Application-Specific Processors for Web-Browsing: An Exploration and Evaluation of the Design Space

Gabriel Yessin, Lubomir Riha, Tarek El-Ghazawi
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Electrical & Computer Engineering, George Washington University, USA
{gyessin, lubomir, tarek}@gwu.edu

David Mayhew
Advanced Micro Devices, Inc.
USA
david.mayhew@amd.com

— The current trend in computing has been to add more and more to the CPU; especially bigger and bigger caches and more cache levels. Based on these observations, we sought to see if bigger is always better. We test this by performing an architectural design space exploration of various cache and frequency configurations for ARM processors. Analyzing the data, we made the surprising discovery that bigger is not always better and we should in fact be taking a step back in the architectural evolutionary roadmap for some applications.

In this study, we performed an analysis of the performance of web-browsers versus the architectural configuration and related it to end-user satisfaction. In the end, we were able to determine that a scaled back modern core would not only be sufficient, but improve the performance of the web-browser.

In doing this, we have also developed GW-GEM5 a set of tools for the creation, monitoring and analysis of concurrent gem5 simulations on computer clusters for use in design space parameter studies.

—Simulation, performance, application specific, web browser, design space exploration, ARM, Android, gem5

I. INTRODUCTION

As we've moved into and adapted to the digital age, email, social media, ecommerce, and the internet as a whole have become integral parts of our day-to-day lives and a never ending source of knowledge, but also of frustrations. The most used application to this end is the computer web-browser [1]. Unfortunately, it has been shown that average webpage load time experienced by users of the top 2000 websites was 10.0 seconds (median 8.4 seconds), and a recent research study found that the average computer user is often unwilling to wait for more than three seconds for a web page to load, with ~57% of users abandoning a webpage before the 4 second mark [2]. Clearly, there is a disparity between user preferences and reality, a disparity which causes frustrations for users and could cost emerging E-businesses new customers and therefore money [3]. While not possible to redesign all websites, it is possible to tailor the architecture of the new slew of tablets and smartphone devices to cater to this gap by designing them to load webpages faster.

There has been a recent trend in computing with movement towards heterogeneous multicores, or more specifically, weakly heterogeneous multicores [4]. They are weakly heterogeneous in that they are identical in ISA and most major microarchitectural features, but vary in some key features. A key example of this architecture is NVIDIA's Tegra 3 and upcoming Tegra 4's variable SMP architecture; their so-called *4-PLUS-1*architecture which makes use of four high-power cores and a separate low-power companion core. The companion core is used to save on power when the system does not require the power of all 4 cores, such as for displaying already-rendered web-pages. The companion core is nearly identical to the other cores except that it runs at a much lower frequency and is made using a special low-power process. The net result is performance equivalent to the high-power quad cores, but with less net power consumption [5] [6].

There has been a good deal of research performed using simulators, including design space explorations [4]; but up until now previous work involving design space exploration has only looked into the effects of varying one parameter, was limited to the running of synthetic benchmarks (which are of little importance to users and user-satisfaction) and/or testing out one new architectural parameter [7] [8]. This shows a significant lack in research with regards to leveraging the full capabilities of performance modeling in these simulators. This is especially true with regards to gem5 [9], the simulator used in this study, which has proven to be extremely accurate [10].

It is due to these observations that we decided to conduct this exploration of the design space for processors in order to determine and recommend what architectures chip makers should look into for designing a browser application-specific core for inclusion in a set of weakly heterogeneous multicores.

As of submitting this paper, we know of no user satisfaction-oriented design space exploration parameter studies that make use of highly accurate architectural simulators.

The structure of the paper is as follows: Section II discusses the methodology used for the generation of simulations and for the collection and analysis of their output data; in section III we show the runtime results collected for the various architectures as well as an explanation for the variations in runtimes, and in section IV we conclude by making design recommendations to chipmakers for use in a browser-oriented core.

II. METHODOLOGY

We utilize the simulator gem5 [9], as it allows us to test different hardware configurations at minimal cost, it provides a large amount of useful statistical data for run-time analysis, and because it provides a full-system simulation which can be used for a true performance assessment [10]. Bbench was chosen because it is a fully self-contained web-rendering benchmark that represents many of the popular websites existing today. Of the sites in Bbench, we chose to run Amazon, Craigslist, eBay,

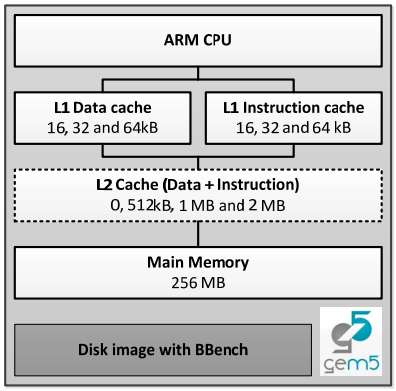

Fig. 1. Datapath configuration when there is an L2 cache.

Google, MSN, and Twitter, which are respectively number 8, 52, 16, 1, 19, and 10 on Alexa.com's list of the world's most popular websites and because they represent a good cross-section of the type of websites available on the web – E-commerce, search engines, email, social media, and etc.

Our simulations were run on the ARM CPU architecture provided by gem5 using Android 2.3 (Gingerbread) and the native browser provided therein. ARM was chosen due to its 95% dominance in the smartphone market [11] and Android 2.3 because it is currently being run on 45.6% [12] of all Android phones, and Android currently encompasses 75% of all smartphones worldwide [13]. There are of course, many microarchitectural features which can be varied, but we focused on core frequency, L1 cache size, and L2 cache size. Frequency was chosen because it is strongly correlated to the relative processing power of a core. L1 and L2 caches were chosen because web browsing requires significant amounts of data to be carried from memory to the CPU, making the memory hierarchy key to performance, as per the suggestions in the study of Kim et al [14]. All configurations were chosen to be representative of common parameters in devices existing today, since that is what chipmakers can most easily and cheaply produce. Overall we collected 288 data points representing the full permutations of the 6 webpages from Bbench, three L1 cache sizes (32kB, 64kB, 128kB), four L2 cache sizes (none, 512 kB, 1024kB, 2048kB), and four core frequencies (500MHz, 750MHz, 1.0GHz and 1.5GHz). The datapath configuration used with L2 caches can be seen in Fig. 1; systems without an L2 cache are identical except that the L1 caches go directly to main memory. It should be noted that L1 cache size was split equally between an L1 instruction cache and an L1 data cache, as can be seen in the figure.

From the collected data points, we sought to beat the 8.4 second median load time discussed earlier by determining which architectures were able to render all six websites in less than 8.1 seconds each, assuming an average consumer's network (bandwidth 5 Mbps down / 1Mbmp up, RTT 28ms). From this we were able to reduce the design space from 36 architectural possibilities to 10 "acceptable" architectures and from there define a singular optimal architecture.

Simulation was carried out using one of gem5's ARM CPU models, specifically the 'detailed' model which models a modern, Out-of-Order processor [15]. Bbench was modified to run one web page per simulation to efficiently utilize parallel environment of our computer cluster, which has hundreds of processors. Each web page was loaded twice so that the effects of both hardware and browser caching could be better understood, the first loading being referred to as the "cold render" and the second as the "warm render." This led to 6 websites being run with 48 configurations, resulting in 288 simulations overall.

For each website, a delay was then added to simulate latency due to the transfer of the website across a real-world network. For this, we used WebPagetest.org, an open-source webpage rendering test and analysis suite maintained by Google [16], to determine what typical network utilization and performance was to be expected for the websites used, so that the run-time reported is the actual expected time it would take from a user clicking on a link or launching a web-browser until the time the web-page is fully loaded and readable. For our testing, a cable network connection (5 Mpbs down / 1 Mbps 28ms RTT) was chosen, as defined by WebPagetest.org. This configuration was chosen because according to NetIndex.com, a results aggregator for speedtest.net, a popular internet connection speed test tool, the average connection speed worldwide was 13.09 Mbps over the past 30 days when the mean distance between the client and the server was less than 300 miles. It should be noted, that while we estimate realistic networks delays and transfer times, we employed a simplistic model which attempts to model the intricate dynamics between network loading and browser rendering, since elements can be loaded while others have already begun rendering. Therefore they should be taken as tools used to estimate feasible total loading times, and not absolute truths.

Next, results were analyzed to determine which architectural configurations were able to run all six websites in the allotted time for a real network, respectively

To determine which parameter causes the biggest performance gain for Bbench websites, we also analyzed the output by keeping all but one parameter constant, measuring the percent decrease in run-time per website, per configuration, then averaging across all websites and then all parameter sets and then determining which parameter overall, when varied, cause the largest percent decrease in runtime for the benchmark (see TABLE I).

Finally, after further analysis, we are able to make some interesting observations regarding the performance of the benchmark and the size of the caches.

TABLE I. AVERAGE PERCENT CHANGE IN RENDER TIME FOR EACH PARAMETER

Parameter	Average Speedup	Standard Deviation
Core Frequency	23.98%	8.65%
L1 Cache Size	5.43%	2.45%
L2 Cache Size	10.69%	6.86%

III. RESULTS AND ANALYSIS

The results we obtained were very instructive. As there are too many design points to show them all here, two representative examples can be seen in Fig. 2 and Fig. 3 which show the render times (excluding network load times) for Google and Amazon which are a small, simple website with a few elements and a large, complex site with many elements, respectively. Labels along the y-axis are in the format "Site-ID:Frequency:L1-Size:L2-Size;" for example "G:1.50G:128K:2.0M" represents Google run on a 1.50 GHz core with a 128kB L1 cache (64kB data and 64kB instruction). This leads to some interesting performance patterns. In order to save space, only two frequency configurations are shown, but the pattern of results is roughly the same for all frequencies. As can be seen with Google, being so small, speeds up greatly during its warm run, whereas Amazon experiences little to no improvement for its warm render. While interesting, warm runs

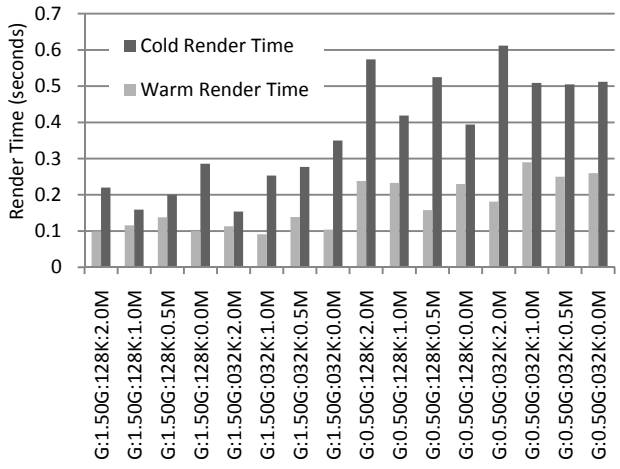

Fig. 2. Render times per architectural configuration for a 'small' webpage. Y-axis label format: "Site-ID:Frequency:L1-Size:L2-Size."

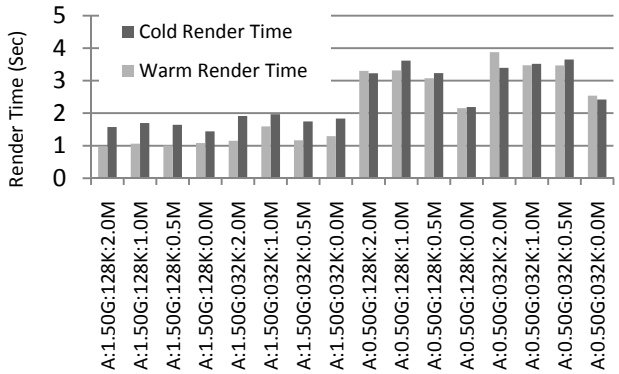

Fig. 3. Render times per architectural configuration for a 'large' webpage.

are of little importance to users and not the direct focus of this study.

Since the parameters are varied in the order of L2 cache size, L1 Cache size, core frequency and then website, it can be seen that the greatest net effect on the core performance is the frequency parameter. This is further seen in TABLE , where we see that going from 500MHz to 1.5 GHz speeds up the load time by 23.98% on average, making it a key performance indicator.

In the case of Google, its worst-case render time was 612ms and best-case was only 154ms, which represents only 7.7% and 1.9% of the 8 seconds page load time goal, respectively, whereas Twitter had a worst-case of 5.57 seconds and a best-case of 2.57 seconds, which are 69.6% and 32.1% of the goal. Clearly, while all sites can benefit from optimization it is more critical for some, especially larger ones which contain many elements, and therefore necessarily have longer load times.

Fig. 4 shows a cube representing the three-dimensional design space in which we were working. Squares represent failed architectures (unable to load in the allotted time). Stars represent successful architectures and the big, white star represents an optimal configuration for chipmakers, as it minimizes both frequency and cache size [17].

It may seem odd that 64kB L1 caches generally fail when any L2 cache is present, even at 1.5GHz, but upon analysis of the output statistics files from the gem5 simulations, we determined that this is due to a "Goldilocks" effect of the mismatch between L1 and L2 size being just right, so as to cause a larger relative number of cache misses. In fact, the existence of high L2 cache miss rate is the main cause of such high render times for all architectures with an L2 cache. This can be observed in Fig. 5. L2 cache miss rates for data were found to be as high as 71.5% in some cases (ave 49.8%, stdev 10.8%). Instructions cached far better, with a 9.9% max miss rate, (7.8% ave, 0.8% stdev).

The reason behind this is that webpages, data-wise, do not cache well at all because they are essential a stream of new data, where little to none of it is reused. On the other hand, instructions cached reasonably well. Since L1 caches are

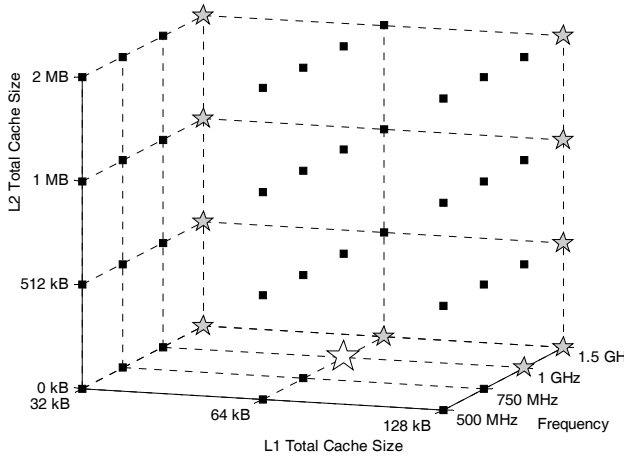

Fig. 4. 3-D Visualization of the results

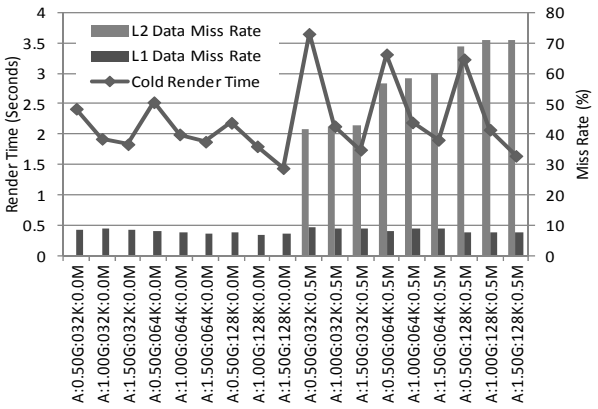

Fig. 5. Comparison of render times to L1and L2 data cache misses for Amazon.

distinct, data does not affect the caching of instructions; but the L2 cache is unified and this allows the browser to pollute the cache with data, affecting the fast caching of instructions. It also adds additional delay to the data path, and with L2 miss rates greater than 70%, greatly adds to the total run time. It is for this reason that we recommend "upgrading" core designs by removing the L2 cache altogether.

IV. Conclusion

In conclusion, we have gleaned some very interesting details about the design space for web browser applications. We found that core frequency is still the key parameter in terms of determining performance, as one would expect. It was also proven that if the number of levels in the memory hierarchy is kept the same, but the cache sizes are increased, performance does in fact improve, especially for warm render times for smaller websites.

Most interestingly, we found that web browsers have poor data caching, since their memory accesses are essentially a stream of new data with little reuse, L2 caches are actually a large detriment to the performance of web browsers. We recommend its removal for a web-browser-oriented core design. From this, we were able to recommend a unique architecture which is performance optimal. We found that a lower-frequency core with an appropriately-sized L1 cache and no L2 cache can outperform a higher frequency chip with large, 2-level caches. We have also developed a rich set of tools, referred to as GW-GEM, for working with gem5 design space explorations and parameter studies on very large clusters.

Acknowledgment

This work was supported in part by Advanced Micro Devices, Inc. and the I/UCRC Program of the National Science Foundation under Grant Nos. IIP-0706352 and IIP-1161014.

References

[1] Bureau of Labor Statistics, U.S. Department of Labor, "Most common uses for computers at work," 2 September 2005. [Online]. Available: http://www.bls.gov/opub/ted/2005/aug/wk5/art05.htm. [Accessed 15 February 2013].

[2] C. Rheem, "Consumer Response to Travel Site Performance," PhoCusWright, 2010.

[3] Strangeloop Networks, Inc., "2012 Annual State of the Union: E-Commerce Page Speed and Website Performance," January 2012. [Online]. Available: http://www.strangeloopnetworks.com/assets/PDF/downloads/2012-Annual-State-of-the-Union-Report.pdf. [Accessed 15 February 2013].

[4] E. Tomusk and M. O'Boyle, "Weak Heterogeneity as a way of Adapting Multicores to Real," in *3rd International Workshop on Adaptive Self-tuning Computing Systems*, Edinburgh, 2013.

[5] NVIDIA Corporation, "Variable SMP (4-PLUS-1 (TM)) - A Multi-Core CPU Architecture for Low Power and High Performance," NVIDIA Corporation, 2011.

[6] A. Kingsley-Hughes, "Nvidia Tegra 4 processor details leaked | ZDNet," 18 December 2018. [Online]. Available: http://www.zdnet.com/nvidia-tegra-4-processor-details-leaked-7000008961/. [Accessed 15 February 2013].

[7] A. Basu, M. D. Hill and M. M. Switch, "Reducing Memory Reference Energy with Opportunistic Virtual Caching," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, Portland, 2012.

[8] D. Zoni, S. Corbetta and W. Fornaciari, "HANDS: Heterogeneous Architectures and Networks-on-Chip Design and Simulation," in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, Redondo Beach, 2012.

[9] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News,* vol. 39, pp. 1-7, May 2011.

[10] A. Butko, R. Garibotti, L. Ost and G. Sassatelli, "Accuracy evaluation of GEM5 simulator system," in *7th International Workshop on Reconfigurable Communication-centric Systems-on-Chip*, York, 2012.

[11] T. P. Morgan, "ARM Holdings eager for PC and server expansion," The Register, 1 February 2011. [Online]. Available: http://www.theregister.co.uk/2011/02/01/arm_holdings_q4_2010_numbers/. [Accessed 15 February 2013].

[12] Android Developers, "Dashboards," 4 Febrary 2013. [Online]. Available: http://developer.android.com/about/dashboards/index.html. [Accessed 15 February 2013].

[13] R. Llamas, K. Restivo and M. Shirer, "Android Marks Fourth Anniversary Since Launch with 75.0% Market Share in Third Quarter, According to IDC," International Data Corporation, 1 November 2012. [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS23771812#.UR3XBaXBOSo. [Accessed 15 February 2013].

[14] H. Kim, N. Agrawal and C. Ungureanu, "Revisiting Storage for Smartphones," in *Proceedings of the 10th USENIX conference on File and Storage Technologies*, Berkeley, 2012.

[15] A. Saidi, "ARM Implementation - gem5," gem5, 10 May 2011. [Online]. Available: http://www.m5sim.org/ARM_Implementation. [Accessed 15 February 2013].

[16] Google, "WebPagetest - About," [Online]. Available: http://www.webpagetest.org/about. [Accessed 15 February 2013].

[17] A. Das, B. Ozisikyilmaz, S. Ozdemir, G. Memik, J. Zambreno and A. Choudhary, "Evaluating the Effects of Cache Redundancy on Profit," in *41st IEEE/ACM International Symposium on Microarchitecture*, Lake Como, 2008.