Open access • Proceedings Article • DOI:10.1109/IISWC.2008.4636090

PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on Chip-Multiprocessors — Source link

Christian Bienia, Sanjeev Kumar, Kai Li

Institutions: Princeton University, Intel

Published on: 30 Sep 2008 - IEEE International Symposium on Workload Characterization

Topics: Benchmark (computing) and Suite

Related papers:

- The PARSEC benchmark suite: characterization and architectural implications
- The SPLASH-2 programs: characterization and methodological considerations
- Pin: building customized program analysis tools with dynamic instrumentation
- Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset
- Simics: A full system simulation platform

PARSEC vs. SPLASH-2: A Quantitative Comparison of Two Multithreaded Benchmark Suites on Chip-Multiprocessors

Christian Bienia†, Sanjeev Kumar‡ and Kai Li†
† Department of Computer Science, Princeton University ‡ Microprocessor Technology Labs, Intel
cbienia@cs.princeton.edu

ABSTRACT
The PARSEC benchmark suite was recently released and has been adopted by a significant number of users within a short amount of time. This new collection of workloads is not yet fully understood by researchers. In this study we compare the SPLASH-2 and PARSEC benchmark suites with each other to gain insights into differences and similarities between the two program collections. We use standard statistical methods and machine learning to analyze the suites for redundancy and overlap on Chip-Multiprocessors (CMPs). Our analysis shows that PARSEC workloads are fundamentally different from SPLASH-2 benchmarks. The observed differences can be explained with two technology trends, the proliferation of CMPs and the accelerating growth of world data.

Categories and Subject Descriptors
D.0 [**Software**]: [benchmark suite]

General Terms
Performance, Measurement, Experimentation

Keywords
benchmark suite, performance measurement, multithreading, shared-memory computers

1. INTRODUCTION
The Princeton Application Repository for Shared-Memory Computers (PARSEC) was recently released [1]. This collection of multithreaded benchmarks extends the spectrum of parallel workloads researchers can choose from. It is the outcome of a joint venture between Intel and Princeton University that seeks to provide the research community with an up-to-date collection of modern workloads for studies of chip-multiprocessors (CMPs). The new benchmark suite immediately aroused the interest of researchers around the world who have started to use it for their work. The first papers with results obtained through PARSEC have been submitted. This motivates a fundamental question:

> What distinguishes PARSEC from other benchmark suites?

Several multithreaded benchmark suites are available. SPLASH-2 [16] and SPEC OMP2001 include workloads from different domains but focus on High-Performance Computing. BioParallel [7] is composed of bioinformatics programs. ALPBench [10] is a suite of multimedia workloads. MineBench [11] was created to study data mining. With PARSEC, researcher now have a new option and need to understand how the selection of this benchmark suite can impact their results. Other scientists might face the challenge to interpret PARSEC results and seek ways to apply their existing knowledge of other workloads to the new benchmarks.

To help other researchers understand PARSEC we compare the suite to the SPLASH-2 selection of programs. SPLASH-2 is probably the most commonly used suite for scientific studies of parallel machines with shared memory. According to Google Scholar, its characterization study [16] was cited more than 1400 times. Like PARSEC it is one of few parallel suites that are not limited to a single application domain. Its wide use and the thorough understanding researchers have of these workloads make it an excellent candidate for a comparison. PARSEC aims to improve over SPLASH-2 with more up-to-date programs from a broader range of application domains.

This paper makes three contributions:

- We compare SPLASH-2 with PARSEC to determine how much the program selections of the two suites overlap. Significant differences exist that justify an overhaul of the popular SPLASH-2 benchmark suite.
- We identify workloads in both suites that resemble each other that can help researcher to interpret results. A few benchmarks of the two suites have similar characteristics.
- We demonstrate how current technology trends are changing programs. The direct comparison of the PARSEC suite with SPLASH-2 shows that the proliferation of CMPs and the massive growth of data have a measurable impact on workload behavior.

The scope of our paper is the parallel aspects of the behavior of multithreaded workloads on CMPs. Moreover, we do not study single-core characteristics because this is not the intended use of either benchmark suite. The focus of our study is on redundancy within and between the suites. The remainder of the paper is structured as follows: In Section 2 we give an overview of both benchmark suites. Section 3 outlines the methodology that we used and explains the statistical methods that we employed. In Section 4 we present our comparison and show that SPLASH-2 and PARSEC contain fundamentally different types of programs. An interpretation of these results is given in Section 5, where we demonstrate how the objectives of PARSEC have lead to the inclusion of workloads with different properties. Future and related work are summarized in Sections 6 and 7. We conclude our work in Section 8.

978-1-4244-2778-9/08/$25.00 ©2008 IEEE

Program	Application Domain	Problem Size
barnes	High-Performance Computing	65,536 particles
cholesky	High-Performance Computing	tk29.O
fft	Signal Processing	4,194,304 data points
fmm	High-Performance Computing	65,536 particles
lu	High-Performance Computing	1024×1024 matrix, 64×64 blocks
ocean	High-Performance Computing	514×514 grid
radiosity	Graphics	large room
radix	General	8,388,608 integers
raytrace	Graphics	car
volrend	Graphics	head
water	High-Performance Computing	4096 molecules

Table 1: Overview of SPLASH-2 workloads and the used inputs.

Program	Application Domain	Problem Size
blackscholes	Financial Analysis	65,536 options
bodytrack	Computer Vision	4 frames, 4,000 particles
canneal	Engineering	400,000 elements
dedup	Enterprise Storage	184 MB data
facesim	Animation	1 frame, 372,126 tetrahedra
ferret	Similarity Search	256 queries, 34,973 images
fluidanimate	Animation	5 frames, 300,000 particles
freqmine	Data Mining	990,000 transactions
streamcluster	Data Mining	16,384 points per block, 1 block
swaptions	Financial Analysis	64 swaptions, 20,000 simulations
vips	Media Processing	1 image, 2662×5500 pixels
x264	Media Processing	128 frames, 640×360 pixels

Table 2: Overview of PARSEC workloads and the `simlarge` input set.

2. OVERVIEW

The SPLASH-2 suite is one of the most widely used collections of multithreaded workloads [16]. It is composed of eleven workloads, three of which come in two implementations that feature different optimizations. We provide an overview in Table 1. When SPLASH-2 was released at the beginning of the 90s, parallel machines were still a relatively uncommon and expensive branch of computers. Most of them were owned by well funded government and research institutions where they were primarily used to work on scientific problems. The composition of the SPLASH-2 suite reflects that. The majority of workloads belong to the High-Performance Computing domain.

PARSEC is a new benchmark suite that was released at the beginning of 2008 [1]. It was rapidly adopted by researchers around the world. Within the first 6 months since its release it was downloaded over 500 times. PARSEC has the following five main features:

Multithreaded Applications All PARSEC workloads have been parallelized to take advantage of multiprocessor computers with shared memory.

Emerging Workloads The suite focuses on new applications that require a significant increase in processing power. These applications pose a challenge that has to be solved in order to fully take advantage of their capabilities.

Diverse PARSEC includes programs from a wide range of application domains and usage models in order to capture the increasingly diverse ways in which computers are used.

Employ State-of-Art Techniques The included workloads implement the latest algorithms and techniques in their field.

Support Research The benchmark suite supports research by providing an infrastructure to allow the instrumentation and manipulation of its workloads and to assist detailed microarchitectural simulations.

The workload composition of the PARSEC suite differs significantly from SPLASH-2. We provide an overview in Table 2. Since the release of SPLASH-2 parallel computing has reached the mainstream. The wide availability of CMPs has turned multiprocessor machines from an expensive niche product into a commodity that is used for problems from an increasingly wide range of application domains. This fact has influenced the PARSEC program selection. The suite includes benchmarks from many different areas such as enterprise servers, data mining and animation.

Unlike SPLASH-2 PARSEC already includes input sets that reflect current computing problems. The reference input set of SPLASH-2 however cannot be considered representative for modern problem sizes anymore due to its higher age. Where possible we used a combination of profiling and timing on real machines to determine inputs for SPLASH-2 that have computational demands similar to the PARSEC inputs. In order to preserve the comparability of different programs we used the same input for workloads that solve the same problem, even if the computational requirements would have allowed us to choose a bigger problem size. In each case the two versions of `lu`, `ocean` and `water`, but also `barnes` and `fmm` have therefore the same input size. Table 1 and Table 2 show the inputs that we chose for this study.

3. METHODOLOGY

We used the following methodology to gather and analyze data about the behavior of the workloads: First, we identified a set of interesting characteristics (Section 3.1). We then used execution-driven simulation to obtain the data relevant for the characteristics (Section 3.2). Finally, standard statistical methods were applied to the data to compute the similarity of the workloads (described in Sections 3.3 and 3.4).

3.1 Program Characteristics

Both SPLASH-2 and PARSEC are aimed at the study of parallel machines. A comprehensive benchmark suite for single processor systems already exists with SPEC CPU2006[14]. The focus of our study is therefore on the parallel behavior of the programs. We primarily chose characteristics that reflect how threads communicate with each other on a CMP and how data is shared. Our selection of interesting program characteristics and how we measured them largely follows the methodology established by previous work on characterization of multithreaded programs [16, 6, 10, 1].

Characteristic	Type
Floating point operations per instruction	Instruction
ALU operations per instruction	Instruction
Branches per instruction	Instruction
Memory references per instruction	Instruction
Cache misses per memory reference	Working Set
Fraction of cache lines shared	Sharing
Fraction of cache lines shared and written to	Sharing
Accesses to shared lines per memory reference	Sharing
Writes to shared lines per memory reference	Sharing

Table 3: Characteristics chosen for the redundancy analysis. Instruction metrics are based on totals across all cores for the whole program.

The characteristics that we chose are given in Table 3. To capture the fundamental program properties we included a set of four instruction characteristics that were normalized to the total number of instructions: The number of floating point operations, ALU instructions, branches and memory accesses. Threads running on a CMP use shared caches to communicate and share data with each other. Another five characteristics were thus chosen that reflect properties related to data usage and communication such as the total working set size or how intensely the program works with the shared data. These characteristics are the data cache miss rate, what percentage of all cache lines is shared for reading and what percentage for writing, the ratio of memory references that reads from shared cache lines and the ratio that writes to them.

One difficulty in extracting properties related to cache usage is that the behavior of the program might change with the cache size. For example, shared data might get displaced by more frequently used private data [1] if the cache is too small to contain the whole working set. It is therefore necessary to collect data for a sufficiently large range of cache sizes. In order to avoid that unrealistic architecture parameters skew the data towards aspects of the program behavior not relevant for future CMPs, we limited our experiments to 8 cache sizes ranging from 1 MB to 128 MB. This approach results in the following 44 characteristics:

Instruction Mix 4 characteristics that describe which instructions were executed by the program

Working Sets 8 characteristics providing information about working set sizes

Sharing 32 characteristics describing how much of the working set is shared and how intensely it is used

3.2 Experimental Setup

We obtained our data with Pin[13]. Pin is comparable to the ATOM toolkit for Compaq's Tru64 Unix on Alpha processors. It employs dynamic binary instrumentation to insert routines into the instruction stream of the program under analysis. To obtain information about the impact of different cache sizes, we employed CMP$im [7]. CMP$im is a plug-in for Pin that simulates the cache hierarchy of a CMP.

The numbers for the working set and sharing characteristics were collected by simulating a single shared cache for a CMP. We used this method because we are interested in fundamental program properties, not processor features. This approach abstracts from the architectural details of the memory hierarchy while still capturing the fundamental properties of the program. It is a common method for the analysis of multithreaded programs [16, 6, 10, 1].

We simulated an 8-way CMP with a single cache shared by all cores. The cache was 4-way associative with 64 byte lines. Its capacity was varied from 1 MB to 128 MB to collect the characteristics for different cache sizes. The experiments were conducted on an 8-way SMP with Intel 64-bit CPUs running a `Linux 2.6.9` kernel. All programs were compiled with `gcc 4.2.1`. We chose `gcc` as our compiler because of its wide-spread use. It is usually the compiler of choice for many non-scientific workloads. The entire runtime of all programs was simulated.

3.3 Removing Correlated Data

Characteristics of real-world programs might be correlated. For example, the behavior of programs on CMPs with two caches of similar size might be almost identical. Correlated characteristics can skew the redundancy analysis. It is therefore necessary to eliminate correlated information with Principal Components Analysis (PCA) [3]. PCA is a common method used for redundancy analysis [4, 5, 15, 8, 12]. First, the data is mean-centered and normalized to make it comparable. PCA is then employed to remove correlated information and reduce the dimensionality of the data. PCA computes new variables – the Principal Components (PCs) – that are linear combinations of the original variables. The vectors computed in that manner have decreasing variance, i.e. the amount of information in each vector decreases. In order to decide objectively how much information to keep, we use Kaiser's Criterion to choose how many PCs to eliminate. This approach keeps only the top few PCs that have eigenvalues greater than or equal to one. The resulting data is guaranteed to be uncorrelated while capturing most of the information from the original variables.

3.4 Measuring Similarity

We employ hierarchical clustering to group similar programs into clusters. The Euclidean distance between the program characteristics is a measure for the similarity of the programs. This approach is a common way to process the output of a PCA [5, 15, 8, 12]. Hierarchical clustering works as follows:

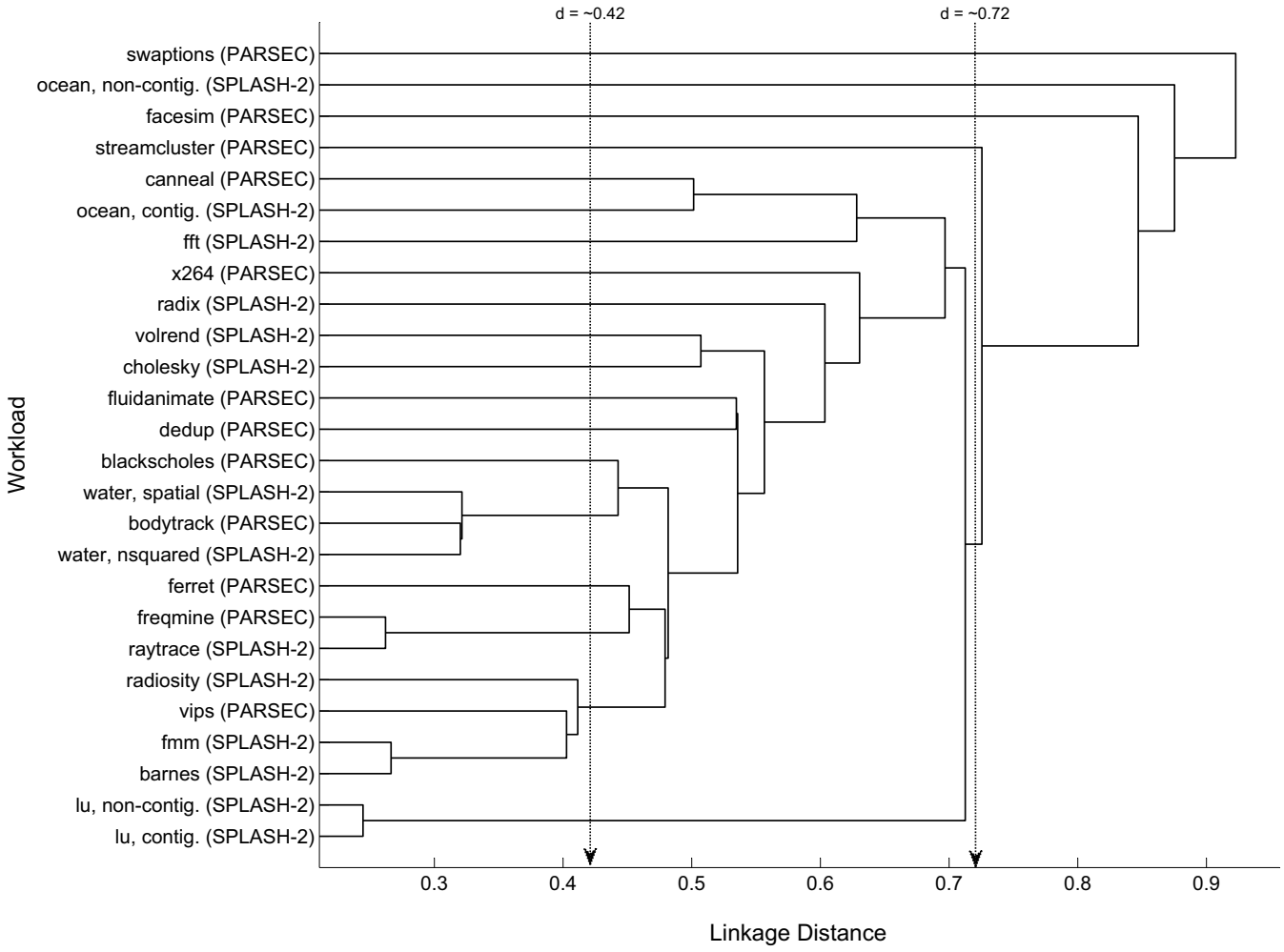

Figure 1: Similarity of SPLASH-2 and PARSEC workloads. The two vertical arrows are used for illustration purposes. SPLASH-2 codes tend to cluster early (distance $d <\sim 0.42$), PARSEC includes a larger number of diverse workloads (distance $d >\sim 0.72$).

1. Assign each workload to its own cluster.
2. Compute the pair-wise distances of all clusters.
3. Merge the two clusters with the smallest distance.
4. Repeat steps 2 - 3 until only a single cluster is left.

The output of the hierarchical clustering algorithm can be visualized by a dendrogram. The vertical axis lists all workloads, the horizontal axis is the linkage distance. Each joint in the dendrogram corresponds to a merge step of the clustering algorithm. Its projection onto the horizontal axis shows how similar two clusters were when they were merged. Clusters with very dissimilar workloads will be merged late, their joint will be close to the root. Programs with very similar characteristics on the other hand will be merged early. Their joint will be close to the leaves, which represent the individual workloads.

4. REDUNDANCY ANALYSIS RESULTS

In this section we employ PCA and hierarchical clustering to analyze how redundant the SPLASH-2 and PARSEC workloads are. We are interested in answers for the following three questions:

- How much do the two program collections overlap?
- In particular, which workloads of the PARSEC suite resemble which SPLASH-2 codes?
- Which benchmark suite is more diverse?

We obtain answers to those questions by analyzing the redundancy within and between the two benchmark suites.

Our first step was to analyze both benchmark suites separately to measure their diversity by computing the total variance of their characteristics. It is almost the same for both suites: SPLASH-2 characteristics have a variance of 19.55, for PARSEC the value is 18.98. However, the variance does not take into account how programs add diversity. Moreover, workloads with almost identical characteristics that deviate substantially from the mean will artificially inflate the variance without contributing much beyond the inclusion of only one of these programs. We will see that this is the case with the two `lu` codes. A more detailed analysis is therefore necessary before conclusions can be drawn.

Our second step was a direct comparison. To compare the suites directly with each other, we analyzed all workloads jointly using a single PCA. This approach guarantees that the PCA weighs all characteristics equally for all programs, since different data will generally result in different correlation and hence different PCs. Having equal weights for the characteristics of all workloads allows us to compare the benchmarks directly. The PCA chose 10 Principal Components that retain 74.73% of the variance. Figure 1 shows the result as a dendrogram containing all SPLASH-2 and PARSEC workloads. PARSEC exhibits substantially more diversity than SPLASH-2. The dendrogram shows that several SPLASH-2 programs form clusters early on. As one would expect, most of the SPLASH-2 programs that come in two versions exhibit significant amounts of redundancy (`lu` and `water`). Only the two `ocean` codes are noticeably different. In fact, the non-contiguous version of `ocean` is the one least similar to any other SPLASH-2 workloads. This is mainly a consequence of intense inter-core communication that is caused by its two-dimensional data layout. With a 4 MB cache, 46% of all memory references of the non-contiguous version of `ocean` go to shared data. About one in three of these references is a write. That is a more than three times higher ratio of shared writes than the next highest one of any SPLASH-2 program. This difference is caused by optimizations for machines with distrubted shared memory and will be discussed in more detail in Section 5.1.1.

Before any PARSEC workloads start to form clusters with each other, the algorithm has already identified 3 groups of workloads containing 7 SPLASH-2 programs in total that exhibit similar characteristics. These clusters have a linkage distance less than $d = \sim 0.42$. We mentioned earlier that workload pairs consisting of two versions of the same program tend to form clusters (`lu` and `water`). Obviously the differences between both versions do not noticeably affect their characteristics. The programs `radiosity`, `fmm` and `barnes` form another cluster. `vips` is the PARSEC workload most similar to them. These benchmarks tend to use a limited number of branches (no more than 11.24% of all instructions). They have medium-sized working sets and benefit from additional cache capacity up to 16 MB. At that point their miss rates fall below 0.1%. `bodytrack` is identified as the PARSEC workload most similar to the `water` programs. Programs of that cluster use about the same amount of floating point operations (between 29.88% and 31.97% of all instructions) and memory accesses (between 27.83% and 35.99% of all instructions). About half of all memory references are used to access shared data once the cache capacity is sufficiently large.

The first pair of PARSEC workloads to be assigned to the same cluster are `bodytrack` and `blackscholes`. They exhibit a linkage difference of about 0.45. The algorithm then identifies a large number of workloads with similar distances. By the time cluster pairs with a distance of about 0.72 are considered, most workloads have been assigned to larger clusters. A distance within that range is the common case for both suites. No obvious similarities can be found anymore between programs clustering in that range.

Several workloads exist that are very different from all other programs in both suites (`swaptions`, the non-contiguous version of `ocean`, `facesim` and `streamcluster`). These programs have a high linkage distance of more than 0.72 to any other cluster of programs and can be considered unique within the analyzed program collection. All but one of these workloads are PARSEC programs. If the two `lu` kernels are treated as a single program, they can also be added to this enumeration, however they have a significantly lower distance to the remainder of the suites than e.g. `swaptions` or the non-contiguous version of `ocean`.

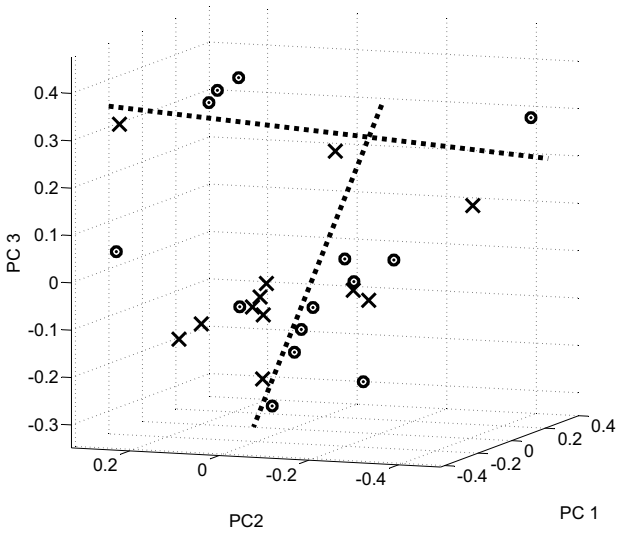

Figure 2: Scatter plot of all workloads using the first three PCs of all characteristics. SPLASH-2 (dots) and PARSEC workloads (crosses) tend to populate different regions of the characteristics space.

Only two PARSEC programs were identified that resemble some of the SPLASH-2 codes (`bodytrack` and `vips`). The similarity within clusters of SPLASH-2 workloads is often greater than the similarity to most other PARSEC workloads. This finding indicates that the two suites cover fundamentally different types of programs. Figure 2 is a scatter plot of all workloads using the first three PCs. It visualizes only a subset of the available information and should hence not be used to infer information about workload similarity. Programs appearing close in the plot might in fact be far away if all relevant dimensions are considered. However, the plot is useful to visualize how programs cluster. We added lines to indicate the regions that we refer to. These lines are not meant to be boundaries and their exact location is not relevant for our conclusions. They are only an aid to visualize tendencies. From Figure 2 it can be seen that all but three PARSEC workloads group in the lower left part of the chart, while all but two SPLASH-2 programs are located in the remaining space. Obviously, SPLASH-2 and PARSEC have little overlap.

Our analysis shows that on modern CMPs, the PARSEC suite contains significantly more diversity than SPLASH-2. Benchmarks of the SPLASH-2 suite tend to cluster early while the PARSEC suite contains a larger number of unique benchmarks. Moreover, PARSEC and SPLASH-2 workloads are fundamentally different, as shown by the scatter plot.

4.1 Multiple Differences

To identify the reason why the two suites differ we performed an analysis of subsets of the characteristics. We broke all metrics up into groups reflecting the instruction mix, working sets and sharing behavior of the programs, which we analyzed separately from each other. This approach allows us to determine which types of characteristics are the reason for the differences. Our results are

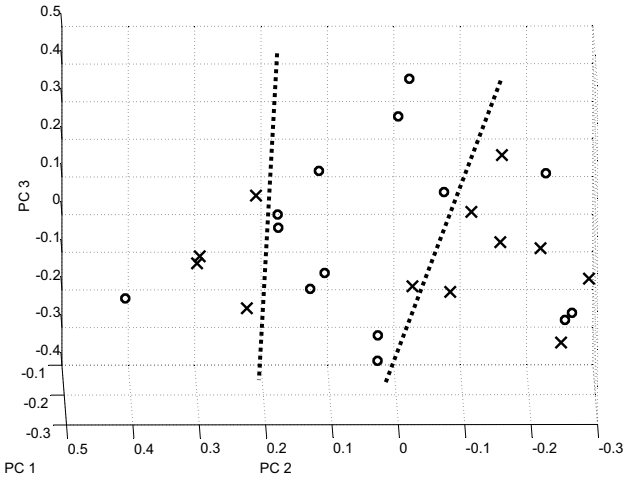

Figure 3: Scatter plot using only the instruction mix characteristics. SPLASH-2 workloads (dots) form a single, major cluster in the first three PC dimensions that contains virtually no PARSEC programs (crosses).

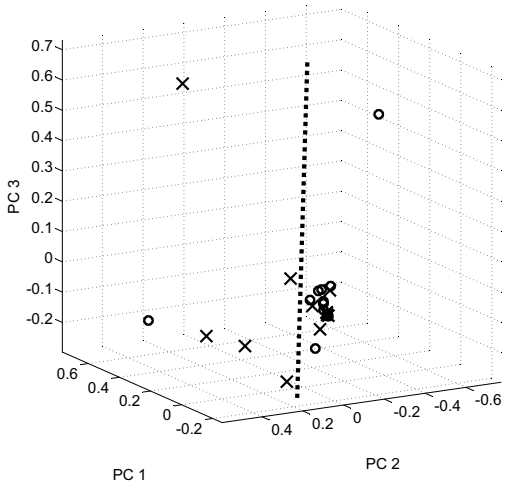

Figure 4: Scatter plot using only the working set characteristics. At least half of the PARSEC workloads (crosses) have substantially different working set properties in the first three PC dimensions, whereas only one of the SPLASH programs (dots) is visibly unique.

presented in Figures 3 - 5. We again added lines to identify the discussed regions.

We will first focus on the instruction mix of the workloads. The instruction mix of SPLASH-2 and PARSEC differs significantly. In Figure 3 we present a scatter plot of the first three PCs derived from the four instruction mix characteristics. As can be seen from the figure SPLASH-2 codes tend to populate the area in the middle. PARSEC programs can primarily be found in the outer regions. The overlap of the suites is small. This result is not surprising considering that the two suites include programs from different domains.

The next aspect which we analyze is the working sets. About half of all PARSEC workloads have noticeably different working set characteristics from all other programs. A scatter plot based on the eight miss rates can be seen in Figure 4. Our analysis of the first three PCs shows that there exists a tight cluster in the first three dimensions to which almost all SPLASH-2 codes and many PARSEC workloads belong. Only one SPLASH-2 program is visibly different, but multiple PARSEC workloads have noticeably different working set properties.

The sharing behavior is one of the most important properties of a multithreaded program on a CMP. Similarities between the two suites seem to exist, albeit with different tendencies. Figure 5 shows a scatter plot with only the 32 sharing characteristics. Benchmarks of the SPLASH-2 suite can predominantly be found in the area on the left side of the figure. PARSEC programs tend to populate the areas on the right and bottom half. Some overlap seems to exist in the lower half of the figure around the horizontal line where a sufficient number of workloads of both suites is located to indicate that commonalities might exist. However, these similarities can only exist between approximately half of the programs in each suite.

Our analysis shows that there is no single source for the differences of the two suites. The program collections exhibit dissimilarities in all studied characteristics. No single property can be identified that can be considered the main reason for the differences.

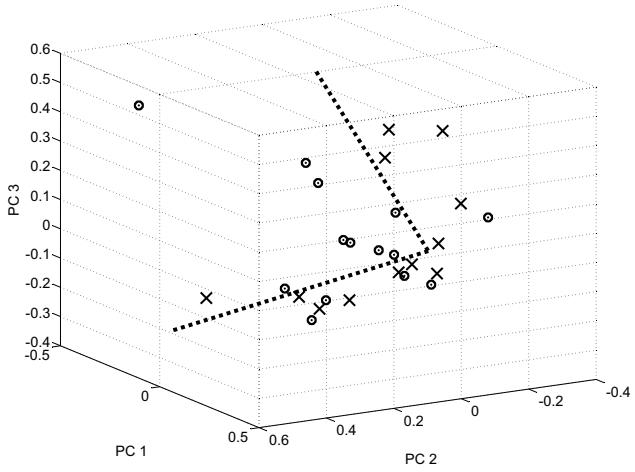

Figure 5: Scatter plot using only the sharing characteristics. SPLASH-2 programs (dots) and PARSEC benchmarks (crosses) tend to populate different areas of the first three PC dimensions of the characteristics space.

5. OBJECTIVES OF PARSEC

PARSEC was designed with the objective to capture recent trends in computing. The remarkably small overlap between SPLASH-2 and PARSEC indicates that these developments might have an impact on workloads that fundamentally alters their characteristics. In this section we will analyze our data from the perspective of two technology trends that have influenced the PARSEC suite: The proliferation of CMPs and the growth of the world's data. PARSEC workloads have been optimized to take advantage of CMPs. Its input sets capture the increasing amount of data that is globally available. SPLASH-2, on the other hand, was created before either of those trends could have an impact on its composition or the design of its programs. We will discuss how both trends are affecting programs. Our goal is to provide a basic understanding of the results of the last section.

Figures 6 and 7 show the miss rates and ratio of shared writes to all accesses of the workloads. We will use this information for our analysis of the impact of the trends. Only a fraction of the information considered by the clustering algorithm for the redundancy analysis can be shown here because the amount of data exceeds what can be comprehended by humans. We show the averages across all cache sizes and details for a selected cache size for both characteristics. We had to use different cache sizes for the detailed breakdowns in the two cases because the shared write ratio is positively correlated with the cache size, unlike miss rates, which are negatively correlated. Different cache sizes from opposite ends of the used spectrum reveal more information about the program behaviors.

5.1 Chip-Multiprocessors

CMPs have become ubiquitous. They integrate multiple processing cores on a single die. The implications of this integration step are twofold: First, it has turned multiprocessors into a widely available commodity that is used to run an increasingly diverse spectrum of programs. The PARSEC suite takes this into account and includes workloads from a wider range of application domains. This is one of the reasons for the increased diversity of the suite. Some characteristics such as the instruction mix seem to be directly affected by that. Second, the trend to CMPs changes the cost model that is employed for program optimizations: On-chip traffic between cores is fast and inexpensive. Off-chip accesses to main memory are costly and usually limited by the available off-chip bandwidth. PARSEC workloads have been adapted to this cost model. SPLASH-2 programs, however, have been optimized for systems with distributed shared memory. They assume a large amount of local memory that can be accessed at relatively little cost while communication with other processing nodes requires I/O and is expensive. This is the opposite of the CMP cost model and can have a significant negative impact on the behavior of the program on CMPs as we will demonstrate using the `ocean` codes.

5.1.1 High Impact of Optimizations

`Ocean` is a program that simulates large-scale ocean movements. SPLASH-2 provides two versions that solve the same problem but employ a different memory layout: The non-contiguous implementation manages the grid on which it operates with two-dimensional arrays. This data structure prevents that partitions can be allocated contiguously. The contiguous version of the code implements the grid with three-dimensional arrays. The first dimension specifies the processor which owns the partition so that partitions can be allocated contiguously and entirely in the local memory of machines with distributed shared memory. Figures 6 and 7 show that this optimization lowers the number of shared writes at the cost of a much higher miss rate on CMPs. The effect is significant. The contiguous version of `ocean` has a shared write ratio that is about 3-5 times lower than the contiguous version for all cache sizes. Its miss rate, however, is about two orders of magnitude higher for small caches. It decreases to 0.13% compared to only 0.03% for the non-contiguous version if the cache capacity is increased to 128 MB. This makes the contiguous `ocean` code the SPLASH-2 program with the worst cache behavior on CMPs. Optimizations like that are used throughout the SPLASH-2 suite. In the case of `ocean` the two available versions allowed us to make a direct comparison, but an evaluation in all other cases is likely to require a rewrite of most of the SPLASH-2 suite. The high miss ratio is the reason why the contiguous version of `ocean` was identified as most similar to `canneal` by the clustering algorithm.

5.1.2 Intense Sharing More Common

Shared writes can be used as an approximation for the amount of communication between threads that takes place through a shared cache of a CMP. Figure 7 shows that the communication intensity is about the same for both suites. It is more concentrated in the case of SPLASH-2, where the two `lu` workloads are responsible for most of the shared writes within the suite. The large growth in shared writes when the cache capacity is increased from 4 MB to 8 MB is almost entirely caused by these two programs. Their ratios increase from 0.88% to 26.18% and from 2.80% to 27.40%. They remain on that level for all other cache sizes. This increase coincides with a drop in the miss rate from 0.11% to 0.01% in both cases when the cache becomes big enough to keep the shared part of the working set that is less frequently used by the programs. This unusual behavior is the reason why both `lu` programs have been identified as different from all other workloads by the redundancy analysis. The fact that the program is contained twice in two very similar versions artificially inflates the average shared write ratio of SPLASH-2. A larger number of PARSEC workloads show increased sharing activity.

5.1.3 Inclusion of Pipeline Model

Another difference between SPLASH-2 and PARSEC is the inclusion of workloads that employ the pipeline programming model in PARSEC. The programs `dedup` and `ferret` use pipelines with functional decomposition, i.e. the various pipeline stages have their own specialized thread pools that handle all the work for their assigned pipeline stage. Unlike in the case of workloads such as HPC programs, the threads of these programs execute different parts of the code. This programming model is frequently used to develop commercial workloads because it allows to break down the problem into independent tasks that can be assigned to different development teams, resulting in lower overall complexity and development cost. As the data streams through the pipeline, it is handed from thread to thread, which perform different operations on the data until the overall goal of the program has been accomplished. A consequence of this model is that in general all communication between the pipeline stages is also communication between different cores. In extreme cases this can be as much as all of the input data, making efficient communication channels with high bandwidth between cores a necessity. Shared caches of CMPs satisfy these requirements. Figure 7 shows that both `dedup` and `ferret` make use of this aspect of CMPs.

5.2 Data Growth

World data is currently doubling every three years[2]. This trend is expected to accelerate further. With this huge amount of information comes the need to process and understand it. An example for a class of programs that deal with this vast amount of data are RMS programs. These workloads employ models that allow them to have a basic understanding of the data they process[2]. For example, the `bodytrack` program employs a model of the human body to detect a person being shown in multiple video streams.

The use of models is nothing new for computing. What has changed is the order of magnitude of the data that must be handled. Both SPLASH-2 and PARSEC contain programs that employ models, but only the algorithms and input sets of PARSEC workloads capture the large increase of data volume that is currently taking place. The compressed archive that contains the whole suite with all inputs is 16 MB in the case of SPLASH-2. For PARSEC, it is 2.7 GB. How does this affect workloads?

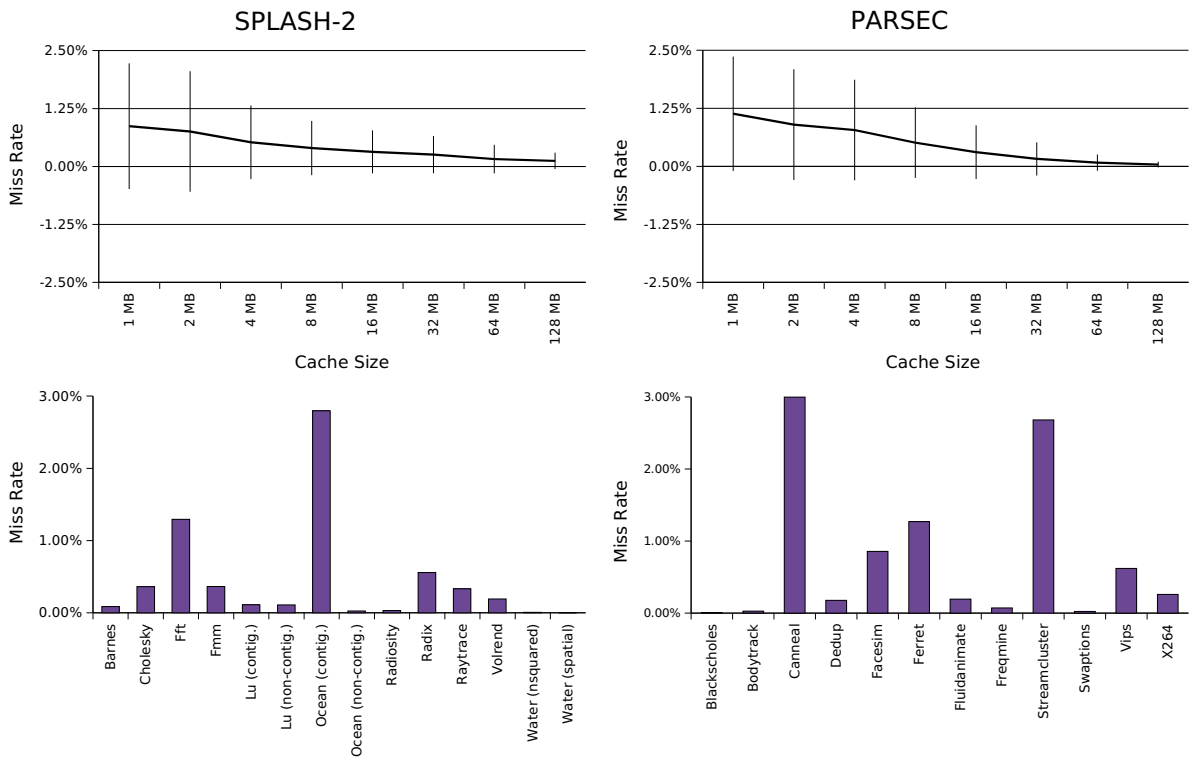

Figure 6: Miss rates of SPLASH-2 and PARSEC. The top charts show the averages of the workloads and the standard deviation for all cache sizes. The bottom charts give the detailed miss rates of the programs for a 4 MB cache. Miss rates of PARSEC workloads are noticeably higher for caches up to 16 MB.

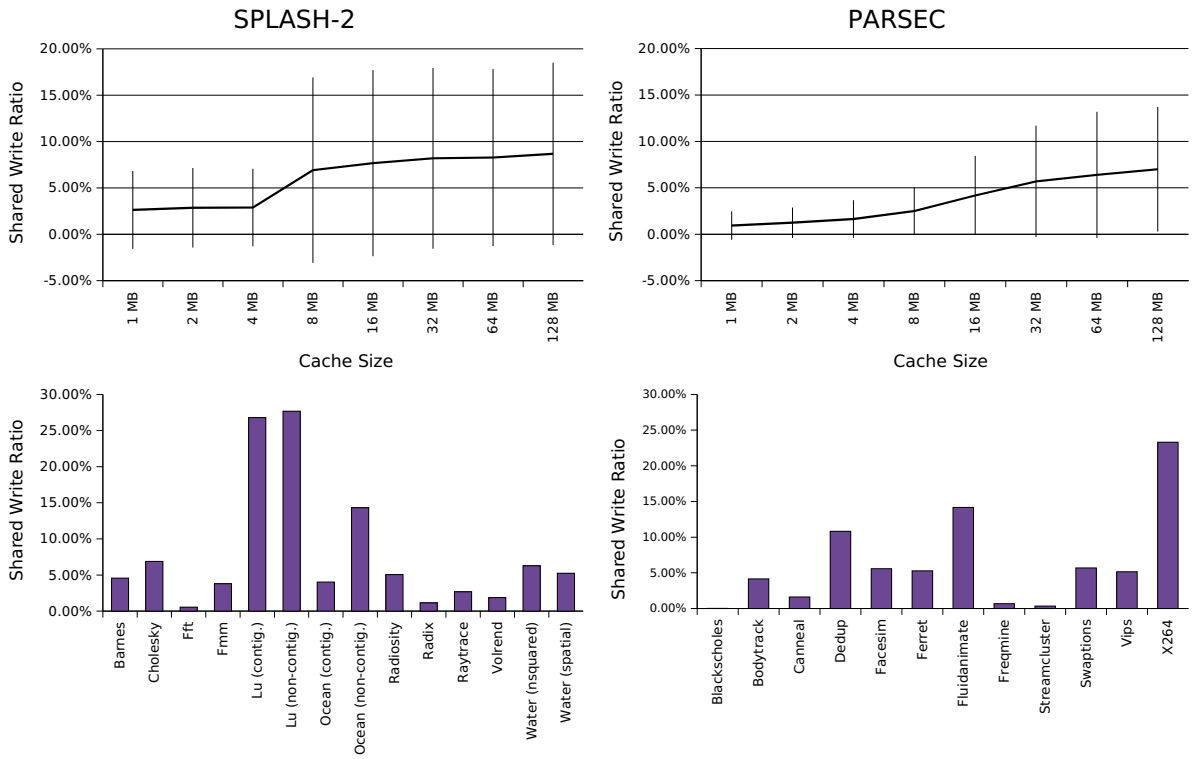

Figure 7: Ratio of shared writes to all memory accesses of SPLASH-2 and PARSEC. The top charts show the averages of the workloads and the standard deviation for all cache sizes. The bottom charts give the detailed ratios of the programs for a 64 MB cache. The ratio of SPLASH-2 is dominated by the two `lu` workloads.

5.2.1 Large Working Sets More Common

Larger input sets are likely to result in larger working sets or require streaming program behavior. Figure 6 shows the miss rates of SPLASH-2 and PARSEC workloads. For smaller caches PARSEC workloads have a significantly higher average miss rate. The difference is 0.26% for a 1 MB cache, approximately one fourth more. It decreases to 0.11% for an 8 MB cache. SPLASH-2 workloads have an average miss rate 0.02% higher than PARSEC workloads if 16 MB caches are used. This trend continues to the end of the spectrum of cache sizes.

A closer look reveals that most of the SPLASH-2 misses are caused by only one program, the continuous version of `ocean`. We already explained in the last section that this behavior is the consequence of optimizations for distributed shared memory machines that negatively affect the program on CMPs. The contiguous version of `ocean` should not be used on CMPs in favor of the non-contiguous version. This makes `fft` the only program with a noticeably higher miss rate. Other SPLASH-2 programs have miss rates that are within a very narrow range on a lower level. PARSEC captures a greater range of miss rates in comparison. It includes workloads such as `blackscholes`, which has the lowest miss rate of all programs (0.01% with 4 MB caches), up to `canneal`, which has the worst cache behavior of all benchmarks (miss rate of 3.18% with 4 MB caches). A total of four of its programs have a noticeably high miss rate (`canneal`, `facesim`, `ferret` and `streamcluster`).

Both SPLASH-2 and PARSEC contain workloads that can generate their own input. Can this feature be used to generate inputs for SPLASH-2 benchmarks that have large working sets comparable to PARSEC workloads? Unfortunately, this is not the case in practice. Most SPLASH-2 codes have runtime requirements that grow superlinearly with the size of the input, whereas its working sets grow no more than linearly in most cases[16]. Time limitations will thus constrain how large input problems can be in practice. For this study, we already sized the SPLASH-2 inputs such that the runtime of all benchmarks was about the same.

In Table 4 we have summarized how time and space requirements grow with the input size for all SPLASH-2 programs that have a parameter to scale their inputs. Only `fmm`, `radix` and the spatial version of `water` employ algorithms for which the computational requirements grow no faster than the memory requirements. `barnes` can also be included in this list for practical purposes since the $\log N$ factor can be considered constant. This limits the number of workloads that can be tuned to reflect the enormous growth of data to only four, too few for most scientific studies. Computationally intensive workloads with relatively small working sets are a characteristic of the SPLASH-2 suite.

Code	Time	Space
`barnes`	$N \log N$	N
`fft`	$N^{1.5} \log N$	N
`fmm`	N	N
`lu (contig.)`	N^3	N
`lu (non-contig.)`	N^3	N
`ocean (contig.)`	N^3	N^2
`ocean (non-contig.)`	N^3	N^2
`radix`	N	N
`water (nsquared)`	N^2	N
`water (spatial)`	N	N

Table 4: Growth rate of time and memory requirements of the SPLASH-2 workloads that have a parameter to scale the input size N. In most cases execution time grows much faster than the working set size.

6. FUTURE WORK

The large difference between SPLASH-2 and PARSEC workloads leaves the question open for a combination of the two program collections. Both suites could improve their diversity by integrating workloads from the other suite. Closely related to this issue is the question of application area coverage. Which program areas should a modern benchmark suite cover, and how well do both suites do from that perspective?

More work is also necessary to understand how these differences can impact performance on CMPs. A thorough study would require the analysis of the workloads on a wide range of designs, since the architectures of CMPs have not matured yet.

7. RELATED WORK

Statistical analysis of benchmark characteristics with PCA and hierarchical clustering is a commonly used method. Eeckhout et al. were first to make use of it for workload analysis [4]. Giladi et al. analyzed the redundancy within the SPEC CPU89 suite [5]. They show that a subset of only six programs is sufficient to capture most of the diversity of the SPEC CPU89 suite. Vandierendonck et al. came to a similar conclusion when they studied the SPEC CPU2000 suite on 340 different machines [15]. Phansalkar et al. presented a study of redundancy and application balance of SPEC CPU2006 on five different machines using six characteristics. Hoste et al. proposed the use of genetic algorithms for comparisons of benchmarks [9]. Their approach is able to reduce the number of characteristics that have to be measured. Joshi et al. compared SPEC benchmarks across generations of the suite [8]. Our selection of characteristics differs from previous work because the focus of our study is on the parallel behavior of programs. We therefore focused on shared-memory characteristics.

8. CONCLUSIONS

In this paper we compared 44 program characteristics that capture instruction mix, communication and memory behavior of the SPLASH-2 and PARSEC benchmark suites on chip-multiprocessors. The redundancy analysis showed that SPLASH-2 and PARSEC are composed of programs with fundamentally different properties. No single reason for the differences could be identified. PARSEC is the more diverse suite in direct comparison.

Some of the observed differences can be explained by the proliferation of CMPs, which have made parallel computing a widely available commodity. Multiprocessors are now used in program domains that have little in common with High-Performance Computing. SPLASH-2 workloads, however, are optimized for distributed shared memory machines. As the example of `ocean` showed this can skew the observed characteristics to the point where the optimization has more impact on the program behavior on CMPs than the actual algorithm. The enormous growth of data has lead to an inflation of working set sizes. Input set and processing time limitations prevent that this behavior can be adequately captured with the SPLASH-2 suite.

So which one is the ”right” suite to use? It depends. SPLASH-2 remains an important and useful program collection. The fact that it is an older selection of HPC programs, which are optimized

for a different machine model, is not necessarily a disadvantage. Once released programs can have a very long life. Our comparison allows researchers to understand the implications of their choice. Whatever their decision, scientists should expect different results.

9. REFERENCES

[1] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, October 2008.

[2] P. Dubey. Recognition, Mining and Synthesis Moves Computers to the Era of Tera. *Technology@Intel Magazine*, February 2005.

[3] G. Dunteman. *Principal Component Analysis*. Sage Publications, 1989.

[4] L. Eeckhout, H. Vandierendonck, and K. D. Bosschere. Quantifying the Impact of Input Data Sets on Program Behavior and its Applications. *Journal of Instruction-Level Parallelism*, 5:1–33, 2003.

[5] R. Giladi and N. Ahituv. SPEC as a Performance Evaluation Measure. *Computer*, 28(8):33–42, 1995.

[6] C. J. Hughes, R. Grzeszczuk, E. Sifakis, D. Kim, S. Kumar, A. P. Selle, J. Chhugani, M. Holliman, and Y.-K. Chen. Physical Simulation for Animation and Visual Effects: Parallelization and Characterization for Chip Multiprocessors. *SIGARCH Computer Architecture News*, 35(2):220–231, 2007.

[7] A. Jaleel, M. Mattina, and B. Jacob. Last-Level Cache (LLC) Performance of Data-Mining Workloads on a CMP - A Case Study of Parallel Bioinformatics Workloads. In *Proceedings of the 12th International Symposium on High Performance Computer Architecture*, February 2006.

[8] A. Joshi, A. Phansalkar, L. Eeckhout, and L. K. John. Measuring Benchmark Characteristics Using Inherent Program Characteristics. *IEEE Transactions on Computers*, 28(8):33–42, 1995.

[9] Kenneth Hoste and Lieven Eeckhout. Comparing Benchmarks Using Key Microarchitecture-Independent Characteristics. In *Proceedings of the IEEE International Symposium on Workload Characterization 2006*, pages 83–92, 2006.

[10] M.-L. Li, R. Sasanka, S. V. Adve, Y.-K. Chen, and E. Debes. The ALPBench Benchmark Suite for Complex Multimedia Applications. In *Proceedings of the IEEE International Symposium on Workload Characterization 2005*, October 2005.

[11] R. Narayanan, B. Özisikyilmaz, J. Zambreno, G. Memik, and A. N. Choudhary. MineBench: A Benchmark Suite for Data Mining Workloads. In *Proceedings of the IEEE International Symposium on Workload Characterization 2006*, pages 182–188, 2006.

[12] A. Phansalkar, A. Joshi, and L. K. John. Analysis of Redundancy and Application Balance in the SPEC CPU2006 Benchmark Suite. In *ISCA '07: Proceedings of the 34th Annual International Symposium on Computer Architecture*, pages 412–423, New York, NY, USA, 2007. ACM.

[13] Pin. `http://rogue.colorado.edu/pin/`.

[14] SPEC CPU2006. `http://www.spec.org/cpu2006/`.

[15] Vandierendonck, H. and De Bosschere, K. Many Benchmarks Stress the Same Bottlenecks. In *Workshop on Computer Architecture Evaluation Using Commercial Workloads*, pages 57–64, 2 2004.

[16] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 24–36, June 1995.