MARVELL

WHITE PAPER

Memory Characterization to Analyze and Predict Multimedia Performance and Power in an Application Processor

Yu Bai
Staff Engineer, APSE
Marvell

November 2011

Introduction:

Nowadays, a memory system can become the main bottleneck for system performance because the speed gap between the ultra-fast central processing unit (CPU) core and the relatively slow memory widens. Low power is another important design consideration, particularly for the growing number of battery-supplied devices. Lower power translates into longer battery life time and extended usability. Memory typically is a significant power contributor in application processors for popular use cases, and memory power tends to increase quickly with its complicated design and large memory sizes and hierarchies. Therefore, reducing memory power presents significant battery life benefits. To better understand various applications' intrinsic behaviors, it is essential to investigate memory characteristics and build a memory model to determine if the application involves intensive memory accesses—and even help predict the application's performance.

This white paper presents a simple and affordable way to dynamically characterize an application's computational and memory composition with acceptable accuracy.

Approaches to Memory Characterization

Without memory involved, the CPU utilization should scale linearly with the CPU core frequency, and the application cost (defined as the multiplication of the CPU utilization and the CPU frequency) should remain constant. But with memory accesses, the CPU utilization would not scale linearly with the core frequency any more. At the higher frequency point, performance tends to be more blocked by the memory since the CPU spends more CPU cycles in waiting for memory response[1]. In this sense, applications can be characterized into two types: computational bound and memory bound.

Next, there are three different approaches to characterize memory composition and help determine the application's CPU utilization. The hardware performance events are collected by probing Performance Monitoring Unit (PMU). Thus, Marvell's methodology applies to any system with the PMU hardware support.

1. Overall Data Cache Miss Rate—Intuitively, higher data cache miss rates imply heavier memory traffic. To obtain data cache miss rate, we need to monitor the total number of accesses and misses in the first level data cache and second level data cache if presented.

2. Main Memory Access Rate—Occupancy rate of main memory controller is a direct indicator of memory utilization. To obtain main memory access rate, two PMU events must be collected: the total number of cycles that memory controller is occupied and the total number of cycles in the monitoring window.

3. Data Stall Rate—Pipeline stalls are primarily due to data dependencies, while the reason of data unavailability is caused by far slower memory accesses compared to the CPU speed. So the number of pipeline stall occurrence reflects memory traffic situation. Moreover, this indicator implies memory access's importance. Not every memory access is critical to the ultimate performance, thus it is quite useful to keep track of memory accesses that introduce data dependencies with performance impact. In this approach, the event occurrence is monitored where the pipeline is stalled due to the data dependency. In addition, the total number of cycles must be recorded for data stall rate calculation in each window.

[1] Here we assume that the memory frequency does not change at different CPU frequencies.

These different approaches reflect memory characterization from different perspectives. We might use single approach or combinations for more effective performance analyses and more accurate prediction with reasonable overhead.

In our tests, we use Marvell's application processor running the Linux-based operating system with the QVGA LCD display as our test platform. It includes two level caches. In this study, we focused on MP3, AAC+, and H.264 decoders.

Figure 1 shows the comparative results of the three different approaches. Each graph includes two curves: one with the second level (L2) cache enabled and one with it disabled, at three CPU frequencies.

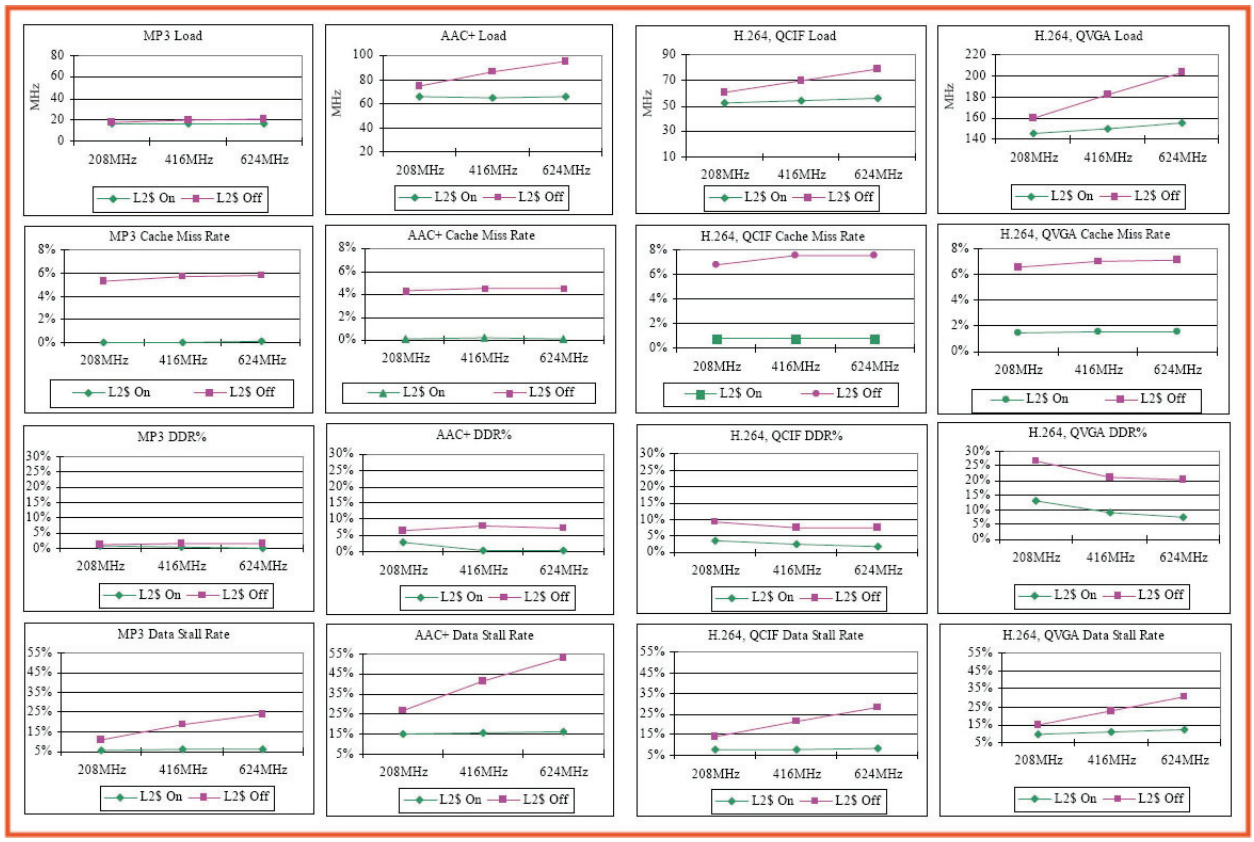


Figure 1: Three approaches of memory characterization

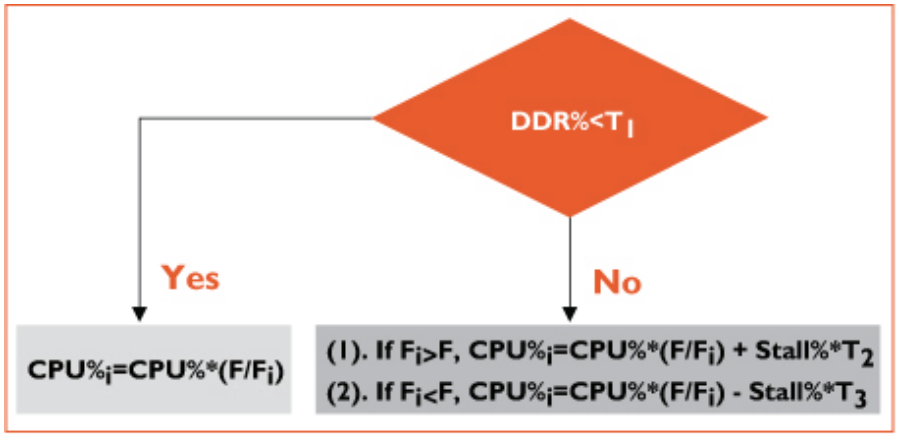
With no heavy memory accesses, the CPU utilization is approximately linear to the core frequency, thus application cost (shown in load curve) should be flat with the core frequency. MP3 and AAC+ decoders are good examples when L2 cache is on because MP3 and AAC+ decoders only introduce a small amount of memory accesses, and most of them are caught by L2 cache. In the cases of no L2 cache, application loads increase with the core frequency. We also found that cache miss rates do not change significantly with the CPU frequency for either L2 cache on case or L2 cache off case, which implies that overall data cache miss rate is an ineffective metric to indicate memory access situation.

Intuitively, the memory access rate contains the similar amount of system information as the cache miss rate because cache misses directly introduce memory accesses. However, our results indicate that this is not true. For example, H.264 QCIF decoder shows a similar cache miss rate trend as H.264

QVGA decoder, but H.264 QVGA decoder spends much larger percentage of the application time in the memory accesses than H.264 QCIF decoder. This again proves that solely monitoring cache miss rate is ineffective. Cache miss rates do not necessarily introduce performance drop if the total amount of cache accesses is trivial. MP3 decoder without L2 cache is a good example of this. On the other hand, a typical cache miss rate may introduce a great amount of memory accesses. H.264 QVGA decoder shows this trend.

Some memory accesses may be on the performance critical path, while others may not. Neither cache miss rate, the total amount of cache accesses, nor main memory access rate can differentiate whether or not the memory accesses are on the critical path. Fortunately, we find that data stall rate is a good indicator. Apparently, data stall rate curves are synchronous with application cost for all applications except MP3 decoder. Data stall rate is the best metric to predict application loads from our experiments. For MP3 decoder, memory access rate is extremely low. Therefore, even if there are some critical memory accesses among overall very few memory accesses, performance impact is still trivial.

Figure 2 presents an algorithm to predict the CPU utilization based on memory characterization. It first checks if the memory access rate is lower than the predefined threshold T_1. If it holds, we predict that the CPU utilization is inversely proportional to the CPU frequency; otherwise, the CPU utilization is predicted in two steps: (1) being inversely proportional to the frequencies and (2) adjusting according to the data stall rate. In the second step, we introduce two more thresholds: T_2 and T_3. To implement this algorithm, we must keep track of both main memory access rate and data stall rate. Therefore, at most three PMU events must be monitored: (i) the total number of cycles that external memory controller is occupied; (ii) the total number of occurrences that the pipeline is stalled due to data dependency; and (iii) the total number of cycles during the monitoring window. (i)/(iii) gives DDR% and (ii)/(iii) gives Stall%. This algorithm is easy to incorporate into the power management framework.



* F is the CPU frequency for now
* F_i is the CPU frequency that the system chooses to switch to in the future
* CPU% is the CPU utilization measured for now
* $CPU\%_i$ is the CPU utilization that needs to be predicted if the system runs at the frequency F_i in the future
* DDR% denotes the main memory access rate for now
* Stall% is the data stall for now
* $T1$, $T2$, and $T3$ are three thresholds depending on the applications and need to be defined and adjusted by experiments

Figure 2: A simple algorithm of performance prediction

Conclusion

In summary, the CPU utilization can be predicted to be inversely proportional to the CPU frequency if overall memory access rate is negligible. Data stall rate should be used together for predicting the CPU utilization when the memory access rate becomes non-trivial. This white paper is based on the conference paper [1] that proposes an algorithm to improve the performance prediction by studying three memory indicators to help characterize memory composition. More future work may include experiments with additional applications and optimize our algorithm by designing dynamically adaptive thresholds based on user inputs and/or more system feedbacks.

As a leader in the development of storage, communications and consumer silicon solutions, Marvell applies high-performance, low-power technology across its diverse product portfolio. To learn more about Marvell and its solutions, visit www.marvell.com or contact your Marvell sales representative.

REFERENCE

[1] Yu Bai, Priya Vaidya, "Memory characterization to analyze and predict multimedia performance and power in embedded systems", *ICASSP 2009*

About the Author:

Yu Bai
Staff Engineer, APSE, Marvell

Yu Bai is a Staff Engineer at Marvell Semiconductor, Inc., where she works on power analysis and power modeling for the company's application processors. She joined Marvell through the company's acquisition of Intel's Xscale™ product line in 2006. Yu received her master and doctorate degrees in Electrical Science and Computer Engineering from Brown University. Her graduate research focused on high performance and low power computer architecture design. Yu has submitted three U.S. patent disclosures and has published more than 10 journal and international conference papers on power management and power optimization.

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
www.marvell.com

Characterization-002_whitepaper 2/2012