MO601 - Project 2 Report

093311 - Alberto Arruda de Oliveira[1]

[1]Institute of Computing – University of Campinas (UNICAMP)

{alberto.oliveira}@ic.unicamp.br

Abstract. *In this project, we examine the virtual memory system employed in modern computers, particularly focusing our attention on Translation Lookaside Buffer (TLB), used to optimize such systems. The TLB is a special cache used to speedup the translation of virtual addresses into physical addresses, avoiding the need to access the main memory every time a virtual addresses has to be translated. Each computer system employs two TLBs: one for Instructions and one for Data. We aim at studying the impact that the Physical Page size has in the number of memory accesses required due to TLB misses and Cache misses, using 4kB and 4MB pages, in addition to the overall impact of the TLB in the performance of a computer system. A Pintool designed to simulate Caches and TLBs, as well as main memory accesses, together with SPEC2006's [SPE] benchmarks and toy benchmarks designed specifically for this project.*

1. Introduction

Virtual memory is employed in modern computer systems as means of virtually increasing the amount of memory available to application, as well as offering means of more protection, by isolating the memory usage of different applications. With virtual memory, *virtual memory addresses* seen by an application are mapped into *physical addresses* in the main memory. This mapping is performed by dividing the main memory into contiguous blocks called *pages*, which are indexed by a page table. Each application has a page table, which is indexed using part of the virtual address seen by the application. Once a page is retrieved using the page table, another part of the virtual address, the offset, is used to find the physical block page.

However, the virtual memory system as described above has two important issues. First, the page table as is, requires too much memory space and is unfeasible for multiple applications. Moreover, each translation requires a lookup in the page table, adding one memory access for each instruction (for fetching), and even more if the instruction deals with storing or loading memory content. The first issue is solved by using *multi-level page table*, in which each entry in a higher level page table table indexes a position in a lower level one, and the lowest level table indexes the page itself, allowing the application to only have parts of the page table loaded into memory in a given moment. This solution, nevertheless, accentuates the second issue, as each page table lookup now requires an additional of n memory access, where n is the number of page table levels.

Another optimization, aiming at solving the second issue, is the addition of the *Translation Lookaside Buffer*(TLB). The TLB is a cache used to store translated virtual addresses. It is indexed by the virtual address, and returns the translated physical address. Each hit in the TLB negates the need for the n memory accesses described above. Like the first level of cache, there is one instruction TLB and one data TLB.

Table 1. Memory Statistics with 4kB and 4MB pages

		IM		IT		IP		DM		DT		DP	
BM name	**input**	**4 kB**	**4 MB**	**4 kB**	**4 MB**	**4 kB**	**4 MB**	**4 kB**	**4 MB**	**4 kB**	**4 MB**	**4 kB**	**4 MB**
403.gcc	**166.in**	79396806	345136	26380863	3	79142589	9	1503347135	198237044	436102156	12366	1308306468	37098
403.gcc	**200.in**	279193514	2452045	92318751	3	276956253	9	3694007144	207005239	1165078483	3244405	3495235449	9733215
403.gcc	**c-typeck.in**	111009769	668114	36747413	3	110242239	9	1872216860	223672250	546909639	1088539	1640728917	3265617
403.gcc	**cp-decl.in**	75092240	460578	24878826	4	74636478	12	1575603959	166080856	470300084	121127	1410900252	363381
403.gcc	**expr.in**	63566749	389530	21083382	4	63250146	12	1792151426	254589010	484279973	721097	1452839919	2163291
403.gcc	**expr2.in**	100001418	582002	33155824	3	99467472	9	2288060088	404444601	636173330	3138100	1908519990	9414300
403.gcc	**g23.in**	93618324	741586	30974751	3	92924253	9	2715095648	983219620	579391024	1551249	1738173072	4653747
403.gcc	**s04.in**	102195440	329014	33604910	3	100814730	9	2949593035	873120033	693926483	1490725	2081779449	4472175
403.gcc	**scilab.in**	138623397	1335317	45723748	4	137171244	12	1323926336	40329999	427915043	47	1283745129	141
416.gamess	**cytosine.2**	12344669	10382	4111315	7	12333945	21	1021788145	2456279	1771433037	17	1019331815	51
416.gamess	**h2ocu2+.gradient**	6620035	55822	2200812	7	6602436	21	1168231591	9254682	3251229764	18	1163754700	54
450.soplex	**pds-50.mps**	6051339	1060085	956258	6	2868774	18	3945747605	3450531394	233295757	156524	699887271	469572
450.soplex	**ref.mps**	4110378	253525	1284844	6	3854532	18	11692542247	8717521825	985190908	2958018	2955572724	8874054
456.hmmer	**nph3.hmm**	15377140	7708	5123236	4	15369708	12	540301346	58694661	159604263	16	478812789	48
462.libquantum	**control**	1964	1496	160	4	480	12	56549189182	53774731522	924819841	64	2774459523	192
464.h264ref	**foreman**	11128567	122818	2798461	4	8395383	12	1611682660	21781459	526048662	16	1578145986	48

This project has two main tasks: (1) analyzing the impact of the TLB in optimizing the virtual memory translation, and (2) examine the impacts that different sized physical memory pages (4kB and 4MB) have in the number of memory accesses required by an application, considering the usage of the TLB. For this task, we have used PIN [Pin] to run a Pintool that simulates Caches and TLBs to count the total number of memory accesses required by the application, the number of TLB misses, and the total number of memory accesses for translation, considering a 3-level Page Table. This Pintool have been run with some of the SPEC2006's [SPE] benchmarks, as well a few toy benchmarks developed specifically for this project.

2. Impact of page size in the number of memory accesses

Choosing a page size is crucial for the performance of virtual memory systems, especially when considering TLB's translation of virtual addresses. Bigger pages lead to less pages overall, meaning that the number of possible translations of a virtual address is smaller. Thus, the likelihood of a hit when checking the TLB is lower, leading to less memory accesses with the purpose of checking the page table. However, the trade-off of using bigger pages is more internal fragmentation.

In this experiment, we compare the performance of a virtual memory system with Instruction and Data TLBs when using pages of 4kB and 4MB. We consider a page table with 3 levels, and TLBs with 32 rows each and full associativity. We were tasked to compare three statistics of the virtual memory system: *total number of memory accesses* (IM for instructions and DM for data), *TLB misses* (IT for instructions and DT for data), and *total number of page table accesses* (IP for instructions and DP for data). IT and DT are given by the instruction and data TLBs respectively, while $IP = 3 * IT$ and $DP = 3 * DT$, considering a 3-level page table. Finally, $IM = IP + IL$ and $DM = DP + DL$, where IL and DL are the number of instruction and data L3-cache misses respectively. The values of IM and DM are computed this way because each miss in the L3 cache requires a new memory access. Table 1 displays the aforementioned memory statistics for virtual memory systems using 4kB and 4MB pages.

As table 1 shows, it is clear the impact that the size of the page has in the number of misses in the TLB, in particular for instructions. This happens because larger pages can house much more content, so the number of pages required by an application is much smaller, reducing the likelihood of TLB misses. The number of required instruction and data accesses for 4MB page systems ends up being much smaller than the required for

Table 2. Time impact of TLB in systems with 4MB Pages

	With TLB		**Without TLB**		**Increase Factor**
BM name	**Page Table Accesses (Inst. and Data)**	**Time Consumed (s)**	**Page Table Accesses (Ins. and Data)**	**Time Consumed (s)**	
501.toy-bm-1	27	0.00000003	120616	0.000120616	4467.26
502.toy-bm-2	30	0.00000003	217128	0.000217128	7237.60
403.gcc <<200.in	9733224	0.00973322	2130813598	2.130813598	218.92

4kB page systems.

3. Performance impact of TLB

In the previous section, we have seen how the page size impacts the number of memory accesses an application has to do. This section goes further and explores the impact of removing the TLB of a computer system, in terms of memory accesses. In practice, removing the TLB means that each Instruction and Data TLB access will require now n memory accesses, where n is the number of page table levels. Considering a 3-level page table, and that each memory access takes approximately 60ns, Table 2 displays the time impact caused by all the memory accesses with and without the TLB. The table considers a 4MB page size, and the two toy benchmarks listed were made specifically for this project. Toy benchmark 1 consists in adding two integers and storing the result in a third, while Toy benchmark 2 consists in storing values in a vector, then storing the multiplied values of the first vector in a second, and printing both.

In all cases, removing the TLB increased the required time to access the main memory to check the Page Table in more than 200 time in comparison to using the TLB. In the case of bigger applications, like GCC, a process which took less than 1/100 of a second, ended up taking more than 2 seconds to complete. This alone shows how important the TLB is for the proper functioning of the paging system.

4. Conclusion

In this project, we studied the Virtual Memory used in modern computers, with particular interest to the *Translation Lookaside Buffer* (TLB), a hardware optimization employed to reduced the required number of main memory accesses in such systems. Using Virtual Memory, a before finding some data in main memory, a translation process is required to turn virtual addresses into physical addresses. Because the translation structure is in main memory, each of such translations would required accesses to the main memory. The TLB is a cache to store previously translated addresses, so they can be retrieved without requiring accessing the main memory in the future.

Not only the TLB impacts the overall performance of the system, by drastically reducing the number of required memory accesses as shown in Section 3, but it also is dependent of further characteristics of the virtual memory system, such as page size, to achieve even better performance. We have seen that when removing the TLB, the time of accessing the main memory to translate virtual addresses alone increases more than 200 times, some times largely trespassing 1000 times more, showing how crucial such hardware is for the proper functioning of virtual memory.

Moreover, the page size also greatly impacts the effectiveness of the TLB. By using larger pages, as seen in section 1, the number of physical pages available is much smaller, and thus a larger percentage of their translations fit in the TLB. This increases the likelihood of hits, improving the performance. This is particularly notable for instructions, as in all cases a number of misses which greatly surpasses the thousands can be reduced to less than tens.

References

Pin - a dynamic binary instrumentation tool. `https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool`. Accessed: 2016-09-15.

Spec cpu 2006. `https://www.spec.org/cpu2006/`. Accessed: 2016-09-15.