THE ANDREW & ERNA VITERBI FACULTY OF ELECTRICAL ENGINEERING

CRML Control Robotics and Machine Learning Laboratory

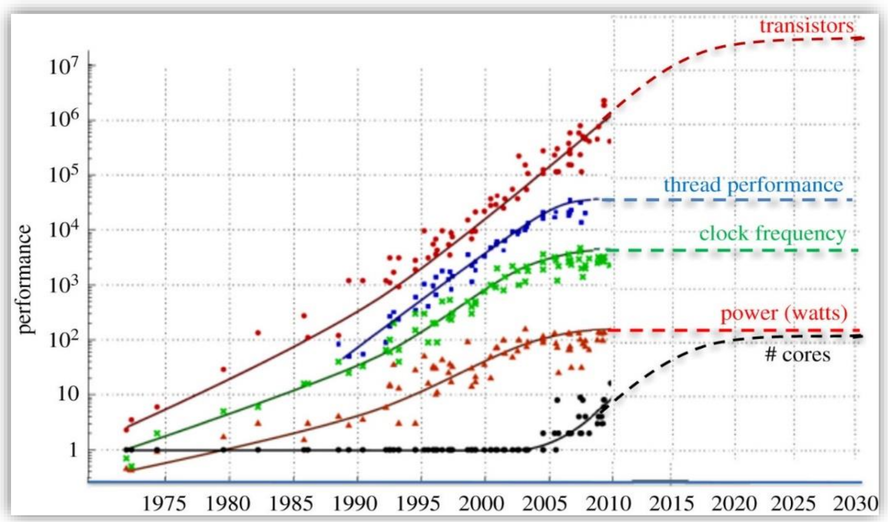
TECHNION Israel Institute of Technology

Machine Learning-based Linux Cache Handler

Netanel Rothschild and Roy Maor Lotan, Supervised by Dr. Roman Kaplan

Introduction

- The 21th century is well known as the "data century" (Moore's law)
- We have more data on multiple devices (pictures, docs, music, etc.)
- Today's computers suffers from accessing data
- A lot of effort invested towards solving this problem
- Average pc waste 95% of the time waiting for I/O



Goals

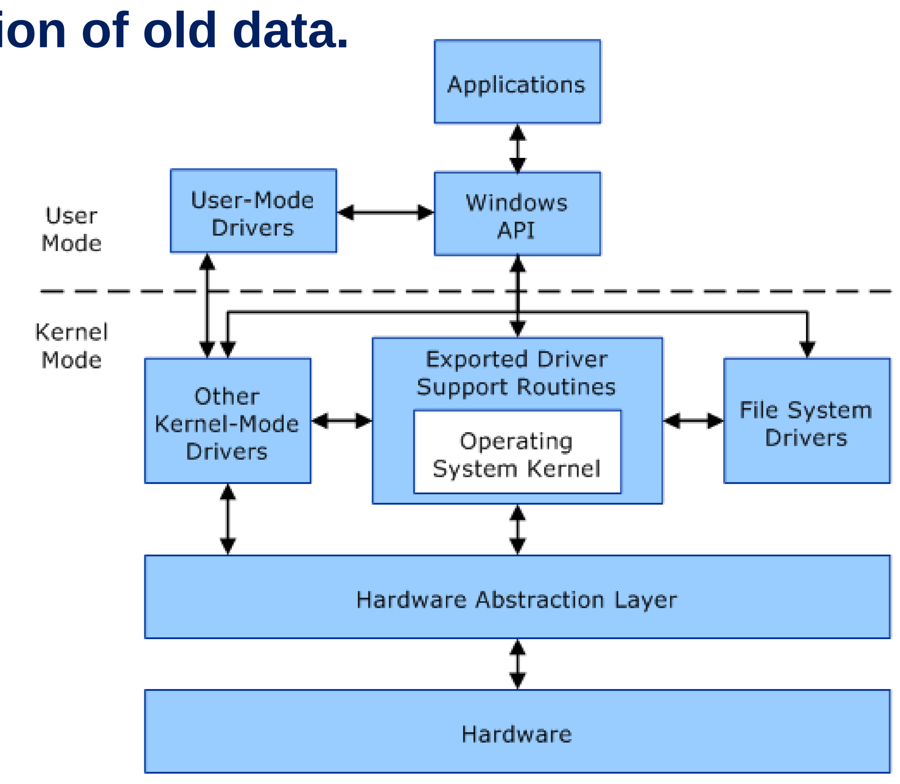
- Learn the user behavior in order to predict the flow of data
- Gain higher performance using the knowledge about the user

Challenges

- Collecting the date from real Operating System in real time
- Evaluate the correlation of the data and the target knowledge

Operating Systems Cache

- The module that manage the data within systems called page cache
- This module is a part of the OS Kernel manage the I/O request and manage the eviction of old data.



Data Extraction

Method 1: Snapshooting

- Taking a snapshot of the cache
- Sample entire image, associating each page with its pid (owner)
- Exanimating the cache action in RT
- Sample clean groups (OS association) for training model

Method2: Trace Processes

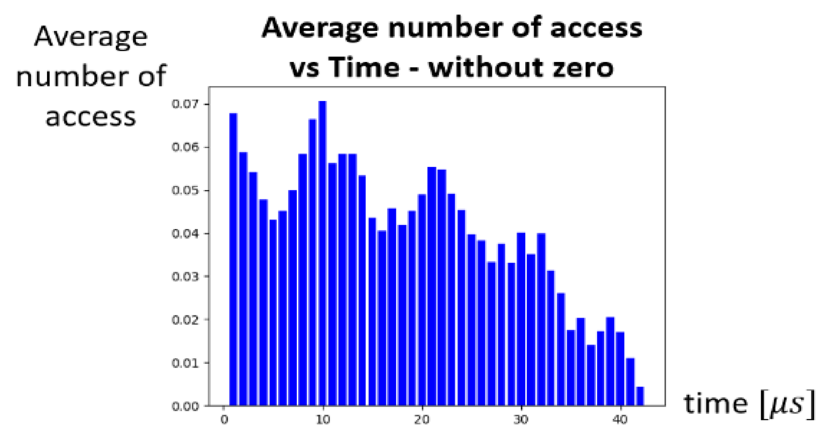
- Tracing one process, monitoring it's access to the memory
- Repeat this action until we have enough data to characterize the user usage of data
- Speculate the next step of a process before evicting his page

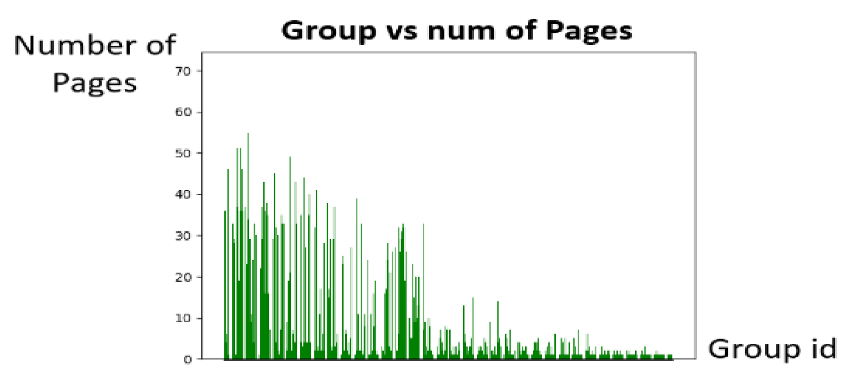
Method3: Bus Monitoring

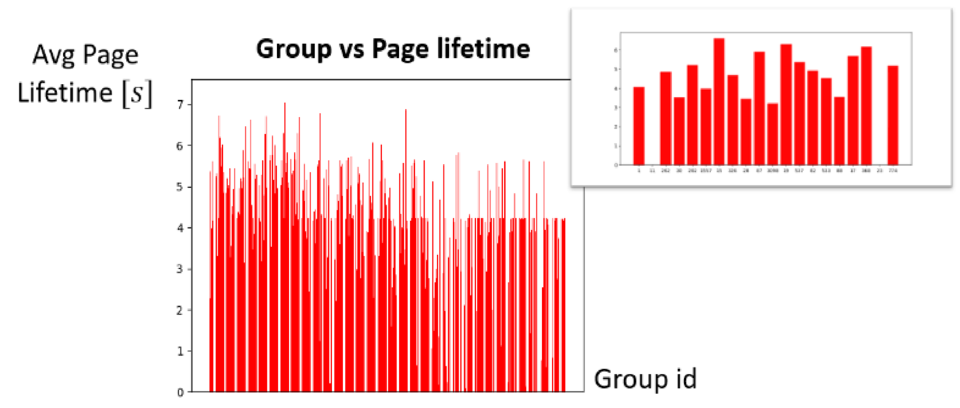
- Starting with time 0, we listening to the requests on the bus
- Any access will be written in a Non-Cacheable buffer and will be written to file is scheduled moments.
- To get an accurate picture of the cache the module should have Kernel permission

Data analysis

- An examine workload points that there is strong connection between the following features and the reappearance of data







Problem Definition

The probability $p_{y|x}(y|x) \triangleq \sigma(\mathcal{O}(x;\theta))$

Given a Dataset $\mathcal{D}_N = \{X_i, y_i\}_{i=0}^{N}$

The loss function is binary cross-entropy:

$$\mathcal{L}(\mathcal{O}(X_i;\theta), y_i) = y_i \log(p(y_i)) + (1-y_i)\log(p(1-y_i))$$

We want to minimize (ERM):

$$\theta_{\mathcal{D}}^{*} = \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\mathcal{O}(X_i;\theta), y_i)$$

Chosen Models

Shallow

Input: 2 $Neurons$
Layer 1: 16 $Neurons$, activation: ReLU
Layer 2: 8 $Neurons$, activation: ReLU
Output: 1 $Neurons$, activation: Softmax
Trainable parameters: 193

Deep

Input: 5 $Neurons$
Layer 1: 128 $Neurons$, activation: ReLU
Layer 2: 64 $Neurons$, activation: ReLU
Layer 3: 64 $Neurons$, activation: ReLU
Layer 4: 8 $Neurons$, activation: ReLU
Output: 1 $Neurons$, activation: Softmax
Trainable parameters: 13,457

Results

- Shallow model Acc: 84.54%
- Deeper model Acc: 87.91%
- Both model affine function of a state that can be saved by the module, in any change (using basic operation in a single clock)
- Both models outperform current existing paging algorithms.
- Training Time, and resources satisfy a normal pc background use

Conclusions

- Choosing the right data in shallow model, with the highest correlation to the target, in our case result in almost the same performance 10^3 scaled deep model
- Machin Learning-based algorithms can play an important role in Operating System management, in order to gain better performance as it works in understanding user experience

April 2021