arm
Research

HPCA ’19, Washington DC

BRB

Mitigating Branch Predictor Side-Channels

UNIVERSITY OF
Southampton

Ilias Vougioukas
Nikos Nikoleris, Andreas Sandberg, Stephan Diestelhorst
Bashir M. Al-Hashimi, Geoff V. Merrett

19/2/2019

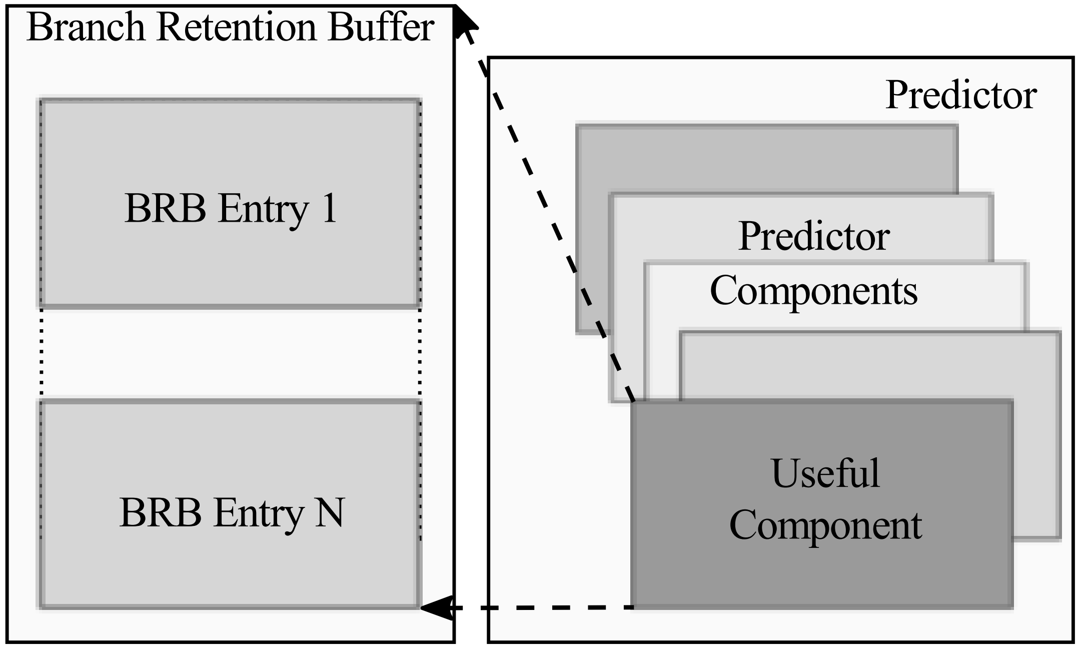
2018: The year of HW exploits

- Transient execution hardware side channel attacks.
- Uses speculative execution to leak information
- In 1 year 13 Spectre and 14 Meltdown attacks
- Effects still being assessed

How do we deal with such threats in the future consistently?

Contribution: BRB

- Flushing the BP between context switches
- Isolates contexts…. but very costly
- Proposal isolate per context : Branch Retention Buffer
- Keep minimal state, trim diminishing returns
- Recover some of the lost performance



Threat Model

Formalise the attack space with a threat model:

- Victim and attacker applications
- Both reside in the same core and share the same BP
- Victim can be slowed to detect behaviour of a branch.
- Attacker can force victim code to execute, targeting vulnerable code.
- Attacker can poison BP entries of the victim, forcing a misprediction.

Isolation can prevent the attack

Flush most BP state between switches…

…retain only useful parts per context

Security and Context Switching

Limit study

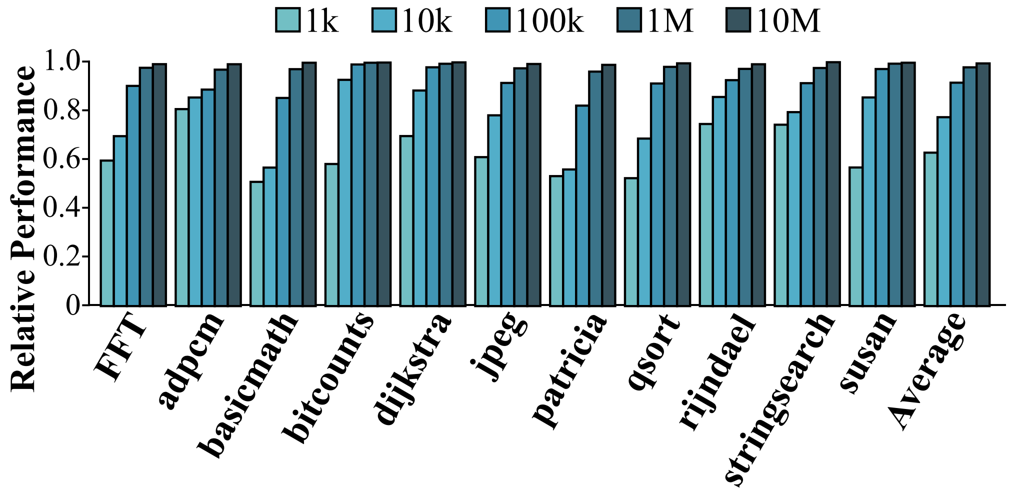
1. **Context switch happens more frequently than you think….**
   - Measured on Pixel phone
   - We measured switches as fast as 1 every 64k instructions.
   - Online streaming services have reported similar numbers

That's roughly 1 every 12k branches!

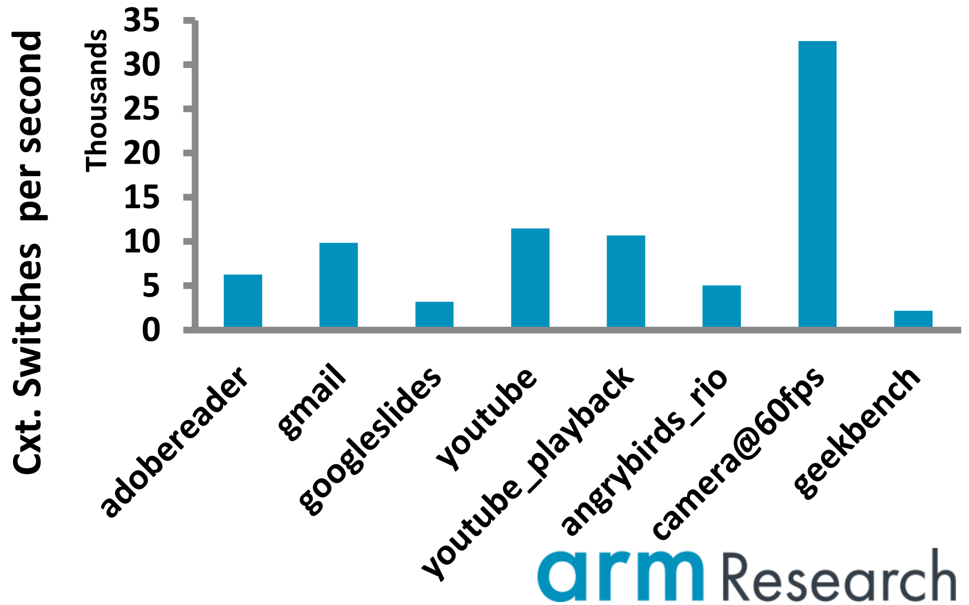
2. **Focus on Spectre type attacks**
   - Fixes not trivial
   - Software
     - Very circumstantial
     - Not always possible
     - System stability issues
   - Hardware
     - Shadow structures
     - Hard Partitioning
     - Flushing / Disabling / Tagging

But the performance drop can be as much as 30%

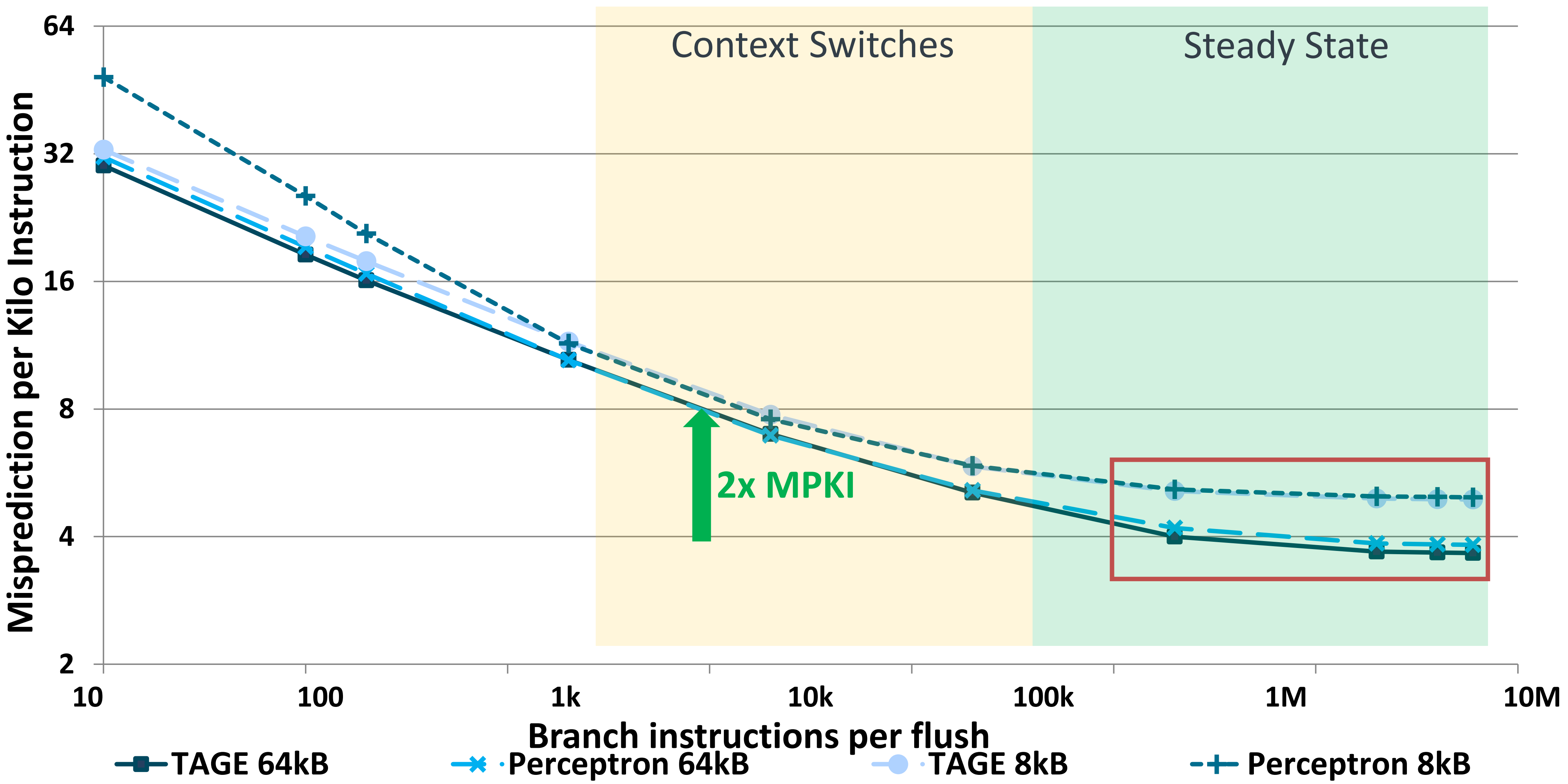
Performance Loss when Flushing the BP



Pixel phone context switches

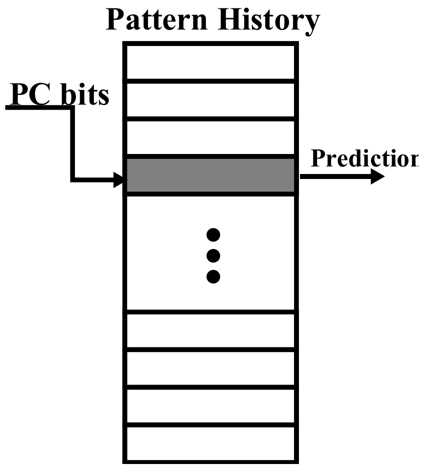


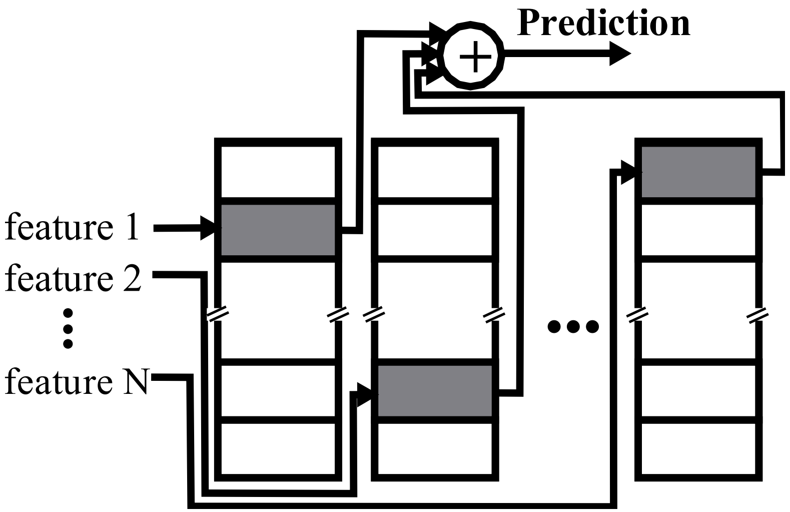
Flushing: Steady State vs Transient State



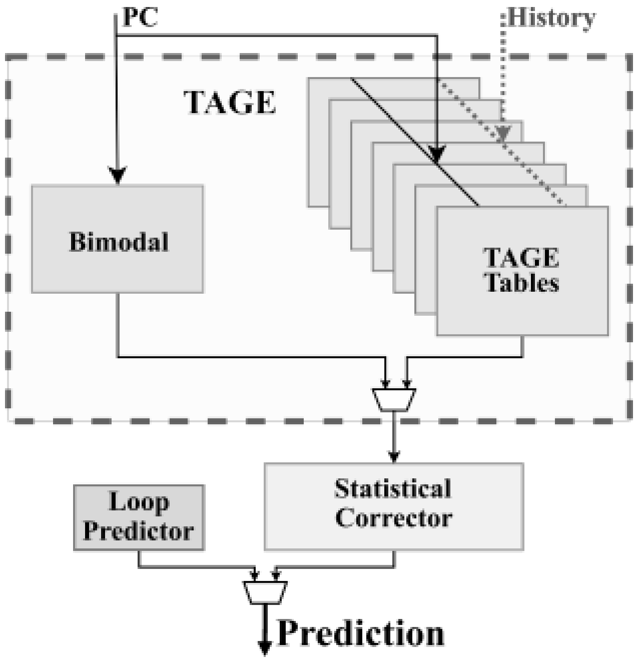
Setup

- Use Championship Branch Prediction Framework 2016 (CBP)
  - Over 250 traces: long/short, mobile/server
- Modify CBP to flush the BP design per component on demand





D. A. Jimenez, “Multiperspective Perceptron Predictor,” JWAC-4: Championship Branch Prediction, 2014.



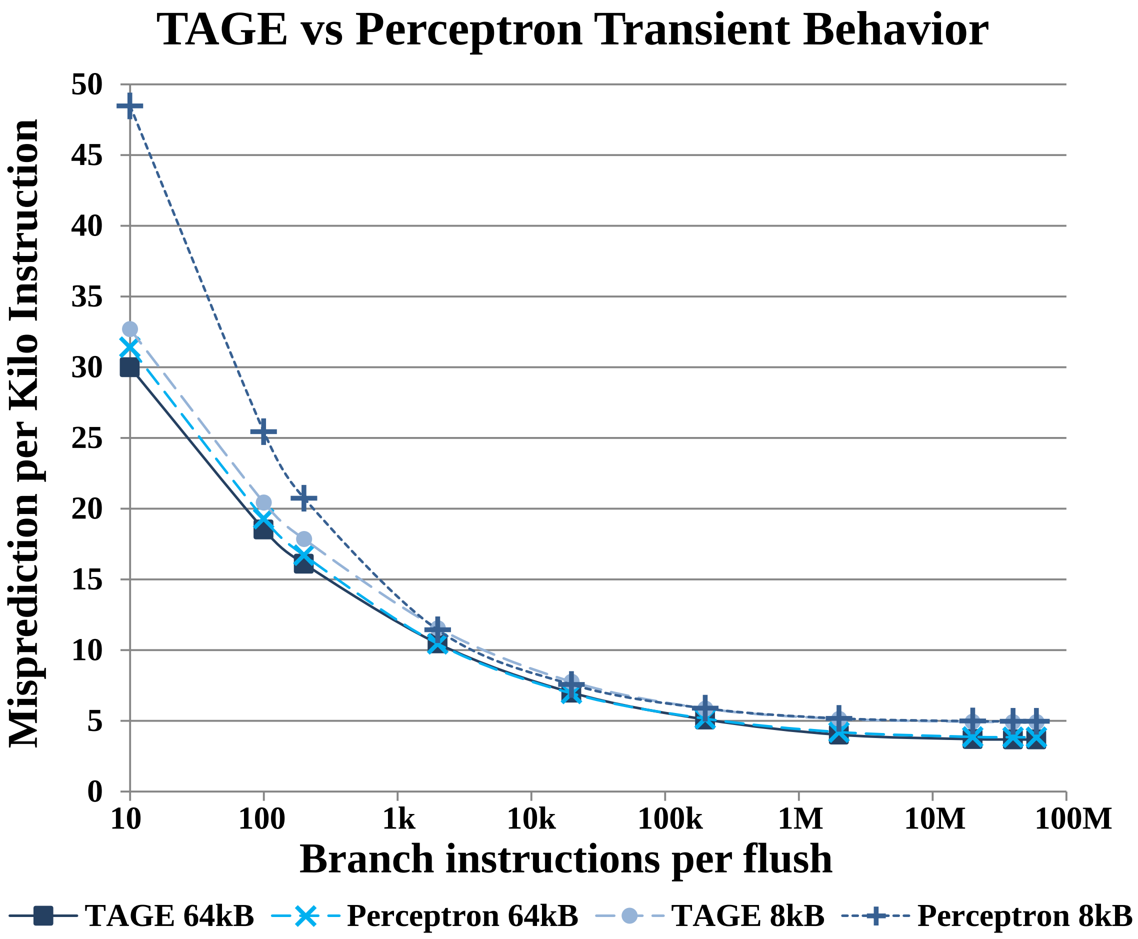
A. Seznec, “TAGE-SC-L branch predictors,” JWAC-4: Championship Branch Prediction, 2014

TAGE vs Perceptron

Transient State Analysis

First look at transient analysis

- TAGE 64 and Perceptron 64 similar accuracy (3.6 MPKI)
- Same for 8kB versions (5 MPKI)
- Perceptron 8kB low accuracy when flushing frequently

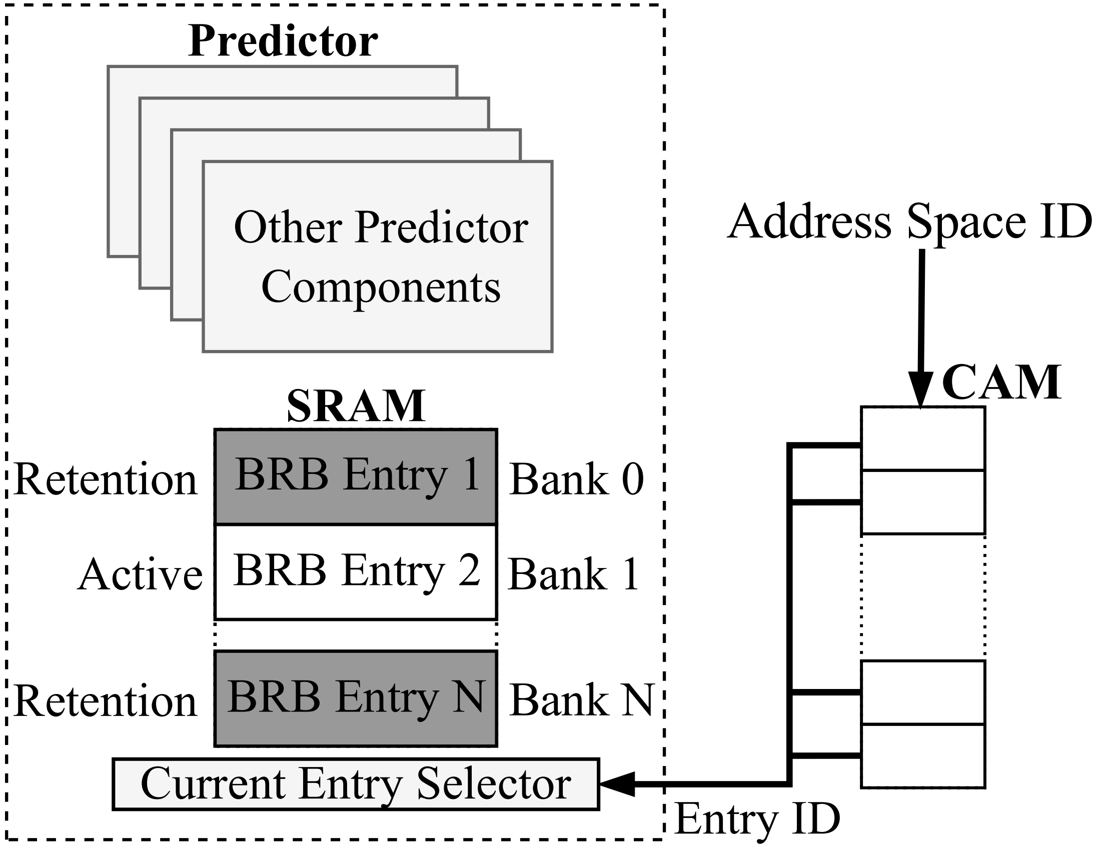


Storing (Partial) State

Branch Retention Buffer (BRB):

- Retains state per context
- Store minimal state
- Change active entry when context switching
- ASID points to active entry
- No overhead during predictions
- 2 entries for userspace, 1 for OS

Focus on TAGE



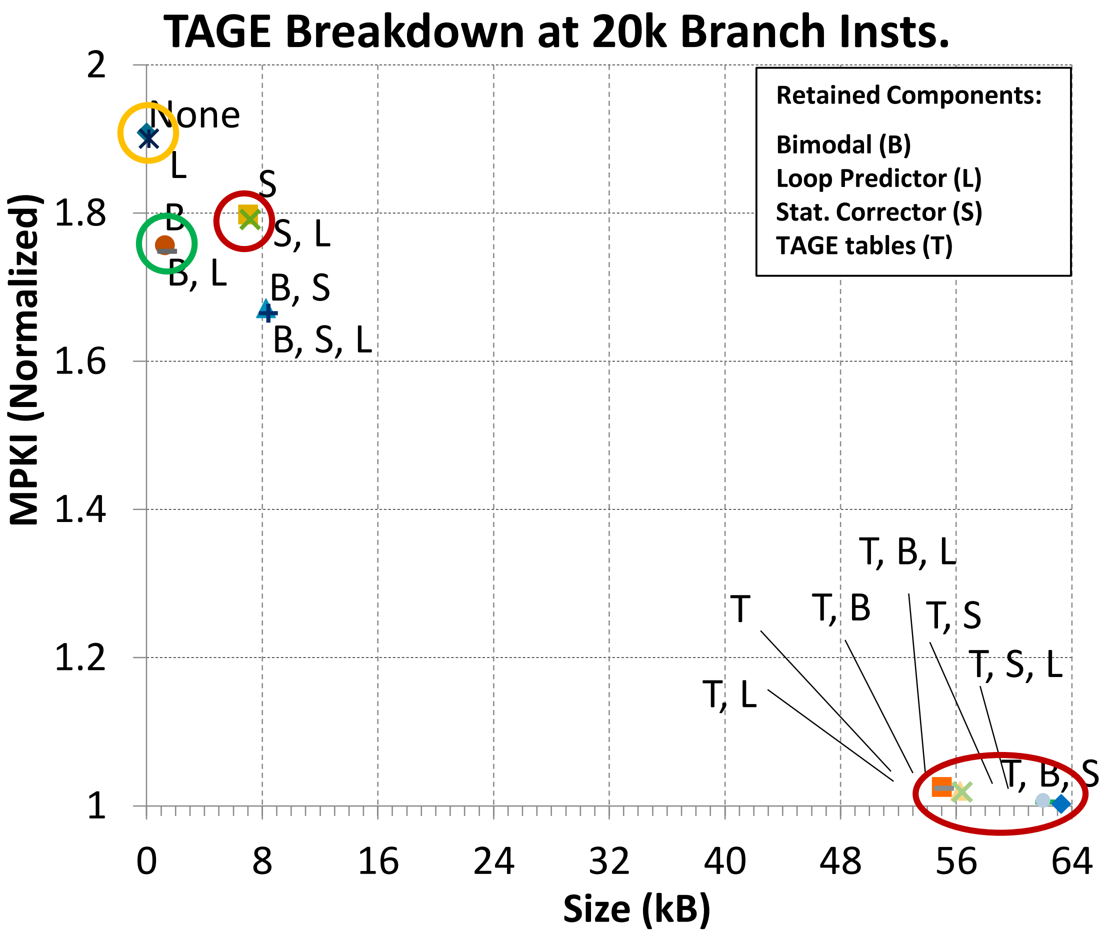
TAGE Accuracy Breakdown

Identifying the components that increase the accuracy the most

Break down how components contribute to accuracy.

- Retaining no state **doubles** the misprediction
- The TAGE tables are most of the accuracy.
- The statistical corrector is not useful for steady state or transient.
- Bimodal: best bang for buck prediction.

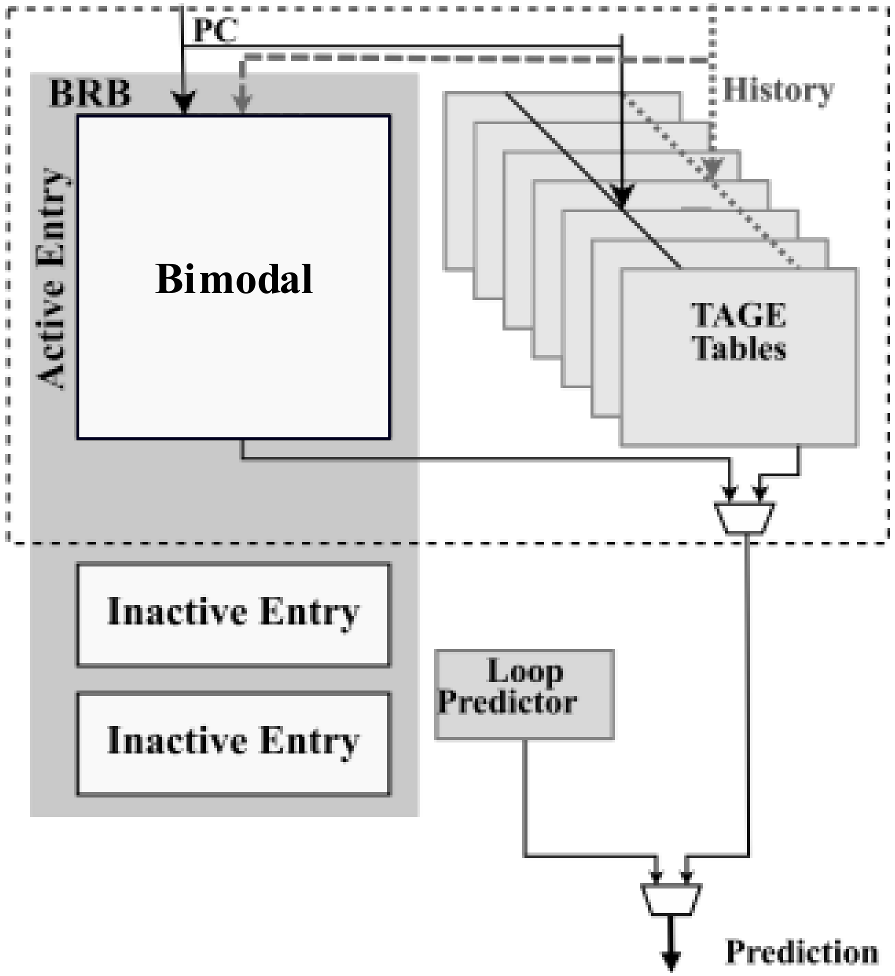
Storing the bimodal can help solve the transient accuracy drop.


TAGE Breakdown at 20k Branch Insts.

Using BRB with TAGE

Preserving partial state per context can improve transient accuracy.

- Use a Branch Retention Buffer (BRB) to store minimal state.
- Can have multiple BRB entries for multiple contexts.
- Store only the bimodal base predictor.

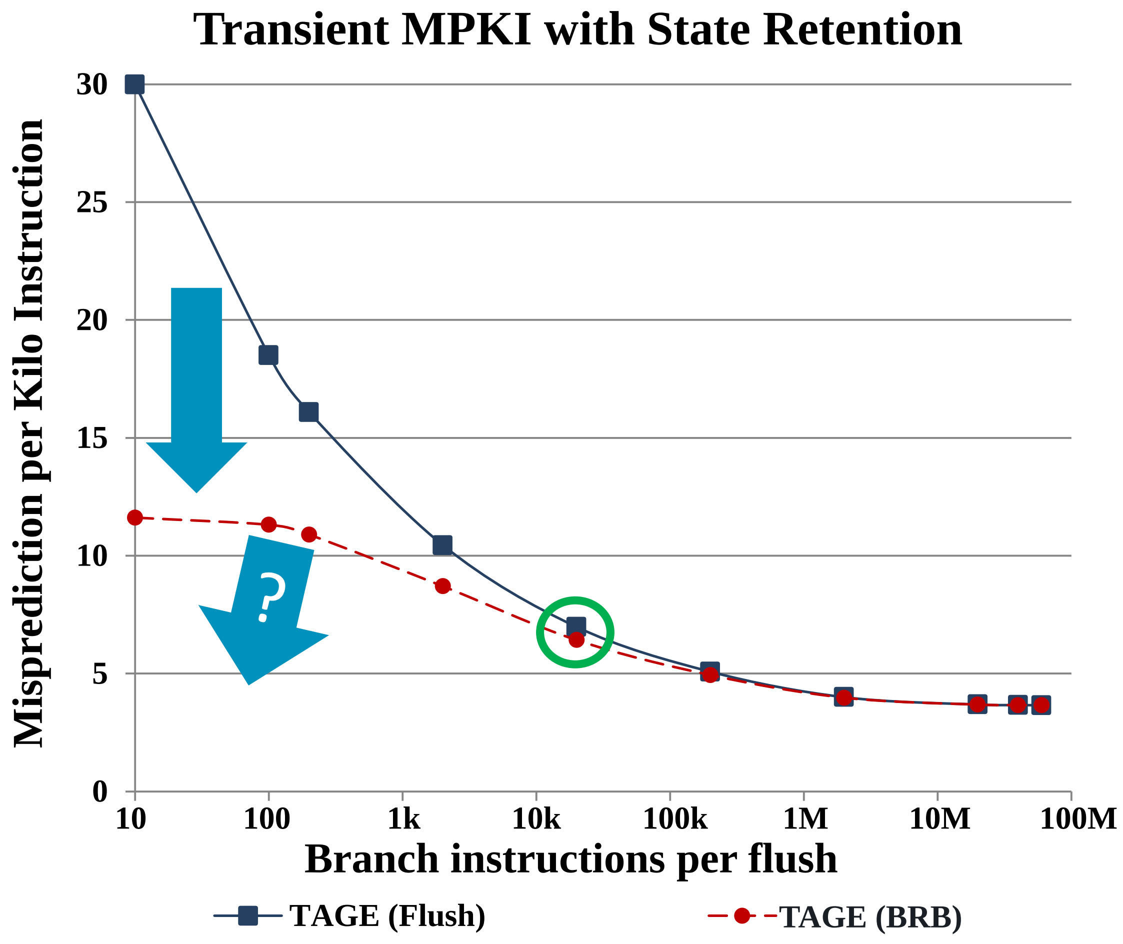


Improving the Transient state

Retaining the bimodal in TAGE

- Bimodal retention improves transient accuracy
- Small benefits at the 12k branch mark
- Transient misprediction could improve

Need to get better accuracy from Bimodal



Comparing the Bases

Perceptron vs bimodal

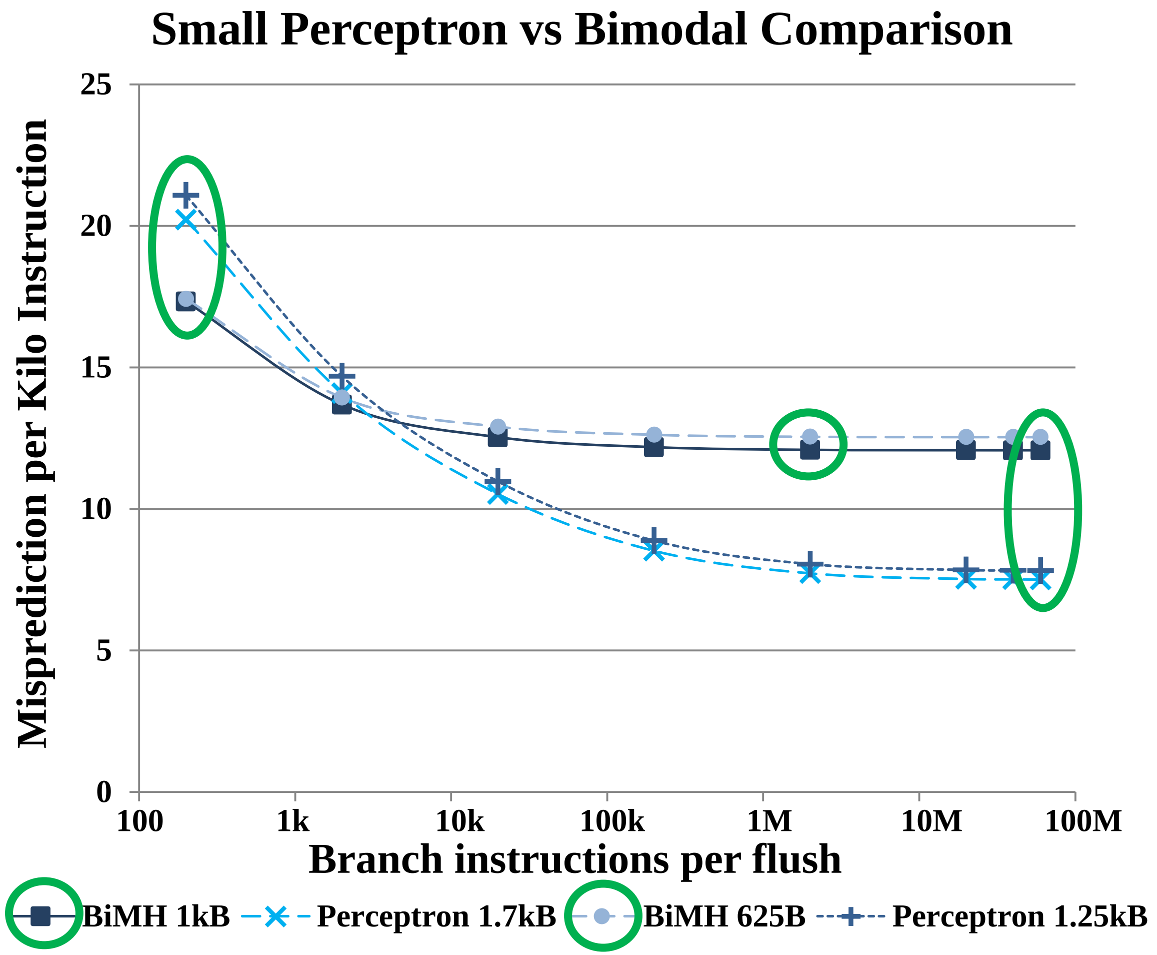
Interesting things in small sizes…

- Bimodal maximum accuracy 11MPKI
- Bimodal accuracy not affected by size

Baseline accuracy for TAGE (BRB)

- Perceptron has worse transient accuracy...
- But much better steady state predictions!

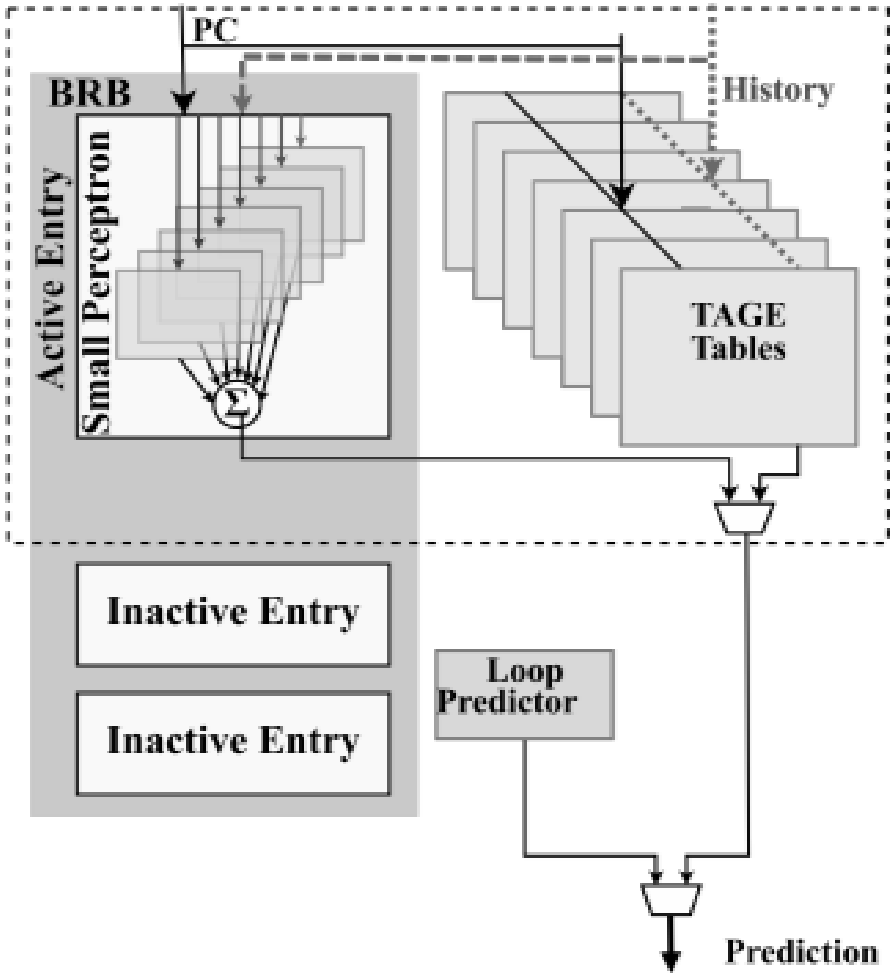
Retention only cares about steady state!



ParTAGE

Swapping the Bimodal for a Perceptron

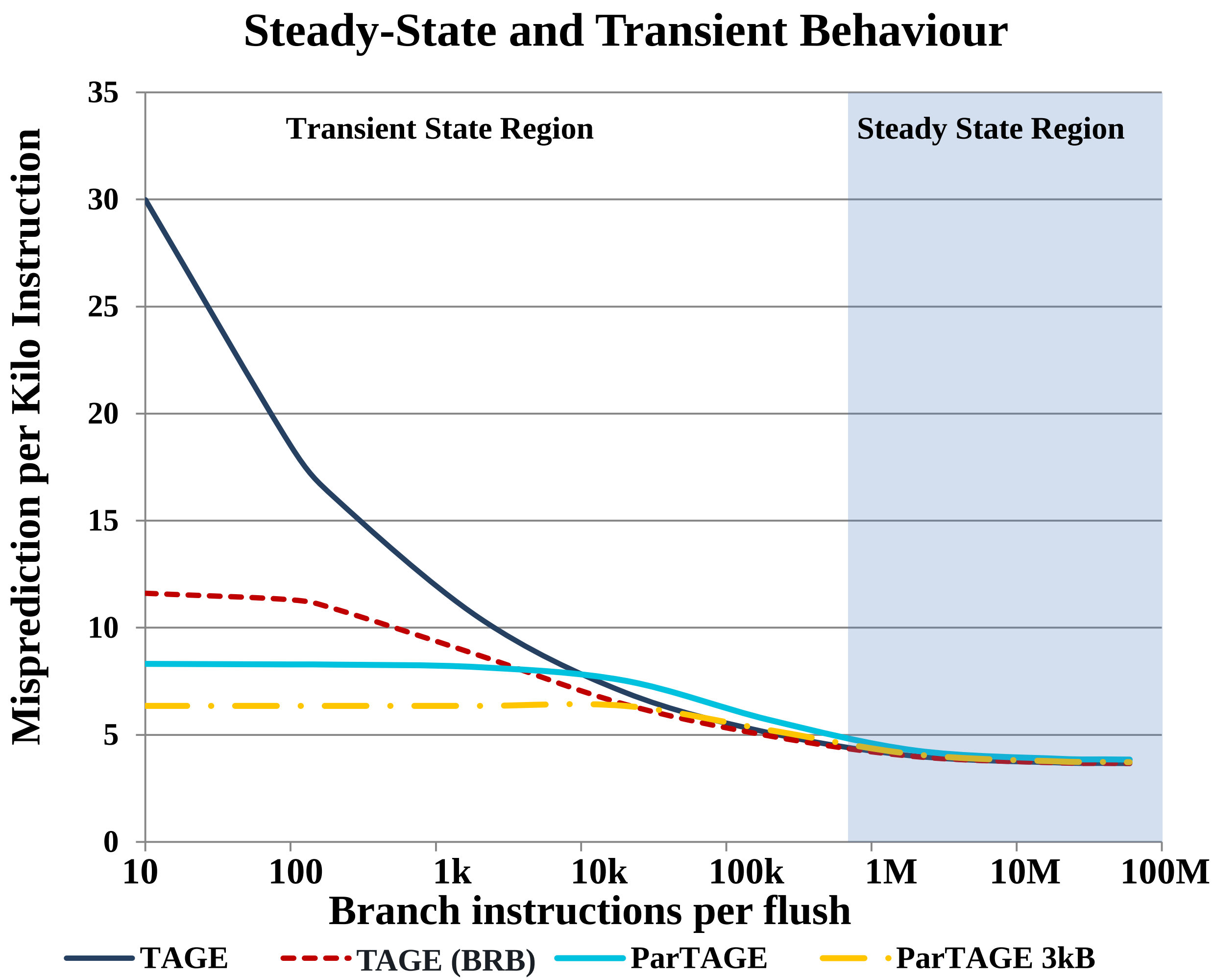
- New hybrid design, TAGE with Perceptron base
  - ParTAGE: Perceptron 8 tables, 1.25kB BRB entry size
  - ParTAGE 3kB: 3kB entry size, no statistical corrector



ParTAGE results

Comparing to empty TAGE again

- All version of ParTAGE are significantly better at transient state
- Notable improvements for 12k branch periods
- No effect at steady state

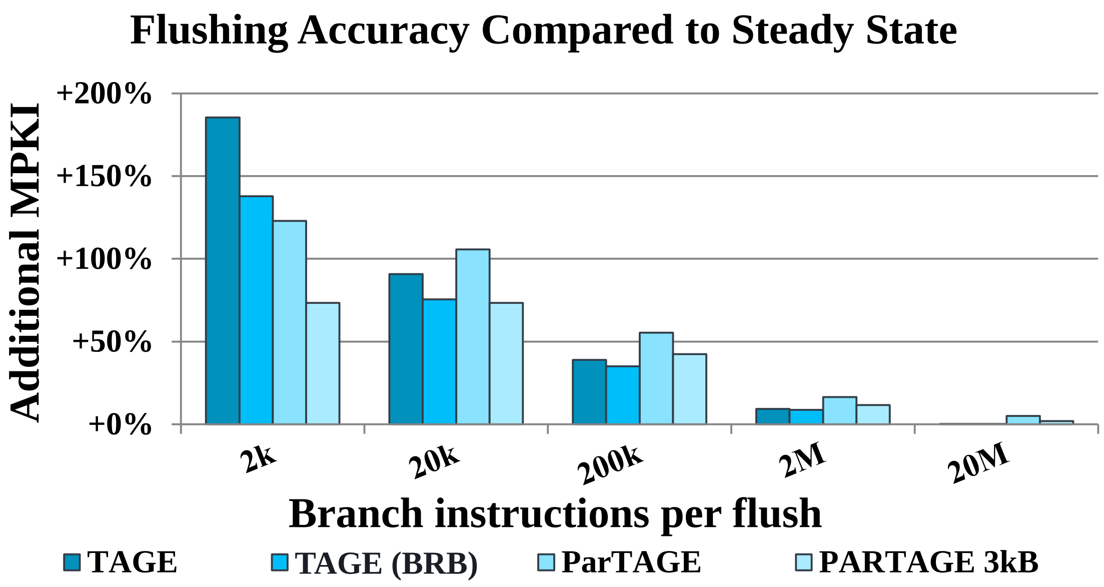


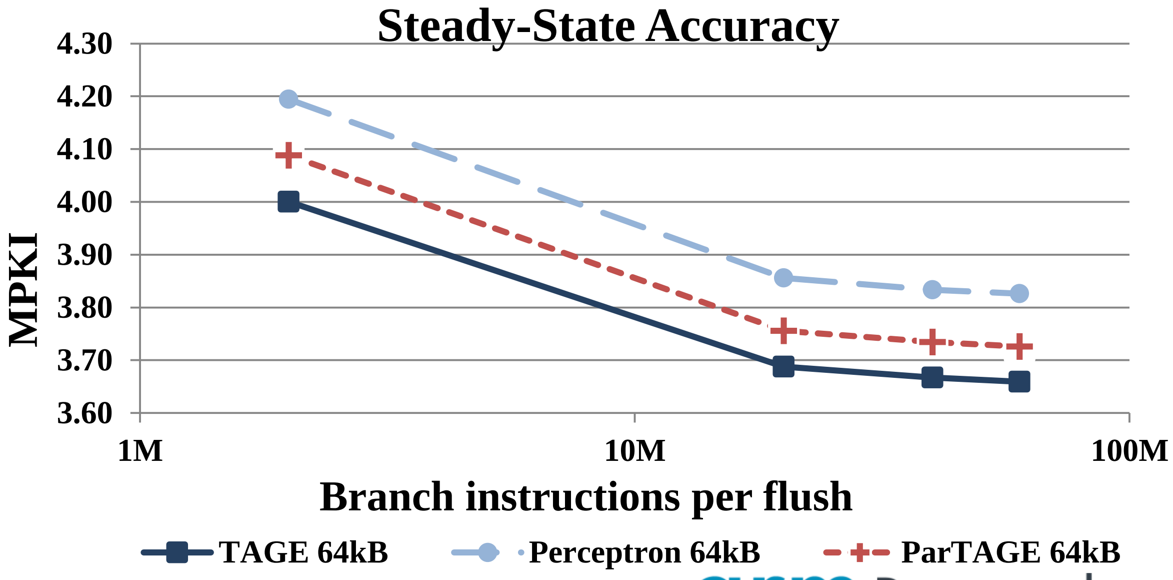
A Closer Look

Break down how components contribute to accuracy.

- TAGE (BRB) improves accuracy by 15%.
- ParTAGE delivers 20% better accuracy.
- Base predictor steady state ∝ Overall transient state.

ParTAGE steady-state on par with current designs.





Final Thoughts

Scan Me!

New balance: Area v Performance v Security

1. Predictors often operate at a transient state.
2. Isolation improves security, but costly: solution **BRB!**
3. ParTAGE better transient prediction.

Motivation for the future to improve small predictors

arm Research Summit

Discovery through Diversity: Addressing tomorrow’s challenges, together

September 15-18 2019 | Austin, TX

Call for submissions open now!

Submit your presentation, poster, workshop or demo and share your research with colleagues from around the world

Deadline: 1 May 2019

arm.com/summit

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה