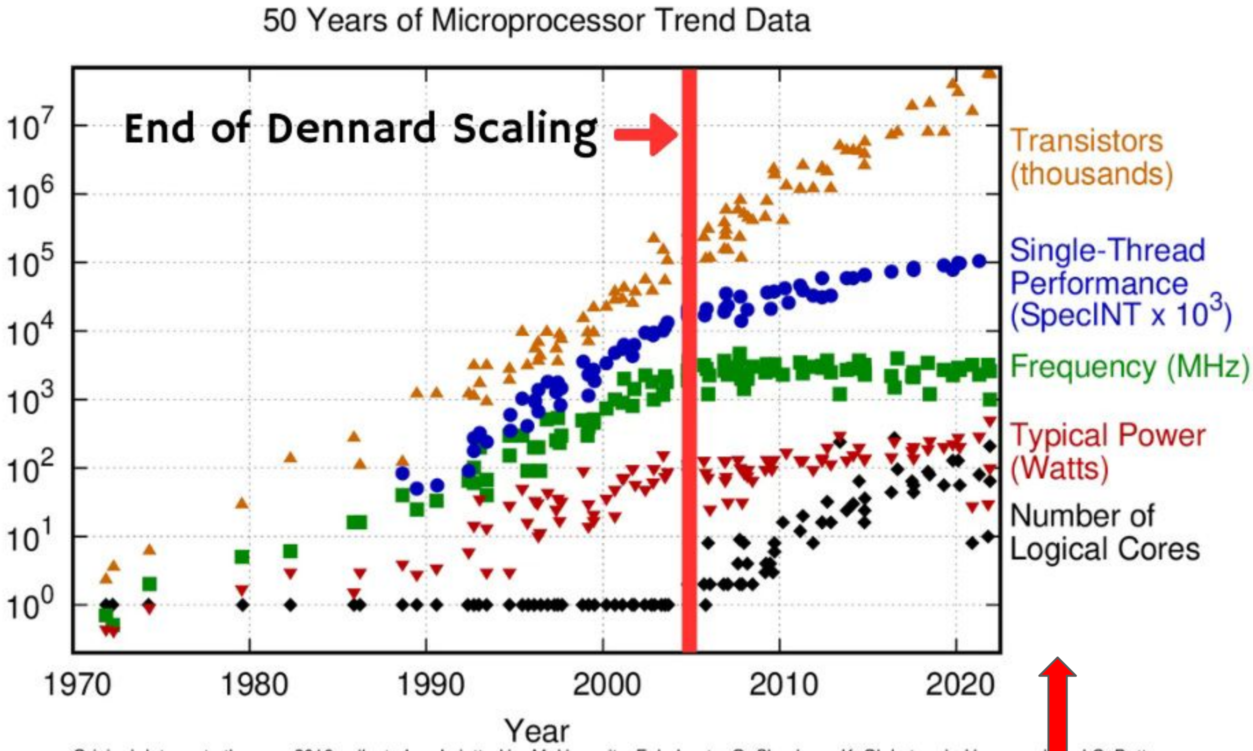
Rethinking Prefetching for Intermittent Computing

Rayyan, Liam, David



50 Years of Microprocessor Trend Data

End of Dennard Scaling

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

Author Backgrounds

Gan Fang: Fourth-year Ph.D. candidate in Computer Science at Purdue University

Jianping Zeng: Assistant Professor at Purdue, ex-Samsung researcher in memory solutions lab, Ph.D. from Purdue

Aditya Gupta: Undergraduate at Purdue, currently working at Theom

Changhee Jung: Associate Professor at Purdue, NSF CAREER Award (2018) & MICRO Hall of Fame (2021)

EHS/Low Power Systems in General

- No batteries or constant power source
    - Ex: Some wireless IoT, Solar powered, Energy-harvesting satellites, kinetic powered switches
- Finite energy in the system
    - Maximize energy harvested usage for productive work
    - Minimize wasted energy/work(even if potentially useful)
- Compute is intermittent
    - Could lose state at any point
    - Need to save current state to maintain program correctness
- Memory is non-volatile, meaning it has a much slower access time than traditional DRAM
- We can use volatile SRAM caches to make up for it

Low Capacity & Frequent Power Outages

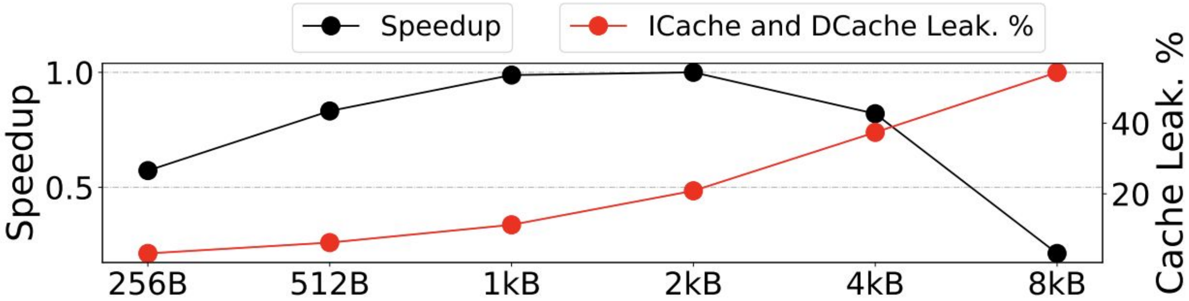


Figure 1: Speedup over baseline (2kB each for ICache/DCache) and cache leakage energy (over total energy consumption); the leakage percentage accounts for both ICache/DCache.

Larger caches have higher leakage, leading to them being less performant for these EHS systems

Why Prefetching Is So Important In EHS

- Memory is non-volatile
  - Much slower than DRAM
- Cache is smaller than is traditional, leading to more misses
- Missing in volatile cache means expensive pipeline stalls while retrieving data
- These stalls cause the processor to consume precious leakage energy waiting for non-volatile memory to return a value
- Conventional way to reduce stalls caused by cache miss is to prefetch both instructions and data

The Problem

- Prefetching is has a greater opportunity cost for EHS systems vs traditional systems
    - Smaller Caches leading to more potential to evict useful blocks
    - Unstable power which prefetchers are consuming
- Prefetched data is in volatile memory, coupled with power failures means there’s a limited usage time period
- Energy is wasted if no data is prefetched on stalling
- More Energy is wasted if data is uselessly prefetched
    - Leakage + prefetch energy

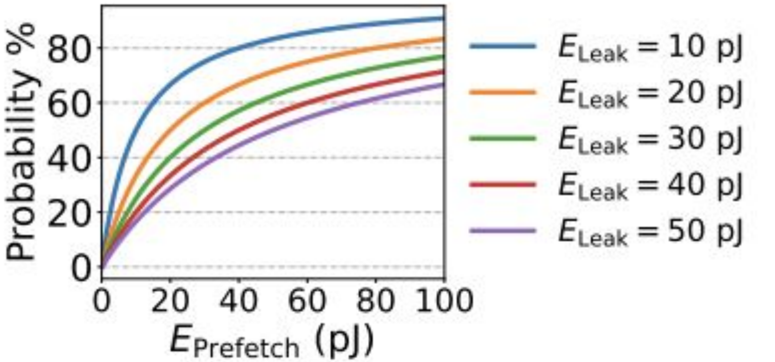
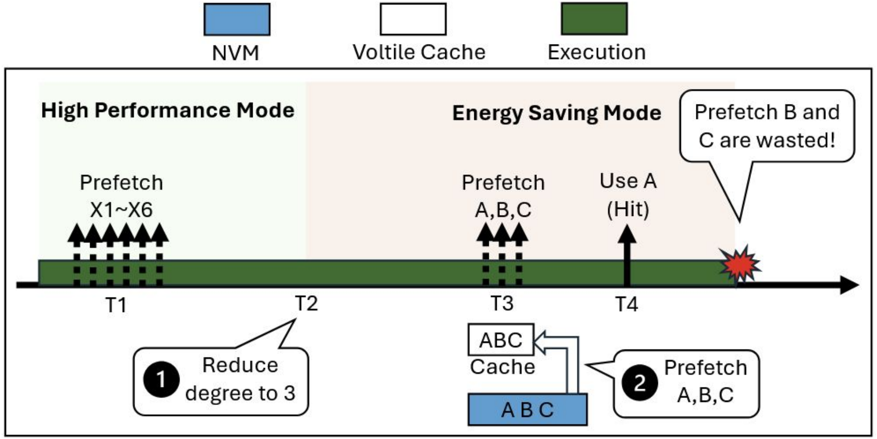


Figure 4: Relationship between minimum required P, $E_{prefetch}$, **and** E_{leak}.

The Naive Solution

- Use 1 voltage threshold to predict incoming power failure
  - Too simple to sufficiently reduce wasted energy
- Differentiate soon and very soon power failure
  - Potentially need multiple levels of degree changes at different voltage levels



IPEX(**I**ntermediate-aware **P**refetching **EX**tension) Adaptation

- IPEX sets voltage thresholds for the capacitor and adjusts prefetching degree when V_{Cap} crosses one.
- After a reboot, IPEX adjusts its thresholds.
  - High throttle rate in previous cycle -> lower threshold
  - Low throttle rate in previous cycle -> raise threshold

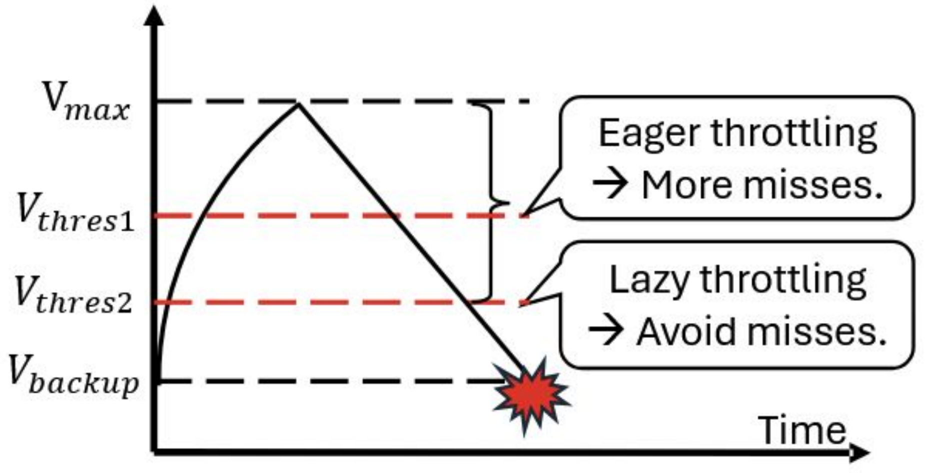
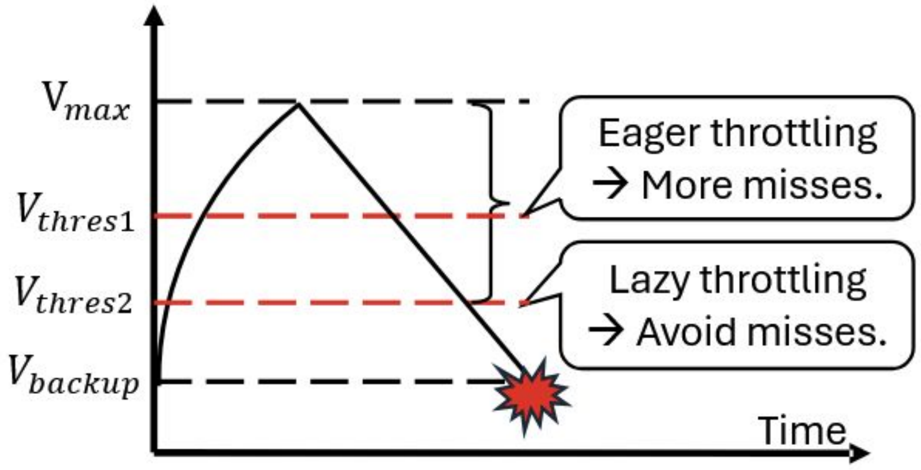

V_{max}
V_{thres1}
V_{thres2}
V_{backup}
Eager throttling → More misses.
Lazy throttling → Avoid misses.
Time

Figure 3: Varying prefetch degree upon crossing one of V thresholds.

IPEX(**I**ntermediate-aware **P**refetching **EX**tension) Adaptation

- Eager throttling
  - Missed performance opportunities
  - Increased cache misses
- Lazy throttling
  - Wasted energy on unused prefetches
- Energy conditions likely to vary



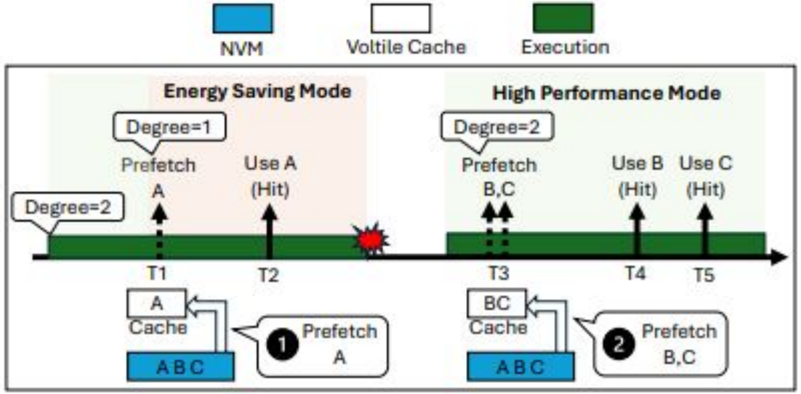
Eager throttling → More misses.

Lazy throttling → Avoid misses.

Figure 3: Varying prefetch degree upon crossing one of V thresholds.

IPEX Bi-modal Operation

- Switch between two modes based on power failure likelihood
  - Energy Saving Mode(Low Voltage)
  - High Performance Mode(High Voltage)
- Use capacitor voltage level as proxy for power failure likelihood
- Reboot to higher performance mode to avoid mode-lock



IPEX Degree Adjustment Algorithm

- Halves the prefetch degree when voltage drops below voltage threshold
- Double the prefetch degree when voltage rises above the voltage threshold
- Degree determines the number of blocks fetched from memory
- Exponential/aggressive scaling for more timely responses
- Finding the right threshold voltage:
    - Keep track of throttling rate(R_tr = throttled / total prefetches)
    - If throttling rate >= 5% -> decrease threshold by 0.05V
    - If throttling rate <= 5% -> increase threshold by 0.05V

Evaluation and Experimental Analysis

- IPEX requires 4 volatile registers per cache, resulting in a total overhead of .0018% of chip area.
- IPEX was implemented on top of GEM5 alongside other simple prefetchers
  - The paper makes the argument that more complex prefetchers would also benefit from IPEX (untested)
- IPEX outperforms the baseline by an average of 3.73% (DCache prefetcher) or 8.96% (both ICache and DCache prefetchers).
- IPEX reduces memory traffic by 2-5%.

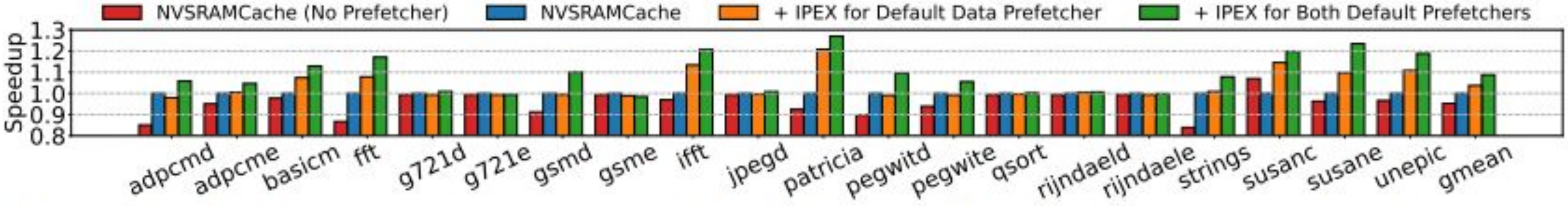


Figure 10: Normalized performance speed to NVSRAMCache (baseline) with RFHome power trace; default prefetchers are enabled for NVSRAMCache.

Performance and Energy Analysis

- IPEX reduces overall energy consumption by 3.42% (DCache only) or 7.86% (Icache and Dcache).
- Prefetch accuracy, increases by 35% for ICache and 22.8% for DCache, while having only 3% and 5% reduction in the coverage for ICache and DCache, respectively.
- Performance improvements plateau at 3 voltage thresholds.
- Performance improvements were higher on more aggressive prefetchers.
- Performance improvement is higher for smaller cache sizes.

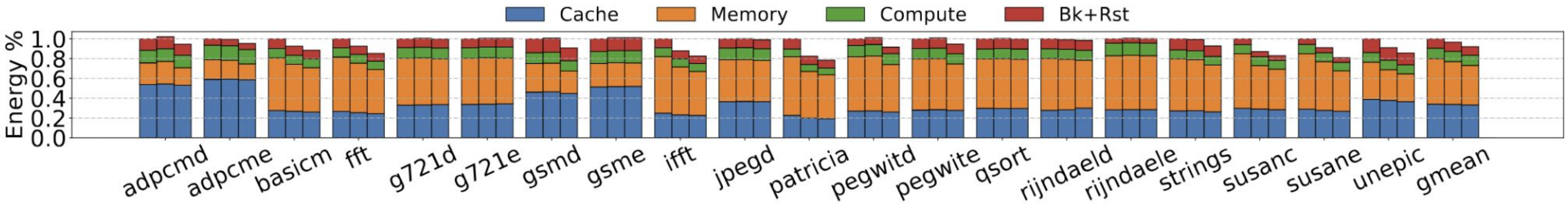


Figure 14: Normalized energy breakdown to NVSRAMCache (baseline) with RFHome power trace. There are 3 bars for each application. ***From left to right: NVSRAMCache, + IPEX for DCache prefetcher, and +IPEX for Both DCache and ICache prefetchers***

Known Limitations

- IPEX is sensitive to a lot of factors, and may not add performance
    - Cache size, capacitor size, main memory size.
- Little to no performance gain for systems with infrequent power interruptions.
    - Due to large capacitor or stable energy conditions.
    - In practice, EHS systems frequently experience power outages.
- Overaggressive throttling can increase cache misses -> performance reduction
- If memory access is a proportionally low percentage of total energy usage, IPEX offers very little improvement.

Further Questions

- What are other potential factors that inhibit IPEX?
- IPEX uses capacitor voltage as a metric to judge proximity to power failure. What are some limitations to this?
- Why is exponential scaling a good approach for adjusting the voltage threshold? Why not use linear scaling?
- Other standard computer architecture optimizations that might need changes for intermittent computing beyond prefetching?
- How could IPEX be applied to non-intermittent systems?