Journal of Computer Science

Original Research Paper

A Novel Method to Predict Processor Performance by Modeling Different Architecture Parameters

Joseph Issa

Department of Electrical and Computer Engineering, Notre Dame University, Zouk Mosbeh, Lebanon

Article History
Received: 31-12-2019
Revised: 24-03-2020
Accepted: 20-04-2020

Email: joseph.issa@ndu.edu.lb

Abstract: Predicting processor throughput and performance is one of the essential aspects of computer architecture. It is crucial to model processor performance behavior for future architectures based on the existing data set. Modeling processor performance for a given workload enables architects to enhance processor features to meet specific performance targets for a given benchmark. Developing an estimation method to predict performance using one micro-architecture parameter is limited, given the need to model multiple parameters simultaneously. In this paper, we propose a novel performance prediction method for SPEC CPU 2006 and HDxPRT 2014 benchmarks based on a combination of measured and estimated performance data. The performance project model predicts processor performance while altering multiple microarchitecture parameters simultaneously such as memory speed, number of cores and the core frequency. We also present a detailed timing analysis for each processor sub-component. The model is verified to project performance with less than 5% error margin between projected and measured baseline.

Keywords: Performance Analysis, Performance Estimations, Processor Architecture, Microarchitecture, Computational Modeling

Introduction

Accessing processor performance is critical for the effectiveness of the entire system combining both hardware and software. The task of performance estimation is challenging, given that performance depends on different software and hardware variables. Given the complexity of this task, it is still essential to predict processor performance for a given benchmark and be able to change the micro-architecture parameters so that to estimate future performance numbers. The first task to achieve this is to understand what determines the processor performance. The two apparent settings in processor performance are throughput and latency. Unlike transaction-processing workloads, some workloads are incredibly diverse in their use and stress on different server sub-systems. Some are Central Processing Unit (CPU)-bound and others are strongly memory-bound. There is a big difference between CPU-bound vs. memory-bound workloads. The most important characteristic affecting the performance of any workload on any system is the number of primary memory transactions it does.

For the CPU-bound workloads, the performance is gated by activity on the processor chip. The critical performance parameters are core frequency, latencies and bandwidth from processor caches. The unimportant parameter is the memory subsystem. Usually, systems are cheaper to build for CPU-bound workloads. The memory-bound workloads are the opposite of CPU-bounded workloads. The performance is mainly determined by the off-chip events, primarily how many main memory transactions can be completed per unit time. CPU-bound workloads have few main memory transactions and are constrained by core frequency, cache latency/bandwidth, cache design and pipeline. Memory-bound workloads have many principal memory transactions and are limited by memory bandwidth and sometimes by memory latency.

In this paper, we present a novel performance prediction method based on a mathematical regression approach, which takes as inputs different processor microarchitecture parameters simultaneously to predict performance for SPEC CPU (2006) and HDxPRT benchmarks. The measured baseline is a Nehalem processor in which the measured data is used for the model. We propose a method to develop a projection model that utilizes measured and mathematical methods using regression data analysis and Amdahl’s Law. The measured data assures that we are capturing the proper effect of the workload behavior and its architecture capabilities. The performance contributions from the processor and memory can be mathematically determined using Amdahl's Law and carefully crafted through experiments. We can look at the impact of different architectural features for CPU and Memory by studying them individually. The data regression techniques are used to find mathematical relationships in the data that can be used in developing extendable models to predict performance for CPU configurations which cannot be measured.

The method presented in this paper is analytical, which means it does not require simulation data or sampling traces for simulation. The simulation approach requires developing a software-based simulator and capturing significant traces based on Cycles-Per-Instruction (CPI) and other architecture constraints that resembles the entire benchmark. The paper is structured as follows: In Section II, we discuss the motivation behind developing the proposed analytical model. In Section III, we discuss SPEC CPU 2006 and HDxPRT benchmarks. In section IV we review previous work in which we compare our analytical model to other published modules, which estimate processor performance using a systematic approach. In section V we present the performance and sensitivity analysis for SPEC CPU 2006 and HDxPRT using Nehalem processor. In section VI we present the proposed performance projection model supported by experimental results and we conclude in Section VII.

Motivation

Modeling processor performance is essential for processor engineers and designers using an analytical approach as compared to a simulation approach. The important feature is to evaluate different hardware configurations and predict the performance for a benchmark without using a trace-based simulator. This approach will help processor architects in designing and fine-tuning different architecture parameters for future processors. The model can give an estimate for performance indicators for SPEC CPU 2006 and HDxPRT workloads by selecting the desired processor architecture parameters. For example, what is the change in performance when the number of cores increases? This can provide processor engineers a leading edge to estimate performance without having to measure the benchmark on a processor that is not yet developed. The model also enables evaluating performance for different benchmarks projected performance score (Unit less performance metric) for different processors, given they are within the same family architecture. The score variable used in this paper is inversely proportional to the time domain for performance measurements.

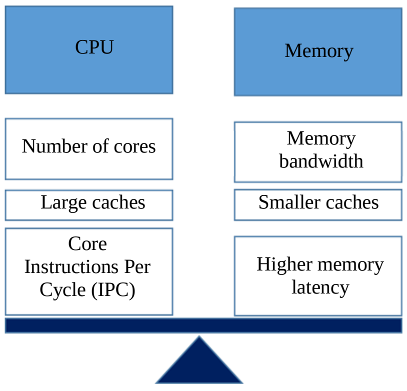


Fig. 1: CPU and Memory architecture parameters

Usually, we expect an increase in benchmark performance for future processors, given that more technology, hardware features and capabilities are added throughout the processor roadmap. Some of the essential elements are an increase in the number of cores, an increase in memory speed and memory capacity, or an increase in the core frequency itself. In the model proposed in this paper, we covered all the critical features that will enable developers to get an early projected performance number for SPEC CPU 2006 and HDxPRT fora future processor configuration. A similar approach can be developed for a different workload. We chose SPEC CPU 2006 and HDxPRT because they are CPU intensive (compute-bound) workload; other workloads can be more memory intensive (memory-bound). In order to develop a new module for a different workload, a new set of processor sensitivity analysis is required. Measurement provides the expected performance of the workload on a given set of architecture settings. The benchmark characteristics consist of a collection of measured data, defining a set of architecture parameters of interest and statistical output for the architecture parameters. The concept in Amdahl's Law allows us to determine the contribution of the CPU and Memory to the overall performance and how specific elements change the component contribution. We do this by running experiments where we keep one side constant and vary parameters, on the other hand, as shown in Fig. 1.

Benchmarks Overview

The SPEC CPU2006 was released by the Standard Performance Evaluation Corporation (SPEC). It's a standardized processor and memory benchmark, which is what we need for our performance projection model. It is designed to stress the CPU and memory subsystems and provides a comparative measure of compute-intensive performance by measuring integer and floating-point performance. This benchmark is widely used in the industry by several computers and processor manufacturers to test their processor performance. It's also used for comparing the performance of different processors by different vendors to decide what processor to purchase based on performance and other factors. It is also used to compare the high-end processor versus the low-end processor's performance; this is used to determine the cost of each processor segment. There are two metrics to measure processor performance, the first metric is time and the second metric is throughput. Time determines the execution time, which is how fast a task is completed per unit time. Another parameter is the throughput, which is to measure how the amount of computation achieved per unit time. In SPEC CPU 2006, we used throughput as performance metrics and also execution time. SPEC CPU 2006 is categorized as a compute-intensive workload, which means it's a compute-bound workload or bounded by the number of cores. Every workload belongs to these two categories, a compute-bound workload or memory-bound workload or in between. For compute-bound workloads mean that the workload is only sensitive to the number of cores and the core frequency. This also means that if memory bandwidth and capacity increases, the performance will not increase. Memory-bound workloads mean that the workload is bounded to the capacity of memory and memory speed. So, any increase in the number of cores and/or the core frequency will not be translated into an increase in performance even though the computation power increased. Workloads can have different sensitivity; for example, some workloads are sensitive to memory bandwidth and memory speed as compared to being sensitive to the number of cores, core frequency or the total number of threads. The performance contributions coming from the CPU and memory can be mathematically determined using a measured baseline. The impact of performance change from different parts of processors and memory and be analyzed individually. A regression method is used to determine the relationship to performance in order to construct the performance projection model. In this paper, we propose a performance equation as a function of different microarchitecture parameters, which includes the number of cores, CPI, core frequency, memory frequency and memory latency. This enables the processor engineer to change different microarchitecture parameters and estimate the change in processor performance.

HDxPRT scoring benchmark is divided into two sub-categories. The first category consists of creating the HD score, which in turn includes an edit and convert videos from camcorder, edit photos and video from a digital camera and prepare media for portable devices. The second category is the HD video playback, which consists of HD video (1080 p, H.264) and HD video online (1080 p with Flash).

Related Work

Researchers have developed different prediction models to predict processor performance for a given benchmark using an analytical approach instead of a trace-based simulation. The analytical model presented in this paper enables the performance projection of relative performance with a <10% error margin difference between measured and estimated performance scores using the SPEC CPU 2006 and HDxPRT benchmarks.

In our previously published papers, we proposed a performance estimation model using Amdahl's law regression method in (Issa and Figueira, 2010). Amdahl's law is based on the law of diminishing returns, which means increasing the number of processors or the number of cores, do not lead to a proportional increase in the same amount in performance. The definition for Amdahl's law states that the performance improvement gained from implementing a faster mode of execution is limited by the fraction of the time the quicker mode can is used. Amdahl's Law states that a system's overall performance increase is limited by the fraction of the system that cannot take advantage of the enhanced performance. The method published in (Issa and Figueira, 2010) predicts benchmark performance with less than a 10% error margin. The way presented in (Issa and Figueira, 2010) is limited, given that it can only accept only one architecture parameter change at a time to estimate performance for different values of that same parameter. The method requires at least two measured data points to establish a measured baseline and this enables performance estimation for microarchitecture parameters that cannot be measured on the processor under test. Note that the measured baseline and the projected performance must be of the same microarchitecture parameter, for example, the number of cores or the core frequency.

This paper is also a continuation of the work we published in (Issa, 2016) for our initial work on this project. In this paper, we added more elaboration, fine-tuned the regression method and added the timing analysis in the results section to show the breakdown in time between the core time and the memory time for SPEC CPU 2006.

Saavedra and Smith (1996) proposed a method for a given benchmark to characterize the machine performance and the program execution. The paper focuses on determining the execution time of the benchmark. The difference between our method and the method published in (Saavedra and Smith, 1996) is that our approach is more general and can be used for any processor by changing different microarchitecture parameters. Krishnaprasad (2001) presented various ways of using Amdahl's law in a different form. Our method presented in this paper has the same objectives, but we use a regression approach instead.

Hoste *et al*. (2007) presented a method based on computing a set of microarchitecture parameters independent characteristics and weights these independent characteristics resulting in locating the application of interest in benchmark space. The performance prediction is implemented by weighting the performance number of a benchmark in the neighborhood of the application of interest. In our method, we do not apply any weighting mechanism for a given benchmark to predict performance, as this may change and becomes different for a different benchmark.

Phansalkar *et al*. (2007) proposed a simulation-based approach for SPEC CPU 2006 by calculating the CPI for cache and Translation Lookaside Buffer (TLB) misses. The paper concludes that a larger TLB size can reduce the cache and TLB miss rates, which in turn will reduce the CPI and may improve performance.

Jens (1996) proposed a different performance estimation method for the Linpack benchmark based on predicting the runtime using a message-passing approach. Our estimation model approach is different in a way we analyze different processor architecture parameters and developed an empirical formula to predict relative performance.

A significant amount of work has been done (Ganesan *et al*., 2008; Prakash and Peng, 2008) using different performance metrics to analyze and optimize the performance of different workloads. These research papers are highly dependent on microarchitecture parameters that are tight to a specific Instruction Set Architecture (ISA) which makes it bias to a specific architecture. It is used to find performance bottlenecks for different benchmarks.

Khan *et al*. (2012) presented a novel method for cache segmentation replacement technique that works independently from Least Recently Used (LRU) replacement method. The method is tested with different cache sizes for Last Level Cache (LLC) sizes using intensive memory subsets of SPEC CPU 2006. This shows the importance for cache performance modeling for memory intensive subsets of SPEC CPU 2006.

Issa and Figueira (2010) proposed a performance estimation model using Amdahl's Law regression method. The method is limited as it requires changing one microarchitecture parameter such as core frequency or memory frequency while keeping other processor parameters fixed. The technique also requires having a measured baseline with a minimum of three measured data points to enable performance projection using the measured baseline. The approach presented in this paper allows performance prediction by changing several architecture variables simultaneously.

Hoste and Eeckhout (2007) presented different metrics for characterizing benchmarks based on microarchitecture-independent characteristics. It is based on instrumenting program binaries to describe diverse instruction mix, ILP, working set size and branch predictability. This is based on the simulation of ISA traces to module performance.

Baghsorkhi *et al*. (2010) proposed an analytical method to predict the performance of the general-purpose application on a GPU architecture. The technique identified how kernel affects different GPU microarchitecture parameters.

Hong and Kim (2009) presented an analytical model for GPU architecture with an emphasis on memory-level and thread-level parallelism. In our analysis, we analyzed the sensitivity of HDxPRT with respect to different cache sizes and the number of cores.

Sensitivity Analysis

a) SPEC CPU 2006

SPEC CPU 2006 benchmark includes twenty-six different benchmarks executed to stress the processor and memory. The output of the benchmark is one number, which is referred to as the performance score (SPEC rate). It is important to design the right experiment so that the performance data can be analyzed accordingly. The objective of the performance model presented in this paper is to combine all the regression measurements into a single empirical formula to predict performance for a SPEC CPU 2006. This enables us to perform a multivariable regression. It is important to mention that all measured data presented contains a common configuration, which means that all performance data presented is referenced to a normalized measured baseline, which is equal to one ‘1’. The remaining measured data are referenced to this '1', which is known as the normalized measured baseline. The main factor contributing to lower processor performance are summarized as follows:

- A low number of cores
- Small cache size
- Low core Instruction-Per-Cycle (IPC). IPC is usually reduced (lower performance) in case of an increase in cache misses structural hazards, control hazards, or data hazards

There are different memory factors that contribute to lower performance such as, lower memory bandwidth, smaller cache size and high memory latency. All the performance measurements used for sensitivity analysis are based on relative performance with respect to the Intel Nehalem Xeon processor with 8 cores, 2800 MHz core frequency and 400 MHz memory speed. It is implemented by taking the measured data for a given workload and analyze the sensitivity performance curve with respect to one performance parameter (number of cores) while keeping all other parameters fixed.

When a benchmark score is larger for higher performance numbers as shown in Fig. 2, inverting the performance parameters, in this case, it's the number of cores along with the score often provides linear lines in a plot, as shown in Fig. 3.

The model output for the relative score with respect to the number of cores is calculated using regression as = 1/(M*(1/# of cores)+B) where M and B are the regression slope and intercept. Linear relationships help in simplifying the predictive model, but this does not always happen.

Some elements of performance end to be well behaved in producing a linear relationship to performance using this technique. The lists of architecture parameters that work well for SPEC CPU 2006 benchmark are:

- Frequency vs. Score (Fig. 4)
- Core count vs. Score
- Memory Bandwidth vs. Score (Fig. 5)
- Memory Latency vs. Score
- IPC improvements vs. Score

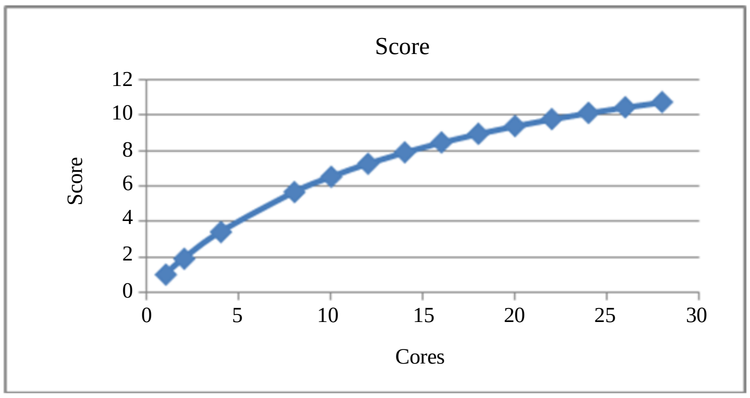


Fig. 2: Score vs. # of cores

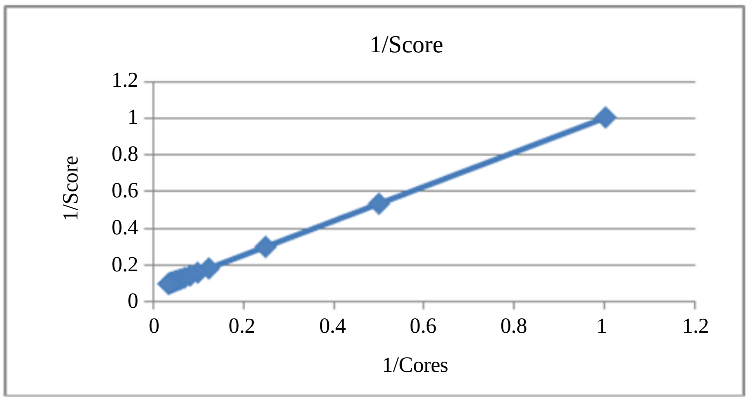


Fig. 3: Linear plot for 1/score vs. 1/cores

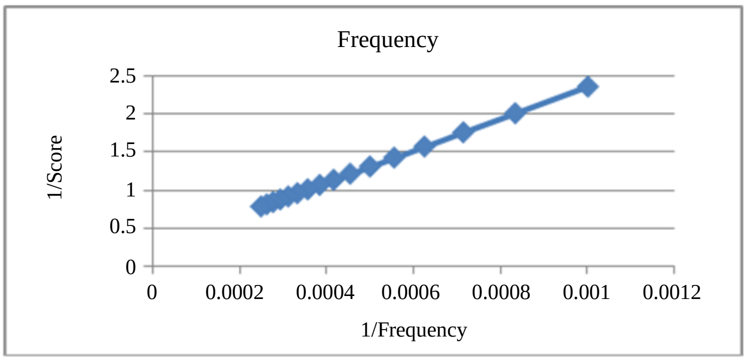


Fig. 4: 1/Score vs. 1/Frequency

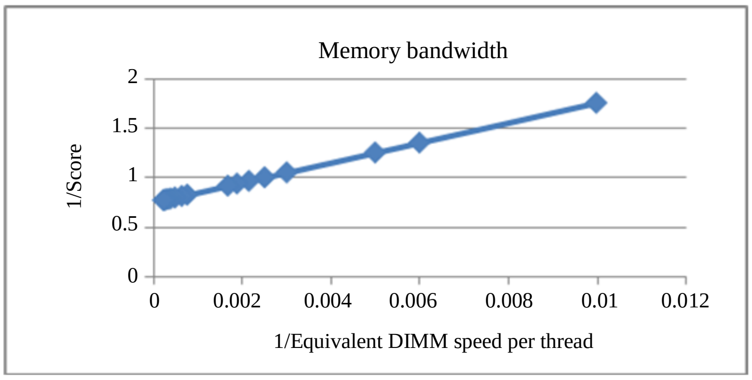


Fig. 5: 1/Score vs. 1/DIMM speed per thread

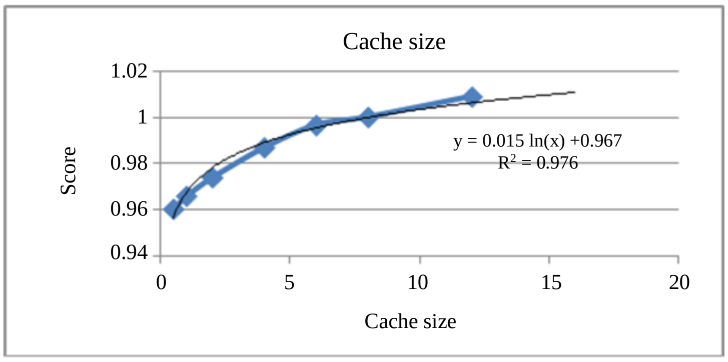


Fig. 6: Cache size vs. performance score

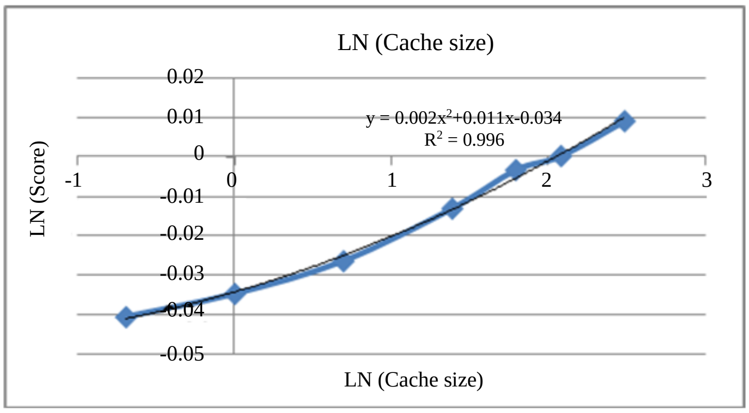


Fig. 7: LN (Cache Size) vs. LN(Score)

Best conditions are with heavily threaded homogeneous workloads and they vary per benchmark.

Some regressions don't work well using this approach. Some relationships are harder to work with. The cache size tends to be one of those. It will shift work from the main memory to the CPU as you increase the work size. We can modify the input parameters in the experiment and regress this into a relationship. In this case, a 3rd order polynomial relationship works best, as shown in Fig. 6 to 8. However, what we need here is to know how the memory contribution changes and this relationship is to total performance.

Some regressions don't work well; In this case, we need to simulate two different frequencies for each cache size. From this data, we can extract the CPU and memory contributions for each of the cache sizes. The impact of the cache size on memory contribution can regress into an equation that is used to modify the memory component, as shown in Fig. 9. The reference configuration cache size should be set to 1 (normalized). All values should be in reference to the normalized baseline (1).

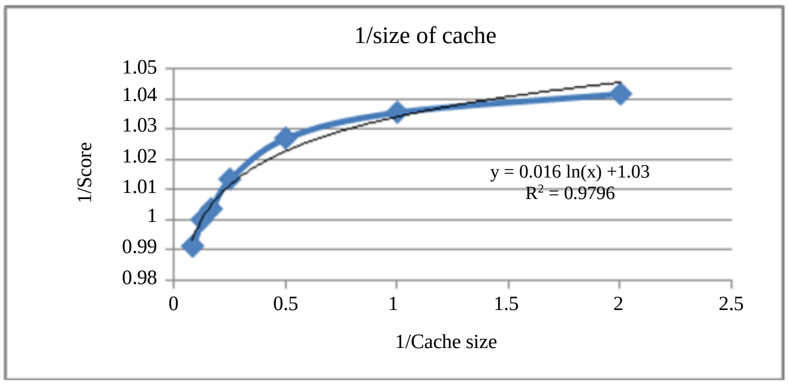


Fig. 8: 1/Cache Size vs. 1/Score

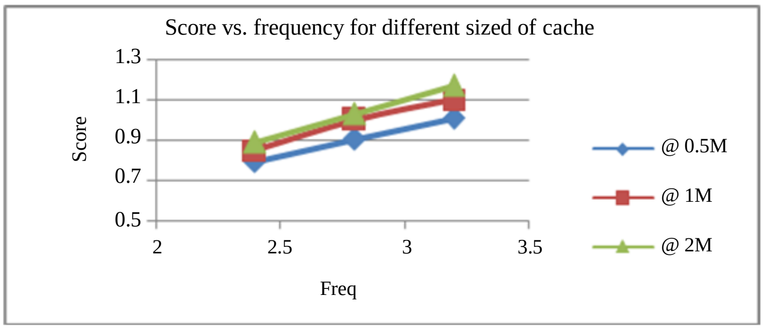


Fig. 9: Performance score vs. different frequency for different cache sizes

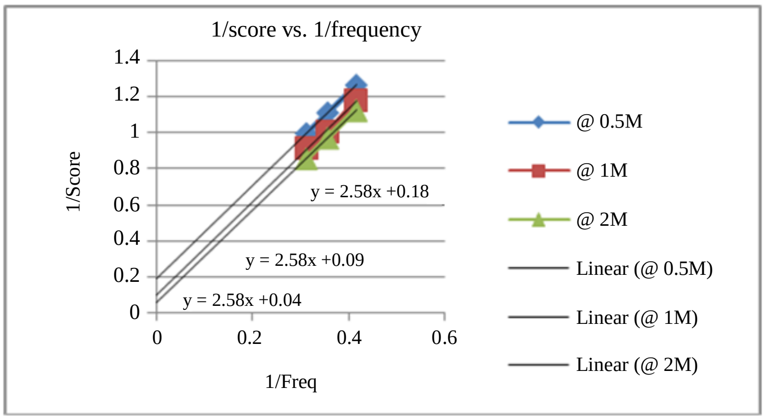


Fig. 10: 1/Score vs. 1/Frequency for different cache sizes

From Fig. 10, the slope for 1/score vs. 1/Freq for all sizes of the cache memory is the same, which is 2.588; however, the intercept part differs. This is expected, given the different cache sizes.

Designing the right experiments simplifies the analysis. With the right experiment sets, we can combine the regression data into one formula to project performance. We can conduct a multivariable regression to do this. Some rules used in conducting a measured experiment used to develop the performance model are:

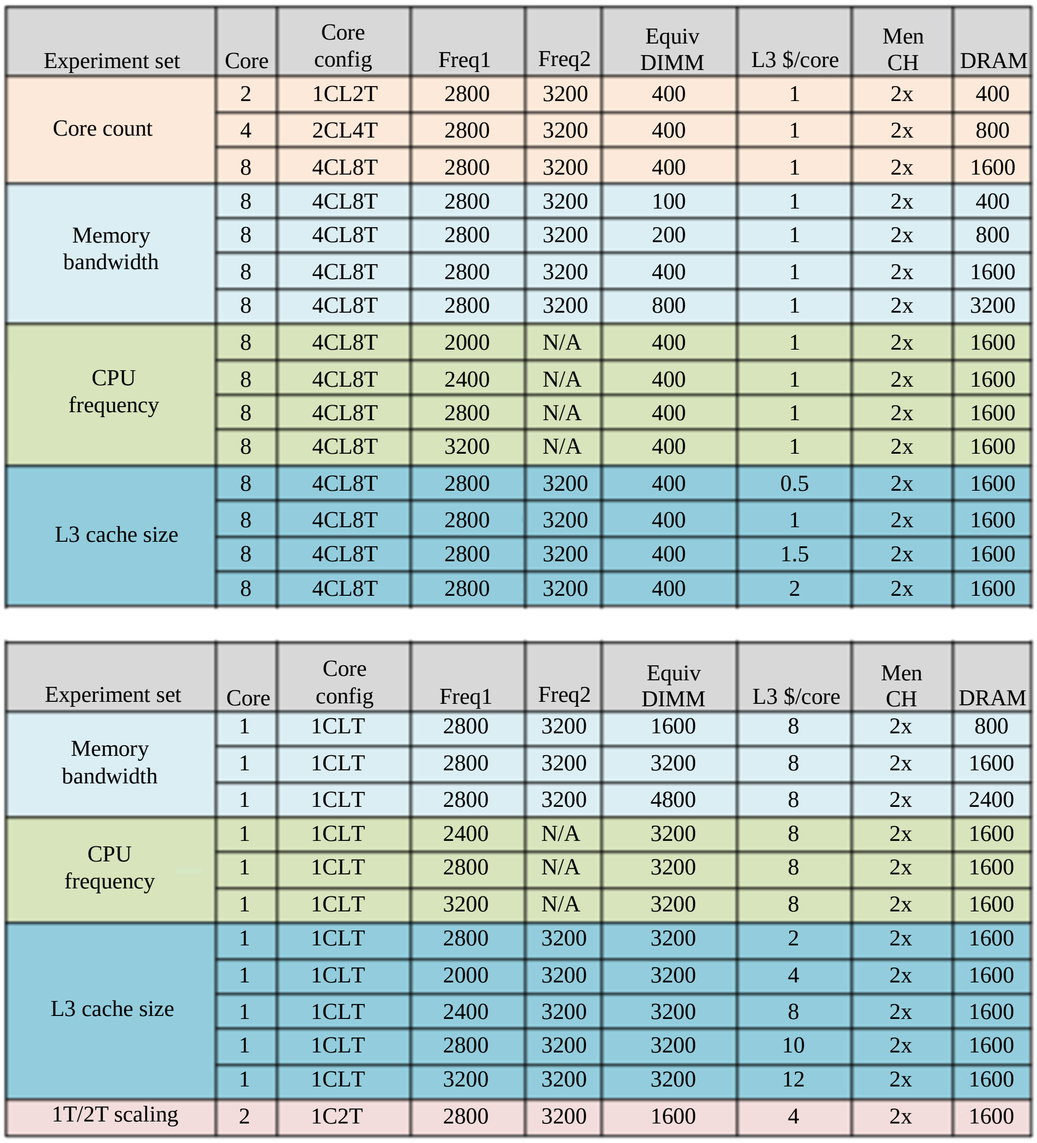
Rule 1: All experiment sets must contain a standard configuration
Rule 2: Experiment sets should have a minimum of 3 configurations. More is always better.
Rule 3: Always measure two different frequencies for sets producing non-linear relationships (i.e., cache size)

The experiment set for option 1 used is shown in Fig. 11. This would be the best experiment set, a single thread set is used for better projections of single-thread benchmarks. Note that in every set, there are duplicates from other sets. The total number of experiments is less than the number of configurations shown.

The experiment set of other options is described as option 1 and option 2. For option 1, represent a method to use the least amount of Multi-thread measurement. The frequency and cache scaling experiments are done with 1 thread measurement. The output for this option provides the least measurement time with the least accurate method.

For option 3, these sets recognize the behavior of a module (core pairs) needs to be modeled. It is a compromise of Option 1 and Option 2. A Single thread set is used for better projections of single-thread benchmarks. Note that in every set, there are duplicates from other sets. The total number of experiments is less than the number of configurations shown in Fig. 11. Deriving the relative performance is with respect to the Intel Nehalem Xeon processor, configured with eight cores, with 2800 MHz core frequency and memory speed bus of 400 MHz. First, we take measured data from SPEC CPU2006 and analyze the sensitivity performance curve with respect to one performance parameter, for this case, it's the number of cores while keeping all other architecture parameters fixed. The measured performance curve is shown in Fig. 12. Relative performance derived is shown in equation (1) as:

$$\textit{Relative Preformance} = 1/\left(\textit{Slope}\times\left(\frac{8}{\#\textit{of cores}}\right)+\textit{Intercept}\right), \qquad (1)$$



Experiment set	Core	Core config	Freq1	Freq2	Equiv DIMM	L3 $/core	Men CH	DRAM
Core count	2	1CL2T	2800	3200	400	1	2x	400
	4	2CL4T	2800	3200	400	1	2x	800
	8	4CL8T	2800	3200	400	1	2x	1600
Memory bandwidth	8	4CL8T	2800	3200	100	1	2x	400
	8	4CL8T	2800	3200	200	1	2x	800
	8	4CL8T	2800	3200	400	1	2x	1600
	8	4CL8T	2800	3200	800	1	2x	3200
CPU frequency	8	4CL8T	2000	N/A	400	1	2x	1600
	8	4CL8T	2400	N/A	400	1	2x	1600
	8	4CL8T	2800	N/A	400	1	2x	1600
	8	4CL8T	3200	N/A	400	1	2x	1600
L3 cache size	8	4CL8T	2800	3200	400	0.5	2x	1600
	8	4CL8T	2800	3200	400	1	2x	1600
	8	4CL8T	2800	3200	400	1.5	2x	1600
	8	4CL8T	2800	3200	400	2	2x	1600

Experiment set	Core	Core config	Freq1	Freq2	Equiv DIMM	L3 $/core	Men CH	DRAM
Memory bandwidth	1	1CLT	2800	3200	1600	8	2x	800
	1	1CLT	2800	3200	3200	8	2x	1600
	1	1CLT	2800	3200	4800	8	2x	2400
CPU frequency	1	1CLT	2400	N/A	3200	8	2x	1600
	1	1CLT	2800	N/A	3200	8	2x	1600
	1	1CLT	3200	N/A	3200	8	2x	1600
L3 cache size	1	1CLT	2800	3200	3200	2	2x	1600
	1	1CLT	2000	3200	3200	4	2x	1600
	1	1CLT	2400	3200	3200	8	2x	1600
	1	1CLT	2800	3200	3200	10	2x	1600
	1	1CLT	3200	3200	3200	12	2x	1600
1T/2T scaling	2	1C2T	2800	3200	1600	4	2x	1600

Fig. 11: Experiment sets used for single-threaded and multi-threaded benchmarks

The constant ‘8’ used in Equation (1) is derived from a measured baseline using the Intel Xeon processor with eight cores. Processor configuration with eight cores is used as the normalized baseline and all other measurements are relative to this baseline. The slope and intercept values are derived using regression. The performance in Figure 12 shows a non-linear relation between the number of cores and relative performance. Taking the inverse will give us a linear relationship, as shown in Figure 13.

We implement the linearity method for memory DIMM speed per thread. The relative memory performance is derived in Equation (2):

$$\begin{aligned}&\mathit{Relative\ Preformance}\\&=1/\left(\mathit{Slope}\times\left(\frac{400}{\mathit{DIMM\ Speed}}\right)+\mathit{Intercept}\right).\end{aligned}\qquad(2)$$

The slope and intercepts derivations are discussed and derived in the results section. The reason we have 400 in the equation is that for the measured baseline we used a memory speed of 400 MHz. We repeat the same experiments for DIMM speed versus the memory relative performance, also the inverse of memory speed versus the inverse of relative performance as shown in Fig 14 and 15.

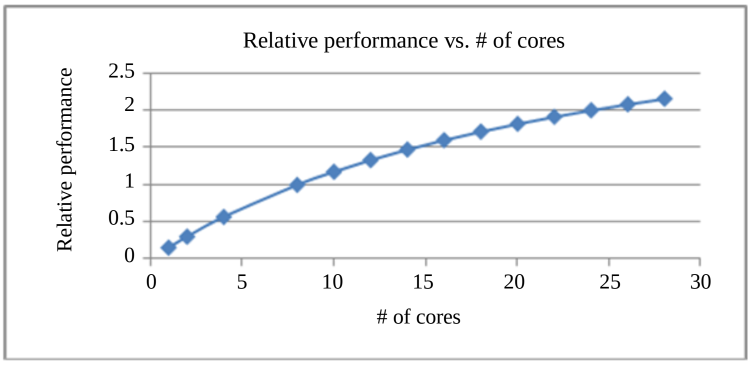


Fig. 12: Relative performance versus # of cores

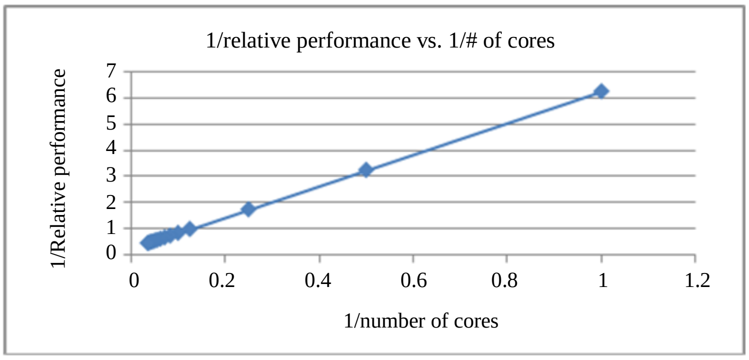


Fig. 13: 1/relative performance vs. 1/# of cores

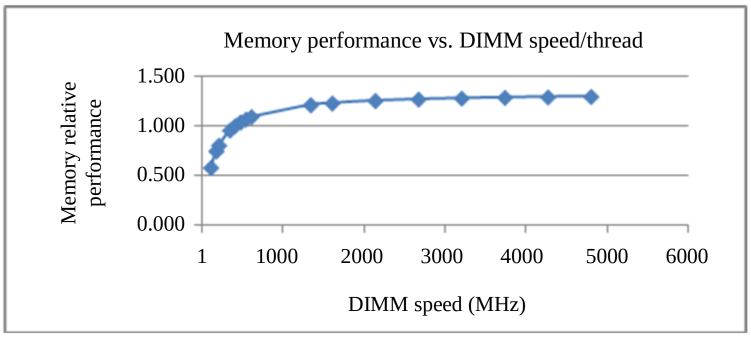


Fig. 14: DIMM speed/thread versus memory relative performance

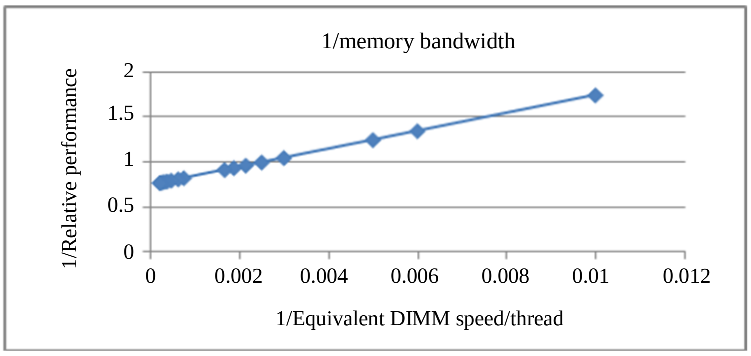


Fig. 15: Inverse of memory DIMM speed and relative performance

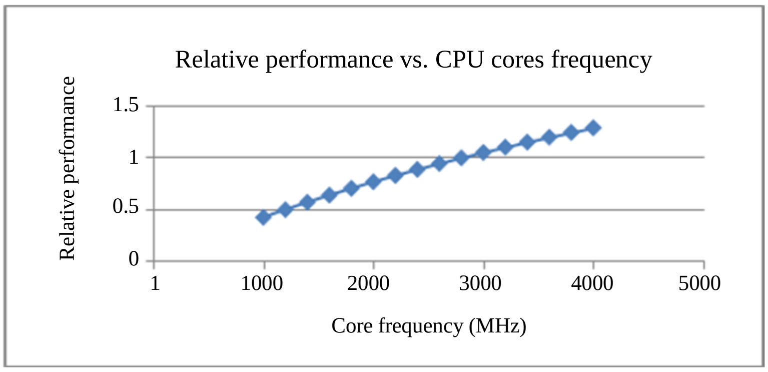


Fig. 16: Relative performance Vs. CPU core frequency

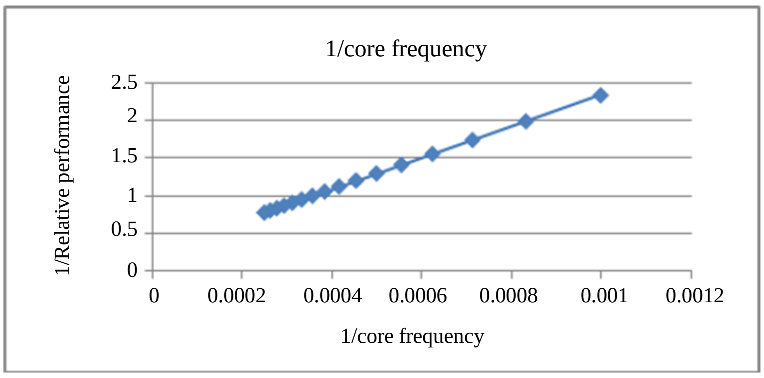


Fig. 17: 1/relative performance versus the 1/core frequency

By repeating the same analysis for the core frequency generates a linear relationship between the inverse of the relative performance versus the inverse of the core frequency, this is shown in Fig. 16 and 17.

b) HDxPRT

For HDxPRT, the performance score which consists of the three sub-categories (convert videos from camcorder, edit photos and video from a digital camera and prepare media for portable devices) is derived using the GeoMean of the three components as follows:

$$score = 100 \times \sqrt{\Pi\left(\frac{Tref}{Trun}\right)} \quad (3)$$

In our sensitivity analysis, we conclude that there is a 40% scaling for the change in the number of cores, minimum sensitivity to cache and <3% sensitivity to Simultaneous Multi-Threading (SMT), as shown in Table 1.

The core sensitivity for different subcategories are shown in Fig. 18.

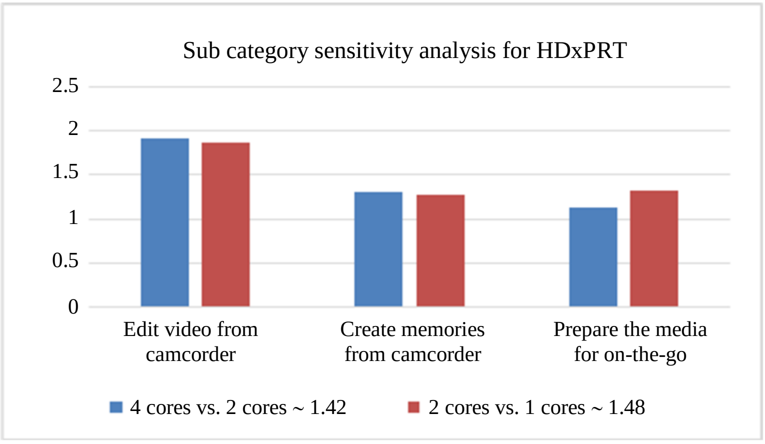


Fig. 18: Sub-category sensitivity analysis for HDxPRT

Table 1: Sensitivity analysis for HDxPRT

Number of Cores scaling		Cache Sensitivity		SMT ON vs. SMT OFF	
2core vs. 1 core	1.48	1MB vs. 3MB	1.02	2 cores 4 threads vs. 2 cores 2 threads	1.01
4 cores vs. 2 cores	1.42	3MB vs. 6MB	1.01	4 cores 8 threads vs. 4 cores 4 threads	1.03
6 cores vs. 4 cores	1.04	6MB vs. 8MB	1.01	cores 612 threads vs. 6 cores 6 threads	1.03

There is an 80% performance improvement for Edit and Convert videos from camcorder and 30% improvements for Edit photos and videos from the camera. For HDxPRT benchmark, the performance for projected time is calculated as:

$$T(projected) = (CPI \times \#of\ instructions \times Weight) /\ freq \quad (4)$$

and per component HDxPRT score is computed as

$$Score(per\ component) = T(referenced) / \Sigma T(projected) \quad (5)$$

The overall score is computed using the GeoMean score show in Equation (3).

Experimental Results

Using multivariable regression on the linear relationships, we can get coefficients for the input parameters to predict a score. Additionally, we can compute the CPU and memory component times. The component times can be modified by the non-linear relationship from the L3 measurements. We used four different processor architecture variables, which are the number of cores, the core frequency, the memory DIMM speed, latency and the Instruction-Per-Cycle (IPC). The IPC variable depends on the number of branches misses cache misses and pipeline and structural hazards. The higher the occurrence of these variables, the more cycles are consumed, which will result in a lower IPC, which in turn will result in lower performance. The IPC value is measured for SPEC CPU2006 using a measured reference baseline for Nehalem processor. Given the sensitivity analysis we discussed, the general formula for the relative performance can be derived as in Equation (3):

$$\text{Re}lative\ performance = \frac{1}{\left[\left(\frac{Core\,coefficient}{\#of\ Cores} + \frac{Core\,Frequency\,coefficient}{Core\,Frequency} + Z\right) \times IPC + \frac{DIMM\ speed\,Coefficient}{DIMM\ speed} \times Memory\ Latency\right]} \quad (6)$$

The value for the core coefficient in Equation (3) is derived from the regression coefficient for 1/# of cores. The value for Z in the linear line intercepts and core frequency coefficient is also calculated from regression for 1/(core frequency). The value for the DIMM speed coefficient can be derived from the regression coefficient for 1/DIMM speed. These coefficients are derived using statistical regression analysis for the measured dataset.

For example, for one of the configuration we want to predict performance, the coefficients calculated by the regression statistics are as follows: Z = -0.75, number of cores coefficient = 6, core frequency coefficient =2100 and DIMM speed coefficient = 100. The relative performance equation is set to project relative performance with respect to the measured baseline. For this experiment, the measured baseline is Intel Xeon 8 cores, 2800 MHz core frequency, with 400 MHz DIMM speed. The following table summarizes the relative performance score for different projected processor configurations.

Table 2: SPEC CPU2006Regression coefficients for different configurations

# of Cores	Core frequency (MHz)	DIMM speed (MHz)	Relative perf.	Core time	Core freq. time	DIMM speed time	Z
1	2800	400	0.16	6	0.75	0.25	-0.75
8	2800	400	1	0.75	0.75	0.25	-0.75
10	2800	560	1.28	0.6	0.75	0.17	-0.75
12	2800	560	1.47	0.5	0.75	0.17	-0.75
12	3600	600	2	0.5	0.58	0.16	-0.75

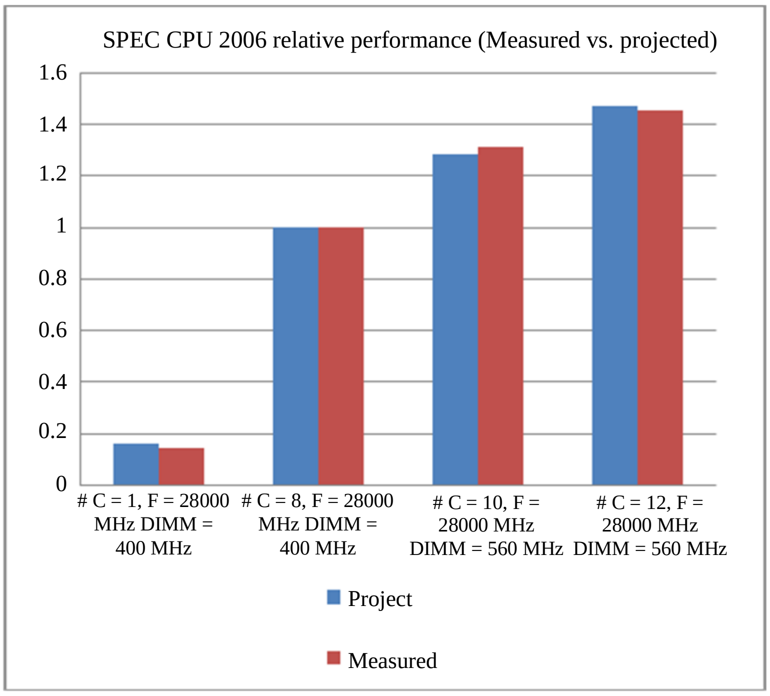


Fig. 19: SPEC CPU2006 Experimental results, Measured vs. Projected Performance

The relative performance shown in Table 2 is normalized '1' with respect to Intel Nehalem Xeon using eight cores, 2800 MHz core frequency and DIMM speed of 400 MHz. This is the normalized baseline and all other measured and projected performance is relative to this measured baseline. The remaining configurations are measured and relative project performance to this normalized configuration. The statistical regression tool enables us to derive the regression coefficients for the number of cores coefficient, core frequency coefficient and DIMM speed coefficient and Z.

The next step is to apply the empirical performance relation in Equation (3) to verify the method with respect to the measured data baseline. We compare relative performance measured with respect to predicted relative performance. The error margin between estimated and measured relative performance is <10% for all test configurations, as shown in Fig. 19. The model is used to cross-validate the estimation of SPEC CPU 2006 performance for different Xeon processor configurations. This enabled performance projection for the future processors that we don't have it yet available for measurement.

In Fig. 19, the performance projection model is used to estimate the relative performance with respect to the Intel Xeon Nehalem baseline. To verify the model, we compare the performance score to a measured score using the same processor configuration as a baseline. The normalized configuration in Fig. 11 is normalized relative to '1', which is done by setting the number of cores to 8, core frequency to 2800 MHz and DIMM speed to 400 MHz. Using the proposed model, if we increase the number of cores to 12, core frequency to 3600 MHz and DIMM speed to 600 MHz, the relative performance increases to about 2.1. The actual measured relative performance is two which is about 5% error margin. Different configurations show an error margin < 5% between measured and projected data.

The timing analysis for multi-variable regression is derived in Equation (7) as follows:

$$\begin{aligned}&Total\,Time = Core\,Time\\&+Ferequency\,Time + DIMM\,Time + Intercept\,Time\end{aligned} \tag{7}$$

The time contribution for the core computation time is the Core_time + Frequency_time + Intercept_Time and the DIMM_time is for the memory time only.

Table 3: SPEC CPU2006 timing analysis for core and memory

				Time Contribution %		Time Contribution	
Cores	Freq	DIMM	Score	Core Time	Memory Time	Core Time	Mem. Time
1	2800	400	0.16	6	0.3	96%	4%
8	2800	400	1	0.8	0.3	75%	25%
10	2800	560	1.284	0.6	0.2	77%	23%
12	2800	560	1.474	0.5	0.2	74%	26%
12	3600	600	2	0.3	0.2	67%	33%

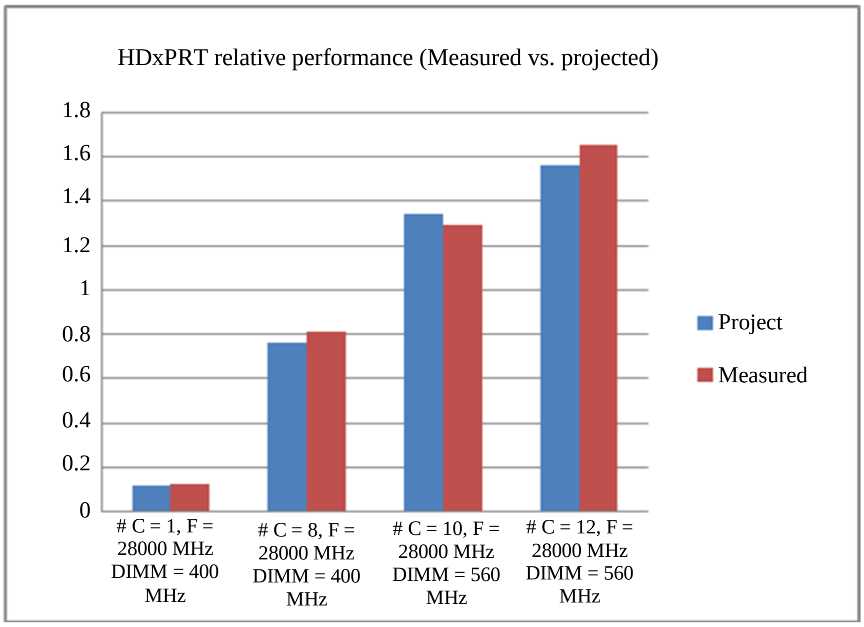


Fig. 20: HDxPRT measured Vs. projected performance

In Table 3, we show the timing breakdown for SPEC CPU 2006 in terms of core time and memory time. The benchmark is more dependent on the core time as compared to the memory time for a lower number of cores. For one core system, the core time is very significant (96%) as compared to the memory time (4%). As the number of cores increases and DIMM speed increases, the memory time contribution also increases. The % core time is derived by taking the ratio of the core_time/total_time. Memory % time is derived similarly by taking the ratio of the DIMM time and the total time. The same method is used to project relative performance for HDxPRT, as shown in Fig. 20.

Concluding Remarks

In this paper, we proposed a novel performance projection method using measured and regression data to predict relative performance for SPEC CPU2006 and HDxPRT using different processor architecture variables that stress the CPU and memory sub-systems. The projection model is independent of underlying ISA; it utilizes regression with a mathematical approach to project relative processor performance. We discovered that the relative performance for the cache is logarithmic rather than linear, while the relative performance for the core frequency, number of cores and memory bandwidth is linear. The estimated relative performance average error margin < 5% compared to the measured performance baseline for Xeon processor configurations. The proposed method in this paper enables the modeling of different processor architecture parameters to estimate relative performance for SPEC CPU 2006 and HDxPRT. The model can be modified by establishing a new measured baseline known as the normalized baseline (normalized to 1) and estimate relative performance from that baseline for different processor architecture. This method does not require any binary or sampled traces used in the simulation for a given benchmark to have an instruction mix that represents well the entire benchmark's instruction mix. For future work, we can implement a sensitivity analysis for different architecture parameters such as the TLB misses, which contributes to lower IPC (higher CPI). Also, the method can be expanded to cover different benchmarks that are used by the industry as a reference to evaluate processor performance and be able to predict performance for future architecture.

Competing Interests

The author declares that he has no competing interests.

References

Baghsorkhi, S.S., M. Delahaye, S.J. Patel, W.D. Gropp and W.W. Hwu, 2010. An adaptive performance modeling tool for GPU architectures. Proceedings of the 15th ACM SIGPLAN symposium Principles Practice Parallel Programming, Jan. 9-14, ACM, Bangalore, India, pp: 105-114. DOI: 10.1145/1693453.1693470

CPU, 2006. Benchmark. https://www.spec.org/cpu2006/

Ganesan, K., L. John, V. Salapura and J. Sexton, 2008. A performance counter based workload characterization on Blue Gene/P. Proceedings of the 37th International Conference on Parallel Processing, Sept. 9-12, IEEE Xplore Press, Portland, OR, USA. DOI: 10.1109/ICPP.2008.57

Hong, S. and H. Kim, 2009. An analytical model for a GPU architecture with memory-level and thread-level parallelism awareness. Proceedings of the 36th Annual International Symposium on Computer Architecture, Jun. 20-24, ACM, Austin, TX, USA, pp: 152-163.

Hoste, K. and L. Eeckhout, 2007. Microarchitecture-independent workload characterization. IEEE Micro, 27: 63-72. DOI: 10.1109/MM.2007.56

Hoste, K., L. Eeckhout and H. Blockeel, 2007. Analyzing commercial processor performance numbers for predicting performance of application on interest. Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Jun. 12-16, San Diego, California, USA. DOI: 10.1145/1254882.1254937

Issa, J. and S. Figueira, 2010. Graphics performance analysis using Amdahl's law. Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication System, Jul. 11-14, IEEE Xplore Press, Ottawa, ON, Canada.

Issa, J., 2016. Processor performance modeling using regression method. Proceedings of the 18th Mediterranean Electrotechnical Conference, Apr. 18-20, IEEE Xplore Press, Lemesos, Cyprus. DOI: 10.1109/MELCON.2016.7495404

Jens, S., 1996. Performance prediction on benchmark programs for massively parallel architectures. Proceedings of the 10th International conference of High-Performance Computer, (HPC' 96).

Khan, S.M., Z. Wang and D.A. Jiménez, 2012. Decoupled dynamic cache segmentation. Proceedings of the 18th IEEE International Symposium on High-Performance Computer Architecture, Feb. 25-29, IEEE Xplore Press, New Orleans, LA, USA. DOI: 10.1109/HPCA.2012.6169030

Krishnaprasad, S., 2001. Uses and abuses of Amdahl's law. J. Comput. Sci. Coll., 17: 288-293. DOI: 10.5555/775339.775386

Phansalkar, A., A. Joshi and L.K. John, 2007. Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite. Proceedings of the 34th International Symposium on Computer Architecture, (SCA' 07).

Prakash, T.K. and L. Peng, 2008. Performance characterization of SPEC CPU2006 benchmarks on Intel Core 2 Duo processor. Proceedings International Conference on Parallel Processing, (CPP' 08).

Saavedra, R.H. and A.J. Smith, 1996. Analysis of benchmark characteristics and benchmark performance prediction. ACM Trans. Comput. Syst., 14: 344-384. DOI: 10.1145/235543.235545

List of Abbreviations

The following is a list of abbreviation used:

CPI: Cycles per Instruction
HDxPRT: High Definition Expert
CPU: Central Processing Unit
HD: High Definition
LRU: Least Recently Used
LLC: Last Level Cache
DIMM: Dual In-line Memory Module
SMT: Simultaneous Multi-Threading
IPC: Instruction per Cycle
CPU: Central Processing Unit
SPEC: Standard Performance Evaluation Corporation
TLB: Translation Lookaside Buffer
LN: Natural Logarithmic