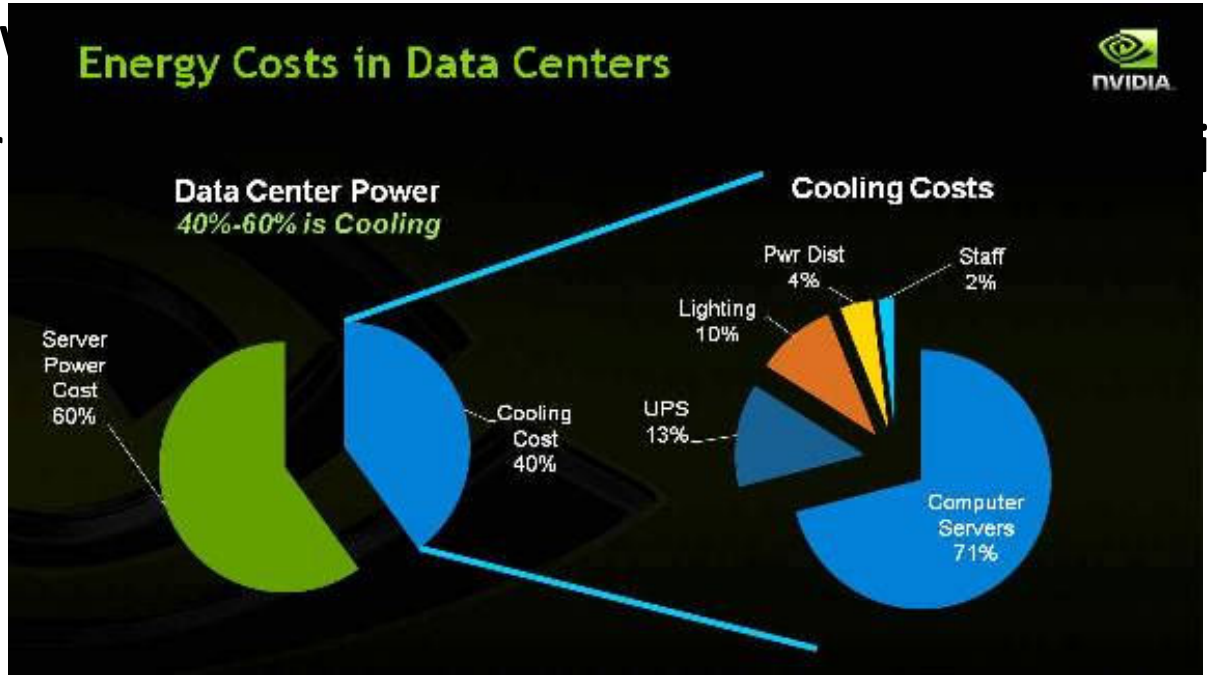
RSVP: Soft Error Resilient Power Savings at Near-Threshold Voltage using Register Vulnerability

Li Tan, Nathan DeBardeleben, Qiang Guan, Sean Blanchard, and Michael Lang

Ultrascale Systems Research Center
Los Alamos National Laboratory, USA

Power and Resilience 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
  - 20M FLOPS)
  - Over ing costs



High Vulnerability of Large-Scale HPC Systems

- Failure Rate Explosion
  - *Small on a single node, increasingly susceptible at scale*
  - Up to 1,700 *ECC memory errors* in 2 months for a 692-node (22,144 cores in total) cluster at PNNL [1]
  - K computer (1st on TOP500 in 2011): *hardware failure* rate of up to 3% + affected by 70 *soft errors* in 1 month [2]
  - A 128,000-node BlueGene/L system: 1 *soft error* in the L1 cache every 4-6 hours due to radioactive decay [3]
  - What about the forthcoming exascale systems in 2020?
- More Error-Prone Components
  - Memory bit-flips, CPU/GPU logic errors, FPGA soft errors

[1] PIC: Pacific Northwest National Laboratory Institutional Computing. https://cvs.pnl.gov/PIC/wiki/PicCompute.
[2] Keiji Yamamoto et al., The K computer Operations: Experiences and Statistics, in Proc. ICCS, 2014, pp. 576–585.
[3] Greg Bronevetsky and Bronis de Supinski, Soft error vulnerability of iterative linear algebra methods, In Proc. ICS, 2008, pp. 155-164.

Interplay: Power Efficiency and Resilience

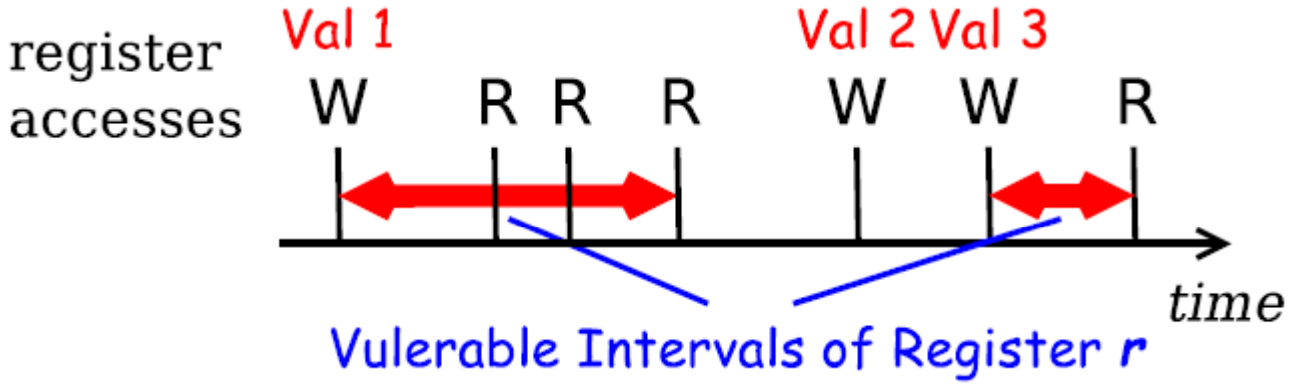
- Motivation
  - There exist *entangled effects* between the two: Improving one does not *necessarily* improve the other
  - *Goal*: The *optimal* configuration that can *balance* the trade-offs (e.g., minimizing power with resilience)
- Solutions and Challenges
  - Operate HPC runs in the *low-power* mode of hardware
  - Trade-off reliability by incurring more errors by aggressive but appropriate *voltage reduction* (e.g., NTV)

RSVP: Register Vulnerability Power Saving

- Background
  - Registers are the *most* frequently accessed component and thus susceptible to *soft* errors
  - Near-threshold voltage computing causes *more* failures
- Approach
  - Build quantitative *register vulnerability* models
  - Investigate the validity of increased failure rates at NTV by the models to *exclude invalid errors* at register level
  - Save the optimal power by ↓ voltage w/o incurring observable number of soft errors during HPC runs

Quantitative Modeling of RSVP

- Register Access Model and Vulnerability Metric



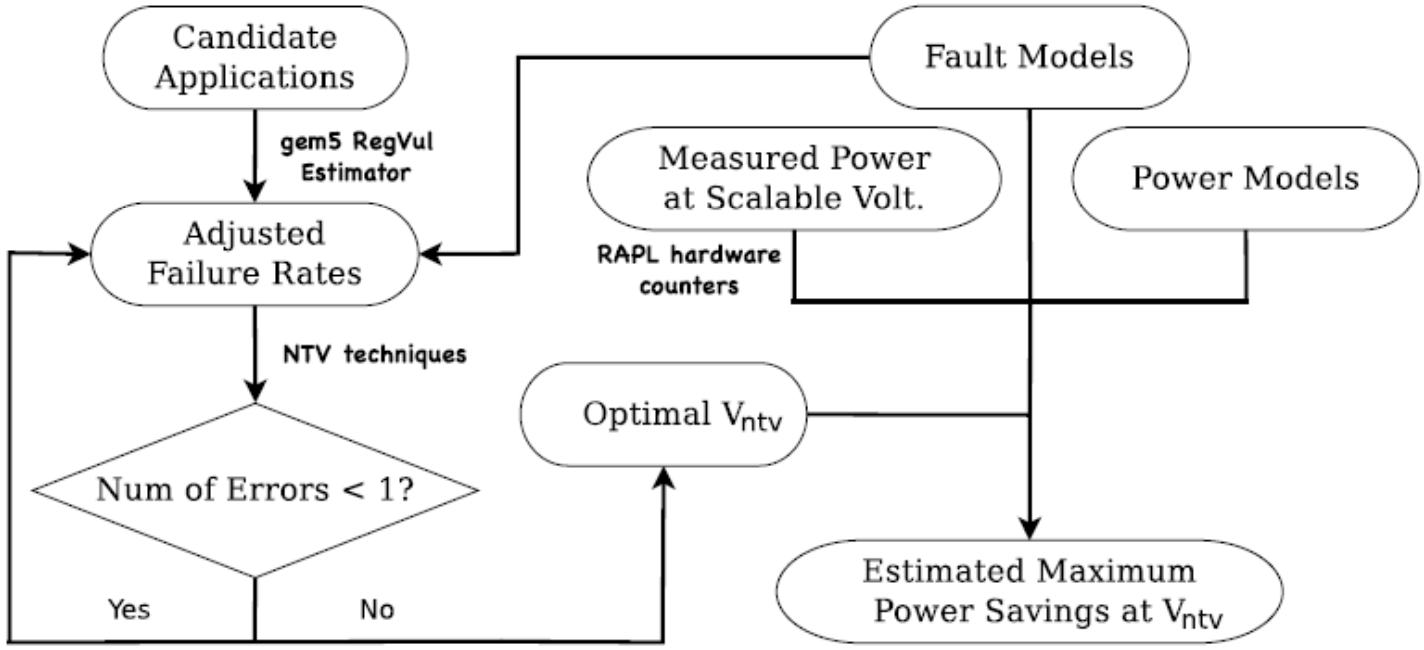
$$\phi^{ntv}_{RegVul} = \phi^{orig}_{RegVul} = \underset{i=0}{\overset{n}{\text{avg}}} \left(\frac{\sum VulTime(r_i)}{LifeTime(r_i)} \right)$$

$$\lambda(f, V_{dd})' = \lambda(f, V_{dd}) \times \phi^{ntv}_{RegVul}$$

  - W-R and R-R intervals are vulnerable to soft errors
  - Adjusted failure rate is for calculating OPT power at NTV

Quantitative Modeling of RSVP (Cont.)

- Overview



  - Dynamic profiling for power data and RVF values
  - Static estimation of OPT power savings at NTV

Quantitat ont.)

- Algorithm

Algorithm 1: Calculation of the Maximum Power Savings at the Optimal NTV Level without Incurring Observable Number of Soft Errors at Runtime

Input: A candidate application app, frequency/voltage pairs used in DVFS: $\{f_h/V_h,\ f_m/V_m,\ f_l/V_l\}$, near-threshold voltage V_{ntv} between V_l and V_{th}, and the calculated failure rate λ_{ntv} at V_{ntv}.

Output: Maximum power savings at the lowest near-threshold voltage V_{ntv}^{opt} without incurring observable number of soft errors.

```
1  begin
2      make the system idle, operating at the nominal f/V
3      for f/V ∈ {f_h/V_h, f_m/V_m, f_l/V_l} do
4          scale to the selected f/V and run app
5          measure power P = ACfV^2 + I_sub V + P_c
6          if f_h/V_h then
7              measure power P_h and execution time T
8      solve AC, I_sub, and P_c, given P_h, P_m, and P_l
9      for V ∈ (V_th, V_l) do
10         scale to the selected V and run app
11         calculate φ^ntv_RegVul and λ(f, V_dd)'
12         if ⌈GetErrNum(λ(f, V_dd)', T)⌉ < 1 then
13             measure power P_ntv
14             V^opt_ntv = V
           else
15             /* Errors arise (may need fault injection) */
16     P_sav ← (P_h − P_ntv)/P_h × 100%
17     return V^opt_ntv
```

Evaluation

- Experimental Setup

TABLE I. HARDWARE CONFIGURATION FOR ALL EXPERIMENTS.

System Size	40 logical cores
Processor	Intel Xeon E5-2660 (10-core)
CPU Frequency	1.2 to 2.6 GHz incremented by 0.1 GHz
CPU Voltage	1.05, 0.65, 0.49, 0.40 V ($V_h/V_l/V_{safe\_min}/V_{th}$)
Memory	128 GB RAM
Cache	640 KB L1, 2560 KB L2, 25600 KB L3
OS	Ubuntu 14.10, 64-bit Linux kernel 3.16.0
Power Meter	Intel RAPL

TABLE II. SIMULATOR CONFIGURATION FOR ALL EXPERIMENTS.

Operation Mode	System Call Emulation and Full System
OS (Full System Mode)	Ubuntu 14.04, 64-bit Linux kernel 3.16.0
Memory (Full System Mode)	512 MB
Instruction Set Architecture	ARM and x86
L1 Cache Size	8 KB (data), 4 KB (instruction)
L1 Cache Associativity	2
L2 Cache Size	32 KB
L2 Cache Associativity	4
Cache Line Size	64

Evaluation (Cont.)

- Benchmarks

TABLE III. BENCHMARK DETAILS. FROM LEFT TO RIGHT: BENCHMARK NAME, BENCHMARK SUITE, BENCHMARK DESCRIPTION AND TEST CASE USED, PROBLEM DOMAIN, MAIN FUNCTION, EXECUTION TIME PERCENTAGE OF THE FUNCTION RELATIVE TO THE TOTAL, AND PARALLELIZATION SYSTEM USED.

Benchmark	Suite	Description and Test Case	Domain	Function	Runtime (in %)	Parallelized by
FT	NPB	Solve partial differential equations using fast Fourier transform, Class A.	Numerical Linear Algebra	cffts1	79.7%	OpenMP
LU	NPB	Solve partial differential equations using Lower-Upper symmetric Gauss-Seidel, Class A.	Numerical Linear Algebra	ssor	89.7%	OpenMP
CRC32	MiBench	Calculate 32-bit CRC for 10 files of total size 3.065 MB.	Coding Theory	main	100.0%	Pthreads
bitcount	MiBench	Perform bit counting functions using bit lookup table with a 75000 bit set.	Coding Theory	main	100.0%	OpenMP
patricia	MiBench	Insert/remove nodes, and search in a Patricia trie for IP addresses and netmasks.	Computation Theory	pat_search	86.9%	OpenMP
sha	MiBench	Perform the Secure Hash Algorithm with a security string file of 311824 bytes.	Cryptography	sha_stream	91.2%	OpenMP
MatMul	Self-coded	Calculate matrix multiplication on two 10k×10k global matrices, saving into a third one.	Numerical Linear Algebra	mmm	99.7%	MPI (gem5 *dist* tool)
blackscholes	PARSEC	Perform option pricing with the Black-Scholes partial differential equation.	Financial Analytics	bs_thread	82.4%	Pthreads
swaptions	PARSEC	Compute prices of a portfolio of 64 swaptions with 20000 Monte Carlo simulations.	Financial Analytics	main	100.0%	Pthreads

Evaluation (Cont.)

- Results (Failure Rates and RVF Values)

TABLE IV. ORIGINAL/ADJUSTED (BITCOUNT) FAILURE RATES AT VOLTAGE LEVELS (UNIT: VOLTAGE (V); FAILURE RATE (ERRORS/MIN.)).

Core Supply Voltage	Original Failure Rate (Calc. by Equation 3)	Adjusted Failure Rate (Calc. by Equation 4)	Observable Num. of Errors (≥ 1)?
1.05	1.33×10^{-5}	0.21×10^{-5}	No
0.95	1.62×10^{-4}	0.26×10^{-4}	No
0.85	1.77×10^{-3}	0.28×10^{-3}	No
0.75	1.70×10^{-2}	0.27×10^{-2}	No
0.65	0.14	0.02	No
0.55	1.06	0.17	No
0.49	2.79	0.45	Yes
0.45	6.72	1.07	Yes

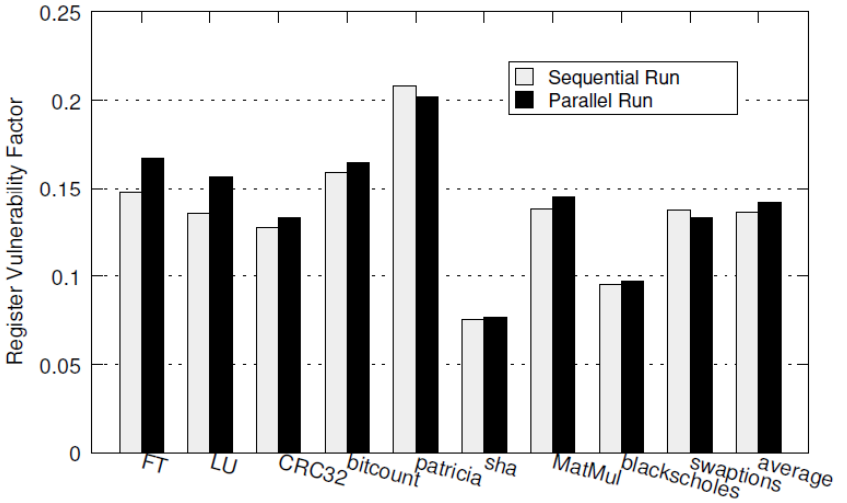


Fig. 4. Register Vulnerability Factors of Seq./Para. Runs for All Benchmarks.

- Adjusted failure rates by an example RVF value (*bitcount*)
- Data contention↑ → vulnerable register intervals↑

Evaluation (Cont.)

- Results (Power Savings and Performance Loss)

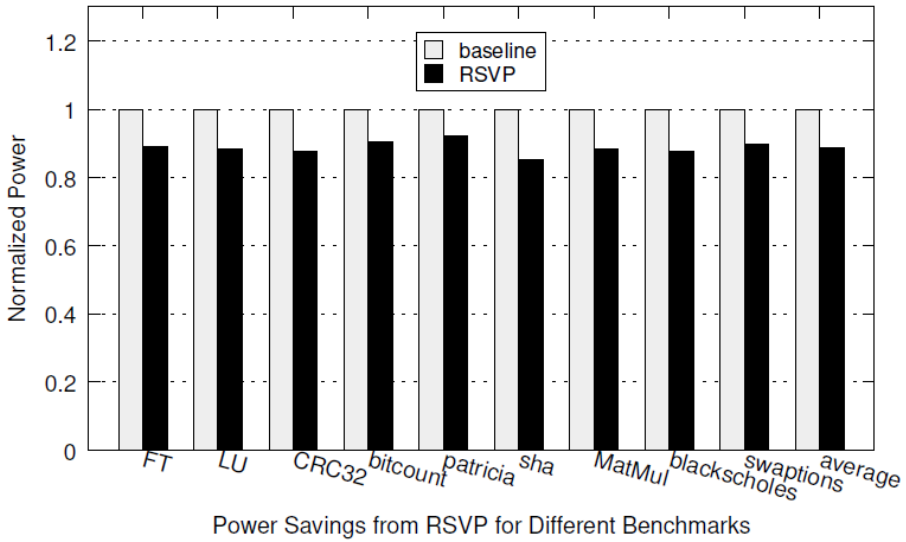


Fig. 5. Maximum Power Savings at V_{ntv}^{opt} for All Benchmarks.

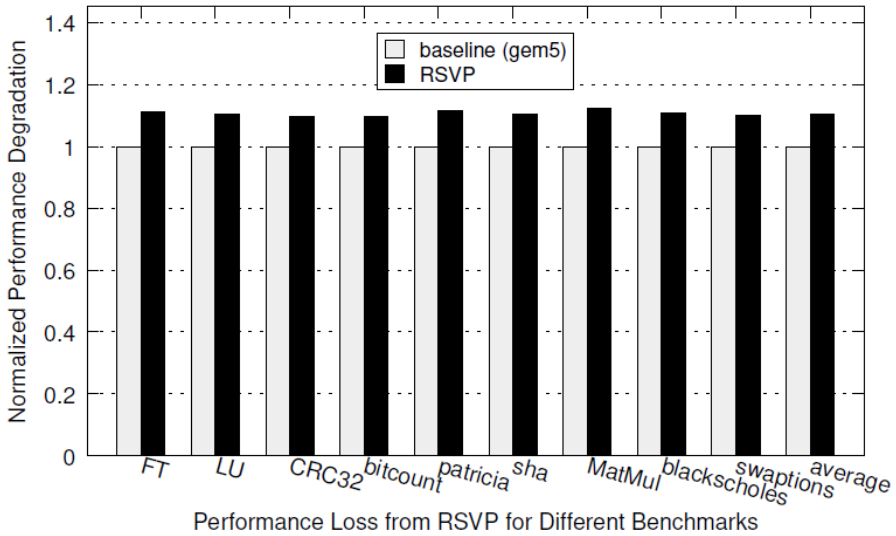


Fig. 6. Performance Loss from RSVP for All Benchmarks.

– 11.2 % system-wide power savings (RVF↓ → OPT_V↓)
– 10.6 % overhead tracking ARCH components for RVF

Conclusions

- Power Savings based on Register Vulnerability
  - Soft errors may *not necessarily* manifest in register access
  - Quantify the validity of failure rates using RVF → *adjusted* failure rates to rule out *invalid* soft errors
  - Identify the *optimal* NTV level w/o incurring observable soft errors → power savings

- Systematic Evaluation on gem5
  - A wide spectrum of HPC applications (parallelizable)
  - Accurately obtain RVF values for both sequential and parallel runs on a power-aware simulated platform