopted.

Conclusions and Future Work

- Future work
  - Extending Iso-energy-efficiency model [Song, IPDPS'11] to include voltage as a parameter for scalability study at scale.
  - Focusing on designing highly-efficient software-level fault tolerance techniques, e.g., low-overhead checksum mechanisms to cover wide of iterative methods.

- Acknowledgement

DOE ASCR Beyond Standard Modeling Project (BSM)

- Refer to the following paper for details of the presentation:

"Investigating the interplay between energy efficiency and resilience in high performance computing", Tan Li, Shuaiwen Leon Song, Panruo Du, Zizhong Chen, Rong Ge, Darren Kerbyson, IPDPS'15.