Scheduling Multithreaded Applications onto Heterogeneous Composite Cores Architecture

Hossein Sayadi, Houman Homayoun
Department of Electrical and Computer Engineering
George Mason University
Fairfax, VA, USA
{hsayadi, hhomayou}@gmu.edu

Abstract— **Composite Cores Architecture (CCA), a class of dynamic heterogeneous architectures, enables the system to construct the right core at run-time for each application by composing cores together to build larger core or decomposing a large core into multiple smaller cores. While this architecture provides more flexibility for the running application to find the best run-time settings to maximize energy-efficiency, due to interdependence of various tuning parameters such as the type of the core, run-time voltage and frequency and the number of threads, it makes it more challenging for scheduling. Past research mainly addressed the scheduling problem in composite cores architecture by looking at one or two of these tuning parameters. However, as we will show in this paper, it is important to concurrently optimize and fine-tune these parameters to harness the power of heterogeneity in this emerging class of architecture. In addition, most previous work on CCA mainly studied traditional single threaded CPU applications. In this work, we investigate the scheduling challenges for multithreaded applications in CCA. First, through methodical investigation of power and performance results, we characterize various multithreaded applications on a CCA which can be composed into few big or many little cores and demonstrate how the interplay among various application, system, and architecture level parameters affect the performance and energy-efficiency. Furthermore, based on characterization results, a highly accurate regression-based model for energy-efficiency prediction is developed to guide the scheduling decision. Using the predictive model, we developed a scheduling scheme for effective mapping of multithreaded applications onto CCA. The results show that the proposed scheduling scheme on average achieves close to 94% efficiency as compared to the Oracle scheduling.**

Keywords— Heterogeneous architecture, Composite cores, Energy-efficiency, Scheduling, Multithreaded applications, Regression model

I. Introduction

Heterogeneous multicore processors offer significant advantages over homogeneous designs in terms of both performance and power by executing workloads on the most appropriate core type. By running multithreaded applications on heterogeneous architecture, each thread is able to run on a core that matches its resource needs more closely than one-size-fits-all solution [11,27]. Commercially available heterogeneous architectures include Intel Quick IA [6], ARM's big.LITTLE [8], and Nvidia Tegra 3 [7] that integrates a high performance big core with low power little core on a single chip.

Although heterogeneous architectures take advantage of application characteristics variation at run-time and improve energy-efficiency, they create unique challenges in effective mapping of threads to cores. As the core configurations in heterogeneous multicores become more diverse, they become more difficult to program effectively. In other words, the effectiveness of heterogeneous architectures significantly depends on the scheduling policy and how efficiently we can allocate applications to the most appropriate processing core [1,3,4,9,28,29]. Applying ineffective scheduling decisions can lead to performance degradation and excess power consumption in such architecture [1,11,17,24].

Composite cores architectures can provide further benefits by allowing the system to construct a right core for each running application. Several designs have been proposed that provide some level of heterogeneity. These proposals include Core Fusion [17], TFlex [18] and Composite Cores [1,11]. In [11,17] the concept of composite cores is proposed where a big core architecture can dynamically be decomposed into a smaller little core architecture. The authors in [1] adapted the concept of composite cores in 3D by further enabling the core composition and decomposition at a low granularity of processor building blocks such as register file and load and store queue. Their proposed architecture allows multiple smaller cores to be composed together to build a larger core or vice versa, as needed. While composite cores architecture provides more opportunity to construct the right core for the running applications, it is making the scheduling a difficult problem.

Previous studies have mainly examined the advantages of using single threaded applications in CCA [1,2,13,17]. However, running multithreaded applications on CCA and composing ideal processor architecture for energy-efficiency is a more challenging problem, considering the possible number of cores and threads, type of core microarchitecture, or combinations of core types. Furthermore, the challenge of how many and what type of core to compose for each multithreaded application becomes even more complicated considering the impact of other tuning parameters on energy-efficiency such as operating voltage and frequency. In this work, we are exploring the CCA in the context of multithreaded applications. In particular, we investigate the scheduling challenges for multithreaded applications for CCA. We focus on the benefits of running multithreaded applications and how this architecture provides opportunities to improve the energy-efficiency.

The main challenge for scheduling is to effectively tune system, architecture and application level parameters in CCA when running multithreaded applications. The particular parameters that are critical to performance and power and are considered in this work include core type, voltage/frequency settings and the number of running threads. While there has been number of work on mapping applications to heterogeneous architectures, to the best of our knowledge, there has been no prior effort to analyze and characterize multithreaded applications on composite cores with its unique architecture and develop effective scheduling scheme. In addition, previous studies on mapping applications to multicore architectures have focused primarily on 1) homogeneous architectures 2) static heterogeneous architectures where the number and type of cores are fixed at design time, and 3) configuring individual or a subgroup of tuning parameters at a time, such as application's thread counts [5,9,21,22], voltage/frequency [2,3], or core type [4,8,11,17,18,19,22] and

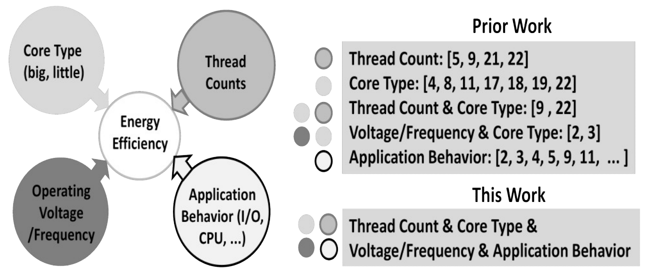

Core Type (big, little)
Thread Counts
Energy Efficiency
Operating Voltage /Frequency
Application Behavior (I/O, CPU, ...)
Prior Work
Thread Count: [5, 9, 21, 22]
Core Type: [4, 8, 11, 17, 18, 19, 22]
Thread Count & Core Type: [9 , 22]
Voltage/Frequency & Core Type: [2, 3]
Application Behavior: [2, 3, 4, 5, 9, 11, ...]
This Work
Thread Count & Core Type & Voltage/Frequency & Application Behavior

Fig.1. Tuning parameters influencing energy-efficiency in composite cores architecture and prior work on scheduling using these tuning parameters

they have ignored the interplay among all of them. This study indicates that these parameters individually, while important, do not make a truly optimum configuration to achieve the best energy-efficiency on a CCA. The best configuration for a multithreaded application can be effectively found, only when these parameters are jointly optimized. Fig. 1 illustrates the tuning parameters influencing the scheduling decision in CCA. Also, recent prior work as well as the contribution of our work is shown in this figure.

In this paper, through methodical investigation of power and performance, and comprehensive system and microarchitectural level analysis, we characterize various multithreaded applications from SPLASH-2 [14] and PARSEC [16] multithreaded benchmark suites on CCA to understand the power and performance trade-offs offered by various configuration parameters and to find how the interplay of these parameters affects the energy-efficiency. Our study is focusing on a CCA where many little cores (base) can be configured into few big cores (composed) and vice versa. Also, the metric that we use to characterize energy-efficiency is the energy delay product (EDP). The experimental results support that there is no unique solution for the best configuration for different applications. Given the dispersed pattern of optimum configuration, we have developed an accurate energy-efficiency prediction model to guide scheduling of multithreaded applications and fine-tuning parameters in order to maximize the energy-efficiency.

Contributions: This paper in brief makes the following contributions:

- We investigate the performance and energy-efficiency sensitivity of various standard multithreaded benchmarks with various frequency settings, core types (base vs. composed), and number of threads in heterogeneous composite cores architecture. We observed that application performance and energy-efficiency sensitivity to one optimization parameter is significantly influenced by other optimization parameters. For instance, increasing the thread counts reduces the performance sensitivity to operating frequency or multithreaded application's performance on base core shows to be more sensitive to frequency variation than on composed core.
- We evaluate the interplay of tuning parameters on performance and energy-efficiency in our studied heterogeneous CCA. The studied parameters that are critical to performance as well as power and energy-efficiency are core type, voltage/frequency settings and the number of running threads at microarchitecture, system and application levels, respectively.
- Based on conducted workloads characterization, a linear regression model is proposed for energy-efficiency prediction of various configurations of application, system and architecture level parameters to guide the scheduling decisions. We evaluate two different regression models and determine that the quantile linear regression model [20] leads to an accurate estimation of the energy-efficiency.

The remainder of this paper is organized as follows. The background and previous work are briefly discussed in section II. The methodology and experimental setup details are given in Section III. Section IV presents the characterization results and provides the performance and energy-efficiency analysis of multithreaded applications on CCA. Then, the energy-efficient scheduling model is proposed in section V. Finally, Section VI presents the conclusion of this study.

II. Background and Related Work

A. Heterogeneous Architectures

The static heterogeneous architecture in [9] and [23] enables efficient thread-to-core mapping and permits a change in the mapping across phases of execution through thread migration. Prior research has shown that the potential benefit of a static heterogeneous architecture is greater with fine-grained thread migration than with coarse-grain migration [11]. In [10], an Intel Xeon is integrated with an Atom processor. Code instrumentation is used at the function or loop level to schedule different phases of the application on each processor. However, the separate core and memory subsystems in static heterogeneous architectures incur power and performance overheads for application migration, which makes dynamic mapping ineffective for fine-grained migration [11].

Unlike static heterogeneous architecture where the number and type of cores are fixed at run-time, dynamic heterogeneous architectures can be configured at run-time. This provides more opportunity to map an application to a core which matches its resource needs more closely. Some of the first efforts to provide this kind of heterogeneity include Core Fusion [17] and TFlex [18]. Composite cores proposed a dynamic heterogeneous architecture where a big core can dynamically be decomposed into a smaller core [11]. The work in [1] and [2] extended the concept of composite cores into 3D stacking which enables fine-grain sharing of resources between cores on a stacked chip multiprocessor architecture. Their proposed architecture permits multiple smaller cores to be composed together making a larger core, given the performance and energy requirements of the running application. Previous work on dynamic heterogeneous architecture and specifically on composite core, has mainly studied mapping of single threaded applications. This work is different as it mainly focuses on multithreaded applications and how they would benefit from such architecture to maximize the energy-efficiency.

B. Scheduling Challenges in Heterogeneous Architectures

As mentioned before, a main challenge for heterogeneous architectures is the mapping and scheduling decision, which finds the most efficient application-to-core match at run-time. The work in [9] and [23] address the problem of dynamic thread mapping in static heterogeneous many-core systems. Prior research aimed to maximize performance under power constraints [3,9,12,23]. Our work is different as it first targets dynamic heterogeneous architectures where core size can be adapted at run-time, and second it aims to maximize the energy-efficiency by reducing the EDP. It is important to note that the power and performance of an application on different cores at various frequencies must be known for proper mapping. Traditional designs suggest selecting the best core based on a small sampling of applications on each core [21]. Other techniques [3,11,12,25,26], estimate core performance and adapt the resources without running applications on a particular core type using learning models. The work in [4] and [11] provide a model for performance estimation on two core types (i.e., big and little cores). The complexity of application mapping on a heterogeneous architecture increases exponentially by increasing number of core types and applications [3,23,24].

While previous studies have mainly examined the advantages of using single threaded applications, there is no prior work on effectively mapping multithreaded applications onto heterogeneous composite cores architecture. There have been several studies on mapping multithreaded applications on homogeneous architectures. The work in [5] suggested a framework called "Thread Reinforcer" to determine the appropriate number of threads for a multithreaded application on a homogeneous architecture. It examines the mapping between number of threads and number of cores to find the optimal or near optimal number of threads to minimize the execution time. The research in [4] proposed a scheduling method to predict application to core mappings that enhances performance. Using profiling parameters, it estimates performance and examines whether the workload needs to run on different core type. The work in [9] proposed a mapping strategy for multithreaded applications on static heterogeneous multicore architecture by initializing a maximum throughput mapping and iteratively performing a thread swap on adjacent types of cores until the power constraint is met. The research in [3] took a closer look at joint optimization of voltage and frequency as well as the microarchitecture. It proposed a platform, which is capable of scaling resources, i.e., bandwidth, capacity, voltage, and frequency, based on single-threaded application performance requirements at run-time while reducing EDP.

As shown in Fig.1, our paper addresses the importance of joint cross-layer tuning of core type, frequency, and thread counts to maximize the energy-efficiency of multithreaded application running on CCA. It will then propose an accurate regression model for energy-efficiency prediction to guide the scheduling decision in a CCA.

III. Experimental Setup and Methodology

This section provides the details of our experimental setup. We use Sniper [13] version 6.1, a parallel, high speed and cycle-accurate x86 simulator for multicore systems for simulation. McPAT [15] is integrated with Sniper and is used to obtain power consumption results. We study SPLASH-2 and PARSEC multithreaded benchmark suites for simulation. We use R, a free software environment for statistical analysis and developing energy-efficiency prediction models. For architectural simulation, we modeled a heterogeneous composite cores architecture based on the proposed work in [17,18]. Our study is focusing on a CCA where two little cores (base) can be configured into one big core (composed) and vice versa. We collect performance counters data on each architecture for characterization and drive the scheduling and mapping algorithms. We use these data to extract and evaluate the actual behavior of applications (I/O, CPU or memory intensive) for predicting energy-efficiency and assist scheduling decision.

TABLE I. Architectural specification

Microarch. Parameter	**Base**	**Composed**
Number of Cores	4/8	2/4
Issue-Commit width	2	4
INT instruction queue	16 entries	32 entries
FP instruction queue	16 entries	32 entries
Reorder buffer	32 entries	64 entries
Branch penalty	7cyc	14cyc
iL1-dL1 Cache	16KB/4-way/2cyc	32KB/4-way/2cyc
L2 Cache	4MB/8-way/32cyc	4MB/8-way/32cyc

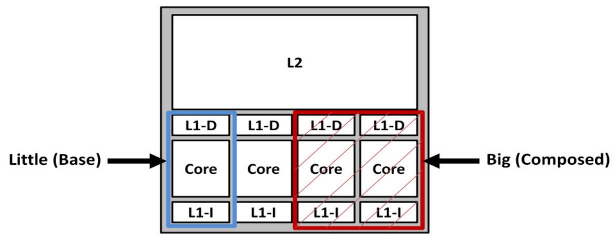


Fig. 2. Conceptual structure of a four core CCA [17]

Table I shows the microarchitectural configuration of base and composed core of CCA in our experiment. In this architecture, the L1 Cache size is 16KB for base core and 32KB for composed core which is constructed by composing two base cores. Contention and latency for composing the core includes two cycles wire delays for cross-core communication [1, 17]. Fig. 2 provides a conceptual overview of a four core CCA. In this paper, we investigate two baseline heterogeneous CCAs which consist of multiple base and composed cores: 1) 8base/4comp, and 2) 4base/2comp. It is important to note that for benchmark simulation we applied the binding (one-thread-per-core) model with #threads == #cores to maximize the performance of multithreaded applications [4, 5].

IV. Characterization Results

In this section, we evaluate the applications performance and energy-efficiency sensitivity to tuning parameters of operating frequency, number of running threads, and the choice of microarchitectures (base vs. composed) in heterogeneous composite cores architecture. The studied parameters not only directly impact the power and performance of the processor, but they also influence one another. The optimal system and microarchitecture configuration to maximize energy-efficiency varies based on the characteristics of the application, which all together influence the best tuning parameters. Therefore, it is essential to investigate the interplay of these parameters to guide the optimal mapping and scheduling decision in CCA. These observations form the basis for developing the prediction model presented in section V.

Note that the entire set of benchmark analysis results is quite extensive. Therefore, due to space limitations we only present the results for a limited number of representative benchmarks shown in Fig. 3. This figure depicts the overall performance in terms of execution time (represented as a bar graph) and EDP results (represented as a line graph) for four different multithreaded benchmarks across different core types, frequencies and number of threads. In this section, first we discuss the impact of changing each parameter on energy-efficiency and next we perform a joint analysis to investigate the interplay of studied parameters and their influence on energy-efficiency in heterogeneous CCA.

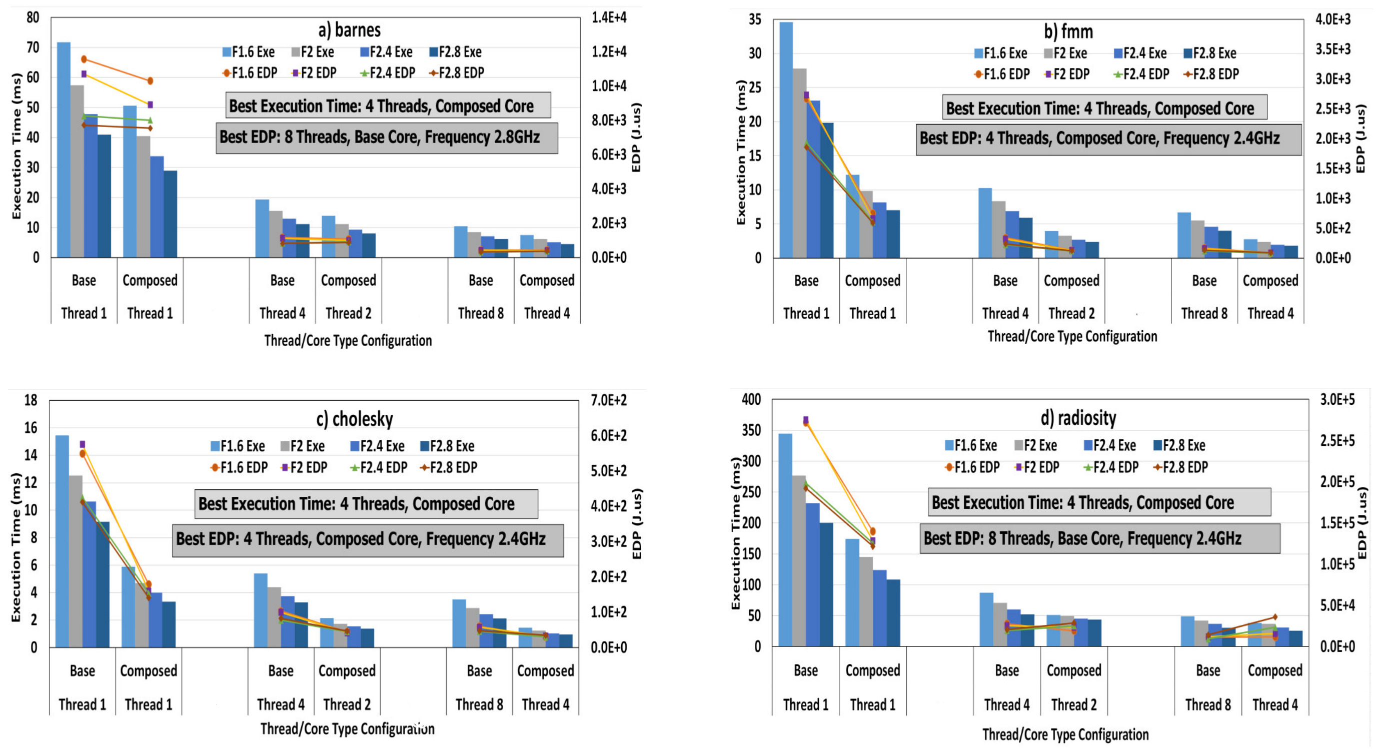


Fig. 3. Execution Time and EDP of a) barnes, b) fmm, c) cholesky, d) radiosity with various Core Types, Threads, Frequencies

A. Frequency Sensitivity

All benchmarks were simulated using a baseline composed core running with only a single thread. The operating frequency is swept from 1.6 GHz to 2. 8GHz with a step of 400MHz and the voltage is changed between 0.7, 0.8, 0.9, and 1V, respectively. As can be seen in Fig.3, some benchmarks are very sensitive to changing the frequency. For instance, in *fmm* and *cholesky* reducing the frequency almost linearly reduces the overall performance. Overall, as expected, as the frequency increases, the performance increases accordingly. The EDP results show that the higher frequency results in lower EDP.

The next observation is that increasing the number of threads interestingly reduces the sensitivity to frequency. In other words, increasing the number of running threads increases the performance gain due to parallelization. Consequently, the overall performance as the number of threads increases is more influenced by the speedup gain as a result of parallelism rather than operating at higher frequency. Moreover, the results show that the base core is more sensitive to frequency scaling than the composed cores. This is also an interesting observation as the composed core has a large pipeline, allowing it to tolerate performance cost due to alterations in cache access latency as a result of frequency scaling. Note that changing clock frequency changes the number of cycles it takes for the processor to communicate with the cache.

B. Core Type Sensitivity

In this section, the results are reported for a baseline configuration with a core running a single thread at the highest frequency of 2.8 GHz and operating voltage of 1V. The changing parameter is the core type, which varies between a base core and a composed core architecture. Core type demonstrates constant behavior with regards to EDP. As shown in Fig. 3, there is a clear gap between the big composed and little base cores (in Thread1 and F2.8), with composed core having lower EDP. In these cases, the performance benefits of the composed core outweigh the energy savings of the base core.

C. Thread Count Sensitivity

Finally, each benchmark is simulated with varying number of threads. In this step, each simulation was performed at the same frequency of 2.8 GHz and operating voltage of 1V, when changing the number of threads from 1 to 8. As shown in Fig. 3 (2.8GHz cases with varying threads), increasing the thread counts leads to better performance. Moreover, there is a large gap between the EDP values of base and composed core at lower number of threads. Particularly, we observe that by increasing the thread counts, the corresponding gap between different core types diminishes and makes the base core competitive to the composed core in terms of EDP.

V. Energy-Efficient Scheduling Framework

A. Joint analysis of (Core Type, Freqeucny, Thread Count)

To understand the interplay among various tuning parameters and find the optimum configuration for maximizing the energy-efficiency, in this section all permutations of the parameters were simulated. We test four voltage/frequency settings on two core types and execute each multithreaded benchmark with 1 to 4 or 8 threads (depending on the core type), where each thread is assigned to a single core. These results are illustrated in Fig 3. Due to space limitations, we only demonstrate the results for 1, 4 and 8 running threads. Furthermore, the best evaluated execution time and EDP for each application is shown in each figure.

As mentioned earlier, we examine two different heterogeneous CCAs consisting of multiple base and composed cores: 1) 8base/4comp, and 2) 4base/2comp. Table

TABLE II. OPTIMAL CONFIGURATION WITH OPTIMIZATION TARGET OF EDP FOR DIFFERENT ARCHITECTURES

Benchmark	8Base/4Comp					4Base/2Comp				
	Best-base		Best-composed		Var (%)	Best-base		Best-composed		Var (%)
	Freq. (GHz)	*#Thread*	*Freq. (GHz)*	*#Thread*		*Freq. (GHz)*	*#Thread*	*Freq. (GHz)*	*#Thread*	
barnes	2.4	8	2.8	4	-444.8	2.8	4	2.8	2	-475.4
fmm	2.4	8	2.4	4	2.2	2.4	4	2.8	2	-2.9
cholesky	2.4	8	2	4	28	2.4	4	2.8	2	5.8
radix	2.8	8	2.8	4	-138.7	2.8	4	2.8	2	-236.2
radiosity	2.4	8	1.6	4	-102.8	2.4	4	1.6	2	-128.1
raytrace	2.4	5	2	4	-28.9	2.4	4	2.8	2	-152
fft	2	4	2	2	36.3	2	4	2	2	36.3
lu.cont	2.4	8	2.8	4	27.2	2	4	2	2	7.8
blackscholes	2.8	6	2.4	4	83.85	2.8	4	2.4	2	77.08
bodytrack	2	7	2	3	41.23	2	3	2	2	34.1
ferret	2	6	2	4	62.4	2	4	2	2	54.3

II presents the optimal set of results for both architectures. This table includes benchmarks name, followed by the best core configuration parameters (Core, Freq., #Thread) in terms of EDP across base and composed cores. We have also calculated the relative EDP variation for each benchmark, which indicates the relative difference between energy-efficiency of the best configuration parameters in base and composed cores. We quantify variation parameter as follows:

$$Var = \left(\frac{Base_{best\_EDP} - Composed_{best\_EDP}}{Base_{best\_EDP}}\right) \times 100 \qquad (1)$$

The variation parameter (Var) indicates whether it is justified to compose cores. For this purpose, a variation threshold is defined that decides what type of core architecture should be selected for executing the corresponding multithreaded application more energy-efficiently. The user-defined threshold can be adjusted based on the architecture and available resources as well as the cost of core composition. Note that composing base cores to a big composed core is not free and comes with power as well as core utilization overhead. The core utilization overhead is in fact due to using additional cores to build bigger cores. When cores are composed to build a bigger core, fewer cores will be available for incoming or co-scheduled applications. In this work, we assume a 20% variation threshold. As a result, if the variation percentage between best-base and best-composed architectures is found to be lower than 20%, we use the base core for scheduling instead of composing to avoid power as well as core utilization costs.

As can be seen from Table II, for most studied applications the best running thread count is equal to the maximum available cores. For instance, *barnes* performs with 2.4 GHz and 2.8 GHz on base and composed cores, respectively, while the best number of running threads on these two architectures are equal to 8 and 4, respectively. As shown, the variation for this application has negative value in some cases, which indicates it is more energy as well as core-utilization efficient to run the application on base core. Therefore, given that the variation value is lower than pre-defined threshold, rather than running the application on costly big composed core, we schedule the multithreaded application onto cost-effective little base core. From these observations, we conclude that while we can obtain significant performance gains, power and core utilization costs could be drastic when running application on big composed core. As a result, in those cases we choose the little base core as the optimal core configuration.

In order to perform a comprehensive EDP characterization of studied architectures, we classified all possible configurations (core types and number of threads) into four classes. The first two are Fully-Base and Fully-Composed configurations that are referred to cases in which the lowest EDP is achieved with full utilization of the base and composed core, respectively. In other words, the optimum number of threads is equal to the maximum number of existing base/composed cores. On the other hand, we use Partially-Base and Partially-Composed configurations when the best number of threads is lower than maximum available cores.

The diversity of optimum configurations across various applications demonstrates that when running a given multithreaded workload on a heterogeneous CCA, depending on the application and energy-efficiency optimization metric, different core configuration parameters (Core Type, V/Freq., #Thread) lead to the best energy-efficiency. In other words, simulation results support that there is no unique solution as the best configuration across various applications. As a result, we can see various configurations for maximizing energy-efficiency across different applications. This dispersed pattern of optimum results implies the necessity of developing a prediction method to guide scheduling decision of unknown multithreaded applications onto heterogeneous composite cores architecture to enhance the energy-efficiency.

B. Prediction Model for Energy-efficiency

1) Model selection: Recent studies have proposed ordinary least squares regression (OLSR) modeling to estimate the power [10] and performance [3,11,19] of a processor at run-time. In this work, we show that OLSR is not the best suited algorithm for performance and power estimation as outliers in particularly in heterogeneous CCA can mislead the model. In fact, various applications experience different phases with different behavior. In addition, superscalar processors are complex, which makes it difficult to develop a general model for their power/performance estimation. OLSR models are highly sensitive to the outliers and potentially produce misleading results as even a single point of data substantially impacts on the regression efficiency. Thus, in this paper, based on a comprehensive characterization of various applications, we evaluate a more robust regression algorithm in addition to OLSR referred as Quantile Linear Regression (QLR) model [20], to predict the energy-efficiency for various configurations in our studied CCA. The main advantage of QLR as compared to OLSR is its robustness against outliers. QLR model is useful to obtain a more comprehensive analysis of the relationship between variables and provides a richer characterization of the data, allowing us to consider the impact of a covariate on the entire distribution of target variable.

TABLE III. HARDWARE PERFORMANCE DATA USED FOR REGRESSION MODEL

Category	Hardware performance counter
Memory subsystem	L1 D-cache access, L1 D-cache miss, L1 I-cache access, L1I- cache miss, L2 cache access, L2 cache miss, I-TLB miss, D-TLB miss
Instructions	Integer instruction issue, Integer floating point issue
Branch	Branch instruction, Branch misprediction

OLSR is a statistical method used to model the relation between a set of predictor variables and a response variable. It estimates the mean value of the response variable for given predictor variables. In linear regression, the regression coefficient represents the increase in the response variable produced by a single unit increase in the predictor variable associated with that coefficient. A more comprehensive picture of the effect of the predictors on the response variable can be obtained by using quantile regression. QLR model estimates the change in a specified percentile (quantile) of the response variable produced by a single unit change in the predictor variable. This allows comparing how some percentiles of the target variable may be more affected by certain estimator characteristics than other percentiles. This variation is reflected in the change in the size of the regression coefficient. For the QLR model, a specific quantile of data is set instead of the mean value. We set the quantile of 0.1, which results in minimizing the median of the error values.

Although the use of non-linear regression or neural network models potentially provides more accurate energy-efficiency of an application, they increase complexity of the design. The overhead in area, power and performance of implementing linear regression model in hardware is minimal and shown to be easily integrated into a core [11]. We show that the QLR model achieves higher accuracy as compared to OLSR. Fig. 4 illustrates a comparison between the derived coefficients of two different predictors using ordinary linear regression and quantile linear regression. As shown in each graph, black dotted line is the slope coefficient for the QLR and the red lines are the least squares estimate for OLSR and its confidence interval. The figure shows how the lower and upper quantiles are well beyond the least squares estimate. The figure further illustrates how the effects of L2 cache access and branch misprediction vary over quantiles, and how the magnitude of the effects at various quantiles differs considerably from the OLSR coefficient, even in terms of the confidence intervals around each coefficient (58% for L2-access and 30% for branch miss predictor). This figure highlights that an ordinary least squares regression is not an optimal solution to capture the actual behavior of applications

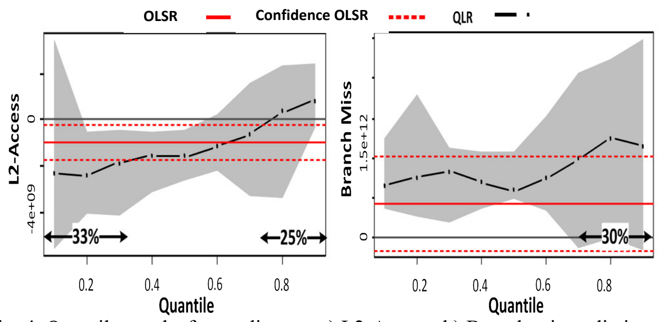


Fig. 4. Quantile graphs for predictors: a) L2-Access, b) Branch misprediction

to predict the energy-efficiency.

2) QLRM derivation and training: To derive the prediction model for energy-efficiency, we need to develop the training data set to train the prediction model. For this purpose, we consider a subset of applications from SPLASH-2 and PARSEC multithreaded benchmark suites. The studied multithreaded applications represent divers compute, memory and I/O intensity behavior. For each benchmark, we collect twelve pieces of hardware performance counter data on all possible configurations of core types, voltage/frequency operating points and number of threads. These microarchitectural parameters are listed in Table III. These features represent pipeline front-end, pipeline back-end, cache subsystem, and main memory behaviors and are influential in the performance of standard applications.

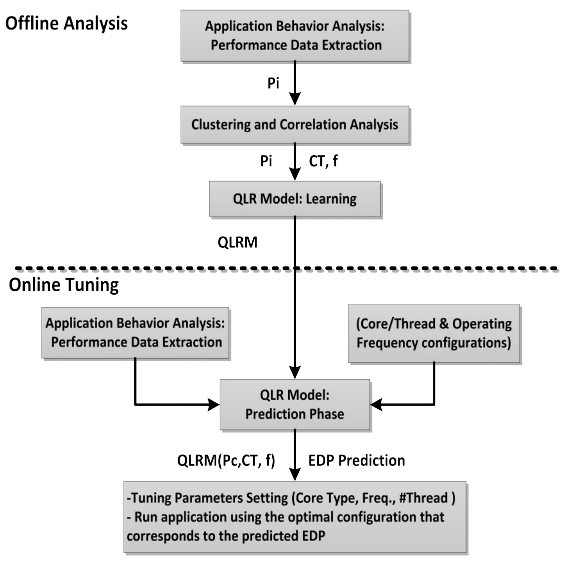
The inputs to QLR classifier include a set of workload metrics which are processor performance counters and the current configuration parameters (Core Type, Freq., #Threads). The quantile linear regression is trained using a subset of benchmarks (less than two third). The QLR model weights are estimated using training data. Given the twelve hardware performance counters, we use Principle Component Analysis (PCA) and correlation analysis on our training set to monitor the most vital microarchitecture parameters to capture application characteristics. By applying the attribute reduction method, we determine the four most related performance counters including L1 D-cache access, L2 cache-access, L2 cache-miss and branch misprediction. These performance counters are included in QLR model as input parameters. Since the main purpose of this model is to predict energy-efficiency across various application, system and microarchitectural parameters, we need to consider these tuning parameters in our model as well. Therefore, along with the identified key performance counter parameters, we include three tuning parameters (Core Type, Freq., #Thread) as input variables into our model to enable predicting the EDP for each configuration resulted by changing the core type, operating frequency and thread counts.

After identification of the four key hardware performance parameters and considering the tuning parameters, we formulate the proposed EDP prediction model using quantile linear regression as follows:

$$QLRM = \left(\beta 0 + \sum_{i=1}^{4} \beta i.Pi\right) + \beta 5.CT + \beta 6.f \qquad (2)$$

where $\beta 0$ is the intercept, βi denotes the corresponding coefficients of the regression model, and Pi are extracted hardware performance counters. Moreover, the core/thread configurations are given by CT, and f represents the frequency on the corresponding core architecture. The βi coefficients can be interpreted as the expected change in EDP per unit change in L1 D-cache access, L2 cache-access, L2 cache-miss, branch misprediction, core/thread and frequency setting.

This model predicts continuous values representing energy delay product as a function of performance counter inputs and tuning parameters, which is then used to make the scheduling decisions at run-time. During run-time, given an unknown application, the QLR model can then predict the EDP of all possible configurations based on a single run data. The configuration corresponding to the lowest estimated EDP is then selected for the run.



Offline Analysis
Application Behavior Analysis: Performance Data Extraction
Pi
Clustering and Correlation Analysis
Pi
CT, f
QLR Model: Learning
QLRM
Online Tuning
Application Behavior Analysis: Performance Data Extraction
(Core/Thread & Operating Frequency configurations)
QLR Model: Prediction Phase
QLRM(Pc,CT, f)
EDP Prediction
-Tuning Parameters Setting (Core Type, Freq., #Thread)
- Run application using the optimal configuration that corresponds to the predicted EDP

Fig. 5. Proposed scheduling scheme with energy-efficiency prediction

C. *Eenrgy-Efficient Scheduling Algorithm*

Fig. 5 provides an overview of our scheduling scheme using the regression-based prediction model. As illustrated, our scheduling algorithm is split between an offline step and an online step. In offline analysis, we first extract an extensive set of data from multithreaded applications. The next step involves clustering and correlation analysis of extracted features that decreases the microarchitecture parameters to the ones that have the most impact on energy-efficiency. Using the selected performance data and various tuning parameters configuration, we train and develop the QLR classifier.

In online tuning step, we run a multithreaded application with the most aggressive configuration setting where all tuning parameters are set at max (maximum number of threads, highest frequency, and for composed core). We then extract the hardware performance counters by profiling the multithreaded application. This is done by running the application and collecting the performance counters information for maximum setting. Profiling stage helps understanding the run-time characteristics and resource utilizations of the application. The regression classifier takes the key performance counter parameters and configuration settings as inputs, and outputs the system energy-efficiency for the given configuration. As a result, we schedule the application with tuning parameters selected by the quantile linear regression model to minimize the EDP.

Note that the linear weights are estimated using training data set. Given the input configuration parameters during run-time, the QLRM can predict the optimal energy-efficiency. The output with optimal energy-efficiency and corresponding new configuration is then chosen as the current operating point at run-time. The predictive model, by observing run-time behavior of a multithreaded application running with a specific configuration, predicts the right configuration parameters namely, the number of threads, operating voltage and frequency, and core type - whether composed or base - to achieve the maximum energy-efficiency. It is important to note that the QLRM can be simply trained for other objectives such as ED^{2}P optimization.

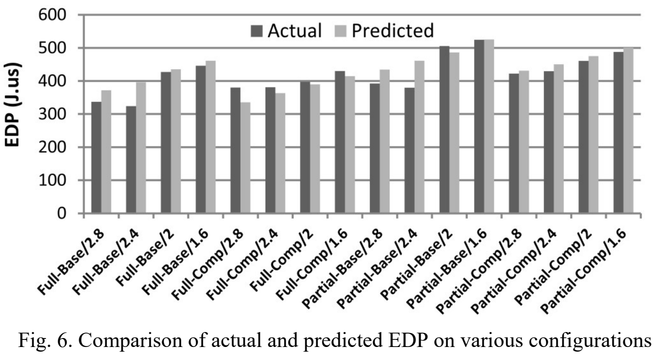


Fig. 6. Comparison of actual and predicted EDP on various configurations

D. *Evaluation Results*

Fig. 6 shows the comparison between the observed and predicted maximum energy-efficiency across various operating frequencies and core/thread configurations. In order to evaluate the accuracy of our prediction model, we calculate the value of relative mean absolute error defined as $\frac{|estimated\ value - actual\ value|}{(actual\ value)} \times 100\%$. This metric indicates the relative difference between the predicted and observed maximum energy-efficiency. In order to validate our QLR learning model, we applied percentage split method to divide the dataset into two sets, using 60% (known applications) of the data to train the model and 40% (unknown applications) to simulate and test.

In Table IV, average relative errors of applied linear regression model for energy-efficiency estimation of different operating points are presented. As shown, we characterized all possible configurations to 16 operating points consisting of various frequencies and core/thread configurations. Given the results, the proposed prediction classifier is most accurate in estimating the energy-efficiency of Fully-Comp architecture and Fully-Base architecture, respectively, both operating at 2 GHz. Moreover, our regression model achieves an average error of 6.85% across all testing data samples and possible configurations. The presented results assist the scheduling decision of multithreaded applications on heterogeneous CCA including composing cores, setting operating voltage and frequency, and adapting the number of running threads.

The performance overhead of implementing the QLRM in hardware and calculating values at each interval is negligible. The power overhead of implementing the QLRM is estimated at 5uW, which is further reduced by gating idle units during each interval [11]. In order to evaluate the efficiency of our prediction model, the following scheduling schemes are studied for comparison:

- *Composite-Oracle*: This model is based on the composite cores architecture with an ideal application energy-efficiency predictor, where all future behavior of the application as well as the power and performance for various configurations are known in advance. Therefore, the Composite-Oracle adapts the cores, operating frequency and application thread counts and exploits all opportunities to maximize energy-efficiency. Since this scheme provides the upper bound for energy-efficiency, it is used to normalize and compare other schemes.

- *Composite-QLRM*: This scheme is based on the composite architecture with our proposed quantile linear regression to estimate the EDP for various core sizes, frequency/voltage

TABLE IV. AVERAGE RELATIVE ERROR OF QLR MODEL

	Core/Thread Configurations			
Freq.	**Full-Base**	**Partial-Base**	**Full-Comp**	**Partial-Comp**
2.8 GHz	10.5%	10.74%	11.69%	2.03%
2.4 GHz	22.49%	21.4%	4.67%	4.87%
2.0 GHz	1.9%	3.9%	1.74%	3.1%
1.6 GHz	3.35%	2.2%	3.6%	2.61%

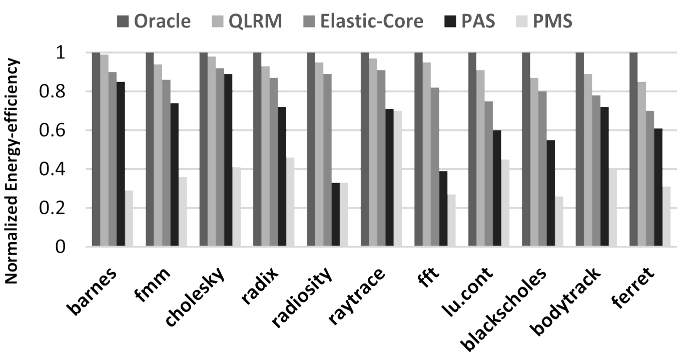


Fig. 7. Normalized energy-efficiency of applications on various scheduling schemes relative to Oracle scheduling

points and number of threads.

- *Elastic-Core* [3]: This dynamic scheme proposed recently is closest to our work, and uses a linear regression model to predict the power and performance of single-threaded applications as a function of core type and frequency settings. Although it does not take number of threads into account, we set the thread counts to maximum values to better evaluate our model by fairly comparing it against a recent dynamic scheduling.

- *Performance Aggressive Scheduling (PAS)*: In this scheme, all tuning parameters are set at maximum value to achieve the maximum performance. Therefore, each application is executed on the high performance composed core with frequency of 2.8 GHz running with 4 threads.

- *Power Minimized Scheduling (PMS)*: This scheduling attempts to minimize power consumption. In other words, the cores always remain at little base core (no composition). Also, the frequency and number of threads are set to their minimum values.

The energy-efficiency results of studied applications normalized to the Composite Oracle are shown in Fig. 7. The Composite-QLRM on average achieves close to 94% efficiency as compared to the Oracle model. The Composite-QLRM has an improved energy-efficiency as compared to the Elastic-Core and PAS schemes by an average of 10% and 30% across all benchmarks, respectively. Also, it outperforms the PMS scheduling by 54% improved energy-efficiency. The results prove the accuracy of our proposed prediction model and effectiveness of proposed scheduling scheme to harness the power of heterogeneity in a composite cores architecture to significantly enhance its energy-efficiency.

VI. Conclusion

Emerging heterogeneous composite cores architectures are complex processors with various tuning optimization knobs for improving performance and energy-efficiency. Scheduling multithreaded applications in these architectures is a challenging problem, given various optimization parameters at application (number of running threads), system (operating voltage and frequency), and architecture (core type, namely big vs. little) levels. In particular, the interplay among these tuning parameters and their influence on energy-efficiency, make the scheduling and tuning even a more challenging problem. In this paper, we respond to this challenge by developing a scheduling and tuning solution for this class of architecture. The space for tuning configuration parameters in a composite cores architecture is large with no unique solution for the most energy-efficient configuration for different multithreaded applications, calling for developing a model to predict energy-efficiency. We developed a predictive model for estimating the energy-efficiency of multithreaded applications in composite cores architecture. Our proposed model achieves an average 6.85% error rate across all testing data samples and possible configurations. Based on the predictive model, we developed a scheduling scheme for effective mapping of multithreaded applications to heterogeneous CCA by setting the tuning parameters to maximize the energy-efficiency. The results show that the proposed scheduling scheme on average achieves close to 94% efficiency as compared to the Oracle scheduling.

Acknowledgment

This work was supported in parts by the National Science Foundation under grant CSR-1526913 and CPS-1329829.

References

[1] H. Homayoun et al., “Dynamically heterogeneous cores through 3D resource pooling”, *HPCA-12*, pp. 1-12, February 2012.
[2] V. Kontorinis et al., “Enabling dynamic heterogeneity through core-on-core stacking”, *DAC-14*, pp. 1-6, June 2014.
[3] M. K. Tavana et al., “ElasticCore: enabling dynamic heterogeneity with joint core and voltage/frequency scaling”, *DAC-15*, pp. 1-6, June 2015.
[4] K. V. Craeynest et al., “Scheduling heterogeneous multi-cores through performance impact estimation (PIE),” *ISCA-12*, pp. 213-224, 2012.
[5] K. Pusukuri et al.,“Thread reinforcer: dynamically determining number of threads via OS-level monitoring”, *IISWC-11*, pp. 116-125, 2011.
[6] N. Chitlur et al., “QuickIA: Exploring heterogeneous architectures on real prototypes”, *HPCA-12*, pp. 1-8, February 2012.
[7] NVidia. The benefits of multiple CPU cores in mobile devices. http://www.nvidia.com/content/PDF/tegra_white_papers/Benefits-of-Multi-core-CPUs-in-Mobile-Devices_Ver1.2.pdf, 2010.
[8] P. Greenhalgh. Big.LITTLE processing with ARM Cortex_A15 & Cortex_A7:http://www.arm.com/files/downloads/big_LITTLE_Final_Final.pdf, September 2011.
[9] G. Liu et al., “Dynamic thread mapping for high performance, power-efficient heterogeneous many-core systems”, *ICCD-13*, pp. 54-61, 2013.
[10] J. Cong et al., “Energy-efficient scheduling on heterogeneous multi-core architectures,” *ISLPED-12*, pp. 345-350, 2012.
[11] A. Lukefahr et al., “Composite cores: Pushing heterogeneity into a core”, *MICRO-12*, pp. 317-328, 2012.
[12] R. Cochran et al., “Pak & Cap: Adaptive DVFS and thread packing under power caps”, *MICRO-11*, Decemeber 2011.
[13] T. E. Carlson et al., “An evaluation of high-level mechanistic core models”, *TACO-14*, vol. 11, October 2014.
[14] S. C. Woo et al., “The SPLASH-2 programs: characterization and methodological considerations”, *ISCA-95*, pp. 24-36, 1995.
[15] S. Li, et al., “McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures”, In *MICRO,* 2009.
[16] C. Bienia et al., “Parsec 2.0: A new benchmark suite for chip-multiprocessors,” *in 5th Annual Workshop on Modeling Benchmarking and Simulation*, June 2009
[17] E. Ipek et al., “Core fusion: Accommodating software diversity in chip multiprocessors”, *ISCA-07*, June 2007.
[18] C. Kim et al., “Composable lightweight processors,” *MICRO*, 2007.
[19] B.C. Lee et al., “Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction”, *SIGPLAN-06*.
[20] R. Koenker, “Quantile regression,” No. 38, Cambridge university press.
[21] R. Kumar et al., “Single-ISA Heterogeneous Multi-core Architectures for Multithreaded Workload Performance,” *ISCA-04*, June 2004.
[22] M. Becchi et al., “Dynamic thread assignment on heterogeneous multiprocessor architectures”, *ACM CF,* 2006.
[23] G. Liu et al., “Procrustes: Power Constrained Performance Improvement Using Extended Maximize-Then-Swap Algorithm”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 34, No. 10, pp.1664–1676, October 2015.
[24] M. K. Tavana et al., “Realizing complexity-effective on-chip power delivery for many-core platforms by exploiting optimized mapping”, *ICCD-15*, September 2015.
[25] C. Dupach et al., “Dynamic adaptation using machine learning”, *ACM TACO-13*, Vol. 10, No. 4, December 2013.
[26] J. Martinez et al., “Dynamic multicore resource management: A machine learning Approach”, In *MICRO*, 2009.
[27] M. Malik et al., “Big vs Little Core for Energy-Efficient Hadoop Computing”, *DATE-17,* March 2017.
[28] K. Neshatpour et al., “Big Data Analytics on Heterogeneous Accelerator Architectures”, *CODES+ISSS*, 2016.
[29] H. Homayoun, “Heterogeneous Chip Multiprocessor Architectures for Big Data Applications” *ACM CF*, 2016.