Rethinking RAM: Testing alternative models of computation

David Lachut
dlachut1@umbc.edu

Kaustav Lahiri
klahiri1@umbc.edu

Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County

15 May 2013

Abstract

This is where the abstract will briefly summarize how great our project is. After all, we discovered that the VAT/RAM model is superior to the RAM/VAT model of computation.

Keywords RAM Model, computational models, benchmarks, algorithms

1 Introduction

The Random Access Machine (RAM) model of computation is widely used for analyzing the performace of algorithms. Some, such as Jurkiewicz and Mehlhorn question whether this model accurately represents the complexities of modern hardware. They propose the Virtual Address Translation (VAT) model of computation to account for added complexities, such as the memory address translation needed to refference ever larger amounts of system memory in modern computers.

This paper analyzes a set of algorithms using both models, VAT and RAM, and presents the results of benchmarking these algorithms on real hardware. The goal of this is to then use the results of the analysis and benchmarks to determine which model more accurately reflects the real hardware.

The novel contribution includes not only the benchmarking of these procedures, but also the comparative analysis of the two models. This work could be very significant if VAT is shown more accurate than RAM. We would be the first to independently verify this and we would be contributing some of the first analysis of common, reference algorithms. (We're still working on results.)

This paper is organized as follows: Section 2 presents the paper's motivation. Section 3 addresses previous work. Section 4 details how the benchmarks and analyses were done. Section 5 presents the results. Section 6 discusses the results, selecting the more accurate model. Section 7 and 8 address the potential for future work and conclude the paper.

2 Motivation

Analysis of algorithms on abstract machines serves as a crucial element of computer science, which allows incredible increases in the speed and utility of computer programs. Until now, computer scientists have used the RAM and the EM models to perform algorithmic analysis accurately. But Jurkiewicz and Mehlhorn observe discrepancies with some experimental findings and attempt to push forward the VAT model to account for these discrepancies.

Before algorithm analysts shift to this new model, they must carefully verify its claims and correctness, it's utility. This is where Jurkiewicz and Mehlhorn's contribution falls short. Their paper does mention that experimental findings tally with the theoretical predictions of the new VAT model, but it does not clearly represent these findings. Additionally the number of test cases is limited. So before scientists readily accept this new model for theoretically analyzing the running time of algorithms the test set must be broadened to verify and compare the experimental results with the model's predictions.

If it can be verifed by experimental comparison that the new VAT model is more accurate, then this will serve as an incentive to researchers to use the proposed model for more accurate estimations of running time of algorithms. More accurate algorithm analysis will help maintain the pace of innovation enjoyed by computer science for a generation.

3 Previous Work

It is evident that most papers on algorithms tend to ignore the cost of virtual address translation, though researchers recognize these costs. This is because of the model that researchers follow, and that is the classic RAM model without virtual memory. Papers like "The cost of virtualization" by

Ulrich Drepper and other materials like "AMD64 Architecture Programmer's Manual Vol 2" describe the implementation of virtual memory and its associated costs during translation; but no study has tried to develop a model that considers these costs for algorithmic analysis prior to the recently proposed VAT model.

Keeping this trend in mind, it is not surprising that there has been little related work that carefully verifies the experimental results and strengthens the new model that accounts for these costs. Our contribution, experimentally examining and comparing the proposed model to the older model, will fill this gap.

4 Methods

Three components are necessary to determine which model of computation yields a more accurate analysis of the performance of algorithms running on real hardware. First, there must be implementations of algorithms running on real hardware. Running time benchmarks establish a ground truth for doing comparisons. Second, the algorithms must be analyzed according to the new VAT model. And third, the algorithms must be analyzed using the RAM model. The authors selected five well-studied algorithms which have been previously analyzed with the RAM model.

4.1 Benchmarking

To generate a standard for comparison, the authors selected a set of five algorithms:Binary Search, Heapsort, Insertionsort, Quicksort, and Permute. These five are well understood, have a variety of running times, and should be straightforward enough to analyze on the new model.

The algorithms were coded in C++ and compiled with GCC and Make. They were run on a VirtualBox virtual machine, with Debian GNU/Linux for a 32 bit x86 processor. The virtual machine was hosted by 64 bit Windows 7 on a 2.0 GHz Core i7 processor. The physical machine has 8 GiB of memory and four processor cores, of which 2.5 GiB and 2 cores were allocated to the virtual machine. Find a link to the source code in the appendix.

The benchmark program ran each procedure 10 times on each input size. The smallest input to each procedure was an integer array of length 1. The input doubled in size after every tenth run. The maximum size input array to Binary Search was 536870912 32 bit integers, the 2 GiB size reaching the limits of the machine's memory capacity. The maximum size input array to

Insertionsort was 1048576 32 bit integers, the execution time on any larger input becoming excessive and overflowing the capacity of the timer. The other three procedures had maximum size input arrays of 268435456 32 bit integers, each array taking up 1 GiB of system memory.

The benchmark program timed each run of each procedure on each size of input. The raw results are available at the link provided in the appendix. The next section of the paper provides a summary presentation.

4.2 VAT Model Analysis

We learned how to use the VAT model. We overview how to do that here, but point you to the J&M article to explain it. We used the VAT model to analyze our set of algorithms.

5 Results

We need actual results to write this section, but it will be divided into three subsections.

5.1 Benchmarks

This subsection will have some text explaining several charts that show empirically discovered running times of our various procedures.

5.2 RAM

This subsection will be fairly brief, with a table listing the runtimes every one 'knows' these reference algorithms have.

5.3 VAT

This subsection will have a little more to it than the RAM section. There will be a table telling about the calculated running times of our algorithms. There might also be some text describing some of the difficulties or nuances of doing the analyses with the new model.

6 Discussion

This is where we discuss results.

6.1 RAM vs Reality

Here is where we will compare the benchmark results with the standard runtimes. This will need a table to compare the two algorithm-by-algorithm.

6.2 VAT vs Reality

Here is where we will compare the benchmark results with the new model's runtimes. This will need a table to compare the two algorithm-by-algorithm.

6.3 VAT vs RAM

The big finale, old vs new, here is where we will declare a winner between RAM and VAT. This will likewise need a table. Really all three tables could be combined into one good chart for the whole section.

7 Future Work

There still remains work to be done making our mathematical models better reflect reality.

Edge case: server/cluster/cloud

8 Conclusions

We did the experiment, because reasons. We found a result, and thought it was pretty nifty.

9 References

The final paper will include our references duly cited using Bibtex. I have to remember how to compile them properly. But from our proposal

1. N. Rahman, "Algorithms for harware caches and TLB," *Lecture Notes in Computer Science*, vol. 2625, pp. 171–192, 2003.

2. Advanced Micro Devices, *AMD64 Architecture Programmer's Manual*, vol. 2: System Programming. 2010.

3. J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. New York: Elsevier, 2012.

4. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, Massachusettes: The MIT Press, 2009.

5. T. Jurkiewicz and K. Mehlhorn, "The cost of address translation," presented at the ALENEX13, New Orleans, Louisiana, 2013, pp. 148–163.

6. U. Drepper, "The Cost of Virtualization," ACM Queue, vol. 6, no. 1, pp. 28–35, 2008.

7. R. K. Ahuja and J. B. Orlin, "Use of Representative Operation Counts in Computational Testing of Algorithms," *INFORMS Journal on Computing*, vol. 8, no. 3, pp. 318–330, 1996.

8. U. Drepper, "What every programmer should know about memory, Part 1 [LWN.net]," 21-Sep-2007. [Online]. Available: http://lwn.net/Articles/250967/. [Accessed: 27-Feb-2013].

Appendix

We will include an Appendix for our source code and data. We might include our efforts at employing VAT analysis. For now, all our materials can be found online at:

https://github.com/dslachut/adv-algo-project