Characterization of SPEC CPU2006 and SPEC OMP2001: Regression Models and their Transferability

ElMoustapha Ould-Ahmed-Vall and Kshitij A. Doshi and Charles Yount and James Woodlee
Intel Corporation, 5000 W Chandler Blvd., Chandler, AZ 85226
{elmoustapha.ould-ahmed-vall,kshitij.a.doshi,chuck.yount,jim.woodlee}@intel.com

Abstract—Analysis of workload execution and identification of software and hardware performance barriers provide critical engineering benefits; these include guidance on software optimization, hardware design tradeoffs, configuration tuning, and comparative assessments for platform selection.

This paper uses Model trees to build statistical regression models for the SPEC CPU2006 and the SPEC OMP2001 suites. These models link performance to key microarchitectural events. The models provide detailed recipes for identifying the key performance factors for each suite and for determining the contribution of each factor to performance. The paper discusses how the models can be used to understand the behaviors of the two suites on a modern processor. These models are applied to obtain a detailed performance characterization of each benchmark suite and its member workloads and to identify the commonalities and distinctions among the performance factors that affect each of the member workloads within the two suites.

This paper also addresses the issue of model transferability. It explores the question: How useful is an existing performance model (built on a given suite of workloads) to study the performance of different workloads or suites of workloads? A performance model built using data from workload suite P is considered transferable to workload suite Q if it can be used to accurately study the performance of workload suite Q.

Statistical methodologies to assess model transferability are discussed. In particular, the paper explores the use of two-sample hypothesis tests and prediction accuracy analysis techniques to assess model transferability. It is found that a model trained using only 10% of the SPEC CPU2006 data is transferable to the remaining data. This finding holds also for SPEC OMP2001. In contrast, it is found that the SPEC CPU2006 model is not transferable to SPEC OMP2001 and vice versa.

I. INTRODUCTION

Analysis of a workload’s execution and the identification of its software and hardware performance barriers provide critical engineering benefits. Unfortunately, the rapidly escalating complexity of modern processors makes it very difficult to identify these barriers and to quantify their impact [1]. Statistical approaches, including regression modeling, are often employed to identify a workload’s sensitivities and the relationship between hardware and software events and its performance.

Model trees is one such approach that can produce accurate, yet interpretable models. This paper uses Model trees for characterizing the behavior of SPEC CPU2006 [2], [3] and the SPEC OMP2001 [4], [5], [6] benchmark suites on a modern processor and to link their performance to the key microarchitectural events on the processor. In particular, these questions are investigated for each suite: Which key architectural resources (such as caches) are most exercised by each suite? What events (e.g., load stalls, cache misses) most directly correlate with changes in performance? How much performance change can be attributed to each?

For each of the two suites, Model trees is used to decompose the suite into a number of distinct linear behavior models, each identifying the key events that affect performance as measured by cycles per instruction (CPI). This analysis is also applied to the constituent benchmarks within each suite.

In general, distinct workloads or dissimilar parts of the same workload can be affected very differently by any one performance factor. Platform designers must nevertheless strive to reach definitive levels of performance across a broad range of applications; typically their recourse is costly but cycle-accurate simulation of candidate designs against a broad variety of workloads. For efficiency, intuitively it is desirable to capitalize upon workload similarities, so that where two workloads excite congruent machine behaviors, a model distilled from one workload can be reused for the other. This consideration motivates the notion of model transferability as sketched next.

Informally, a performance model built using data from workload suite P is considered transferable to workload suite Q if it can be used to accurately study the performance of workload suite Q. The obvious direct benefit is an economy of scale in modeling and simulation investments. A second benefit is confirmation that abstracted relationships do not overfit some peculiarities of any one workload and therefore generalize well. More formally, the paper presents statistical methodologies for assessing model transferability and demonstrates the utility of two-sample hypothesis tests and prediction-accuracy analysis techniques. For the two suites, SPEC CPU2006 and SPEC OMP2001, the MS models are found to be highly dissimilar. Specifically, many of the key events that appear in one tree model do not appear in the other, and the transferability tests confirm the intuition in this case. Conversely, the models built
from randomly chosen data from SPEC CPU2006 (and SPEC OMP2001, respectively) can be expected to apply to the remaining data. The transferability tests confirm this generalizability.

II. RELATED WORK


Benchmark subsetting is one of the main applications of benchmark characterization and comparison. [11] applies clustering techniques to architecturally-independent attributes in order to reduce the size of a benchmark suite trace set for simulation. Several studies investigate the application of PCA (Principal Component Analysis) and clustering to the subsetting of benchmarks: [12] focuses on finding similar architecture-independent phases across benchmark-input pairs; [13] analyzes PMU events for SPEC CPU2006 and applies PCA, clustering and other statistical methods to this data to compare and subset the benchmarks; and [14] compares several subsetting approaches and concludes that PCA and Plackett and Burman (P&B) outperform the others. [11] uses SPEC CPU2000 and ICA (Independent Component Analysis) to identify a representative subset of benchmarks for use in simulation studies and demonstrates the improved performance of ICA over PCA and other approaches.

Recently more complex analytical models have been applied to benchmark performance analysis. [15] compares five machine-learning regression algorithms applied to PMU data for a subset of SPEC CPU2006. Model trees is found to perform as well as both artificial neural networks (ANNs) and support vector machines (SVMs). In addition, Model trees has the advantage of interpretability. [16] describes the application of Model trees for characterizing benchmark performance. [17] studies benchmark representativeness and builds a predictive regression model for SPEC CPU2006. [18] develops a statistically rigorous methodology in applying regression models to the architecture design space. Care is taken to ensure that model attributes are statistically significant and that the models, derived from SPEC CPU2006 and SPECJbb, are accurate in predicting performance and power. [19] applies clustering, association and correlation analysis to program attributes and constructs predictive models using ANNs and polynomial regression. Our work leverages the efforts in this direction by building performance models for detailed characterization of suites of workloads rather than application tuning. We also discuss the application of statistical techniques to evaluate the transferability of performance models.

III. METHODOLOGY AND EXPERIMENTAL SETUP

To determine the key performance factors for a suite of workloads and quantify the impact of each factor, we use Model trees [20]. The model tree algorithm used in this paper is M5’ [21], which is a re-implementation of Quinlan’s original M5 algorithm [20] in the open-source software package WEKA [22]. While most other linear and non-linear regression techniques fit a single function to predict a dependent variable from a set of independent variables, M5’ recursively partitions the input space (training set) into a number of disjoint hyperspaces. The partitioning is reflected by a tree model. The disjoint hyperspaces that are produced become leaf-level linear models, each of which represents a separate class of behavior. High prediction accuracy and model interpretability make Model trees a particularly suitable approach for workload characterization [16], [15].

The model inputs are obtained as processor hardware counter data gathered over short execution intervals of equal instruction length on a machine with an Intel® Core™2 Duo (dual-core) 2.13 GHz processor [23], [24] and 4 GB of memory. Each core has a 32 KB level-one instruction cache and a separate data cache of the same size. The two cores share a level-two cache of 4 MB and use the Microsoft® Windows™XP 64-bit operating system. For modeling, CPI is used as the metric to be predicted as a function of 20 other performance counters as listed in Table I. Both SPEC CPU2006 and SPEC OMP2001 were compiled with baseline rules using Intel Compiler version 9.1. The data is collected for all the benchmarks in SPEC CPU2006 with their reference dataset. For SPEC OMP2001, the data was collected for each benchmark using the medium input set. The processor has five performance counters, three of which are dedicated to the events cpu_CLK_UNHALTED.CORE, INST_RETIRED.ANY and cpu_CLK_UNHALTED.REF. The remaining two reconfigurable performance counters are round-robin multiplexed over the remaining events in table I for periods of 2 million instructions (i.e., the interval size or sample width is 2 million instructions). The event counts are normalized by this multiplexing interval to obtain per-instruction values (such as cycles per instruction, branch mispredicts per instruction, etc.) for use in modeling. Since the analysis in this paper is driven by processor hardware counters for a specific platform and a specific set of benchmark binaries, the results are specific to the architecture, platform, and compiler used.

IV. SPEC CPU2006 PERFORMANCE MODEL

SPEC CPU2006 [2] was released in 2006 to replace CPU2000. It is intended to be a CPU- and memory-intensive benchmark suite. This section describes how a performance model based on Model trees can be used to characterize the performance of the complete benchmark suite and its 29 constituent benchmarks on a particular microarchitecture.
TABLE I
CPU PERFORMANCE METRICS USED IN THIS STUDY

<table>
<thead>
<tr>
<th>Metric</th>
<th>PMU event (each is divided by INST_RETIRED_ANY)</th>
<th>Description (each is per-instruction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPI</td>
<td>CPU_CLK_UNHALTED.CORE</td>
<td>CPU clock cycles</td>
</tr>
<tr>
<td>Load</td>
<td>INST_RETIRED LOADS</td>
<td>Loads</td>
</tr>
<tr>
<td>Store</td>
<td>INST_RETIRED STORES</td>
<td>Stores</td>
</tr>
<tr>
<td>Mispredicted</td>
<td>BR_INST_RETIRED MISPRED</td>
<td>Mispredicted branches</td>
</tr>
<tr>
<td>Br</td>
<td>BR_INST_RETIRED ANY</td>
<td>Branches</td>
</tr>
<tr>
<td>L1DMiss</td>
<td>MEM_LOAD_RETIRED .L1D_MISSES</td>
<td>L1 data misses</td>
</tr>
<tr>
<td>L1HMiss</td>
<td>L1_MISSES</td>
<td>L1 instruction misses</td>
</tr>
<tr>
<td>L2Miss</td>
<td>MEM_LOAD_RETIRED .L2D_MISSES</td>
<td>L2 misses</td>
</tr>
<tr>
<td>DtlbMiss</td>
<td>DTLB_MISSES.ANY</td>
<td>Last level DTLB misses</td>
</tr>
<tr>
<td>LdBlkStA</td>
<td>LOAD_BLOCK .STA</td>
<td>Load block due to store-address events</td>
</tr>
<tr>
<td>LdBlkStd</td>
<td>LOAD_BLOCK .STD</td>
<td>Load block due to store-data events</td>
</tr>
<tr>
<td>SplitLoad</td>
<td>L1D_SPLIT LOADS</td>
<td>L1 data split on loads</td>
</tr>
<tr>
<td>SplitStore</td>
<td>L1D_SPLIT STORES</td>
<td>L1 data split on stores</td>
</tr>
<tr>
<td>Malign</td>
<td>MISALIGN_MEM_REF</td>
<td>Misaligned memory references</td>
</tr>
<tr>
<td>Div</td>
<td>DIV</td>
<td>Divide operations</td>
</tr>
<tr>
<td>PageWalk</td>
<td>PAGE_WALKS .COUNT</td>
<td>Number of page-walks</td>
</tr>
<tr>
<td>Mul</td>
<td>MUL</td>
<td>Multiply operations</td>
</tr>
<tr>
<td>FpAsst</td>
<td>FP ASSIST</td>
<td>Floating point assists</td>
</tr>
<tr>
<td>SIMD</td>
<td>SIMD_INST_RETIRED_ANY</td>
<td>Retired Streaming SIMD instructions</td>
</tr>
</tbody>
</table>

A. Tree Model for the Whole Suite

In this subsection, we describe a model built by applying the M5' algorithm to event data from the SPEC CPU2006 suite. We varied M5' algorithm parameters to achieve a balance between tractable model size and good prediction accuracy; the resulting model is shown in Figure 1. It contains the following components: split nodes in the interior of the tree, linear models at the leaves of the tree, and directed arcs between split nodes and other split- or leaf-nodes in the tree. Each split node (oval) contains the shortened name of a split variable, the percentage of samples that are contained in the subtree rooted at the split node, and the average CPI across each leaf node (square) contains the name of a linear model (LM1 …LM24), its share of samples, and the average CPI across those samples. Each arc describes a criterion for the split variable in the arc’s origin node.

1) Split Variables and Their Thresholds: Each split node, along with the two arcs that lead away from it, defines a bifurcation criterion that is selected (by M5') in order to minimize the variance on each side of the split and maximize the variance between the two sides. Thus, the split event at a given node identifies the parameter to which CPI is statistically most sensitive. The size of the subtree covered by a split node is a qualitative indicator of the importance of the split event at that node.

The root node in Figure 1 segregates samples with more than, and, not more than 0.00019 DTLB misses per instruction. If this threshold is not reached, a sample is directly modeled by LM1. If this threshold is exceeded, a minimum of three other split variables will have to be considered before reaching another linear model. Its root position identifies DTLB misses as the most discriminating performance factor.

The models representing the next two largest sets of samples, LM7 and LM8, are reached when there are more than 0.00019 DTLB misses per instruction, not more than 0.00048 L2 misses per instruction, and more than 0.00045 load blocks due to store address per instruction. Then, LM7 is reached when there are not more than 0.00019 branch mispredicts per instruction or LM8 when there are. It is not surprising to see DTLB and cache-miss events figuring prominently in the model. It is common for memory-hierarchy events to have a large impact on CPU- and memory-bound workloads.
2) Linear Models: Each leaf node in the tree describes a set of samples that can be modeled by a linear equation. Thus, samples that are classified into a given leaf node can be said to follow the same performance behavior model. However, different samples that follow the same linear equation do not necessarily have the same event densities and can vary in absolute performance. Linear model 1 (LM1) described by Equation 1 indicates that the sample’s execution time increases by 4.73 cycles for every L1 miss event, and so forth. Interestingly, even as only samples with few DTLB misses are selected into this model, these samples have high sensitivity to DTLB misses, especially in light of the overall low average CPI of 0.6 in LM1. LM1 accounts for 45.28% of all samples. This is by far the largest percentage for any linear model. Since only one split node occurs above LM1, it is not surprising that this linear model contains a large number of variables when compared to other models.

Note that some instruction-mix variables (e.g., Loads, Stores, DIV, SIMD) in LM1 and other linear models have negative coefficients. While this is counterintuitive, it can be explained by the fact that the machine furnishes dedicated units where specific instructions can execute. In workload phases where a dedicated functional unit is not an execution bottleneck, the fraction of instructions that it retires improves the aggregate instruction counts and reduces the CPI.

LM1: $CPI = 0.53 + 4.73 \times L1DMiss + 7.71 \times Div + 63 \times L2Miss + 0.254 \times Mul + 7.88 \times Misalign + 17.5 \times MisprBr + 4.37 \times LdBlkStD + 15.7 \times PageWalk + 0.046 \times SIMD + 503 \times DtlbMiss + 6.42 \times L1IMiss + 3.22 \times LdBlkStA + 2.98 \times LdBlkOp + 0.128 \times Load − 0.198 \times Store − 0.251 \times Br$

(1)

The next two largest sets fall in LM7 and LM8 described as follows:

LM7: $CPI = 0.24 + 1172 \times L2Miss + 2.72 \times Store + 17.82 \times DtlbMiss + 24.18 \times L1IMiss + 2.37 \times LdBlkOp + 101.67 \times SplitStore + 0.26 \times SIMD$

LM8: $CPI = 0.61 − 7.99 \times Div − 0.23 \times Mul + 13.85 \times MisprBr + 17.44 \times DtlbMiss + 15.20 \times L1IMiss + 1.44 \times LdBlkStD + 11.35 \times PageWalk + 0.16 \times SIMD$

LM7 represents 10.81% of the samples. In LM7, the samples have a combination of high rates of DTLB misses and load block events. It has a large coefficient of 1172 for L2 misses per instruction—much larger than in LM1. At the same time, its coefficient of 17.8 for each DTLB miss is much smaller than in LM1. Such variation highlights the variety of performance behaviors that are observed when events interact at varying densities; thus the coefficient for an event $x$ in each linear model more generally reflects the per-instruction cost of a group or mix of events that is correlated with each occurrence of $x$. Together, LM1, LM7 and LM8 account for 68.04% of the samples. The other 21 linear models are notably simpler. Three equations have one to three variables:

LM10: $CPI = 1.74 − 0.56 \times SIMD$
LM14: $CPI = 1.21 − 1.15 \times Load + 24.11 \times LdBlkOlp$
LM18: $CPI = 0.98 + 16.47 \times L1DMiss + 56.15 \times DtlbMiss + 6.80 \times LdBlkStA$

The remainder of the models are constants whose values can be found in Figure 1; e.g., the model for LM2 is simply $CPI = 1.44$. The absence of a specific event variable in a linear model does not necessarily imply that the effect of that event is hidden by the influence of other events. Rather, its effect may be contained in the constant term if there is insufficient variability of that event across samples classified in the corresponding leaf node.

This section (tree and linear models) has shown a clear sensitivity of SPEC CPU2006 performance to DTLB misses, L2 misses, load blocks and to a lesser degree branch mispredicts. The following section will describe how the same model can be used to show similarities and differences between individual benchmarks within the suite.

B. Characteristics of the Individual Benchmarks within SPEC CPU2006

Once a model tree is constructed, it can be used to characterize other sets of sample data containing the same performance-monitoring events. This is done by classifying each sample based on the split points in the tree. When all samples are classified, a profile results, showing a distribution of the samples over the linear models. Table II shows the result of applying this characterization to the sample data from each of the 29 benchmarks in the SPEC CPU2006 suite. The row labeled “Suite” contains the distribution of linear models across all samples in the suite discussed earlier. Since the number of samples selected for each benchmark is proportional to the number of instructions required to execute that benchmark, the “Suite” value is weighted in proportion to each benchmark’s instruction count. The row labeled “Average” is an average of the distributions across the benchmarks, giving equal weighting to each benchmark.

We can make several observations from Table II. Ten benchmarks have over 50% of their samples classified into the most popular linear model, LM1, and five of these have over 90% of their samples there. Other linear models are more lightly represented overall, but each of them contains at least 10% of samples from one or more benchmarks. Following is a brief description of a selection of the individual benchmarks that have particularly salient profiles from Table II:

- 482.sphinx3: This speech-recognition benchmark written in C is the only workload with non-negligible contribution to LM18 (72.7% of its samples), which is characterized by
a significant number of split loads and a CPI of 1.2, 20% above the suite average. A split load is a load whose target is not contained within one cacheline.

- **471.ommeter**: This discrete-event simulation workload written in C++ is the only workload with non-negligible (79.8% of its samples) contribution to LM24, which is impacted significantly by DTLB misses, L2 cache misses, load blocks due to overlap stores and branches, and has a relatively high CPI of 2.1.

- **470.lbm** and **436.cactusADM**: These workloads are significant because they are the only workloads with non-negligible samples in LMS and LM11, respectively. **470.lbm** is a computational fluid-dynamics benchmark written in C with 52.6% of its samples in LM5. **436.cactusADM**, written in FORTRAN and C, solves the Einstein evolution equations and has 54.6% of its samples in LM11. Both of these LMs are characterized by high SIMD content: at least 77% of instructions for LM5 with a CPI of 1.6, and at least 91% for LM11 with a CPI of 1.2. LM5 has a significant number of load blocks due to overlapped stores and is distinguished from LM11 high in the tree with fewer L2 misses.

The data in Table II can also be used to calculate the similarity between any two benchmarks based on their linear-model distribution. The $L_1$ (Manhattan) distance gives the following equation for the difference between benchmark $j$ and $k$:

$$D_{j,k} = \frac{1}{2} \sum_i |s_{i,j} - s_{i,k}|$$  \hspace{1cm} (4)

where $s_{i,n}$ is the percent of samples from benchmark $n$ classified into linear model $i$.

Table III shows the differences between pairs of a subset of the 29 SPEC CPU2006 benchmarks. Some interesting observations may be made from Table III. We see that there are several pairs of benchmarks with small differences:

- **456.hmmer** vs. **444.namd**: 1.6% difference. This difference may be surprising because **456.hmmer** is an integer benchmark and **444.namd** is floating point. However, both benchmarks are bioinformatic high-performance-computing (HPC) applications: **456.hmmer** uses Profile Hidden Markov Models (profile HMMs) to search for patterns in DNA sequences; **444.namd** is a simulation of large biomolecular systems that calculates inter-atomic interactions.

- **435.gromacs** vs. **444.namd**: 2.0% difference. Here, both benchmarks are HPC floating-point applications. The **435.gromacs** benchmark performs a simulation of the protein Lysozyme in a solution of water and ions.

- **435.gromacs** vs. **456.hmmer**: 3.3% difference. A direct result of the above two similarities.

- **454.calculix** vs. **447.dealII**: 2.8% difference. Both of
these benchmarks are HPC floating-point applications: 454.coucoulexi performs finite element analysis on linear and nonlinear three dimensional structural applications; similarly, 447.deallII uses an adaptive finite element method to solve partial differential equations. It is interesting that 454.coucoulexi uses Fortran 90 and C, while 447.deallII is written in C++. Also note that the difference between 454.coucoulexi and 435.gromacs is only 8.1%.

Referring back to Table II, we see that the five benchmarks listed above are the same five that had LM1 coverage of over 90%.

At the other extreme, we can find benchmarks that are very dissimilar:

- 429.mcf vs. 444.namd: 97.7% difference. 429.mcf is an integer benchmark used for single-depot vehicle scheduling in public mass transportation; expectedly, it is very different from 444.namd, a biomolecular simulator.
- 429.mcf vs. 459.GemsFDTD: 93.6% difference. 459.GemsFDTD: is a floating-point benchmark that solves the Maxwell equations in 3D in the time domain using the finite-difference time-domain (FDTD) method.
- 444.namd vs. 459.GemsFDTD: 96.3% difference. Here, we see that these workloads are not only dissimilar from 429.mcf, they are dissimilar from each other.

We can also see from the last row of Table III that some benchmarks such as 473.astar and 464.h264ref are more similar to the overall SPEC CPU2006 suite than one such as 436.coucoulexi and 470.lbmmcf. By extension, the linear-model profile of any combination of benchmarks can be compared to the overall suite.

V. SPEC OMP2001 Performance Model

The SPEC OMP2001 benchmark suite is designed to reflect compute intensive parallel applications. This section presents the results of applying Model trees to SPEC OMP2001. Figure 2 shows the model tree for SPEC OMP2001. The tree contains 18 linear models. As discussed in Section IV, the size of a subtree covered by a split node indicates the importance of the event at that node. Load block overlapping a store that shows at the root of the tree is an event where some load instruction awaits the retirement of a store instruction. It figures prominently in the SPEC OMP2001 tree model. This need to wait upon store retirement arises when the load and store operands overlap partially or when they occur at identical page offsets. While this condition is rare in general, its presence in a workload can reduce performance because when it happens, the machine cannot forward store results to loads until stores retire. In approximate descending order, the following list identifies other events that have dominant roles in the SPEC OMP2001 benchmark: store instructions, SIMD instructions, L2 misses, branch mispredicts, multiply instruction, misaligned memory references, DTLB misses and page walks. This list includes events from interior and leaf nodes in Figure 2.

A. Linear Models

Linear models 17 and 18 cover more than half of the training set. For both models, load block due to overlapped store occurs at a rate of 0.74% or more per instruction. Additionally, for LM17 stores occur less than or equal to 7.7% per instruction, while for LM18 they occur at a rate of greater than 7.7%. This sensitivity to the number of stores suggests a possible loss of performance when a store is delayed and a dependent load stalls. Whether this arises due to true data dependencies or due to false sharing is not apparent, but the model suggests profiling the benchmark for the occurrence of this event and for evidence of false sharing.

Linear model 17 (LM17) is described as follows:

\[
\text{LM17: CPI} = 0.80 + 39.1 \times \text{L1DMiss} - 0.281 \times \text{Mul} - 0.941 \times \text{Br} + 9.1 \times \text{L1BkStA} + 5.6 \times \text{L1BkOlp} + 34.6 \times \text{PageWalk} + 0.129 \times \text{SIMD}
\]
Linear model 18 (LM18) is described as follows:

\[
\text{LM18: CPI} = 0.95 - 4.7 \times \text{Div} + 2.08 \times \text{Store} \\
+ 53.0 \times \text{PageWalk} + 0.427 \times \text{SIMD}
\] (6)

LM18 exhibits a large number of stores, which stall loads due to data overlaps. Ordinarily, delaying store instructions does not affect CPI due to the machine’s ability to forward stores to the load instructions that follow them. But when a store stalls due to a DTLB miss, the resulting long duration page walk considerably amplifies its effect on a stalled dependent load. The average CPI for this class is a moderately high 1.49, a third higher than LM17, and reflects this sensitivity. Over 95% of the execution time of benchmarks 328.fma3d_m and 318.galgel_m falls into this class. Both benchmarks would be helped considerably by reducing stores or by reducing partial overlaps or aliases between load and store targets.

Samples with a low SIMD instruction count are tested first for L2 cache misses and then for the rate of branch mispredictions. Figure 2 shows that when SIMD instruction rates are low, the rate of L2 cache misses and the rate of branch misprediction are important in further classification, and when SIMD instruction rates are high, the classification is more sensitive to the rate of multiply instructions.

As was the case with SPEC CPU2006, a majority of the other leaf nodes in the suite are quite a bit simpler. Four equations have three or fewer variables:

\[
\begin{align*}
\text{LM2: CPI} & = 0.39 + 3.95 \times \text{Store} \\
\text{LM6: CPI} & = 0.75 + 16.28 \times \text{L1DMiss} + 123.60 \times \text{Misalign} \\
\text{LM15: CPI} & = 0.79 + 23.17 \times \text{LdBlkStA} + 7.28 \times \text{PageWalk} \\
\text{LM16: CPI} & = 0.65 + 9.51 \times \text{L1DMiss} - 1.11 \times \text{Br} \\
& + 1.98 \times \text{SIMD}
\end{align*}
\]

The remainder of the models are constants whose values can be found in Figure 2; e.g., for LM3, the model is simply CPI = 0.53.

The average CPI for LM16 is 2.50, while that for LM11 is 2.79, which are among the highest in the suite. The two classes cover roughly 7.6% of the suite but due to their high CPI, they account for nearly a sixth of the total benchmark execution time. In Figure 2, LM16 is selected for those samples that have a high fraction of SIMD instructions (greater than 84%); with each SIMD instruction in the above equation contributing nearly 2 cycles, it is not surprising to see a high average CPI in this class. A high CPI for a SIMD instruction-rich benchmark does not, in general, indicate poor performance as SIMD units are data parallel and can complete the most frequent 128 bit operations with just a single cycle overhead. The relatively high multiple of 9.51 for L1 data misses shows the importance of keeping the SIMD units well fed in order to achieve their data-parallel potential. Moving to LM11, which has constant CPI, we see that samples reaching this class have moderately high SIMD instruction density and a measurable number of misaligned memory references. This suggests that a non-trivial

In this linear model, L1 data misses contribute 39.1 cycles, page walks with TLB misses 34.6. It is likely that the high contribution of L1D misses to CPI is due to the amplifying effect of load stalls (the root level split event) and due to contribution of downstream misses (i.e. L2 misses that the model equation does not cover.) And, the cost factors from page walks and load DTLB misses should be considered together, since page walks occur during the course of resolving TLB misses. The average CPI for representatively weighted sections from the SPEC OMP2001 suite at this leaf is 1.16, which is a moderate value, and is 10% below the overall SPEC OMP2001 CPI of 1.27. Three quarters of the execution time of benchmarks 314.mgrid_m and 332.ammp_m falls into LM17; these benchmarks are dominated by a high number of loads blocked by overlapping stores combined with a modest number of stores.
portion of the SIMD operations are performed on operands that are not 16-byte aligned and likely cross cache line boundaries.

B. Characteristics of the Individual Benchmarks within SPEC OMP2001

Table IV shows the distribution of samples taken from each component of SPEC OMP2001 across the linear models of the SPEC OMP2001 model tree of Figure 2. The similarity matrix for this suite can be constructed and interpreted in the same way as that for SPEC CPU2006.

Following is a brief description of the characteristics of each of the individual benchmarks on the basis of the above table.

- **324.apsi_m**: This is a molecular mechanics program in C. LM17 accounts for 73%, and LM15 for 17% of its samples. The average CPI of 1.11 is moderate. It is dominated by load stalls due to overlaps with stores.
- **316.applu_m**: This Fortran program solves partial differential equations. LM16 accounts for nearly two-thirds of its samples, while LM13 and LM15 account for 30%. The average CPI of 1.99 is high due to the high average CPI from LM16. The dominant characteristic of this benchmark is the high rate of SIMD instructions and high rate of MUL instructions.
- **320.eqquake_m**: This C program for earthquake modeling uses finite element method. LM14 accounts for nearly 54% of the samples from this benchmark; while LMs 6, 13, 17, and 18 roughly equally account for 35% of the samples. The key performance factors of the whole suite are represented to measurable degrees in this benchmark, and its CPI of 1.37 is within 10% of the suite CPI. Equake dominates LM14.
- **328.fma3d_m**: This large Fortran program, using finite element methods, simulates crashes. Almost all of this benchmark falls into LM18, and its average CPI of 1.46 is close to the average CPI of the whole suite in LM18. As noted earlier, LM18 has a high rate of load stalls due to store overlaps and a high rate of store instructions.
- **310.wupwise_m**: This Fortran program is used to recognize objects in a thermal image. LM1...LM4 account for virtually all of the samples from this benchmark. This makes it a low CPI (0.53) benchmark. Thus this benchmark also adds good variation to the suite.
- **326.gafort_m**: This is a Fortran fluid dynamics program, used for simulating pollution effects in a lake environment. Over half of the samples in this benchmark are from LM5. The dominant characteristics of the suite (high numbers of load stalls due to store overlaps, high SIMD instruction counts, high rates of stores) are absent from this benchmark, which makes it an interesting component of SPEC OMP. It has a moderate CPI of 0.9. Contribution of samples to LM5 in the overall suite seems to be coming almost entirely from 324.apsi_m.
- **330.art_m**: This neural-network program in C is used to recognize objects in a thermal image. LM1...LM4 account for virtually all of the samples from this benchmark. This makes it a low CPI (0.53) benchmark. Thus this benchmark also adds good variation to the suite.

C. Discussion

Load Blocked with Overlapped Store is clearly a significant factor in the execution of SPEC OMP2001 suite since a high value of this event affects half the suite. Out of the SPEC OMP2001 suite, 314.mgrid_m, 328.fma3d_m, 318.galgel_m and 322.ammp_m, dominated by LM17 and LM18, are most affected by this event. Of these, 328.fma3d_m and 318.galgel_m, virtually all of whose samples fall into LM18, are also affected by the amplifying impact of high rates of store instructions. Store instructions dominate other sections of the SPEC OMP2001 suite besides 328.fma3d_m and 318.galgel_m; however, a high rate of store instructions seems to be a performance factor leading to LM5 and LM12, thus affecting benchmarks 324.apsi_m and 310.wupwise_m. Altogether, store instructions affect 21% of the suite.

A high rate of SIMD instructions is another significant performance factor in the suite. Nearly half of the samples, those in linear models 7 through 16, are affected. Benchmarks 316.applu_m and 312.swim_m have nearly 90% of their samples in these linear models, while 65% of 320.eqquake_m and 45% of 310.wupwise_m are affected by this performance factor.
Mispredicted branches affect nearly 15% of the suite, primarily LM2...LM5 and LM14. Even so, the threshold comparisons in the tree (BrMsPr ≥ 0.001 and BrMsPr ≥ 0.003) are relatively insignificant in the direct performance impact of mispredicted branches. This threshold may identify benchmark sections that have relatively fewer vectorizable sections. The only benchmark with a measurable number of sections in LM14, has one of the shortest vector lengths as noted in [25].

VI. TRANSFERABILITY OF PERFORMANCE MODELS

This section discusses the use of statistical techniques to assess the transferability of performance models. Methods of assessing model transferability are presented, and their application is illustrated. We consider two data sets: $L^1 = \{z_1, z_2, ..., z_n\}$ and $L^2 = \{z_1, z_2, ..., z_n\}$. Each sample consists of independent and identically distributed data points drawn from some data generating process ($P_1$ for $L^1$ and $P_2$ for $L^2$). The performance model $f_1(.)|L^1$ is the model trained using the $L^1$ dataset. It follows a statistical distribution $F_1$, which depends on the data generating process $P_1$. Model transferability is assessed by testing the accuracy of the predictions made using the function $f_1(.)|L^1$ on the dataset $L^2$.

Two approaches are used below to assess the prediction accuracy. The first approach is based on hypothesis testing, while the second is based on various regression prediction accuracy metrics.

A. Two-Sample Hypothesis Testing

1) Methodology: Hypothesis testing can be used in two ways. The first is to compare the two sets $L^1$ and $L^2$ with the assumption that if the two represent the same distribution, then the regression model trained on $L^1$ will accurately predict on $L^2$. In this case the Null and Alternative hypotheses can be formulated as $H_0 : P_1 = P_2$ and $H_a : P_1 \neq P_2$. The second is to compare the predicted values on $L^2$ with the corresponding correct values. In this case, the Null and Alternative hypotheses are formulated as $H_0 : P_{1\hat{y}} = P_2$ and $H_a : P_{1\hat{y}} \neq P_2$, where $P_{1\hat{y}}$ is the distribution of the prediction on $L^1$ using the regression model trained on $L^2$.

In both cases, the distributions used in the tests can only be expressed through the sample estimates of their properties such as mean, median and standard deviation. For instance, when we test using the mean of the dependent variable, the first test becomes $H_0 : \mu_1^2 = \mu_2^2$ and $H_a : \mu_1^2 \neq \mu_2^2$, where $\mu_1^2$ and $\mu_2^2$ are the dependent variable means for $L^1$ and $L^2$ respectively. Similar tests can be formulated using one or more of the independent variables. In the same way, the second way of testing is on the dependent variable mean in $L^2$ and the average predicted value using $f_2(.)|L^1$ as follows: $H_0 : \mu_{12} = \mu_{22}$ and $H_a : \mu_{12} \neq \mu_{22}$, where $\mu_{12}$ is the mean of the predicted values.

To conduct the above tests, several hypothesis testing techniques can be used. These techniques can be divided into two broad categories: parametric tests such as the two-sample t-test, and non-parametric tests such as Leven’s and Mann-Whitney tests. In addition to the independence of the two samples, the two-sample t-test assumes normality and equality of variance. The test is robust against non-normality in case of large samples (the Central Limit Theorem). It is also robust against unequal variance when the number of instances $n$ and $m$ in the two samples are not very different, i.e., they do not differ by a ratio of 3 or more. Since we are using independent large samples with $n$ and $m$ having close values, the two-sample paired t-test is appropriate for the study presented in this paper.

Since the true population distribution is not known for both samples, their standard deviation and mean statistics are not available. Instead, unbiased estimates are obtained from the two samples. The means are estimated using the formulae

$$\hat{\mu}_1 = \frac{\sum_{i=1}^n y_i^1}{n} \quad \hat{\mu}_2 = \frac{\sum_{i=1}^m y_i^2}{m} \quad \hat{\mu}_{12} = \frac{\sum_{j=1}^m y_j^{12}}{m} \quad (8)$$

where $Y^1 = \{y_i^1\}$ and $Y^2 = \{y_j^2\}$ represent the sets of dependent variables in $L^1$ and $L^2$, and $Y_{12}$ is the set of variables representing the predicted values. Furthermore, the variances for the two samples $L^1$ and $L^2$ and for the predicted values can be estimated using the unbiased estimators

TABLE IV
SAMPLE DISTRIBUTION ACROSS LINEAR MODELS BY BENCHMARK: CONTRIBUTIONS ABOVE 20% ARE IN BOLD TEXT.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ammp_m</td>
<td>0.8</td>
<td>0.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
<td>16.7</td>
<td>0.0</td>
<td>73.0</td>
<td>6.3</td>
</tr>
<tr>
<td>apsu_m</td>
<td>1.5</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>2.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>12.3</td>
<td>0.0</td>
<td>17.7</td>
<td>63.7</td>
<td>0.2</td>
<td>1.3</td>
</tr>
<tr>
<td>aru_m</td>
<td>0.7</td>
<td>0.2</td>
<td>9.5</td>
<td>4.7</td>
<td>57.2</td>
<td>7.9</td>
<td>12.2</td>
<td>5.1</td>
<td>4.5</td>
<td>6.9</td>
<td>0.0</td>
<td>0.1</td>
<td>1.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>equake_m</td>
<td>0.4</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>9.0</td>
<td>4.6</td>
<td>0.0</td>
<td>2.7</td>
<td>0.0</td>
<td>0.2</td>
<td>9.1</td>
<td>54.0</td>
<td>0.0</td>
<td>2.6</td>
<td>8.7</td>
<td>8.1</td>
<td></td>
</tr>
<tr>
<td>fma3d_m</td>
<td>0.9</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>98.1</td>
</tr>
<tr>
<td>galgel_m</td>
<td>1.2</td>
<td>68.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.3</td>
<td>0.0</td>
<td>0.5</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>21.4</td>
<td></td>
</tr>
<tr>
<td>gafof_m</td>
<td>0.2</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td>0.7</td>
<td>0.0</td>
<td>7.2</td>
<td>0.0</td>
<td>0.9</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2.2</td>
<td>85.6</td>
<td></td>
</tr>
<tr>
<td>mgrid_m</td>
<td>0.4</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3.6</td>
<td>0.4</td>
<td>0.0</td>
<td>3.3</td>
<td>0.0</td>
<td>48.3</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>39.0</td>
<td>5.9</td>
<td></td>
</tr>
<tr>
<td>wupwise_m</td>
<td>14.3</td>
<td>0.1</td>
<td>0.1</td>
<td>7.2</td>
<td>0.0</td>
<td>31.9</td>
<td>2.1</td>
<td>5.1</td>
<td>3.6</td>
<td>0.0</td>
<td>35.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Suite</td>
<td>4.9</td>
<td>3.0</td>
<td>2.2</td>
<td>3.0</td>
<td>6.1</td>
<td>4.8</td>
<td>1.4</td>
<td>1.1</td>
<td>2.4</td>
<td>0.7</td>
<td>2.1</td>
<td>3.6</td>
<td>1.2</td>
<td>1.7</td>
<td>5.7</td>
<td>5.5</td>
<td>37.8</td>
<td>12.8</td>
</tr>
<tr>
<td>Average</td>
<td>4.9</td>
<td>9.3</td>
<td>2.1</td>
<td>2.9</td>
<td>5.2</td>
<td>6.2</td>
<td>1.2</td>
<td>1.0</td>
<td>1.7</td>
<td>0.6</td>
<td>4.5</td>
<td>3.3</td>
<td>2.7</td>
<td>4.9</td>
<td>3.1</td>
<td>9.8</td>
<td>15.8</td>
<td>20.9</td>
</tr>
</tbody>
</table>
\[ \hat{S}_{1}^{2} = \frac{\sum_{i=1}^{n} (y_i - \hat{\mu})^2}{(n-1)} \]
\[ \hat{S}_{2}^{2} = \frac{\sum_{j=1}^{m} (y_j - \mu_{ij})^2}{(m-1)} \]
\[ \hat{S}_{12}^{2} = \frac{\sum_{j=1}^{m} (y_{ij} - \mu_{ij})^2}{(m-1)} \]

Unbiased estimators for standard deviations of the difference between the tested values are given by
\[ \hat{\sigma}_{1-2} = \sqrt{\frac{\hat{S}_{1}^{2}}{n} + \frac{\hat{S}_{2}^{2}}{m}} \]
\[ \hat{\sigma}_{2-12} = \sqrt{\frac{\hat{S}_{2}^{2}}{m} + \frac{\hat{S}_{12}^{2}}{m}}. \]

Finally, the two-sample t-test statistics are given by
\[ t_{n+m-2}^{1-2} = \frac{\hat{\mu}_{1} - \hat{\mu}_{2}}{\hat{\sigma}_{1-2}} \]
\[ t_{2m-2}^{2-12} = \frac{\hat{\mu}_{12} - \hat{\mu}_{1}}{\hat{\sigma}_{2-12}} \]

where \( n + m - 1 \) corresponds to the degree of freedom of the corresponding Student-t distribution.

2) Application to the SPEC CPU2006 and SPEC OMP2001 Models: In this section, we use the two-sample t-test to assess the transferability of the SPEC CPU2006 model built using a training set representing 10% of CPU2006 data to two different test datasets. The first set is randomly selected from SPEC CPU2006 data and is independent of the training set. This set is expected to have the same distribution as the training set used to build the SPEC CPU2006 model. It is, therefore, expected that the model will be found transferable to this set.

The second set corresponds to the randomly selected SPEC OMP2001 set that was used to train the SPEC OMP2001 performance model. Since this set comes from workloads with potentially very different characteristics (e.g., parallelism), the SPEC CPU2006 model is not expected to be transferable to this set.

a) SPEC CPU2006 Transferability to a Random SPEC CPU2006 Test Set: We first use a test dataset that was randomly selected from the overall SPEC CPU2006 independently from the training set used to build the model. The training set representing 10% of the SPEC CPU2006 data corresponds to the \( L^1 \) set described in the previous section. The random test set represents another 10% of the overall data correspond to \( L^2 \).

The training set has the following mean and variance CPI statistics: \( n = 208373, \hat{\mu} = 0.960046 \) and \( S_{1}^{2} = 0.526^2 = 0.279 \). The CPI mean and variance statistics for the test set are given by \( m = 208373, \mu_{1} = 0.960046 \) and \( S_{2}^{2} = 0.528^2 = 0.277 \). These values result in the following test statistics: 
\[ t_{1}^{1-2} = 1.212 \]
\[ t_{2}^{2-12} = 0.96046 \]

In this circumstance, we consider the SPEC CPU2006 training set (described in the last subsection) as corresponding to \( L^1 \) and the SPEC OMP2001 test set as corresponding to \( L^2 \). If we test for the significance of the differences using the dependent variable (i.e., the CPI), we have the following OMP2001 statistics: \( m = 135582, \mu_{2} = 1.21190 \) and \( S_{2}^{2} = 0.604^2 = 0.365 \). This results in the following test statistics: 
\[ t_{1}^{1-2} = 2.5144, \alpha_{1-2} = 0.002 \] and \( t_{2}^{2-12} = 125.384 \). Since \( t_{2}^{2-12} >> 1.960 \), the test rejects the Null hypothesis. This means that significant difference is found to exist between the two samples at 95%. (Similar conclusions can be reached if the above procedure were repeated for several independent variables such as LdBlkOp.)

When testing for significance of the difference between the
predicted and actual values of the CPI on the SPEC OMP2001 set, we have the following statistics for the mean and variance of the predicted CPI: \( m = 135582, \mu_{12} = 1.14642 \) and \( \sigma^2_{12} = 0.4282 = 0.183 \). This results in the following test statistics: \( \frac{\mu^2_{12} - \mu_{12}}{12} = 0.06548, \sigma^2_{12} = 0.002 \) and \( \frac{\sigma^2_{12}}{2m-2} = 32.57 \). Again, the test statistics reject the Null hypothesis and indicate that a significant difference exists between the predicted and actual CPI for the OMP2001 set.

In terms of transferability, these two tests conclude that the SPEC CPU2006 model built using only 10% of the data is not transferable to the SPEC OMP2001 data.

### B. Use of Prediction Accuracy Metrics to Assess Transferability

In this section, we discuss the use of various prediction accuracy metrics to assess model transferability in the context of performance modeling.

1) Methodology: Prediction accuracy metrics can be used to quantify the accuracy of the prediction made using a performance model on a data set. There are many statistical prediction metrics that have been proposed in the literature. All of these metrics compute some statistics based on the difference between predicted and actual values in the test set. Here, we propose to use two metrics:

- **The Correlation Coefficient (C):** As expected, this metric is based on the standard correlation coefficient. It is a dimensionless index that ranges from -1 to 1 with 1 corresponding to the ideal case. It reflects the extent of a linear relationship between the predicted values and the measured values. The correlation coefficient \( C \) is given by
  \[
  C = \frac{\text{Cov}(f(L^2|L^1), Y^2)}{\sigma_{12}\sigma_{2}} \quad (12)
  \]
  where \( \text{Cov}(f(L^2|L^1), Y^2) \) is the covariance between the predicted and the actual values, while \( \sigma_{12} \) and \( \sigma_{2} \) are their respective standard deviations. Of course, as in the previous section, the prediction is made using the performance model built using the dataset \( L^1 \).

- **Mean Absolute Error (MAE):** This metric is measured in the same units as the dependent variable (in this case CPI). It is given by
  \[
  \text{MAE} = \frac{\sum_{j=1}^{m} |f(z_j^2|L^1) - Y_j^2|}{m} \quad (13)
  \]

MAE can take values from 0 to infinity, with 0 representing the ideal case.

The transferability is assessed by comparing the metrics obtained when predicting on the test set with “normal” or expected values that are within thresholds. Acceptable values are domain dependent. In the case of performance modeling, we are considering for illustration that a correlation coefficient of more than 0.85 and a mean absolute error of no more than 0.15 as acceptable.

2) Application to SPEC CPU2006 Model: Here, we use the same two test sets that were used to illustrate the application of hypothesis testing. The first set is a test set randomly selected from SPEC CPU2006 independently from the training set. The prediction accuracy metrics defined in Equations 12 and 13 obtained from application of the SPEC CPU2006 model to this set are as follows:\( C = 0.9214 \) and \( \text{MAE} = 0.0988 \). These two metrics indicate that the SPEC CPU2006 model built using 10% of the data is indeed transferable to a randomly-selected test set. This finding is not surprising and confirms the result of the two-sample t-test. For the SPEC OMP2001 set, the prediction accuracy metrics have the following values: \( C = 0.4337 \) and \( \text{MAE} = 0.3721 \). These values are far from being in the acceptable ranges; thus, the SPEC CPU2006 model is found to be not transferable to the SPEC OMP2001.

Indeed if we look at the model tree built using the SPEC OMP2001 data, we can see that the lack of transferability is attributed to the fact that SPEC OMP2001 has very different performance factors from the ones seen in the case of SPEC CPU2006. In particular, the microarchitectural events that are found most significant, in terms of performance and that come in the first few levels of the tree, are very different for the two suites.

A similar transferability analysis was performed on the SPEC OMP2001 model, yielding similar results: the model built using 10% of SPEC OMP2001 data is found to be transferable to the rest of SPEC OMP2001 data and not at all transferable to SPEC CPU2006 data.

### VII. Summary and Conclusions

We have examined the SPEC CPU2006 and SPEC OMP2001 benchmark suites using the M5' model tree algorithm applied to performance-counter data. The resulting regression models permitted the identification of principal performance factors for each suite and enabled the ability to not only identify constituent benchmarks with unique properties, but also to discern the underlying causes of those distinctions.

We have also described model transferability and its potential benefits. Two methods for studying the transferability of models were discussed: two-sample hypothesis testing and prediction accuracy metrics. Transferability is illustrated by applying these two methods to SPEC CPU2006 and SPEC OMP2001 models.

### References


