TOSHIBA

IEEE IPDPS/iWAPT 2020

Machine Learning-Based Prefetching for SCM Main Memory System

Yusuke Shirota, Toshiba Corporation

For questions, email yusuke1.shirota@toshiba.co.jp

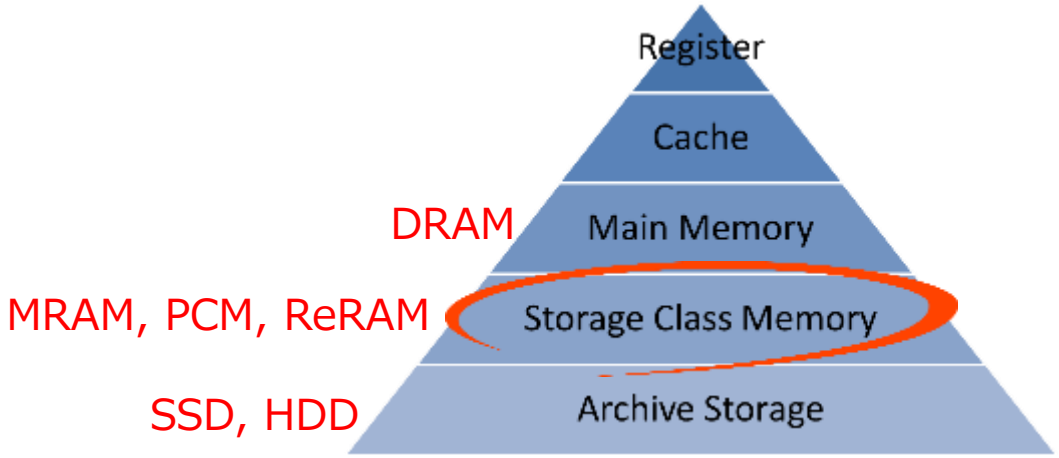
This work was based on results obtained from "Project to develop cross-sectoral technologies for IoT promotion" commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

Outline

- **Introduction**
- **ML-based Auto-tuning Framework for SCM Main Memory Control**
  - Offline Training Phase
  - Online Inference Phase
- **Evaluation**
- **Conclusions and future work**

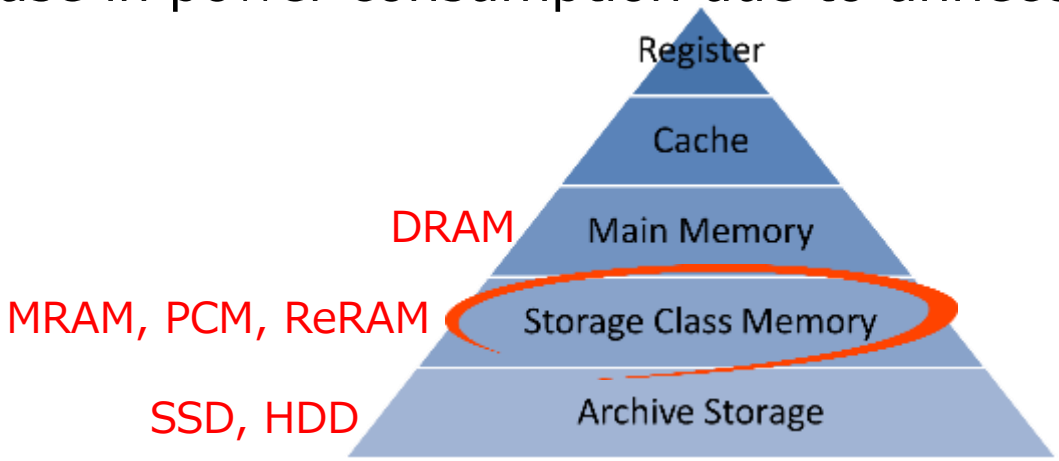
Emergence of SCMs

- Emergence of new applications such as Big Data analytics, deep learning for artificial intelligence (AI) and CPS/IoT have increased demands for in-memory data processing of large-scale data.
- In-memory processing used to require large-capacity DRAM main memory, but DRAM-only approach is hitting the limit due to DRAM's capacity scaling issue and significant background power.
- With new type byte-addressable high-speed non-volatile memories, or storage-class memories (SCMs), we can explore high performance, low power, and scalable main memory system by redesigning memory hierarchical controls.



Memory hierarchical control is important

- SCM features lower standby power and higher density compared with DRAM.
- However, SCM has higher access latency than does DRAM, so simply replacing DRAM with SCM may degrade processing performance.
- This increases the importance of memory hierarchical controls such as prefetch control which mitigates the high latency of SCM.
- In addition, SCM has higher dynamic power than DRAM. Precise prefetch control with higher accuracy is important to suppress increase in power consumption due to unnecessary prefetch.



Outline

- Introduction
- **ML-based Auto-tuning Framework for Memory Control**
  - Offline Training Phase
  - Online Inference Phase
- Evaluation
- Conclusions and future work

Optimum memory control through ML

- SCM-based main memory hierarchy control is complicated!
  - Control that considers execution time and power consumption is required.
  - There are various types of SCM such as MRAM, ReRAM, PCM and 3D XPoint, and differences in access latency and power characteristics between them complicate memory controls
  - When configuring main memory as a SCM–DRAM hybrid, it is necessary to perform hierarchical control to adaptively use both according to the application's memory access patterns.

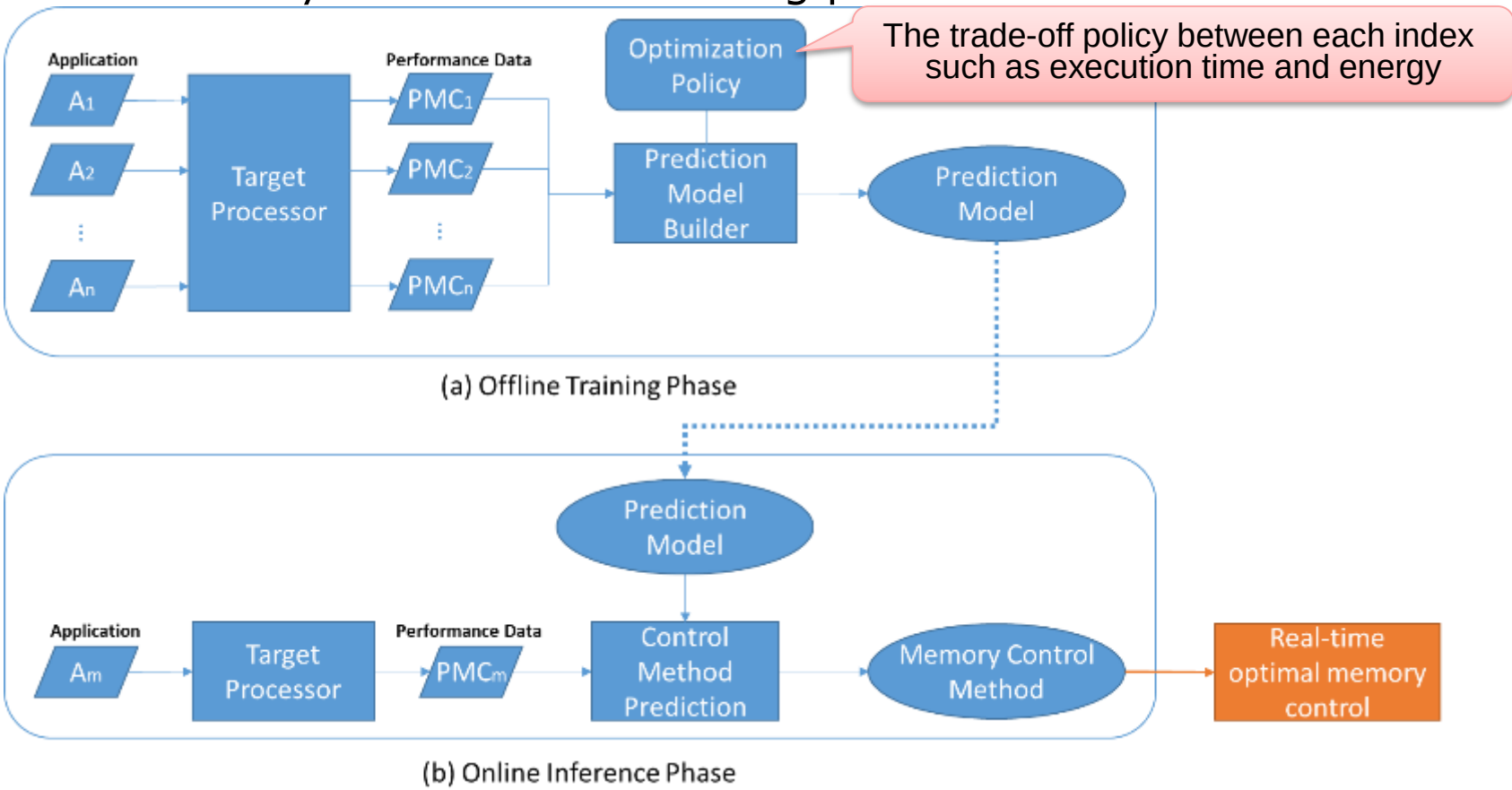
Memory Control Optimization Framework

- Auto-tuning(AT) framework for dynamically optimizing memory controls for various SCM-based main memory configurations, which utilizes machine learning based on system-level time series performance information.
- Aiming at efficient use of SCM with a focus on latency and power, we describe a method for applying the proposed framework to prefetch control.

ML-based AT Framework for memory control

Real time memory control using system level performance data

- **Offline Training Phase**
  - generate prediction models from performance data
- **Online Inference Phase**
  - Predict memory control method using prediction model



ML-based AT Framework for memory control

- **Time series performance data**
  - Collected from performance monitoring counter (PMC) that measures hardware events implemented on Intel® processor (*)
- **Performance Monitoring Counter**

Hardware Event Name	Description
PAGE_WALKER_LOADS.DTLB_MEMORY	Number of DTLB page walker loads from memory
L2_LINES_OUT.DEMAND_DIRTY	L2 modified lines evicted by a demand request
BR_MISP_EXEC.ALL_BRANCHES	Counts all mispredicted near executed branches (not necessarily retired)
MEM_UOPS_RETIRED.ALL_STORES	All retired store instructions
LONGEST_LAT_CACHE.REFERENCE	This event counts requests originating from the core that reference a cache line in the last level cache

* Intel and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

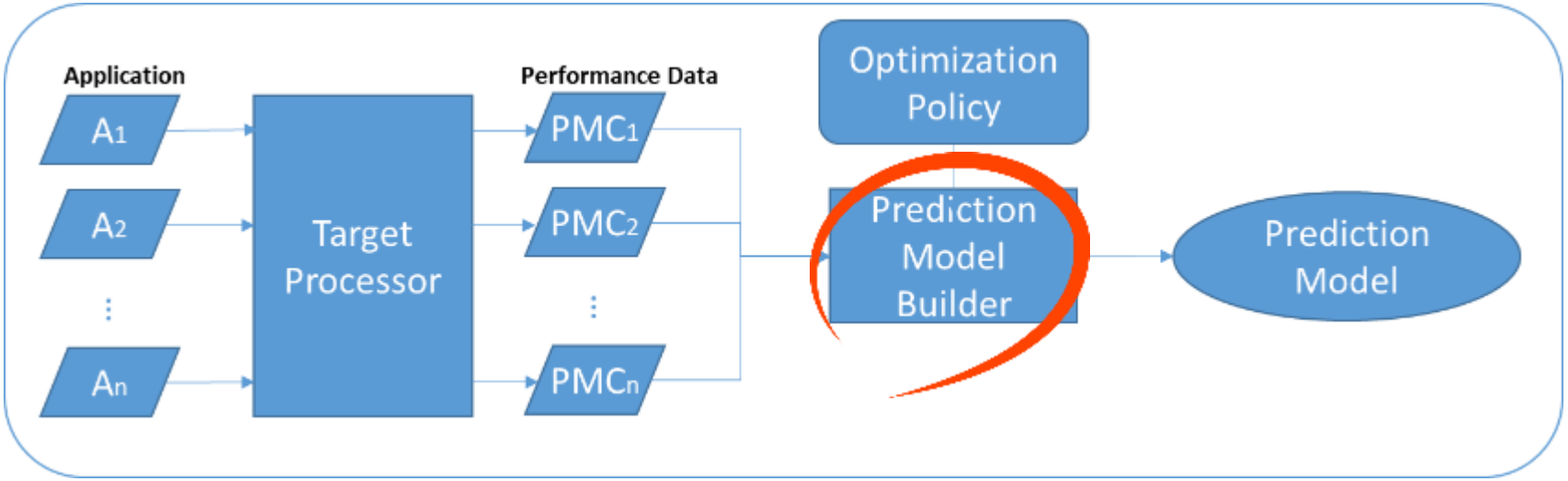
Outline

- Introduction
- **ML-based Auto-tuning Framework for Memory Control**
  - Offline Training Phase
  - Online Inference Phase
- Evaluation
- Conclusions and future work

Offline Training Phase

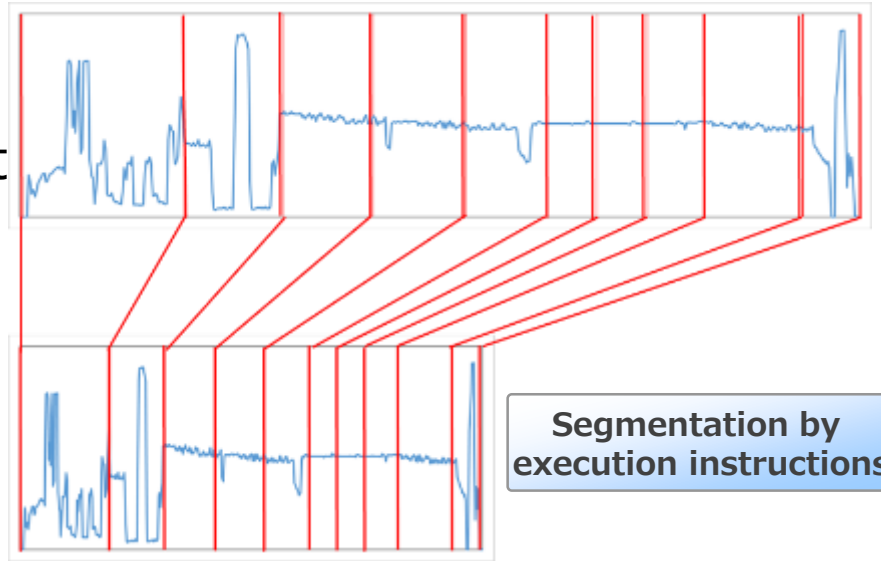
- **Machine learning for prediction model generation**
  - Select events that related on target memory control from huge hardware events
  - The method to automatically generate a prediction model without manually selecting the hardware events

Machine learning to automatically generate rules from input performance data



Prediction Model Builder

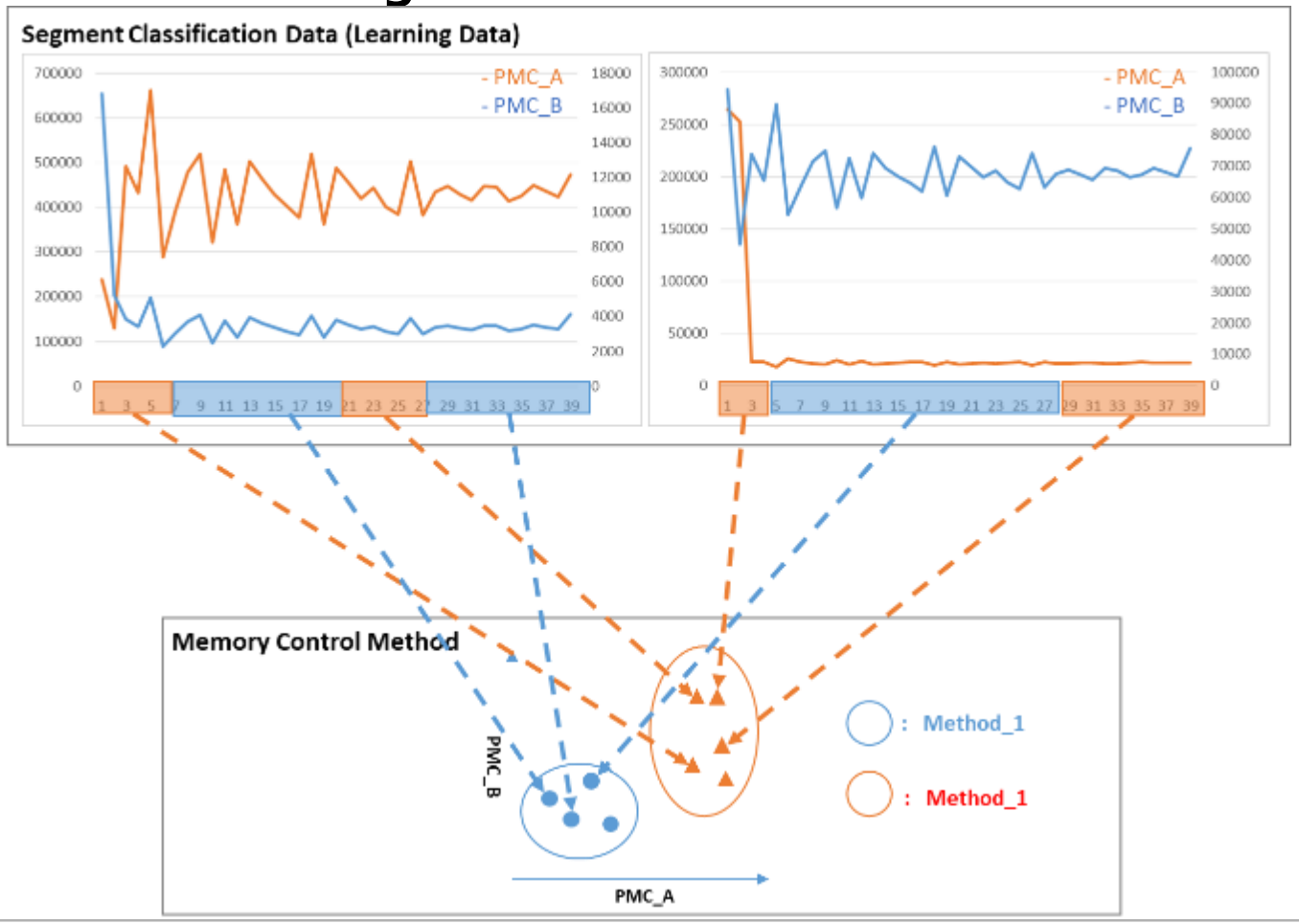
- **Supervised learning for memory control prediction model**
  - classifies unknown input data automatically
  - generates a prediction model by learning the train data
  - predicts classification result using prediction model when unknown input is given
- **Segment divider**
  - Determines the switching point
  - Execution time changes when memory control methods differ
  - segments according to the number of execution instructions



Characteristic extraction not affected by memory control method

Machine Learning

- Classification of each segment by memory control method
- Generate classification criteria of memory control method by machine learning

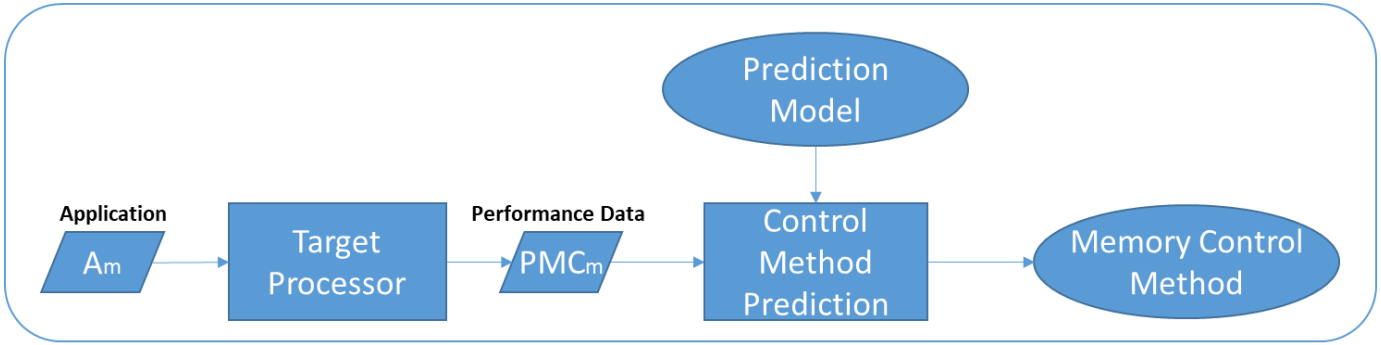


Outline

- Introduction
- **ML-based Auto-tuning Framework for Memory Control**
  - Offline Training Phase
  - Online Inference Phase
- Evaluation
- Conclusions and future work

Online Inference Phase

- **Predict memory control method at real time**
  - Input prediction model and time series performance data



- **Memory control method unset period**
  - The memory control method can not be assigned until the memory control method is predicted

Continuous Method

- **Continue memory control method executed in previous segment**
- **Effective when the characteristics of adjacent segments are similar**

Reset Method

- **Always executed by the same control method from the segment start until the memory control method is predicted**
- **Improves learning accuracy by fixed memory control method**

Prefetch Control

- **Useful for improving memory access performance in SCM main memory sysytem**
- **Intel® Xeon® processor (Haswell) prefetchers**
  - L2 hardware prefetcher
  - L2 adjacent cache line prefetcher
  - DCU (Data Cache Unit) streamer prefetcher
  - DCU IP (Instruction Pointer-based) prefetcher
    - Default setting enables all prefetcher settings
- **SCM main memory access**
  - SCM has higher access power than DRAM
    - Prefetching of unnecessary data due to misprediction increases
    - The power required for ejection and reload also increases because the necessary data is ejected from cache by unnecessary prefetch

Prefetch control considering SCM latency and energy efficiency

Outline

- Introduction
- ML-based Auto-tuning Framework for Memory Control
  - Offline Training Phase
  - Online Inference Phase
- **Evaluation**
- Conclusions and future work

Evaluation

- **Applicability evaluation of the proposed framework to prefetch control**
  - Predictive performance evaluation by reset method
  - Execution time and energy evaluation by reset type
- **Experimental conditions**
  - Evaluate SCM system by emulation using DRAM system.
  - Server
    - Intel® Xeon® Processor E5-2690 v3 (Haswell) 2.6 GHz
    - Ubuntu 16.04.2 (Linux 4.4.0-130-generic)
  - Experimental data
    - 11 applications included in PARSEC (a multiprocessor benchmark collection)
  - Hardware events (Performance counter)
    - Selected 31 hardware events related to TLB, branches, etc., which may have an influence on prefetch control
  - Segment
    - divided at equal intervals for each 3G instructions

Training Data

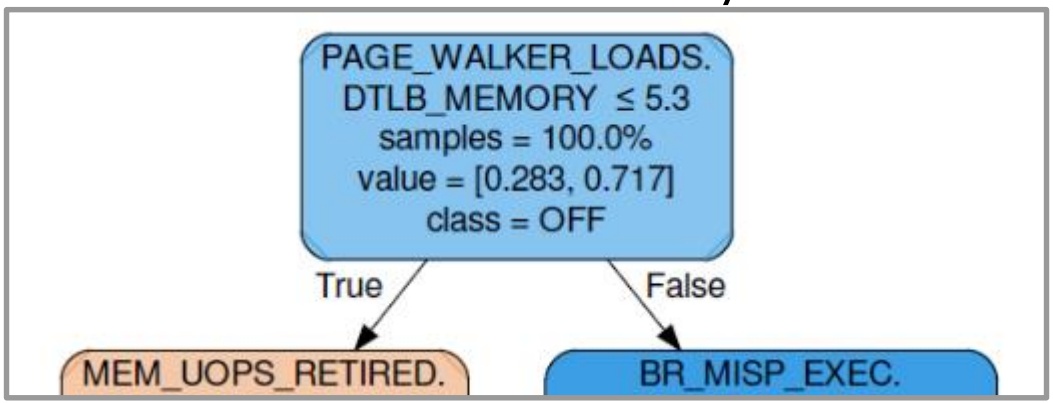
- **Training data generation**
  - Generate for each segment
  - Consists of optimal prefetch control method based on hardware events and optimization policy
  - Optimization policy
    - Energy efficient prefetch control
    - Consider execution time and energy of main memory access
- **Determination of optimal prefetch control method**
  - Select a shorter execution time method
  - Select a lower energy method, if the difference in execution time is small

【Optimal prefetch control method】

```
for i in segment_number:
if time_prefetcher_on[i] <= time_prefetcher_off[i] * 0.95:
  best_prefetch[i] = "ON"
  elseif time_prefetcher_on[i] > time_prefetcher_off[i] * 0.95:
    if energy_prefetcher_on[i] <= energy_prefetcher_off[i]:
   best_prefetch[i] = "ON"
    elseif energy_prefetcher_on[i] > energy_prefetcher_off[i]:
      best_prefetch[i] = "OFF"
```

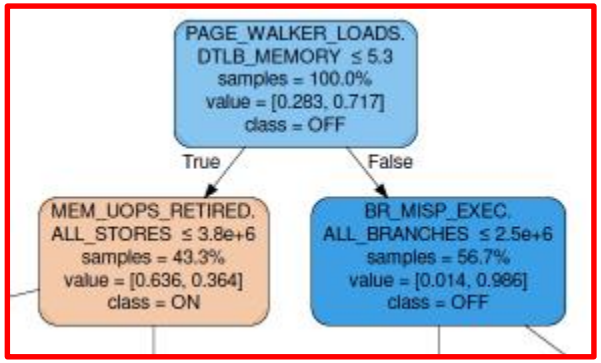

Evaluation of Decision Tree

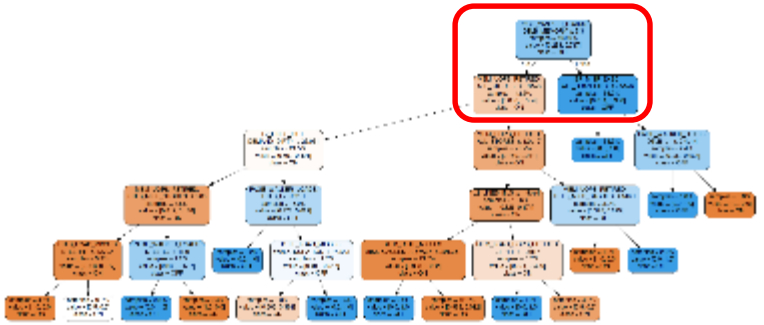
- **Depth 5 decision tree generated by reset method**
  - Fastest prediction using a depth 5 decision tree
    - Prediction accuracy rate does not change in the depth 5-7



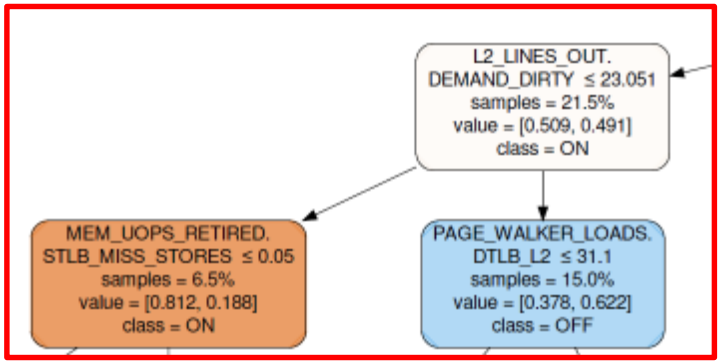
Branches affecting the prefetch prediction

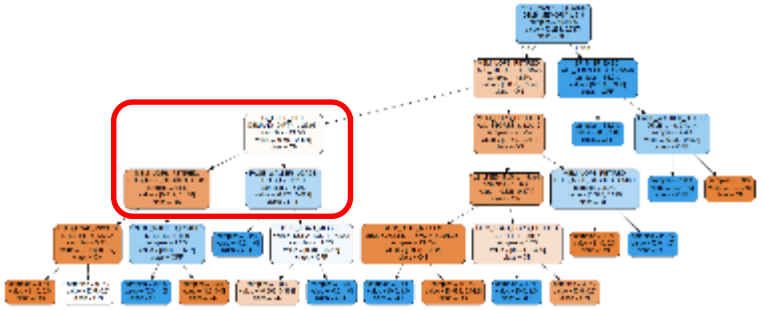
- **Branches affecting the prediction of prefetch control method**
  - Conditional branch using PAGE_WALKER_LOADS.DTLB_MEMORY
    - Number of times the page walk loaded form memory
    - FALSE : Prefetch OFF because memory accesses are spread over a wider





  - Conditional branch using L2_LINES_OUT.DEMAND_DIRTY
    - Number of dirty L2 cache lines ejected by demand requests
    - FALSE : Prefetch OFF because prefetching may further dirty the cache

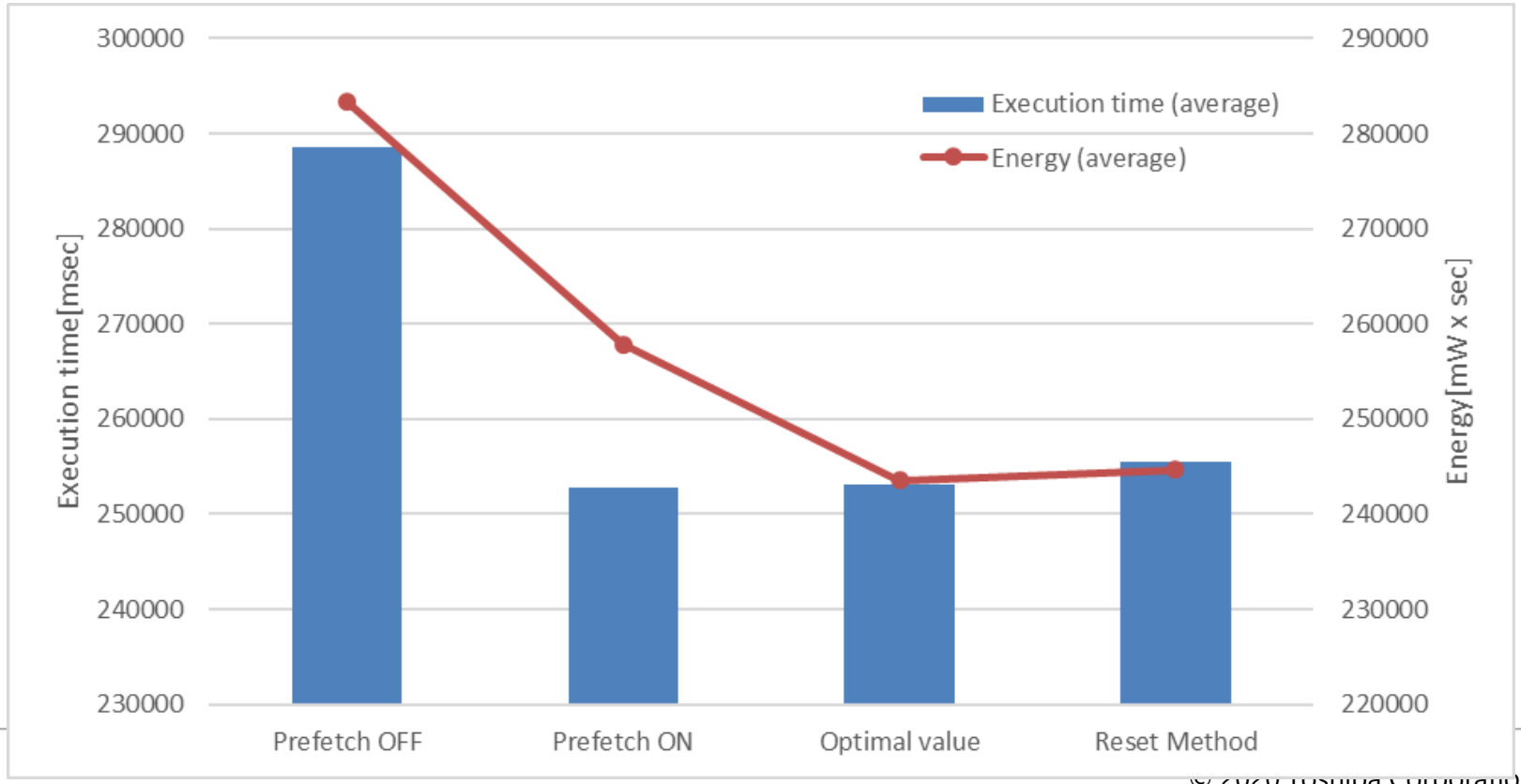




Valid for branches affecting prefetch control

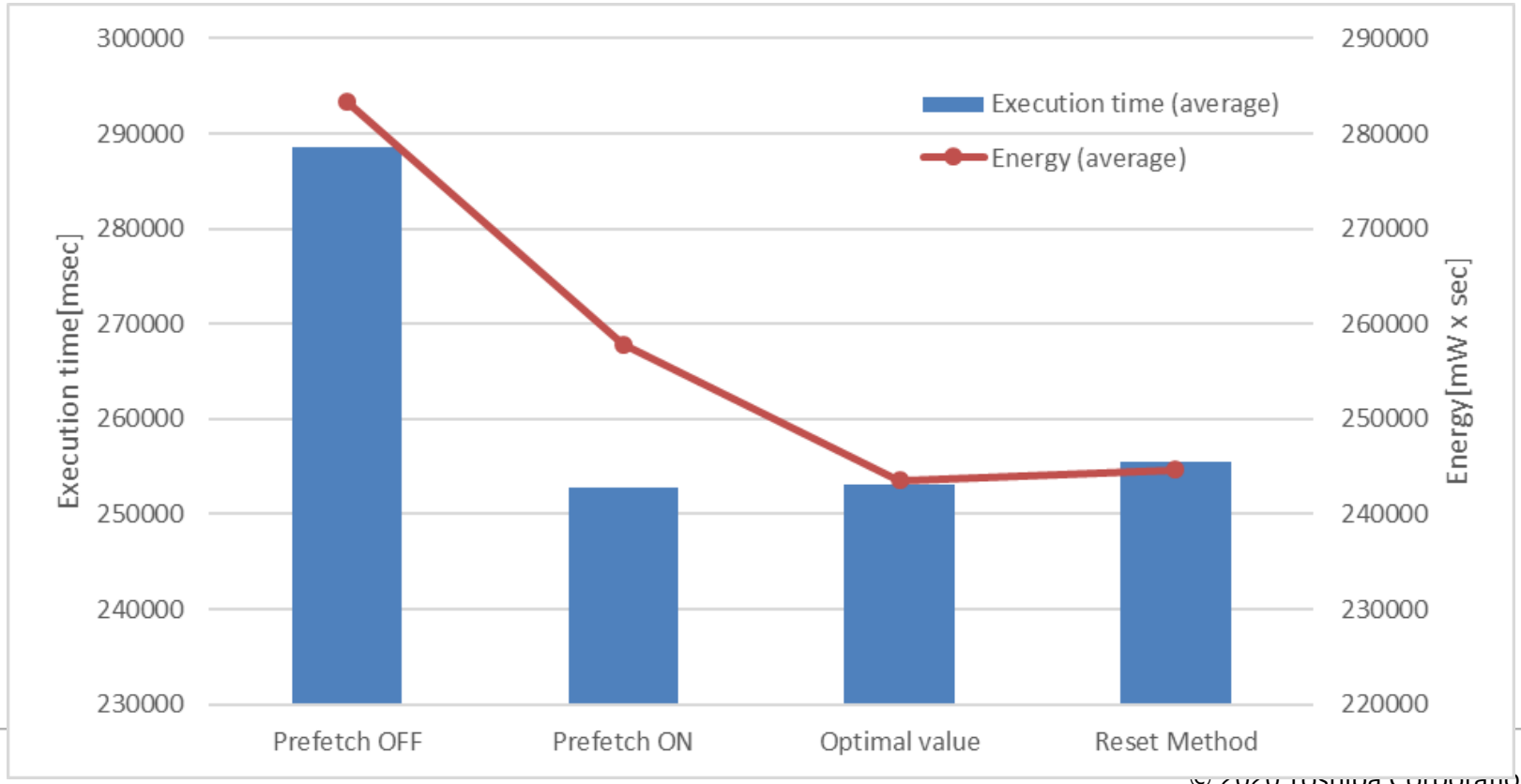
Comparison of execution time and energy

- **Compare execution time and energy of application of reset prefetch method**
  - Prefetch OFF for all segments
  - Prefetch ON for all segments
  - Optimal prefetch control
  - Reset prefetch method (proposed method)



Comparison of execution time and energy

- **Result**
  - Bothe execution time and energy are close to the prefetch optimum value in the reset type (proposed framework)
- **Conclusion**
  - Reduce energy by prefetch OFF to reduce the impact on execution time
  - Execution time of prefetch ON/OFF is value on DRAM, but the differences is further expanded in SCM



Outline

- Introduction
- ML-based Auto-tuning Framework for Memory Control
  - Offline Training Phase
  - Online Inference Phase
- Evaluation
- **Conclusions and future work**

Conclusions and future work

- **Conclusions**
  - Propose a framework to predict memory control method using decision tree which is one of supervised learning
  - Evaluation shows that the proposed framework can learn the application efficiently and generate decision tree, and shows the possibility of fast, accurate memory control
- **Future work**
  - Improve the accuracy of prefetch control
    - Improve decision tree
    - Improve segment division
      - For more accurate extraction of characteristic of time series performance counter data
  - Application to hierarchical control of SCM/DRAM hybrid main memory
    - Framework extension including continuous method