Advanced Operating Systems Final Project Proposal
TLB Behavior Monitoring

Mohammad M. Gharaguzlo, Soheil Fadaee
401206836, 401205189
{moh.gharaguzlo13, soheil.fadaee22}@sharif.edu

A Concise Review of VM Abstraction

Virtual memory emerged in an era whence memory size limitations were a serious hurdle in the way of multi programming.
It enabled the processor to have access to a good number ready-to-execute programs by fitting only the necessary parts of the aforementioned tasks in main memory (aka *Segmentation*), thus improving the CPU utilization.
VM provided better security by *isolating* processes in their own *address spaces.*
It also introduced a number of optimizations. Namely, *copy on write* and synonyms among different virtual addresses(multiple logical addresses which point to the same physical address to prevent data redundancy and save memory).
However, these improvements haven't been achieved without a cost. The process of translating virtual address to physical takes *time* and *space.* Numerous techniques has been introduced over the years to address this issues. For space, the most popular and practical of which is using *Multi Level Page Tables.*
As for the time, traditionally the industry's solution was to employ a cache-like hardware unit called *Translation Lookaside Buffer*(aka TLB) to speed up the translation routine.

New Challenges

With the ever growing size of today's applications the overheads of address translation in both terms of time and space is becoming a real issue, with 5-20% of applications runtime begin spent for translation. This is even more horrendous for *Virtualized platforms* of which the translation can take of up to 50% of runtime.
This is partly rooted in the increased number of processing cores and the need to synchronize their TLBs. Namely, TLB *Shootdowns* and *Flushes* can cause significant performance setbacks for the entire system.
Remember that virtual memory belongs to an era of single core processors with small sized memories. With the ridiculous level of *parallelism* in today's application and also advancement in technology, which has led to *high capacity main memory units* it seems that the traditional abstraction of Virtual memory needs a serious overhaul.
This has motivated both academia and industry to consider new approaches to address these issue, we will explain some of them in detail after reading the related papers.

Our goal

We aim to monitor TLB's behavior and if possible extract a pattern regarding each application working nature to the amount of page walks and shootdowns.
For this, we have selected 5 intensive task from the provided list. Tensorflow, Pytorch, Matlab, Deep500 and Geekbench.
Our initial estimate is that we can run the required apps and analyse the data for bare metal systems by the khordad 7th.
For deep learning frameworks, we've chosen *Vgg 16/19*, *AlexNet* and possibly *ResNet.* Regarding the data sets and *Matlab's* specific running task we're gonna need a more detailed instruction from the teaching team. This also applies to *Geekbehnch.*
The rest of project which includes *Virtualized platforms* and various *page sizes*, will be done in the second stage.