RDCC: A New Metric for Processor Workload Characteristics Evaluation

Tianyong Ao, Pan Chen, Zhangqing He, Kui Dai, Xuecheng Zou

***Abstract*—Understanding the characteristics of workloads is extremely important in the hardware and software design of computer systems. Better metrics and methods for evaluating workloads are continually needed. Here we propose a quantitative metric, namely RDCC (Ratio of Dynamic Computation to Control operations), for evaluating workloads. The RDCC values of a great many benchmarks are measured in various conditions. The evaluation results indicate that the intrinsic properties of the application can be marked with the RDCC value. The applications with lower RDCC values contain more control operations and less computational operations. In contrast, the applications with higher RDCC values consist of less control operations and more computational operations. In addition, the RDCC value of one application is relatively stable with different compiler optimizations and inputs. Furthermore, this paper presents a quantitative approach for identifying computation-intensive and control-intensive applications by RDCC values, and provides five levels of classification based on the K-means clustering of RDCC values. The feasibility is verified by many evaluation results. Therefore, RDCC can be used as a metric to identify and evaluate the characteristics of applications and is much helpful for the design of high-efficiency processor architecture and software optimization.**

***Index Terms*—RDCC, processor, workload, Computation-intensive, Control-intensive.**

I. INTRODUCTION

ENERGY efficiency has become an important companion to performance in modern computer design. Understanding the characteristics of workloads is extremely important in the design of energy-efficient computer architectures [1], [2]. It is very necessary to study the characteristics of computation-intensive applications and control-intensive applications for improving the energy efficiency of the computer system [3], [4], [5]. Differentiating computation-intensive applications from control-intensive applications is helpful for the designers to improve the system energy-efficiency in the design of computer architecture and software, especially in the heterogeneous multi-core system.

Computer architects continually need better metrics and methods for evaluating processors as well as a better understanding of how to use them appropriately [6], [7]. As a fundamental part of computer architecture research, the metrics study has been focusing on multicore processors recently. For example, Michaud and Pierre proposed several different metrics for quantifying the throughput of multicore processors [8], and Otoom elaborated the capacity metric for chip heterogeneous multiprocessors in his doctoral thesis [9]. There are a lot of metrics to evaluate the processor workloads such as ratio of computation to memory access, ratio of computation to communication, basic block size and parallelism [2], [10]. However, few metrics can be used to identify computation-intensive applications and control-intensive applications exactly.

The definitions of control-intensive and computation-intensive applications are mostly qualitative in our survey. Guthaus et al. said that control-intensive applications have a much larger percentage of branch instructions and that computation-intensive applications have a larger percentage of integer or floating point ALU operations [11]. Eeckhout and Bosschere made a conclusion that multimedia applications are computation-intensive workloads because the percentage of computational operations in their instruction mix are higher than that for general-purpose workloads [12]. Oguike et al. thought that a computation-intensive application can be defined as any application of a single processor computer system where the arrival rate of processes into the processor queue is greater than the departure rate of the processes from the processor [13]. This definition tends to regard a CPU-intensive application as a computation-intensive application. John et al. analyzed the relationship between the computation instructions and memory access instructions on high performance RISC architectures [14]. Samuel Williams et al. gave a model named roofline which pointed the relationship between the operational intensity and memory access for multicore architectures [15]. However, there is little information available in their studies about revealing detailed characteristics of the relationship between computational and control operations of workloads.

As different applications have different characteristics, it is necessary to find an exact approach to identify computation-intensive applications and control-intensive applications. Unfortunately, we have not found any papers introducing a quantitative approach to identify them. In many literatures, the authors discussed the methods or algorithms to improve the performance and efficiency of the system based on the assumption that the applications are already known as computation-intensive or control-intensive without any quan-

Manuscript received May 26, 2013; revised November 20, 2013. This work is supported by Natural Science Foundation of Hubei Province of China (No. ZRZ0051) and Key Science and Technology Program of Wuhan (No. 201150699190).

Tianyong Ao is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China and is also with the School of Physics and Electronics, Henan University, Kaifeng, China (e-mail: tyaohust@gmail.com).

Pan Chen is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China (corresponding author, phone: +86-02787611245; e-mail: chenpan.maxview@gmail.com).

Zhangqing He is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China and is also with the School of Electrical and Electronic Engineering, Hubei University of Technology, Wuhan, China (e-mail: ivan_hee@126.com).

Kui Dai is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China (e-mail:daikui@mail.hust.edu.cn).

Xuecheng Zou is with the School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China (e-mail:estxczou@gmail.com).

(Revised on 2 December 2013)

titative analysis. There may be some reasons, for example, it is hard to identify them for lack of exact models or metrics. Therefore, it is worth studying an exact approach to identify them. It is not only useful in the design of computer architectures, but also worthy of software design, especially for hybrid computing systems.

Therefore, the motivations of this paper are as follows: firstly, to reveal the statistical characteristics of the relationship between computational operations and control operations in various applications; secondly, to explore a quantitative approach to identify computation-intensive applications and control-intensive applications exactly; thirdly, to provide some suggestions in the design of energy-efficient computer architecture and software.

To study the characteristics of various applications, the Ratio of Dynamic Computation to Control operations (RDCC) is chosen as a new metric, and the statistical characteristics of RDCC for a great many benchmarks are measured in various conditions. In addition, the Ratio of Static Computation to Control operations (RSCC) is measured to find whether or not there are dependencies between RDCC and RSCC. To the author's knowledge, this paper is a first step in analyzing the statistical characteristics of RDCC of various applications. The contributions of this paper are as follows:

1) The RDCC as a new metric for evaluating workload characteristics is proposed. The statistical characteristics of the RDCC of many benchmarks are measured in various conditions. The evaluation results indicate that some intrinsic properties of applications can be marked with the RDCC value. The results can be used to guide workloads scheduling in heterogeneous multi-cores.

2) A quantitative approach for identifying computation-intensive applications and control-intensive applications by the RDCC metric is presented. In addition, a reference classification with RDCC values is given based on the K-Means clustering analysis.

3) Several interesting results are found. The RDCC values of various benchmarks are quite different, but the RSCC values are much similar, and there are few dependencies between the RDCC and RSCC values. In addition, it is found that strong control-intensive applications usually accompany memory-intensive properties.

The remainder of this paper is structured as follows. The following section introduces the metrics and methodology selected to evaluate benchmark characteristics. The statistical results are shown in section III. The characteristics of RDCC are discussed in section IV. In section V, a quantitative approach to identify computation-intensive and control-intensive applications with the RDCC values is presented in detail, and other applications of RDCC are explored. Finally, some conclusions are drawn in section VI.

II. METRICS AND METHODOLOGY

A. Metrics

To analysis the characteristics of different applications, RDCC and RSCC are used as metrics. RDCC is defined as the ratio of the dynamic computational operations count (*Ncom_dy*) to the dynamic control operations count (*Ncon_dy*) based on the dynamic instruction mix of an application, as shown in (1). RSCC is defined as the ratio of the static computational operations count (*Ncom_st*) to the static control operations count (*Ncon_st*) based on the static instruction mix of an application, as shown in (2).

$$RDCC = Ncom\_dy/Ncon\_dy \tag{1}$$

$$RSCC = Ncom\_st/Ncon\_st \tag{2}$$

Alternatively, RDCC can also be calculated by the percentage of computational operations dividing by the percentage of control operations in the dynamic instruction mix. Likewise, RSCC can also be calculated by the percentage of computational operations dividing by the percentage of control operations in the static instruction mix.

The RDCC value can be used to mark the effective utilizations of computational resources and control resources of a processor while an application is being executed. The hardware resources of a processor can be classified into computational units, control units and memory access units by their functions. The computational units are used to process calculation class instructions, such as floating-point operations, arithmetic operations and logic operations. The control units are used to deal with control operations. In general, the hardware complexity of control units is growing with more control instructions and control-related technologies such as branch predictors, speculation, and dynamic scheduling being adopted in the processor. Memory access units refer to data load and store between the processor and the memory. This paper will focus on the computational and control operations first.

To analyze the relationship between computational operations and control operations with a more abstract level, both floating-point and integer computational operations are regarded as computational operations in the following. Likewise, branch operations and jump operations are esteemed as control operations.

B. Benchmarks

To characterize applications in terms of RDCC and RSCC as well as their effective characteristics for classifying applications, large amount of benchmarks from the SPEC CPU2006 and Mibench benchmark suites were used in our analysis. SPEC CPU2006 is the most widely used industry-standardized benchmark suite for general-purpose computers, which represents a different class of applications developed from real user applications [16]. Mibench is a free, commercially representative embedded benchmark suite [11]. A total number of 24 benchmarks from SPEC CPU2006 and 20 benchmarks from Mibench suite are successfully measured.

C. Tools

To get the RDCC and RSCC values of various benchmarks, the dynamic instruction mix and the static instruction mix of each benchmark are measured. The dynamic instruction mix is obtained from the gem5 simulator. The gem5 simulation infrastructure is the merger of the best aspects of the M5 and GEMS simulators [17]. It provides a highly configurable simulation framework, multiple Instruction Set Architectures (ISAs) and diverse CPU models, and it is an open source community tool. A statistical function to count each type executed instructions has been added into gem5

simulator by us. The static instruction mix is obtained by a script program from the static executable files which are generated by a compiler. The compiler is GCC4.3.2 and the Alpha ISA is used in this study.

D. Data Analysis

Two statistical data analysis techniques are used here to survey the characteristics of applications. First, the correlation coefficient is used to survey the dependent relationship between the results from different cases. The correlation coefficient is usually used to measure the strength and direction of the dependent relationship between two vectors. The higher absolute value of correlation coefficient indicates more dependencies between the two vectors. Second, the K-Means clustering analysis method is used to classify the applications. Clustering techniques are widely recommended tools for classification, and the K-Means is a famous clustering algorithm which aims to partition all observations into several clusters by their locations [17]. The K-Means clustering analysis used in this study is the function named Kmeans in Matlab2008.

III. RESULTS

In this section, the statistical characteristics of RDCC and RSCC of the benchmarks are presented. Firstly, the RDCC and RSCC values of the benchmarks from SPEC CPU2006 and Mibench are presented. Secondly, the relationship between the RDCC and the dynamic instruction mix of different benchmarks are investigated. Thirdly, the relationship between RDCC and basic block size are compared. Finally, the effects of inputs and optimizations on the RDCC values are compared. All the benchmarks are compiled by the compiler GCC 4.3.2 with -O3 optimization options if without special statements.

A. RDCC and RSCC

The RDCC and RSCC values of the benchmarks from SPEC CPU2006 are presented in Fig. 1. For the benchmarks from SPEC CPU2006, the ref input mode is used. The RDCC and RSCC values of the benchmarks from Mibench are shown in Fig. 2, and the large input mode is used for each benchmark. In each figure, the benchmarks are ranked by the RDCC values in ascending order from left to right.

As shown in Fig. 1 and Fig. 2, it can be found that the RDCC values of the various benchmarks are very different, and the values change from 1.2 to 41.4. However, the range of the RSCC values is small and most of the values are round 1.4. Furthermore, the correlation coefficient between the RDCC values and the RSCC values for all the benchmarks is only 0.2. This indicates that there are few dependencies between RDCC and RSCC for a benchmark. For the diversity of RDCC values, the characteristics of RDCC will be analyzed further in the following.

B. RDCC and Instruction Mix

To make more detailed analysis of RDCC and program behaviors, the dynamic instruction mixes of the benchmarks from SPEC CPU2006 and Mibench are presented in Fig. 3 and Fig. 4, respectively. The instructions are classified into four types: control transfer (including jump and branch), computation (including floating-point operations, integer arithmetic and logic operations), memory access operations (load/store) and miscellaneous (including trap barrier and system calls). As the percentages of miscellaneous is too little (all less than 0.5%) to be seen, they are not shown in the two figures.

Several observations can be found from these two graphs. First, if the RDCC value of a benchmark is lower, the percentage of dynamic control operations is greater in most cases. As shown in Fig. 3 and Fig. 4, when the RDCC values are less than 2, the percentages of control operations are nearly 20%; when the RDCC values are greater than 12, the percentages of control operations are less than 5%. Second, if the RDCC value is greater, the percentage of computational operations is greater too in most cases. For example, when the RDCC values are greater than 12, the percentage of computational operations is about 55% on average. In contrast, if the RDCC values are less than 2, the percentage of computational operations is only about 30% on average. Third, when the RDCC value is less than 2, the percentage of memory access operations is more than 50%, but when the RDCC is greater than 12, the percentage of memory access operations is 42% on average.

The correlation coefficient between the RDCC values and the percentage of computational operations is 0.63 in SPEC CPU2006, and the correlation coefficient between the RDCC values and the percentages of control operations is -0.84. The two correlation coefficients are 0.62 and -0.78 respectively in Mibench.

C. RDCC and Basic Block Size

Basic block size, which indicates how many instructions between a pair of branch instructions, is a significant program characteristic because the scheduling capability of a compiler is highly affected by it [10]. The RDCC values and the basic block size of the benchmarks from SPEC CPU2006 are shown in Fig. 5. The basic block size of a benchmark with lower RDCC value is smaller, but it is greater if the RDCC value is higher. There are also the same dependencies between the RDCC values and the basic block sizes of the benchmarks from Mibench. The correlation coefficient between the RDCC values and the basic block sizes is as high as 0.98, indicating that there are very strong dependencies between them. The RDCC value indicates how many computational instructions per branch, and basic block size indicates how many computational and memory access instructions per branch. Both of them reflect how many non-control instructions per branch. Therefore, there are strong dependencies between the RDCC values and the basic block sizes for an application.

D. Effects of the Input on RDCC

To observe the effects of different inputs on RDCC, the RDCC values of benchmarks are measured with different input data. There are two input sets for each benchmark in SPEC CPU2006. They are the ref and the test input modes. The former has larger input data than the latter. The RDCC values of benchmarks in both input modes are presented in

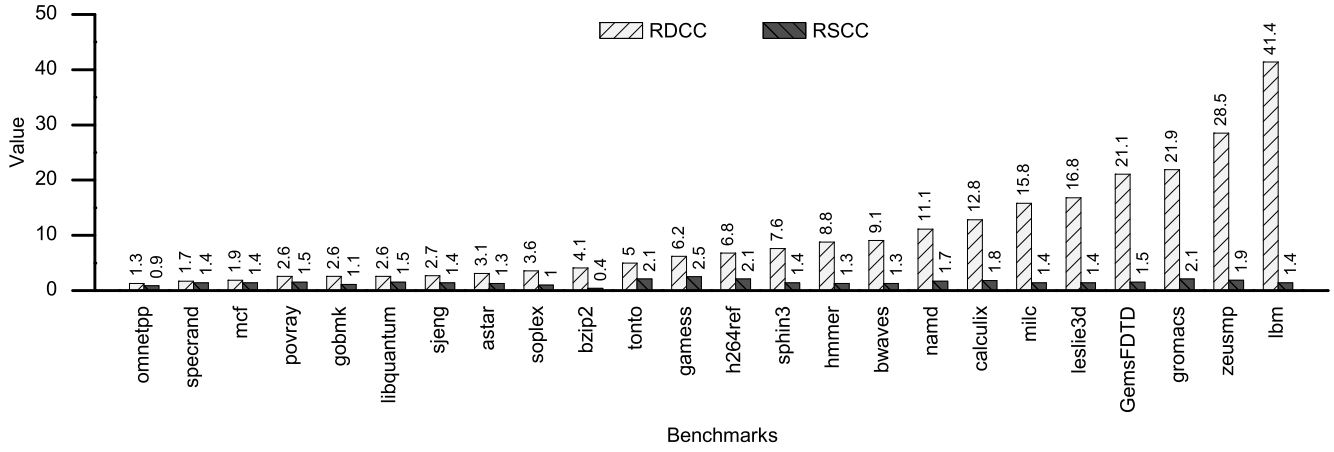

Fig. 1. RDCC and RSCC values of the benchmarks from SPEC CPU2006

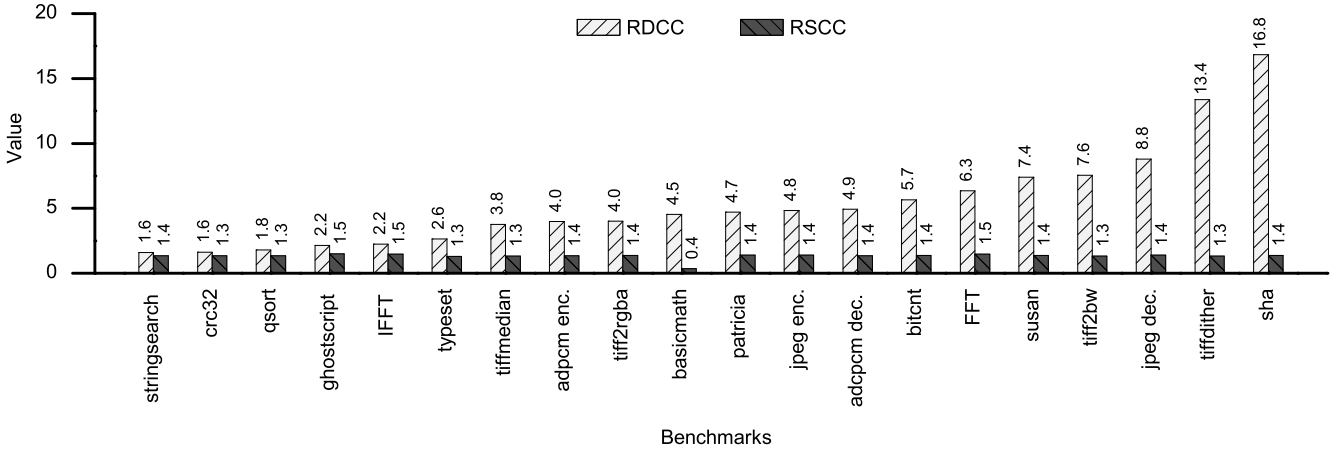

Fig. 2. RDCC and RSCC values of the benchmarks from Mibench

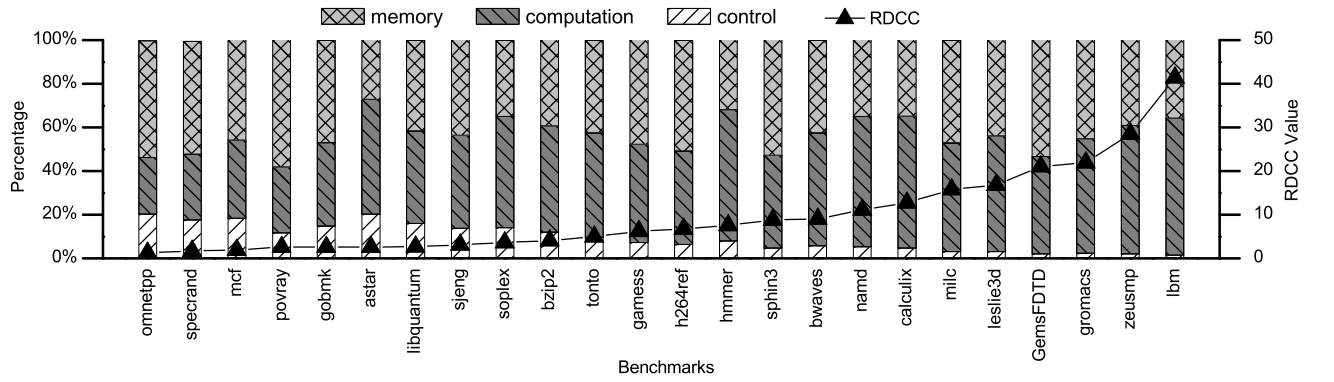

Fig. 3. RDCC and Instruction Mix for SPEC2006 CPU

Fig. 6. The benchmarks are ranked by the RDCC values in ascending order from left to right. As shown in this figure, the RDCC values are relatively stable, especially for the benchmarks with lower RDCC values, although there are a few differences between the two input modes. The RDCC values of the benchmarks with lower RDCC values in the ref mode are still lower in test mode in most cases, and the RDCC values of the benchmarks with higher values in test mode are still greater in ref mode. The correlation coefficient of the RDCC values in between test and ref input modes is as high as 0.77, indicating that there are few changes of RDCC value among different input sets in most cases.

E. Effects of the Optimization on RDCC

To observe the effects of optimization options on RDCC, the benchmarks are compiled with two different optimization options. ”-O1” is used in one case, and ”-O3” is used in the other case. The RDCC values of the benchmarks are measured in the both cases and presented in Fig. 7. The benchmarks are ranked by RDCC values in the O3 case in ascending order from left to right. As shown in this figure, the RDCC values are relatively stable although there are tiny

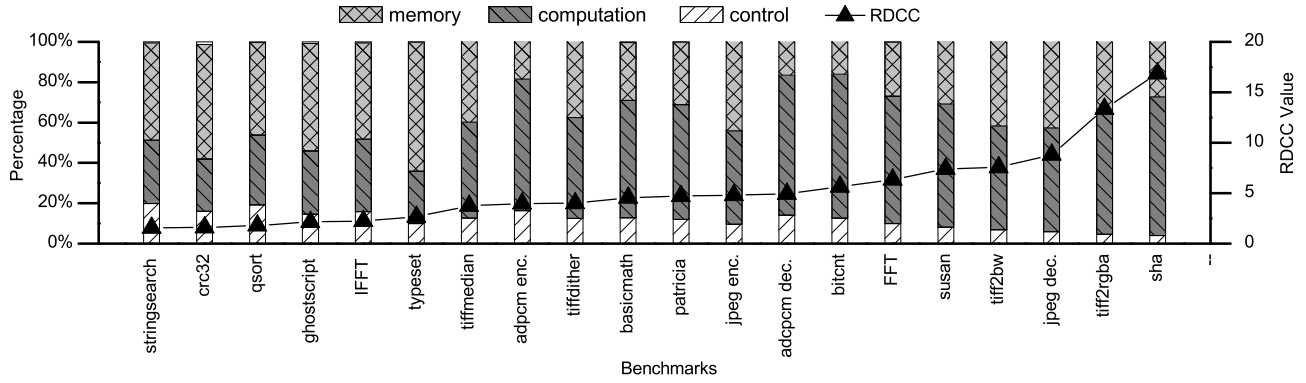

Fig. 4. RDCC and Instruction Mix for Mibench

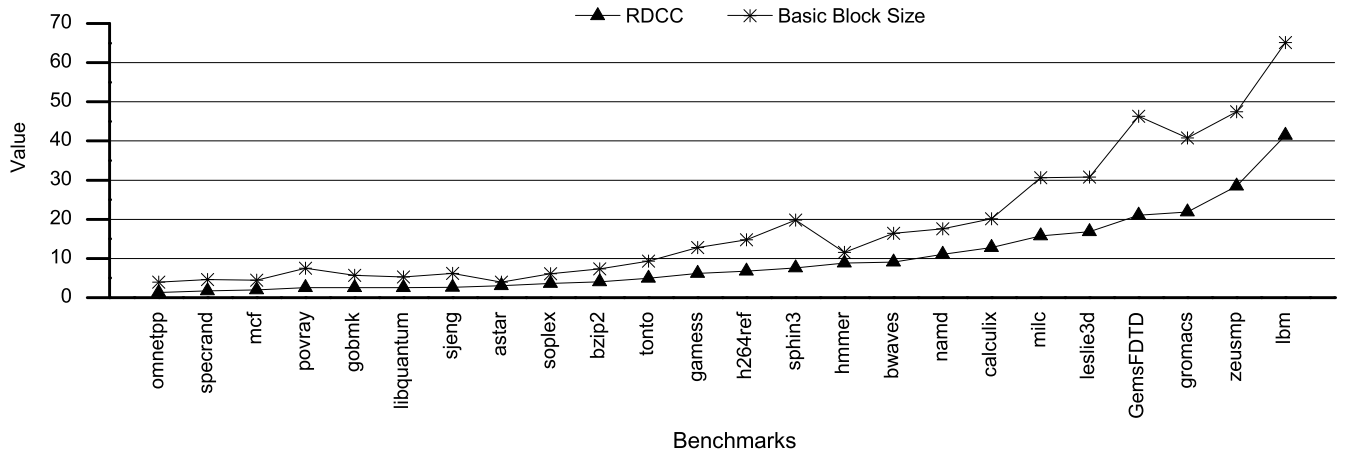

Fig. 5. RDCC and Basic Block Size

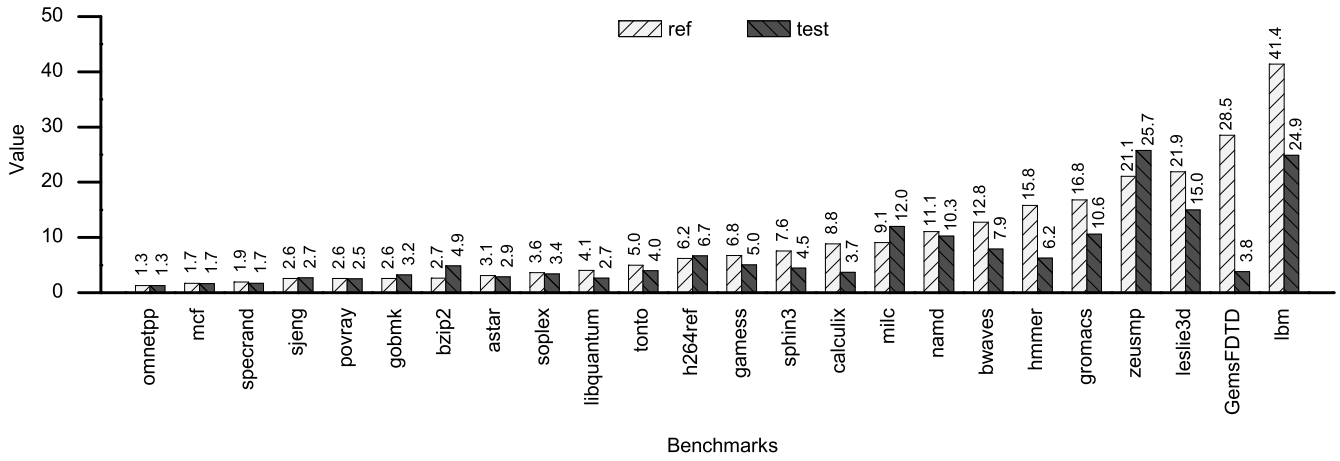

Fig. 6. Effects of the input on the RDCC values

changes between O1 and O3 cases. The benchmarks with lower RDCC values in O1 case are still with smaller RDCC values in O3 condition. Although there are big changes of the RDCC values for a few benchmarks such as leslie3d, GemsFDTD and lbm, the RDCC values of them in both cases are still higher than that of the most other benchmarks. For most benchmarks, there are few changes of the RDCC values between O1 and O3 optimization options. The correlation coefficient of the RDCC values between O1 and O3 cases is as high as 0.92, indicating that there are few changes of the RDCC value between different compiler optimizations for a benchmark.

F. RDCC and Loop unrolling

Loop unrolling is an important optimization technique, as it decreases the loop overhead and increases the opportunities for instruction-level parallelism [18]. In some cases, however, due to lack of further potential for optimizations and constraints of compensation codes for loop unrolling, the utilization of loop unrolling is not always effective [19]. In order to compare the effects of loop unrolling on the RDCC value with that on the performance speedup of

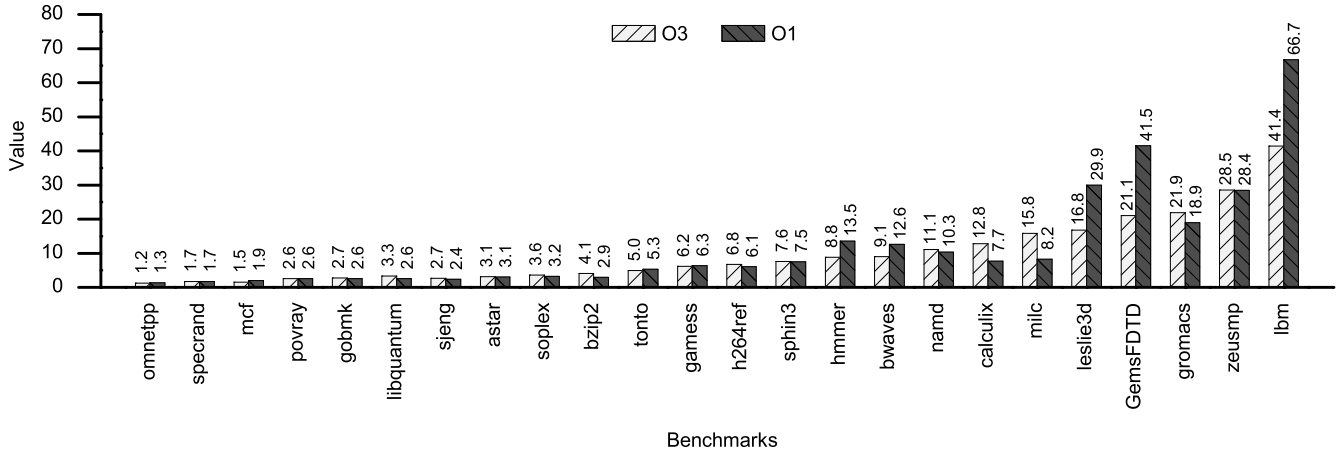

Fig. 7. Effects of Optimization on the RDCC values

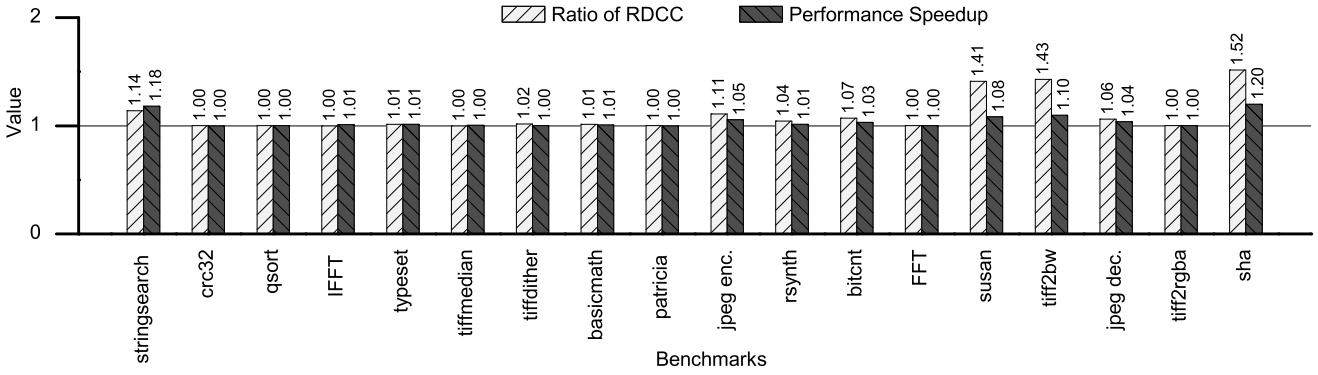

Fig. 8. Comparison of the effects of loop unrolling on the RDCC value and performance speedup

programs, the benchmarks from Mibench are compiled with two different optimizations. Only ”O3” optimization option is used in one case, but ”O3” and ”funroll-all-loops” are used simultaneously in the other case. The RDCC values and the executed times of the benchmarks are measured in both cases, and then the ratios of RDCC and the values of performance speedup are calculated. The performance speedup, which is used to reflect the performance improved by the loop unrolling, is calculated by the executed time of the program compiled without loop unrolling dividing by the executed time with loop unrolling. The ratio of RDCC is the RDCC value with loop unrolling divided by the RDCC value without this optimization. The comparison of them is shown in Fig. 8. As shown in this figure, it can be found that the loop unrolling optimization is not always effective for all programs. Moreover, the RDCC value of a program will be increased if loop unrolling is an effective optimization for the program in most cases. When affected by loop unrolling, the ratios of RDCC and values of performance speedup show strong correlation, as indicated by a high correlation coefficient—0.8. Therefore, it is possible to use the ratio of RDCC to evaluate the effects of loop unrolling on the performance speedup.

IV. CHARACTERISTICS OF RDCC

In this section, the factors influencing on the RDCC values are discussed firstly, and then the relationships between the RDCC values and energy-efficiency are discussed. Finally, the statistical characteristics of RDCC are analyzed and workload characteristics evaluated by RDCC are discussed.

A. Factors Affecting RDCC value

The RDCC value of an application may be affected by many factors. The factors can be categorized as follows.

1) Compilers and optimization techniques: The RDCC may be affected by different compilers and different optimization techniques which are used to generate the executable files. For example, the RDCC value will decrease when some expressions are simplified by the optimization techniques, but the RDCC value may increase if loop unrolling is adopted.

2) Input data: The dynamic instruction mix may be influenced by the input set because the execution traces of applications may depend on the input data.

3) ISAs: Different ISAs may have different functions for computational operations and control operations. For example, if an application with many multiply-accumulate operations, the RDCC value in the ISA included Multiply-Accumulate instructions may be less than in other ISA without this instruction.

4) Algorithms: There may be many different algorithms which can be used to deal with the same task, and there may be some differences of the RDCC values among the different algorithms.

B. RDCC and Energy-efficiency

The static power dissipation becomes an important part of total power consumption for deep submicron technology [20], [21]. In general, more functional units require more transistors, leading to more static power consumption. In addition, if some units do a lot of useless work, the energy efficiency of a processor will be reduced. Therefore, high energy-efficiency usually meets high effective utilization of resources in modern processors.

The Amdahl's law implies that the performance gain is limited by the percentage of potential speedup part [22]. For the applications with high RDCC value, the performance gain obtained from improving computational operations will be higher than that from improving control operations at the same cost, and the effective utilization of computational units is much higher than that of the control units. On the contrary, control operations will be done frequently when the applications with lower RDCC values are being executed. Meanwhile, the average basic block size of those applications is small and the parallel degree of computation is generally low.

The effective resource utilization of computational units and control units can be marked with the RDCC value while an application is being executed. The number of computational operations is larger than that of control operations while the application with high RDCC value is being executed. Consequently, the effective utilization of computational units is higher than that of the control units. Generally, the percentages of control operations in the applications with low RDCC values are more than that in the applications with high RDCC values. The effective utilization of control units for the applications with low RDCC values will be higher than that for other applications. The applications with low RDCC values will make high effective resource utilization of control units, but the applications with high RDCC values will make high effective resource utilization of computational units. Therefore, there are hardly any energy-efficiency gains if more parallelism computational units are used for the applications with low RDCC values. Likewise, better energy-efficiency gains will be hardly obtained if only more complex control technologies are used for applications with high RDCC values.

C. RDCC and Workload Characteristics Evaluation

Although the RDCC values may be affected by many factors, the statistical characteristics of RDCC can be found as follows.

1) The RDCC values of various applications are quite different. This indicates that RDCC could be used to mark different applications.

2) The RDCC values can be used to reflect the characteristics which can be marked with basic block size, because there are strong dependencies between the basic block size and the RDCC value for an application. RDCC value indicates how many computational operations per branch. Basic block size indicates the total number of computational operations and memory access operations per branch. When the RDCC value is greater, the basic block size is larger too. The basic block size can be used in many aspects. For example, basic block distribution is used to find the periodic behavior and simulation points in applications [23], and a framework for power estimation and reduction in multi-core architectures using basic block approach is present in [24]. In general, there is a higher probability to exploit instruction-level parallelism in larger basic block. Therefore, RDCC can be used to reveal the potential probability of instruction-level parallelism. In a word, RDCC can be used to reveal the characteristics of programs as basic block size in many aspects.

3) The RDCC value of one application is relatively stable despite different optimizations and inputs, especially for the applications with lower RDCC values. The correlation coefficients of the RDCC values from different conditions are greater than 0.77, indicating that the RDCC value is relatively stable among different conditions. In addition, it is found that the RDCC values of the applications are relatively stable between ARM and MIPS ISAs according to the instruction mix results in the literature [25]. The applications with high RDCC values in ARM ISA are also with high RDCC values in MIPS ISA.

4) The characteristics of applications can be marked with the RDCC values. For simplicity, the benchmarks with RDCC values less than 2 are named Low-RDCC type and the benchmarks with the RDCC values greater than 12 are named High-RDCC type. The comparison related to the instruction mixes between Low-RDCC type and High-RDCC type is shown in table I. The range and average values of the percentage of each instruction types are presented in the columns. The values of the low-RDCC type and the High-RDCC type are presented in the second and third columns, respectively. The values of all the benchmarks are shown in the last column.

TABLE I
THE COMPARISON RELATED TO INSTRUCTION MIX AMONG BENCHMARKS WITH DIFFERENT RDCC VALUES

Instruction	Percentage /range(average)		
Types	Low-RDCC	High-RDCC	All *
Control	16%-20%(18%)	1.5%-5%(3%)	1.5%-20%(11%)
Computation	25%-36%(31%)	46%-70%(57%)	25%-70%(48%)
Memory	46%-57%(51%)	27%-47%(40%)	16%-64%(41%)

* Low-RDCC represents the benchmarks whose RDCC values are less than 2; High-RDCC represents the benchmarks whose RDCC values are greater 12; ALL represents all the benchmarks.

The following statistical characteristics can be found from table I.

First, the Low-RDCC type benchmarks include more control operations and less computational operations than that for all the benchmarks on average. The percentage of control operations for Low-RDCC type (18% on average) is larger than that for all benchmarks (11% on average). The percentage of computational operations for Low-RDCC type (31% on average) is lower than that for all the benchmarks (48% on average).

Second, the High-RDCC type benchmarks consist of less control operations and more computational operations than that for all the benchmarks on average. The percentage of computational operations is larger for High-RDCC type—57% on average—than that for all the benchmarks. The percentage of control operations for High-RDCC type (31%

on average) is lower than that for all the benchmarks.

Furthermore, the Low-RDCC type benchmarks usually accompany with memory-intensive properties. The percentages of memory access operations for Low-RDCC type benchmarks (51% on average and 46% minimum value) are greater than that for all the benchmarks (41% on average).

The statistical characteristics of RDCC and results in section III indicate that the inherent properties of applications can be marked with the RDCC values. Therefore, RDCC can be used to evaluate characteristics of workloads. For example, the RDCC values can be used to reflect the relationship between control operations and computational operations for applications. The applications of RDCC will be explored in the following section.

V. APPLICATIONS OF RDCC

In this section, a quantitative approach to identify computation-intensive and control-intensive applications by RDCC values is discussed in detail and other applications of RDCC are introduced.

A. Identifying Computation-intensive and Control-intensive Applications by RDCC

1) RDCC and identifying Computation-intensive and Control-intensive Applications

From the qualitative perspective, control-intensive applications have a much larger percentage of control operations, but computation-intensive applications have a larger percentage of integer or floating point ALU operations than general applications [11].

According to the results in section III and table I, it can be found that the applications with lower RDCC values show more control-intensive characteristics and the applications with higher RDCC values show more computation-intensive characteristics. This conclusion not only can be deduced from the average values of control operations and computational operations between High-RDCC type or Low-RDCC type benchmarks and all the benchmarks, but also can be concluded from the range values of control operations or computational operations of the two type benchmarks.

The benchmarks with lower RDCC values show more control-intensive characteristics. The percentages of control operations for all the Low-RDCC type benchmarks are greater than 16%, which is greater than that for all the benchmarks (11% on average). Meanwhile, the percentages of computational operations for all the Low-RDCC type benchmarks are less than 36%, which is lower than that for all the benchmarks—48% on average. In other words, the benchmarks with lower RDCC values have more control operations and less computational operations than the average. Therefore, the applications with lower RDCC values can be seen as control-intensive ones.

The benchmarks with higher RDCC values show more computation-intensive characteristics. The percentages of control operations for all the High-RDCC type benchmarks are less than 5%. The percentages of computational operations for all those benchmarks are greater than 46%, and the average value of them is as high as 57%. In other words, the benchmarks with higher RDCC values include less control operations and more computational operations than the averages for all the benchmarks. Therefore, the applications with higher RDCC values can be seen as computation-intensive ones.

The benchmarks which include not only many computational operations but also many control operations such as adpcm dec., adpcm enc. and bitcnt have both control-intensive and computation-intensive characteristics. Therefore, it is reasonable that those benchmarks appear in the middle position of the ranking which is used to identify control-intensive and computation-intensive applications.

Therefore, RDCC can be used as a metric for identifying computation-intensive and control-intensive applications.

2) A Reference Classification

It is necessary to provide a reference classification for identifying an application whether computation-intensive or control-intensive when its RDCC value is obtained. Although different people may have various views on how to define the classification, it is obvious that an application with a higher RDCC value could not be seen as a strong control-intensive one. For example, lbm whose RDCC value is 41.4 should not be regarded as a control-intensive application, because the percentage of control operations in its dynamic instruction mix is only 1.5%. In order to reflect the degree of intensive properties of different applications accurately, the reference classification is divided into five types in this study. The five types are named as follows: Strong Control-intensive, Weak Control-intensive, Normal, Weak Computation-intensive and Strong Computation-intensive. This method will be more accurate than that only dividing two or three types.

To get the reference classification, K-Means clustering analysis method is adopted. Before doing the clustering analysis, the special mathematical characteristics of RDCC should be addressed. First, RDCC values are always greater than zero and are greater than 1 in most cases. Second, applications with values less than 1 can be regarded as control-intensive since they are relatively small. On the contrary, the RDCC values of computation-intensive applications are usually large and without an upper limit theoretically. Furthermore, it is reasonable that the applications with the RDCC values greater than a special number can be considered as computation-intensive ones. For those characteristics of RDCC, when the RDCC values are directly used as the inputs of the K-Means clustering analysis function, only one benchmark is considered as strong computation-intensive. This is a pitfall in this way. Fortunately, the natural logarithm can make the steep part of a curve become smoothed. It can be used to avoid the pitfalls.

The RDCC values are processed by the natural logarithm function firstly and then the results are used as one input of K-Means function. The other input which indicates the number of clusters is five. The boundary values of the five clusters are extracted based on the results of K-Means function. They are 0.67, 1.13, 1.85 and 2.75, respectively. The RDCC values corresponding to the boundary values are 2, 3, 6 and 16, respectively, which are gotten by the natural exponential function being applied to the boundary values and then rounded to integers. At last, we have gotten the reference classification with the RDCC values, as shown in table II.

According to the reference classification, when the RDCC

TABLE II
THE REFERENCE CLASSIFICATION

Types	The Range of RDCC value
Strong control-intensive	Less than 2
Weak control-intensive	2-3
Normal	3-6
Weak computation-intensive	6-16
Strong computation-intensive	Higher than16

TABLE III
RESULTS OF CLASSIFICATION

Types	Benchmarks (RDCC value)	Description
Strong control-intensive	omnetpp(1.3)	Discrete Event Simulation
	specrand(1.7)	Pseudorandom generator
	mcf(1.9)	Depot vehicle scheduling
Weak control-intensive	gobmk(2.6)	Game playing
	sjeng(2.7)	Game tree search
	povray(2.6)	Computer visualization
Normal	soplex(3.6)	Program Solver
	bzip2(4.0)	Compression
	tonto(4.9)	Quantum crystallography
Weak computation-intensive	h264ref(6.7)	Video compression
	sphin3(7.5)	Speech recognition
	hmmer (8.8)	Gene sequence searching
Strong computation-intensive	gromacs(21.9)	Molecular dynamics
	zeusmp(28.4)	Magnetohydrodynamics
	lbm(41.4)	Computational fluid dynamics

value of an application is less than 2, the application can be regarded as strong control-intensive. The range of the RDCC value for weak control-intensive applications is from 2 to 3. An application can be seen as the normal type if its RDCC value is greater than 3 and less than 6. If the RDCC value of an application is greater than 6 and less than 16, the application can be considered as weak computation-intensive. The applications are strong computation-intensive if their RDCC values are greater than 16.

3) Verification

To further verify the classification approach and effectiveness of the reference classification, many representative benchmarks are mapped into each type based on their RDCC values. The results are shown in table III.

The results of the classification for those benchmarks are very reasonable. This is reflected in a significantly higher percentage of control operations and lower percentage of computational operations of the benchmarks which are classified into strong control-intensive. The percentages of control operations for those benchmarks are nearly 20%, which is higher than that for all the benchmarks—11% on average. The percentage of computational operations is only about 30%, which is much lower than that for all the benchmarks—49% on average. The rationality is also reflected in the dynamic instruction mix of the benchmarks which are classified into strong computation-intensive. Those benchmarks include higher percentages of computational operations and lower percentages of control operations.

The classification results are consistent with the conventional understanding about computation-intensive and control-intensive applications. The benchmarks dealing with the random or discrete events are classified into strong control-intensive rank such as omnetpp, specrand, mcf. Weak control-intensive characteristics are shown in the applications referring to artificial intelligence. The benchmarks named h264ref, sphin3 and hmmer refer to multimedia or information processing applications. The benchmarks named gromacs, zeusmp and lbm relate to scientific computing applications. The more details of those benchmarks can be found in the literature [16]. The applications about multimedia, information processing and scientific computing are generally regarded as computation-intensive applications, and the intensity of computation-intensive for scientific computing applications is stronger than that for other categories [12], [26], [27], [28].

Therefore, the approach to identify computation-intensive and control-intensive applications by RDCC values is feasible, and the reference classification is reasonable, especially for the strong control-intensive and strong computation-intensive ones.

B. RDCC in Other Applications

1) Evaluation of Benchmark Suites: RDCC can be used as one of the indicators to evaluate whether the benchmark suites have similar characteristics with real workloads in the relationship between computational and control operations. One goal of the design of benchmark suites is that benchmark workloads should yield the same distribution of the utilization of system resources as real workloads [29]. It is essential that a subset of benchmarks used to evaluate the architecture, is well distributed within the target workload space rather than clustered in specific areas [30]. Good benchmark suits should have similar characteristics with the real workloads in the RDCC value. In addition, basic block size can be used to character behaviors of applications. For example, Sherwood et al. proposed basic block distribution analysis to find the representative parts of the entire program and they found that basic block could reflect many architectural metrics,such as IPC, branch miss rate, cache miss rate, value misprediction and address misprediction [23]. As the RDCC values highly depend on the basic block sizes, it is reasonable that RDCC is used to evaluate the behavior characteristics of benchmarks like basic block size.

2) Computer Architecture Design: RDCC can be used as a reference indicator which may be helpful for designers to make an appropriate choice in computer architecture design, especially for application specific instruction set processors and reconfigurable computing systems. Heterogeneous chip multiprocessors present unique opportunities for improving system throughput, reducing processor power, and mitigating Amdahl's law [31]. Many computer architects believe that heterogeneous multi-core architectures will be the dominant architectural design in the future. Amin Ansari et al. presented the advantages of adaptive asymmetric multiprocessors [32]. However, the architects usually face the problem that they must make a choice among many design

schemes to balance the performance and power dissipation [33], [34]. To improve energy-efficiency of one computer architecture, the effective resource utilization of function units should be much concerned and the characteristics of its workloads should be analyzed firstly. If the workloads are mainly computation-intensive applications, which are with high RDCC values, it will be reasonable to improve computational units. If the workloads are mainly strong control-intensive applications, which are with low RDCC values, the control ability should be enhanced in the system. The strong control-intensive applications are usually accompanied by memory-intensive attributes as shown in table I. This also offers a reference indicator for the design of the memory system.

3) Task Allocation and Scheduling: RDCC can be used as a reference indicator to improve the task allocation and scheduling for heterogeneous multi-core systems. Compared with the advantages of heterogeneous multi-cores, task scheduling in heterogeneous multi-cores is still a daunting challenge. The successful scheduling approach for heterogeneous multi-cores must be able to map workloads to the most appropriate core in performance and power. In heterogeneous multi-core system, different cores have different specialties. Some cores may be good at processing control-intensive tasks; others may be good at executing computation-intensive tasks. Making wrong scheduling decisions can lead to suboptimal performance and energy-efficiency. To address this scheduling problem, the indicators to guide workload scheduling should be chosen. For example, workload memory intensity, CPI, MLP and ILP profile information are used as indicators to guide workload scheduling [35], [36]. Koufaty et al. proposed the bias scheduling for heterogeneous systems by key metrics that characterize an application bias [37]. As the RDCC values can reflect the characteristics of workloads in control-intensive and computation-intensive, it is reasonable that RDCC is used as a new indicator to guide workload scheduling in heterogeneous multi-cores. The RDCC values can be recorded by offline or online profiling. The scheduling objects can be different levels such as functions, threads, processes or whole programs. The tasks with higher RDCC values will be processed by the cores with strong computing ability and the tasks with lower RDCC values will be allocated to the cores with strong control ability.

4) Compiler Design for Reconfigurable Computing: Compilers for reconfigurable computing are widely used to accelerate applications. Automatic mapping of computations to reconfigurable computing architectures represents a relatively new and very promising area of research [18]. It is very necessary to identify computation-intensive tasks and control-intensive tasks for choosing appropriate speedup methods in the compilers. RDCC can be used as one of the indicators to identify computation-intensive and control-intensive functions or tasks based on profiling texts.

VI. CONCLUSION

As a fundamental part of computer architecture research, better metrics and methods for evaluating workloads are continually needed. This paper presents a new metric named RDCC and the characteristics of the RDCC values of a great many benchmarks in various conditions are explored. The results indicate that many intrinsic properties of workloads can be marked with the RDCC values. The RDCC can be used in the following aspects at least: 1) to identify the computation-intensive and control-intensive applications; 2) to evaluate the effective resource utilizations of computational units and control units of a processor for given workloads; 3) to evaluate whether benchmark suites have similar characteristics with real workloads; 4) to reflect behaviors of workloads and to guide workload scheduling in heterogeneous multi-cores. In addition, this paper presents a novel quantitative approach to identify computation-intensive and control-intensive applications with only one metric—RDCC, and gives a reference classification which is obtained by K-Means clustering analysis. The RDCC gives a new indicator to evaluate application characteristics, and it will be much helpful for processor architecture design, the compiler and software optimization, espacially for heterogeneous multi-core systems. Therefore, the RDCC could be adapted to be a metric to identify and evaluate the characteristics of applications. It may be worth studying how to use the RDCC to improve task scheduling for heterogeneous systems and hardware-software partitioning for reconfigurable computing systems.

REFERENCES

[1] K. Hoste and L. Eeckhout, "Microarchitecture-independent workload characterization," *Micro, IEEE*, vol. 27, no. 3, pp. 63–72, 2007.

[2] L. K. John, P. Vasudevan, and J. Sabarinathan, "Workload characterization: Motivation, goals and methodology," in *Workload Characterization: Methodology and Case Studies*. IEEE, 1999, pp. 3–14.

[3] H. Singh, L. Ming-Hau, L. Guangming, F. J. Kurdahi, N. Bagherzadeh, and E. M. Chaves Filho, "Morphosys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *Computers, IEEE Transactions on*, vol. 49, no. 5, pp. 465–481, 2000.

[4] S. Che, J. Li, J. W. Sheaffer, K. Skadron, and J. Lach, "Accelerating compute-intensive applications with gpus and fpgas," in *Application Specific Processors, 2008. SASP 2008. Symposium on*. IEEE, 2008, pp. 101–107.

[5] M. F. Dossis, "Automatic generation of massively parallel hardware from control-intensive sequential programs," in *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*. IEEE, 2010, pp. 98–103.

[6] K. Skadron, M. Martonosi, D. I. August, M. D. Hill, D. J. Lilja, and V. S. Pai, "Challenges in computer architecture evaluation," *Computer*, vol. 36, no. 8, pp. 30–36, 2003.

[7] F. J. Cazorla, A. Pajuelo, O. J. Santana, E. Fernández, and M. Valero, "On the problem of evaluating the performance of multiprogrammed workloads," *Computers, IEEE Transactions on*, vol. 59, no. 12, pp. 1722–1728, 2010.

[8] P. Michaud, "Demystifying multicore throughput metrics," *Computer Architecture Letters*, vol. PP, no. 99, pp. 1–4, 2012.

[9] M. N. Otoom, "Capacity metric for chip heterogeneous multiprocessors," Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2012.

[10] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Morgan Kaufmann, 2011.

[11] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001 IEEE International Workshop on*. IEEE, 2001, pp. 3–14.

[12] L. Eeckhout and K. De Bosschere, "Quantifying behavioral differences between multimedia and general-purpose workloads," *Journal of Systems Architecture*, vol. 48, no. 6-7, pp. 199–220, 2003.

[13] O. E. Oguike, D. U. Ebem, M. N. Agu, S. C. Echezona, H. O. D. Longe, and O. Abass, "Performance metrics of compute intensive applications of a single processor computer system," in *Modelling Symposium (AMS), 2011 Fifth Asia*, 2011, pp. 243–247.

[14] L. K. John, V. Reddy, P. T. Hulina, and L. D. Coraor, "Program balance and its impact on high performance risc architectures," in *High-Performance Computer Architecture, 1995. Proceedings., First IEEE Symposium on*. IEEE, 1995, pp. 370–379.

[15] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Communications of the Acm*, vol. 52, no. 4, pp. 65–76, 2009.

[16] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, 2006.

[17] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.

[18] J. M. P. Cardoso, P. C. Diniz, and M. Weinhardt, "Compiling for reconfigurable computing: A survey," *ACM Comput. Surv.*, vol. 42, no. 4, pp. 1–65, 2010.

[19] S. M. Freudenberger, T. R. Gross, and P. G. Lowney, "Avoidance and suppression of compensation code in a trace scheduling compiler," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 16, no. 4, pp. 1156–1214, 1994.

[20] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68–75, 2003.

[21] M. Monchiero, R. Canal, and A. Gonzalez, "Power/performance/thermal design-space exploration for multicore architectures," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 5, pp. 666–681, 2008.

[22] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, no. 7, pp. 33–38, 2008.

[23] T. Sherwood, E. Perelman, and B. Calder, "Basic block distribution analysis to find periodic behavior and simulation points in applications," in *Parallel Architectures and Compilation Techniques, 2001. Proceedings. 2001 International Conference on*. IEEE, 2001, pp. 3–14.

[24] M. R. Babu, P. V. Krishna, and M. Khalid, "A framework for power estimation and reduction in multi–core architectures using basic block approach," *International Journal of Communication Networks and Distributed Systems*, vol. 10, no. 1, pp. 40–51, 2013.

[25] M. E. Salehi, S. M. Fakhraie, and A. Yazdanbakhsh, "Instruction set architectural guidelines for embedded packet-processing engines," *Journal of Systems Architecture*, vol. 58, no. 3, pp. 112–125, 2012.

[26] F. Barat, M. Jayapala, P. Op de Beeck, and G. Deconinck, "Reconfigurable instruction set processors: an implementation platform for interactive multimedia applications," in *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on*, vol. 1. IEEE, pp. 481–485.

[27] U. Srinivasan, P.-S. Chen, Q. Diao, C.-C. Lim, E. Li, Y. Chen, R. Ju, and Y. Zhang, "Characterization and analysis of hmmer and svm-rfe parallel bioinformatics applications," in *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*. IEEE, 2005, pp. 87–98.

[28] L. Eeckhout, T. Vander Aa, B. Goeman, H. Vandierendonck, R. Lauwereins, and K. De Bosschere, "Application domains for fixed-length block structured architectures," in *Computer Systems Architecture Conference, 2001. ACSAC 2001. Proceedings. 6th Australasian*, pp. 35–44.

[29] J. J. Dujmovic, "Evaluation and design of benchmark suites," *State-of-the-Art in Performance Modeling and Simulation: Theory, Techniques, and Tutorials*, pp. 287–323, 1998.

[30] A. Phansalkar, A. Joshi, L. Eeckhout, and L. K. John, "Measuring program similarity: Experiments with spec cpu benchmark suites," in *Performance Analysis of Systems and Software, ISPASS 2005. IEEE International Symposium on*. IEEE, 2005, pp. 10–20.

[31] R. Kumar, D. M. Tullsen, N. P. Jouppi, and P. Ranganathan, "Heterogeneous chip multiprocessors," *Computer*, vol. 38, no. 11, pp. 32–38, 2005.

[32] A. Ansari, S. Feng, S. Gupta, J. Torrellas, and S. A. Mahlke, "Illusionist: Transforming lightweight cores into aggressive cores on demand." in *HPCA*, 2013, pp. 436–447.

[33] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas, "Single-isa heterogeneous multi-core architectures for multithreaded workload performance," in *Proceedings of the 31st annual international symposium on Computer architecture*, ser. ISCA '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 64–76.

[34] M. DeVuyst, A. Venkat, and D. M. Tullsen, "Execution migration in a heterogeneous-isa chip multiprocessor," in *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1. ACM, 2012, pp. 261–272.

[35] J. C. Saez, M. Prieto, A. Fedorova, and S. Blagodurov, "A comprehensive scheduler for asymmetric multicore systems," in *Proceedings of the 5th European conference on Computer systems*, ser. EuroSys '10. New York, NY, USA: ACM, 2010, pp. 139–152.

[36] K. Van Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, and J. Emer, "Scheduling heterogeneous multi-cores through performance impact estimation (pie)," in *Proceedings of the 39th International Symposium on Computer Architecture*. IEEE Press, 2012, pp. 213–224.

[37] D. Koufaty, D. Reddy, and S. Hahn, "Bias scheduling in heterogeneous multi-core architectures," in *Proceedings of the 5th European conference on Computer systems*, ser. EuroSys '10. New York, NY, USA: ACM, 2010, pp. 125–138.