Performance Analysis of Multimedia Retrieval Workloads Running on Multicores

Yunping Lu, Xin Wang, Weihua Zhang, Haibo Chen, *Senior Member, IEEE*, Lu Peng, and Wenyun Zhao

Abstract—Multimedia data has become a major data type in the Big Data era. The explosive volume of such data and the increasing real-time requirement to retrieve useful information from it have put significant pressure in processing such data in a timely fashion. However, while prior efforts have done in-depth analysis on architectural characteristics of traditional multimedia processing and text-based retrieval algorithms, there has been no systematic study towards the emerging multimedia retrieval applications. This may impede the architecture design and system evaluation of these applications. In this paper, we make the first attempt to construct a multimedia retrieval benchmark suite (MMRBench for short) that can be used to evaluate architectures and system designs for multimedia retrieval applications. MMRBench covers modern multimedia retrieval algorithms with different versions (sequential, parallel and distributed). MMRBench also provides a series of flexible interfaces as well as certain automation tools. With such a flexible design, the algorithms in MMRBench can be used both in individual kernel-level evaluation and in integration to form a complete multimedia data retrieval infrastructure for full system evaluation. Furthermore, we use performance counters to analyze a set of architecture characteristics of multimedia retrieval algorithms in MMRBench, including the characteristics of core level, chip level and inter-chip level. The study shows that micro-architecture design in current processor is inefficient (both in performance and power) for these multimedia retrieval workloads, especially in core resources and memory systems. We then derive some insights into the architecture design and system evaluation for such multimedia retrieval algorithms.

Index Terms—Multimedia retrieval, benchmarks, architectural characteristics

1 Introduction

Our society has entered into the *Big Data* era, with data volume increasing at exponential rate. Among various data types, multimedia data, such as images and videos, have become one of the major types. Video data occupies 64 percent of the customer Internet traffic in 2014 and was predicted to increase to an 80 percent occupation by 2019 [1]. Among them, more than 400-hour worth new videos were uploaded to *YouTube* every minute in 2015 [2]. In the meantime, *Facebook* hosts more than 240 billion of user-uploaded images [3].

To extract useful information from such data, multimedia retrieval applications are emerging to process such multimedia data, including video recommendation [4], travel guidance [5] and content-based TV copyright identification [6]. To guarantee retrieval accuracy, typical applications usually extract and utilize hundreds of high-dimensional features to represent an image or a video frame. Thus, in contrast to traditional text-based retrieval applications, multimedia retrieval applications are not only more data-intensive but also more computation-intensive, which lead to significant pressure on real-time processing. For example, SURF [7] is one of the most widely-used image or video retrieval algorithms [8]. It can only achieve the process speed of a handful images or video frames per second on general-purpose processors. Although these applications are becoming more and more popular, there are currently no systematic benchmark suites to understand their architectural characteristics, which are critical to design and implement optimizing architectures and systems for such workloads. Most related benchmarks are still text-based [9], [10] or target tradition multimedia processing [11], [12].

As a first attempt, we design and implement a multimedia retrieval benchmark suite (MMRBench) for architecture design and system evaluation, by selecting representative algorithms in the multimedia retrieval fields. We also implement multiple programming versions for these algorithms, such as sequential, parallel and distributed versions. To satisfy varied evaluation requirements, we provide automation tools for adjusting the parameters and generating input. Furthermore, we provide a basic framework including all the major processing stages of multimedia retrieval applications as well as the data transfer interfaces between different processing stages. In this framework, the algorithms in different stages can be easily replaced with different versions or other algorithms. The parameters and inputs can also be adjusted based on the provided tools. Therefore, users can

- *Y. Lu and W. Zhao are with the Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China and the School of Computer Science, Fudan University, Shanghai, China and Parallel Processing Institute, Fudan University, Shanghai, China. E-mail: luyping@sina.com, wyzhao@fudan.edu.cn.*
- *X. Wang and W. Zhang are with the Software School, Fudan University, Shanghai, China and the Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China and Parallel Processing Institute, Fudan University, Shanghai, China. E-mail: xin_wang@fudan.edu.cn, whzhang.fd@gmail.com.*
- *H. Chen is with the Institute of Parallel and Distributed Systems, Shanghai Jiaotong University, Shanghai, China. E-mail: haibochen@sjtu.edu.cn.*
- *L. Peng is with the Division of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA. E-mail: lpeng@lsu.edu.*

Manuscript received 10 Sept. 2015; revised 5 Jan. 2016; accepted 10 Feb. 2016. Date of publication 23 Feb. 2016; date of current version 12 Oct. 2016. Recommended for acceptance by A. Dubey. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPDS.2016.2533606

flexibly evaluate their designs by using different algorithms in MMRBench individually or in integration for a full workflow system evaluation.

Furthermore, we use performance counters to study a set of architectural characteristics of these algorithms, including the characteristics of core level, chip level and inter-chip level. We observe obvious mismatch between the requirement in multimedia retrieval workloads and current predominant processor architecture. We conclude the characteristics of these workloads in three levels. In the *core level*, the programs have the following characteristics: (1) complex branch behavior and small instruction working set, (2) lower instruction-level parallelism (ILP) and memory-level parallelism (MLP), (3) insensitive to floating-point operations; in the *chip level*, they (1) require modestly sized L2 cache which acts efficiently in data fetch and relatively small data working set in last level cache, (2) have abundant thread level parallelism and critical imbalance problem; and in *inter-chip* level, they have low off-chip memory bandwidth utilization. Based on the analysis, some insights into architecture design are derived for these algorithms.

In summary, this paper makes the following contributions:

- Design and implementation of a multimedia retrieval benchmark suite, including state-of-the-art algorithms, different implementation versions, automation tools and a flexible system framework. Such a design enables the algorithms to be used individually or combined together to form an integrated system for evaluation.
- Detailed analysis on the architectural characteristics of multimedia retrieval algorithms compared to conventional benchmark suites and predominant processor architecture.
- Architecture design insights. Based on the above performance evaluation, we provided architecture design insights on the application level, the core level, the chip level, and the inter-chip level. These can be used to design efficient architecture for multimedia retrieval programs in the future.

The paper is organized as follows. Section 2 explains the motivation of this paper and Section 3 discusses the related work. Then Section 4 gives an overview of MMRBench and Section 5 presents the design of the basic framework. Section 6 analyzes the architectural characteristics of MMRBench. Finally, Section 7 concludes the paper.

2 Motivation

Currently, the transistor threshold and voltage scale problems limit the further improvements on single-core processor performance, both in compute density and power efficiency. As more and more emerging workloads dominate in cloud environments, researchers now begin to optimize server systems to meet the performance requirement and power constraints by removing unnecessary components such as graphic chips and using more green power supply. However, modern processor architectures are still inefficient in space and power for cloud workloads such as web search [13]. Considering multimedia retrieval workloads are more data-intensive and computation-intensive than traditional text-based retrieval workloads, we aim to investigate whether there is a mismatch between the characteristics of these workloads and the predominant commodity processors. Further, we intend to analyze the system and architecture characteristics for emerging multimedia retrieval applications, with the goal of gaining insight into designing efficient processor architecture and systems in this area. To achieve such goals, our workload characterization should meet the following targets.

- *Representative system behavior:* A multimedia retrieval workflow typically consists of three stages: feature extraction, feature matching and spatial verification. Algorithms in these stages handle different computation hence have varied characteristics. Therefore, one should include popular algorithms from all these stages to understand the characteristics of multimedia retrieval systems.
- *State-of-the-art techniques:* Varied algorithms and techniques may apply to the three aforementioned workload stages. For example, feature extraction applications may include both global and local feature based algorithms. However, since low precision in matching (more than 30 percent error rate [14]), global feature based algorithms have been rarely used in real applications. As a result, state-of-the-art algorithms and techniques should be used in system evaluation and characterization.
- *Applicable to cloud computing:* As cloud and datacenter computing are gaining increasing momentum, the proposed multimedia retrieval workloads should be able to extend to cloud and datacenter level in computation, storage and communication intensity. In addition, varied input sets are desirable to show an application's characteristics at different system scales.
- *Flexible infrastructure:* We envision an integrated evaluation framework with flexible computation kernel modules. As such, the framework can be applicable to various system designs and experimental environments, e.g., from multi-threading in a multi-core processor design to distributed implementation at the datacenter level.

3 Related Work

In this section, we discuss several existing benchmark suites and prior research on the evaluation of performance and power efficiency of modern processors.

3.1 Benchmark Suites

In order to compare the multimedia retrieval algorithms with other applications, we choose representative benchmark suites designed for different targets for comparison.

- *Traditional benchmarks:* SPEC [15] has designed several benchmark suites, such as SPEC CPU, to satisfy different evaluation scenarios. Splash2 [16] and PARSEC [17] are two most popular benchmarks for parallel system evaluation. Parboil [18] is another parallel benchmark suite focusing on GPGPU-like

architecture. However, emerging multimedia retrieval algorithms are not considered in such suites.

- *Multimedia benchmarks:* Traditional multimedia benchmark suites include ALPBench [11], MiBench [12], SD-VBS [19] and MEVBench [20]. Still, they mainly focus on traditional multimedia *decode* and *encode* processing.[1] Although both of traditional multimedia algorithms and information retrieval ones process multimedia data, the major processing steps are largely different. Multimedia retrieval algorithms focus on how to extract and analyze these multimedia data.
- *Cloud benchmarks:* Benchmark suites aiming to cloud environment are also becoming popular. Intel researchers published MapReduce based HiBench [10]. CloudSuite [9] is designed to cover the emerging scale-out workloads in cloud environments. However, most of these benchmarks target text-based workloads. The characteristics of multimedia-based information retrieval applications are not reflected.

3.2 Architecture Evaluation

In this section we review prior work on performance evaluation and architecture design.

- *Performance evaluation:* Most of performance evaluations on efficiency of modern processors are based on traditional commercial applications. For scientific applications, Tuck and Tullsen indicates that the SMT mechanism can notably improve performance [22]. Nikos et al. characterize a commercial database server running on emerging chip multiprocessor technologies and find the major bottleneck is data cache stalls [23]. Nan et al. propose a model to evaluate the earth system simulation framework [24]. Xu et al. discuss the performance modeling of Sparse Matrix-Vector Multiplication (SpMV) on GPUs [25]. Xue et al. simulate the performance of atmospheric dynamics on supercomputers [26]. To catch up with specific infrastructure trends such as datacenters, Ranganathan et al. discuss the implications for architecture research and suggesting some high-level challenges for the community to address [27].
- *Architecture design:* As multi-core and many-core systems becoming a mainstream, many researches advocate new processor structure design for them. In [28], Hardvellas et al. show that server chips will not scale beyond a few tens of cores such that we cannot afford to power. They then propose use simpler cores and reduce on-die caches to overcome the initial power barrier. Other efforts such as [29] and [30] study optimal cache and DRAM structure for given applications. For emerging cloud workloads, Kozyrakis et al. [31] provide a characterization on large-scale online services workloads and give some insights into such cloud server designs. In CloudSuite [9], [32], Ferdman et al. identify the key micro-architecture needs for scale-out workloads and design efficient scale-out processors. Although there are many architecture optimizations for different workloads, none of them consider the characteristics of multimedia retrieval applications in their designs.

1. Although multimedia retrieval algorithms are considered in SD-VBS [19], only two feature extraction algorithms are included in its design, while most algorithms in it are traditional multimedia algorithms. Ferret [21] is a toolkit to construct content-based similarity search systems. Therefore, it is not designed for performance evaluation. No state-of-the-art algorithms are included in it and no architectural characteristics are analyzed in it. Moreover, it is global feature-based, which has been shown to have low retrieval precision [14]. Therefore, as we described in Section 4.1, local feature-based algorithms are widely used in current real-world applications.

4 MMRBench Overview

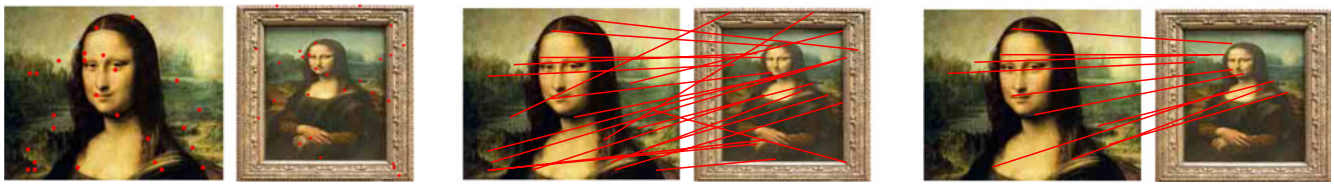
In this section, we first describe the generic workflow of multimedia retrieval applications. We then describe our methodology to select multimedia retrieval algorithms for this work, followed by a brief overview of such algorithms.

4.1 Multimedia Retrieval Workflow

Multimedia retrieval applications generally consist of three stages: feature extraction, feature matching and spatial verification.

- *Feature extraction:* In this stage, feature points are extracted to represent an image or a video frame. Feature extraction algorithms can be divided into two classes: Global Feature Based Algorithms (GFBAs) and Local Feature Based Algorithms (LFBAs). GFBAs use one unique feature to represent an image, while LFBAs adopt hundreds of points to guarantee the results are insensitive to various transformations, such as scaling, rotation and illumination. Due to GFBAs' low retrieval precision [14], LFBAs are widely used in real-world applications. Therefore, we focus on LFBAs in this work.
- *Feature matching:* The way to judge whether two images are similar is to check whether they have enough similar feature points. The similarity metric applies the Euclidean distance. Due to the huge amount of image processing data, most multimedia retrieval applications apply approximate algorithms in their matching stages to avoid excessive computation. When the matched points between two images exceed a threshold, these two images will be considered as matched. Therefore, floating-point precision in feature matching is important. We apply this insight in our analysis in Section 6.
- *Spatial verification:* The matching results are usually polluted by false matching. To filter out these mismatched feature points, spatial verification algorithms are adopted to refine the matching results by checking the spatial relationship of the matched feature pairs from the above stage.

Fig. 1 illustrates the aforementioned workflow with two images of *Mona Lisa* as an example. In this example, we assume the feature points of the image on the left are trained in a backend database and the image on the right will be used to query the database. Many more images will be processed in a real world scenario. In the feature extraction stage, the feature points in images are extracted. The red points in the images represent the positions of extracted



(a) Feature extraction. (b) Feature matching. (c) Spatial verification.

Fig. 1. Example of three-stage (a, b and c) multimedia retrieval.

feature points. In the feature matching stage, each feature point is compared with those in the database. If the number of matching points exceeds a certain threshold, e.g., 20, they will be considered as similar. Note that, multimedia retrieval applications typically do not achieve a total match among all points between two images due to varied image transformations. As a result, a threshold is generally set. More matching points over the threshold would not affect the result. Lastly, spatial verification algorithms are adopted to filter out false matches, e.g., the points on the frame in the right image of Mona Lisa in Fig. 1. Based on the final results of this stage, the matching images are ranked based on their numbers of matching points.

4.2 Workloads

Each processing stage of multimedia retrieval applications contains various algorithms. To choose the most representative ones, we conduct a survey on the top conferences related to multimedia retrieval applications, such as *IJCV*, *CVPR* and *ACM Multimedia*, over the last five years. If an algorithm has been cited more than certain threshold (we set the threshold to 10 in this work), it is considered the state of the art and is selected in this work. As a result, we select eight multimedia retrieval algorithms. The chosen algorithms are shown in Table 1 (T:train/Q:query). Here we also find that SIFT and SURF are the most popular algorithms in feature extraction stage and VOC-Tree is applied with the most frequently in feature matching stage.

- *SIFT (Scale Invariant Feature Transform)* [33] is an algorithm to detect local features in images. SIFT detects features (or points) which are invariant to scaling, rotation, illumination and viewpoint. It consists of a detection stage and a description stage. In the detection stage, scale invariant points are detected. In the description stage, each point is assigned with one or more orientations to achieve rotation invariance. Then, a 128-dimension descriptor vector is computed for each point.
- *SURF (Speeded-Up Robust Features)* [7] is another scale, illumination and rotation-invariant algorithm. The workflow of SURF also contains detection stage and description stage. However, SURF has slightly different ways of detecting feature points. It uses a special structure "integral image" instead of Gaussian pyramid in SIFT, which enables it to process faster while keeping similar distinctiveness of feature points.
- *MSER (Maximally Stable Extremal Regions)* [34] is a feature detection algorithm. Instead of extracting feature points, MSER extracts co-variant regions from an image, called MSERs. MSER has the advantage that features can be reliably matched regardless of the appearance of the surroundings of the actual region.
- *HOG (Histogram of Oriented Gradients)* [35] is an algorithm used for object detection. The technique counts occurrences of gradient orientation in localized parts of an image. It is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization to improve accuracy. Generally, HOG is used for pedestrian detection.
- *KD-Tree* [34] is originally designed for exacting nearest neighbor search in multi-dimensional data. In its training stage, multiple randomized trees are created by selecting the top N dimensions with the greatest variance. In the querying stage, it uses best bin to find a set of approximate solutions.
- *VOC-Tree (Vocabulary Tree)* [36] defines a hierarchical quantization built by recursively doing k-means clustering. In its training stage, the training data are processed through defining k cluster centers and recursively defining quantization cells by splitting each quantization cell into k new parts. The tree is constructed level by level, up to some maximum number. In the querying stage, each feature point traverses through the tree by comparing itself to each of k cluster centers and choosing the closest one until it reaches the leaf node.
- *LSH (Locality Sensitive Hashing)* [37], [38] is a method of performing probabilistic dimension reduction of high dimensional data. When training data, it hashes the items so that similar items can be more probable to be mapped to the same buckets than dissimilar data. It has two parameters: the width parameter k and the number of hash tables L. In the querying step, it hashes the query point q into each of the L hash tables. In each hash table, it iterates over k hash functions. It retrieves the points hashed into the same buckets as q.

TABLE 1
Overview of the MMRBench

Application	Function	Input
SIFT	Feature Extraction	Image/Video Frame
SURF	Feature Extraction	Image/Video Frame
MSER	Feature Extraction	Image/Video Frame
HOG	Feature Extraction	Image/Video Frame
KD-Tree(T/Q)	Feature Matching	Feature Points
VOC-Tree(T/Q)	Feature Matching	Feature Points
LSH(T/Q)	Feature Matching	Feature Points
RANSAC	Spatial Verification	Feature Points

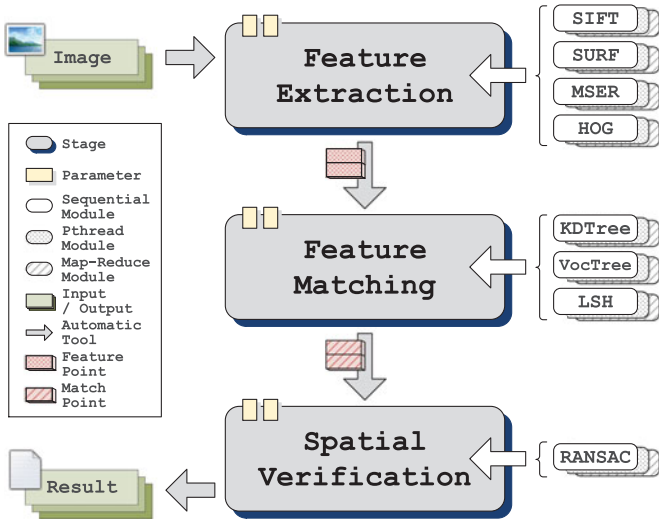


Fig. 2. Infrastructure for multimedia retrieval workloads.

- *RANSAC (RANdom SAmple Consensus)* [33], [39] aims to filter *"outliers"* out of *"inliers"* in a data set. *Inliers* are consistent with the estimated (spatial) model and can be explained by some set of model parameters while *outliers* do not fit the model. The retrieval precision under the spatial consistency can be improved with RANSAC.

5 Methodology

This section present our analysis methodology in infrastructure, metrics, tools and experimental setup.

5.1 Infrastructure

Our goal is to build an evaluation infrastructure that is convenient for users to construct test suites for their varied evaluation needs, e.g., architecture designs for datacenters or system evaluations on multi-core architectures. To achieve this goal, we have provided multiple versions for the selected algorithms, which includes sequential version, multi-threaded version for multi-core architecture and Hadoop (map-reduce) version for large-scale distributed environment. Most of these algorithms have multiple versions of open-source implementation. We select the sequential implementation whose paper has most citations to ensure that the algorithms in our benchmark suite are popular and represent state-of-the-art. The parallel and map-reduce version of these workloads are all implemented from scratch by ourselves.

In addition, we provide both multiple input sets and automation tools that facilitate input data generation and control parameter adjustment. To make it convenient for users to construct a real multimedia retrieval system, we also provide a framework with an interface API between retrieval stages. Fig. 2 illustrates our multimedia retrieval evaluation framework, of which we describe the features in the following.

- *Framework:* The framework includes the computation modules for all the stages in the workflow and the data transfer interfaces between the stages. Each algorithm module can be replaced with another one in the same stage. The intermediate result of an algorithm can be transformed into a format that can be processed by the next stage through the API interface.
- *Input sets:* In our current design, we assemble three input sets: small, medium (standard) and large. The small input set is provided by Mikolajczyk [40] that contains 48 images and has been widely used in various multimedia retrieval applications. The medium data size is the Oxford Building dataset [41] with thousands of images. For large size input, we collect twenty thousands of images of various categories from the Internet. Furthermore, users can generate new input set with the provided automation tool.
- *Multiple versions:* We provide three versions for each chosen algorithm: sequential version, multi-threaded version and map-reduce version. All versions are implemented in C/C++ under Linux environment. The multi-threaded version is implemented based on Linux POSIX API and map-reduce version is based on Apache Hadoop MapReduce framework. The parallelism granularity is at image-level in this implementation.
- *Automation tools:* While evaluating their systems, users may want to generate their own input data for the algorithms and prefer adjusting the parameters or thresholds in the algorithms. To ease the burden on the users, we provide corresponding automation tools for generating input data and adjusting parameters for the algorithms.

Putting it together, it is fairly flexible for the users to construct an evaluation framework and satisfy their varied needs in inputs, control parameters or execution environments. For example, the users can select different versions of an algorithm for their preferred evaluations, such as the parallel version for multi-core design or the Hadoop version for large-scale datacenter environment. They can also adjust parameters or generate their own input. Furthermore, with the provided framework and interface, it is fairly easy for the users to construct a real multimedia retrieval system to evaluate the overall performance or a special-purpose multimedia retrieval applications running in a different environment. Even after the system is constructed, it is also easier for the users to adjust parameters in the system with the interface and the automation tools provided in our framework. They can also choose to use these algorithms with either higher performance or more accurate to replace the one in the existing system.

Based on this framework, we have implemented an image retrieval system, which is mainly based on SURF feature. The database scale has reached tens of millions images and our image retrieval system can achieve a throughput of about five-thousand image queries per second in the feature matching stage on a multi-core server.

5.2 Experimental Methodology

In this section, we explain the methodologies used for our architectural analysis including measurement metrics, tools and experimental setup.

5.2.1 Measurement Metrics and Tools

For architectural characteristic analysis, we use Intel VTune [42] which enables hardware-based sampling by

utilizing processor performance counters. We collect the following metrics through VTune: execution time breakdown, L1 I-cache/L2 instruction misses, IPC (Instruction per Cycle), MPC (Memory Access per Cycle), L2 cache hit rate (data access) and off-chip bandwidth. For the execution time breakdown, we classify the execution time into commit cycle and stall cycle, which indicate whether at least one instruction is committed or not during one cycle. We also profile the memory cycles in order to expose the impact of memory access in a program. Here the memory cycles are computed as the sum of off-core requests outstanding cycles and DTLB miss duration. We note that memory cycles computed in this way is an approximate value because stalls caused by memory accesses will be overlapped in the pipeline execution. This is also the reason that why we do not take L1/L2 cache access time into consideration since we assume that they are already entirely overlapped.

To gain more architectural insights at the chip level, we further evaluate several factors, which will influence transistor resource allocation among different hardware components, such as processor core and last level cache (LLC). In our current evaluation, the factors include computation intensity, LLC cache misses and load balance. To measure computation intensity, we deploy a metric that indicates how many instructions are committed over a byte of input data, called Instructions per Byte (IPB).[2] It measures the computational resources needed when processing unit size input. To collect the cache misses of LLC, Cachegrind in Valgrind Tool Suite [43] is used to do cache profiling for cache sensitivity analysis. It performs detailed simulation of caches and identifies cache misses. We use Cachegrind to get the number of last-level cache (LLC) misses per 1,000 instructions in different LLC size configurations. For load balance, the Coefficient of Variations (CoV) of the execution time among different threads is used to indicate the level of load imbalance in the system.

5.2.2 Statistical Methods

We employ *Principle Component Analysis* (PCA) to calculate the data variance through reducing high-dimensional data into a low-dimensional form. Raw data are normalized before PCA is applied to generate a low-dimensional format. To compare the similarity among the programs, *Cluster Analysis* is employed to group the programs based on the PCA data. The intuition is to cluster programs with similar characteristics into the same category. In our current implementation, we use Matlab to perform PCA transformation and SimPoint [44] to do K-means clustering.

5.2.3 Experimental Setup

Our experimental environment includes a 3.4 GHz 4-core Intel i7 system with 8 GB main memory. Each core employs 4-wide out-of-order execution with a three-level cache hierarchy. L1 and L2 cache are private to each core and their size are 32 KB (split L1 I/D) and 256 KB respectively. LLC (L3) reaches 8 MB which is shared among all cores. The system runs Linux 2.6.32 kernel with GCC compiler (GNU C/C++ Compiler) 4.3.2 with -O2 option. For the load balance analysis, we use a 1.87 GHz 16-core Intel Xeon system with 24 GB 1,333 MHz main memory to run the parallel version. The input set for feature extraction applications, except HOG, is provided by Mikolajczyk [40], which have been widely used as a standard dataset to evaluate different feature extraction algorithms. We use the INRIA Person Dataset [45] for HOG since it specializes in human detection. For feature matching and spatial verification algorithms, we use feature points extracted from the Oxford Building dataset [41], which is often adopted to evaluate retrieval systems.

2. The notion of Instructions per Byte relates to the traditional Bytes per Flop or Bytes/FLOP ratio in the High Performance Computing (HPC) field. We generalize FLOPs to Instructions so that the less contribution of floating-point operations in multimedia retrieval applications (Sections 4 and 6) is taken into account.

TABLE 2
Clustering Results

Cluster NO.	Program
Cluster 0	MPGdec, Sphinx, FaceRec (Train) FaceRec (Recognize), RayTrace, MSER
Cluster 1	MPGenc
Cluster 2	SIFT, HOG
Cluster 3	SURF

6 Characteristic Analysis

In this section, we first compare the multimedia retrieval algorithms with traditional multimedia benchmarks and study the sensitivity to input size of these algorithms. Then we analyze the architectural characteristics of multimedia retrieval algorithms at core level, chip level and out-of-chip level. Furthermore, we give some insights into architecture design and system evaluation for these multimedia retrieval algorithms.

6.1 Application Level Characteristics

In this section, we compare the architectural characteristics of multimedia retrieval algorithms with traditional multimedia benchmarks and study the sensitivity to input size. The following evaluations are based on sequential version of algorithms if not mentioned specifically.

6.1.1 Comparison to Traditional MM Benchmarks

First, we compare the similarity between the algorithms in MMRBench and traditional multimedia applications. Since only feature extraction algorithms use images or frames as the input, we only compare them with traditional ones. For traditional multimedia algorithms, we adopt programs in ALPBench [11], which covers the traditional multimedia algorithms, such as video and image processing applications. We use 29 architecture-independent metrics used in [46], which include instruction mix, branch information, dependence distance, locality metrics, and generate the PCA data. Based on the PCA data, we group the programs with clustering analysis. The clustering results are shown in Table 2.

As the data shows, the characteristics of feature extraction algorithms are different from those of traditional multimedia ones. SIFT, SURF, and HOG are clustered into different groups from traditional multimedia programs. Furthermore, almost all the programs in ALPBench are clustered into the same group, which means that they have similar behavior based on the overall architectural characteristics. To see

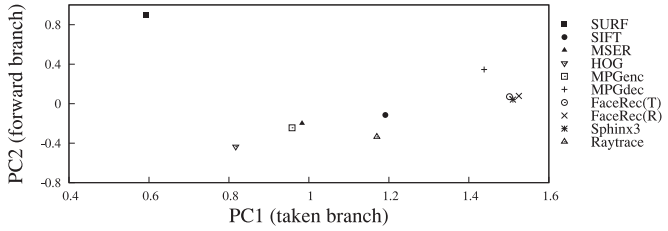


Fig. 3. PCA space built from branch characteristics.

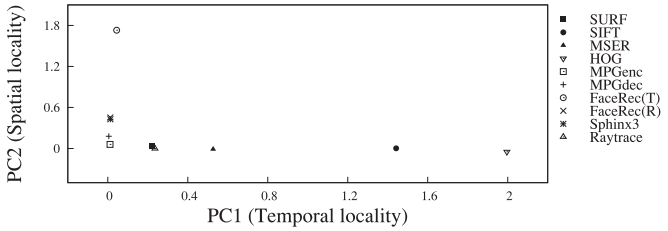


Fig. 5. PCA space built from data locality.

which characteristics lead to the difference, we further study the characteristics of their branch, ILP and data locality.

Note that PCA results have decreasing variance, with the first principle component (PC) containing the most information and the last one containing the least information. Here we retain top two PCs containing more than 90 percent information. In Fig. 3, PC1 measures taken branch information, i.e., a more positive value along PC1 indicates higher branch taken rate and PC2 reflects forward branch information, i.e., a more positive value along PC2 means higher forward branch rate. For ILP data shown in Fig. 4, higher PC1 indicates higher percentage of dependence distance between 2 to 8 while higher PC2 indicates higher percentage of dependence distance equal to 1. Higher percentage of short dependence distance may imply lower potential ILP in a program. For locality data shown in Fig. 5, higher PC1 indicates lower temporal locality. Lower PC2 indicates more benefit from larger cache line size. We observe that SURF has notably different characteristics in branch and ILP when compared to others. As the locality data shows, both SIFT and HOG are positioned in higher PC1 far from than others, which means they have much worse temporal data locality. Above three characteristics are the major factors that lead to the difference between the characteristics of feature extraction algorithms and those of traditional multimedia programs.

Insights: From the analysis above, we can conclude that the feature extraction algorithms have different characteristics from traditional multimedia benchmarks in many significant architectural aspects.

6.1.2 Input Sensitivity Analysis

The input size has great influence on the execution time of an application. When the application is used as the input of a simulation, especial full-system multi-core simulation, a large input may lead to too long simulation time to complete. However, if architecture characteristics of an application do not depend on the input size, a small input size would be enough, which can save a lot of evaluation time. To provide some advices on MMRBench-based evaluation, we measure the input sensitivity in this section.

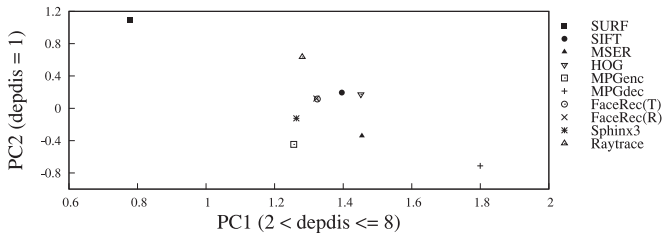


Fig. 4. PCA space built from ILP.

To evaluate the input sensitivity, we collect the 29-dimensional architecture-independent metrics for all algorithms in MMRBench under the input size from 1 to 20 MB. We then analyze them through PCA and clustering. The results are shown in Fig. 6 and Table 3. An application is assigned as an invariant type if its metrics gathered under different input sizes are clustered into the same group. A sensitive type means these metrics are totally clustered into different groups along with input changes. If the metrics of several neighboring input sizes are grouped into the same cluster, the application exhibits moderate change.

As the data shows, feature extraction algorithms exhibit invariant behavior with different input size. This is because they are image-based and process image one by one. Thus, their characteristics will change little when input size scales up. Besides, the other algorithms are sensitive to the input size. Through checking the data, we find data locality contribute the most of the changes. Especially for KD-Tree (Query) and LSH (Query), they are most sensitive because their irregular data access pattern will be greatly influenced when the input size changes. For example, KD-Tree (Query) searches each point through a large tree and then randomly accesses a logged index array whose size is equal to the number of points trained to build the tree. When the input size increases, more part of the tree and the index array will be accessed in a data-dependent way, which leads to worse locality.

Insights: Based on our analysis, a small input size is sufficient for feature extraction algorithms for evaluation. In contrast, the factors of input size should not be ignored when using the other algorithms for evaluation.

6.2 Core Level Analysis

As the basic component, the complexity of a processor core has a great influence on performance, power and chip size. In this section, we analyze the core level characteristics including pipeline execution efficiency, instruction fetch

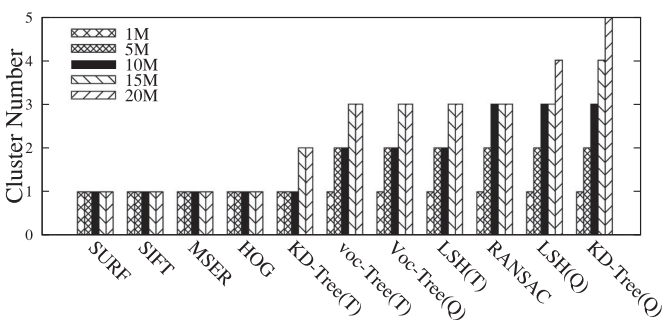


Fig. 6. Cluster result.

TABLE 3
Input Sensitivity

Type	Application
Invariant	SURF, SIFT, MSER, HOG
Moderate	KD-Tree(T), LSH(T), VOC-Tree(T) VOC-Tree(Q), RANSAC
Sensitive	KD-Tree(Q), LSH(Q)

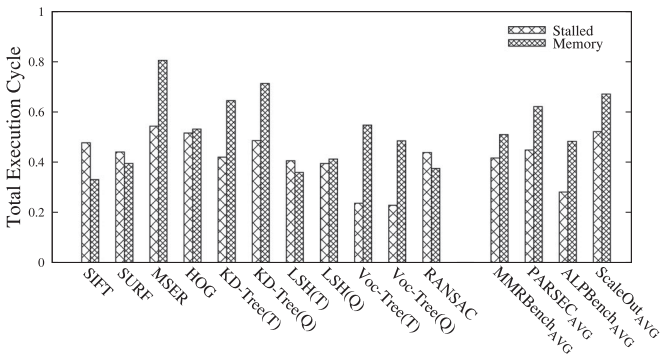


Fig. 7. Execution time breakdown and memory stall cycles.

efficiency, ILP and MLP, and the sensitivity of floating pointing units.

6.2.1 Pipeline Execution Efficiency

The execution cycles of a pipeline can be classified into commit cycles and stall ones. To check the pipeline efficiency, we collect the data of the percentage of stall cycles out of total execution cycles and the stall cycles caused by long latency memory accesses.[3] To make a comparison with traditional and cloud workloads, we also test PARSEC [17], ALPBench [11] and Scale-Out workloads [9].

As the results in Fig. 7 show, multimedia retrieval workloads suffer from high stalls (near 40 percent). It is similar to PARSEC and scale-out workloads but higher than traditional multimedia workloads (ALPBench). We find that the stall cycles occupy a higher percentage in MMRBench than traditional multimedia benchmarks, and memory stalls are not as severe as other benchmark suites. Although some multimedia retrieval workloads (e.g., MSER) have even higher memory cycles, most of them (e.g., SIFT and RANSAC) are mainly stalled due to computation process instead of memory operations.

Since branch misprediction can cause costly pipeline flush, we analyze the branch behavior of these multimedia retrieval workloads with three prediction strategies. The strategies include one-bit, two-bit saturating counter, and two-level adaptive predictor. The two-level adaptive predictor records the history of the last 8 occurrences of a branch (local history table) and has a global history table of 256 entries.

The results are shown in Fig. 8. As the data shows, most of multimedia retrieval algorithms have higher branch misprediction rate (more than 20 percent) under simple prediction strategy, such as one-bit and two-bit saturating counters, due

3. As mentioned in section 5.2.1, these memory cycles are the results of an approximate measurement and may be overlapped by pipeline execution. Therefore, the memory cycles may be higher than the stalled ones.

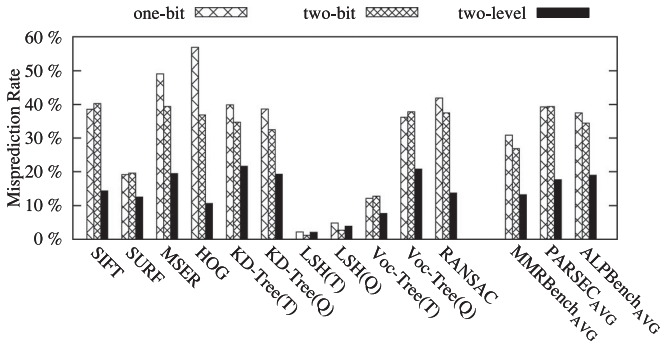


Fig. 8. Branch misprediction rate with one-bit, two-bit saturating counter, and two-level adaptive predictor.

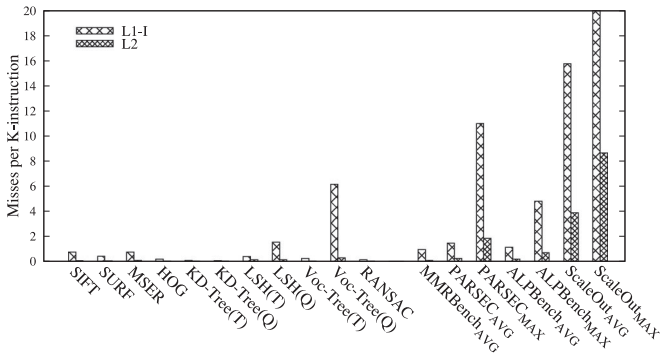


Fig. 9. Misses per K-instructions in L1-I cache and L2 cache.

to their special workflow. A high misprediction rate also exists in computation-intensive benchmarks like PARSEC and traditional multimedia workloads like ALPBench. In SURF, SIFT and HOG, a majority of computation happens in generating descriptors, which consists of many multi-level yet short loops (loop boundaries relate to the boundaries of sub-blocks in an image). For MSER, KD-Tree, VOC-Tree and RANSAC algorithms, there are a lot of data-dependent conditional branches, such as path selection in tree traversal or condition check of neighbor node status. These data-dependent branches make it challenging to achieve a better branch prediction. The only exception is LSH with low misprediction rate in both training and query stage. This is because its main loop traverses points sequentially and few conditional branches take place. In contrast, the prediction accuracy improves greatly with a two-level predictor. Yet, a further improvement would be necessary, since many of them still have over 10 percent misprediction rate.

Insights: High branch misprediction rate indicates that high stalls found in multimedia retrieval workloads are caused by misprediction penalty besides long-latency memory access. To mitigate this issue, a core design should embed sophisticated branch prediction mechanism, which can predict complex branch patterns in these workloads more efficiently.

6.2.2 Instruction Fetch Efficiency

Instruction fetch misses can also influence the efficiency of pipeline execution. To investigate instruction fetch efficiency, we collect the data of misses when fetching instructions from L1-I cache and L2 cache. As the results in Fig. 9 show, most of multimedia retrieval workloads have lower L1 instruction miss rate than other three workloads. Even

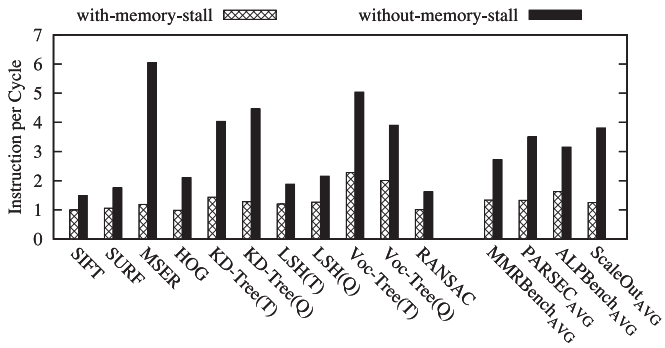


Fig. 10. Instruction committed per cycle in program. In order to evaluate the impact of memory stalls, we also provide the IPC without memory stall cycles, as shown by the black column in the figure.

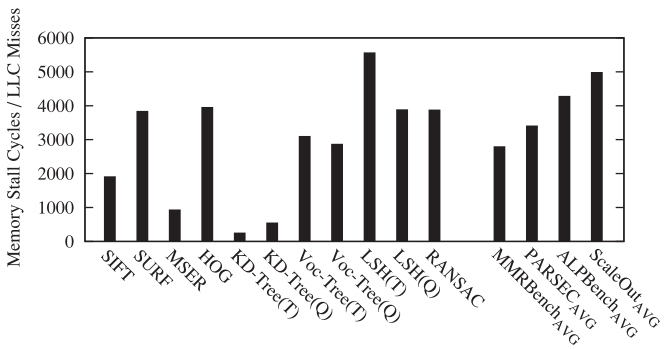


Fig. 12. Memory stall cycles divided by the number of LLC misses. The lower the value, the higher memory-level parallelism.

the highest one (VOC-Tree(Q)) is still lower than the max one in the PARSEC benchmark. All multimedia retrieval workloads rarely suffered from L2 instruction misses, which is very costly due to high latency when fetching instructions from it. In contrast, especially for scale-out workloads, their instruction working sets considerably exceed the L1-I cache capacity, even exceed the L2 cache capacity [9]. We note that this is mainly because most of the execution logic in multimedia retrieval workloads locates in several parts of core code sections, which fit well in L1 instruction cache.

Insights: The instruction working set of multimedia retrieval workloads fit well in current modern processor architecture, which is unlike some scale-out workloads [9]. It means we do not need to further increase the cache size to capture the whole instruction working set, which can save the on-chip real-estate and make the architecture design more power efficient.

6.2.3 ILP and MLP

In order to perform more operations simultaneously per cycle, modern processor architectures generally employ out-of-order technique to execute independent instructions in parallel. More than one instruction can be fetched, decoded and committed in one cycle. This strategy can speedup program's execution theoretically as long as less instruction dependency exists. In Fig. 10, we present the IPC values of multimedia retrieval workloads as well as those of PARSEC and ALPBench.

To evaluate the impact of memory stalls, we also provide the IPC without memory stall cycles. We observe that almost all multimedia retrieval workloads behave low IPC value (near 1.0). Even the highest one (VOC-Tree(T)) is still relatively low, with an IPC value 2.0 out of max value 4.0 (a popular configuration in modern processors). Furthermore, to exclude the memory stall cycles in the pipeline, we deduct the memory stalls from the execution pipeline, and then calculate IPC without memory stalls. The results show that the IPC without memory stalls (2.7) is higher than the IPC of the whole application; yet the average value is still lower than PARSEC, ALPBench, and scale-out workloads. This indicates that the instruction-level parallelism of multimedia retrieval applications is lower than traditional benchmarks, therefore the instruction issue window of pipeline could be scaled down.

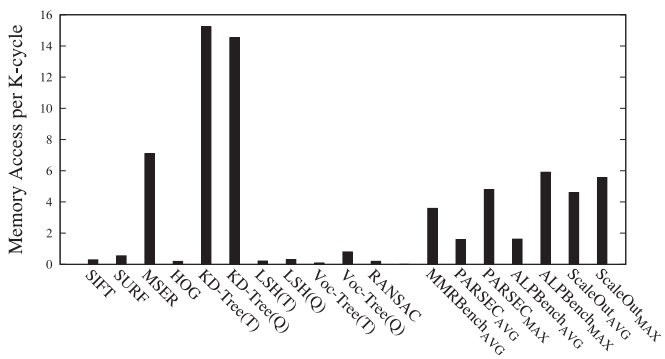


Fig. 11. Memory access per cycle in program.

Similar to exploiting ILP, out-of-order execution also enables memory-level parallelism (MLP) by performing multiple independent memory accesses in a time. This level of parallelism is also important to many programs because it can effectively hide long-latency memory accesses by overlapping them. To measure the potential MLP that a program can reach, we count the total number of last-level cache access and off-chip memory access (MPC) in these workloads. This is because LLC and off-chip memory access have much longer latency which should be hidden by MLP. The higher the MPC, the more beneficial the MLP can gain. The results are shown in Fig. 11. As such data shows, most of the multimedia retrieval workloads have lower MPC than parallel, scale-out and traditional multimedia workloads. This implies that they have lower potential MLP. On the other hand, we note that MSER and KD-Tree(Q/T) have very high MPC, which indicates they are more likely to achieve high MLP. There are two reasons for the high MPC in KD-Tree training and query. First, both the training and query phase of KD-Tree consist of large portions of frequent and irregular tree traversal and node accesses. Both of those operations would incur irregular and intensive memory accesses. Second, in training phase, multiple randomized trees are created by selecting the top N dimensions with the greatest variance to construct a KD-Tree forest, which makes the memory accesses more scattered than VocTree searching which only contains a single tree.

On the other hand, the cache misses typically come into bursts thus the overall MPC cannot reflect the potential benefit that could be achieved by overlapping the memory accesses. In order to evaluate the impact of memory-level parallelism, we calculate the memory stall cycles divided by the

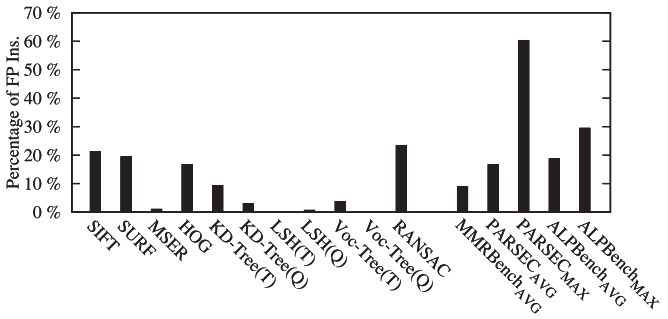


Fig. 13. Floating-point instruction ratio.

number of LLC misses, as shown in Fig. 12. The lower the ratio, the more latency is hidden by memory-level parallelism. The average value is comparatively high since the core frequency is nearly three times as high as memory frequency. From the results we can observe that the average value in MMRBench is lower than those in PARSEC, ALPBench, and scale-out workloads, which implies the LLC miss penalty of multimedia retrieval algorithms can be effectively hidden by memory-level parallelism.

Insights: Both of ILP and MLP techniques need hardware support including large instruction window, multiple decoders and execution units, large load-store queue and reorder buffer. The above results show that multimedia retrieval workloads have comparatively lower instruction-level parallelism than traditional benchmarks. Therefore, the instruction issue window of pipeline could be scaled down to simplify the CPU core structure. The overall MLP is lower than traditional benchmarks yet overlapping memory access operations is still an effective technique to hide the cache miss penalty. Here we do not consider much about MSER and KD-Tree(T/Q) since they are not as popular as others in real applications as mentioned in Section 4.2.

6.2.4 Sensitivity of Floating Point Operation

Floating point operations (FLOPs) exist in various algorithms. However, Floating Point Unit (FPU) is not only complex but also power-hungry compared to fixed-point units. Therefore, if FLOPs can be replaced by fixed-point operations, a hardware design can be simplified and more power-efficient. To analyze FLOP sensitivity, we first collect the percentage of FLOPs in each algorithm. As the data in Fig. 13 shows, KD-Tree (Q) and VOC-Tree (Q) virtually do not have any FLOPs, while others do. The result demonstrates that MMRBench has lower FLOP percentage than other benchmark suites like PARSEC and ALPBench, indicating that the FLOP intensity of MMRBench is lower.

To evaluate the FLOPs sensitivity of these algorithms, we convert all the FLOPs in each algorithm to fixed point operations: N bits are used to indicate the fraction part of the original float point number, and (32-N) bits are used to indicate the integral part. The differences between the results of fixed point version and those of float point versions are then used to evaluate the impact. We apply two metrics in the evaluation. The first one is error rate (ER), which is the deviation percentage between the outputs of two implementations of an algorithm. The second one is error impact rate (EIR), which indicates the difference in the final matching results between two implementations of an algorithm.

TABLE 4
Floating-Point Operation Sensitivity

Application	ER	Best Fractional Bits	EIR
SIFT	1.78%	22	0
SURF	2.28%	13	0
MSER	0%	0	0
HOG	0%	15	0
KD-Tree	3.44%	0	0
VOC-Tree	0%	0	0
LSH	0.01%	19	0
RANSAC	1.64%	7	0

Note that matching points will have to be verified in the Spatial Verification stage for the final results (Section 4).

We test different Ns and evaluate their impact. We only present the best results (lowest error rate) with certain N as shown in Table 4 for brevity. We observe that the deviation of such a transformation is very small for these multimedia retrieval algorithms and the largest error rate is less than 4 percent. Indeed, it is more important to obtain acceptable final results than intermediate ones for multimedia retrieval applications. Interestingly, there exists virtually no difference (zeros in Table 4) between these two implementations in the EIR results. Therefore, the deviation from fixed-point transformation can be mostly ignored for these applications.

The reason for the low FLOP sensitivity is that the calculation result of floating-point number of multimedia retrieval algorithms is mostly used for approximate comparison. For example, in VocTree training and query, each extracted feature is represented by a high-dimensional vector. A large portion of floating point operations are included in the calculation of Euclidean distance between the feature vector and descendent nodes of the current node. However, the program only needs to find the descendent node with the smallest distance, thus most floating-point results are only intermediate and their precision has little impact on the final result. Other algorithms in feature matching stage also have similar characteristics. Therefore, the precision of FLOPs in MMRBench has limited impact on the final matching result. However, such characteristics do not exist in applications where FLOP precision influences the validity of final results to a large degree, such as the MPEG encoding and decoding algorithms in ALPBench.

Insights: FPU is more complex than fixed-point ALU when considering area requirement and power consumption. In many scientific-related applications, FPU is very important since the precision of FLOPs is vital to their final results. In multimedia retrieval workloads, the average percentage of FLOPs is lower than other benchmarks, and more than half of the algorithms have extremely low (<5 percent) percentage of FLOPs. Besides, the precision of FLOP has little impact on the final result. Therefore, when designing hardware for multimedia retrieval applications, FPUs can be partially replaced by fixed-point units to achieve more cost efficiency and make the core structures even simpler.

6.3 Chip Level Analysis

To understand the characteristics of chip level, we analyze different factors which will influence the resource allocation among different hardware components, such as core and

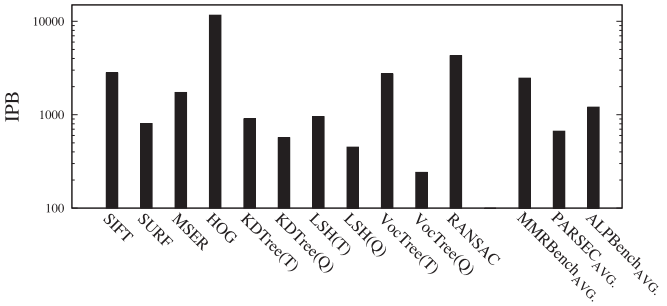


Fig. 14. Computation intensity over input size. IPB stands for Instructions per Byte.

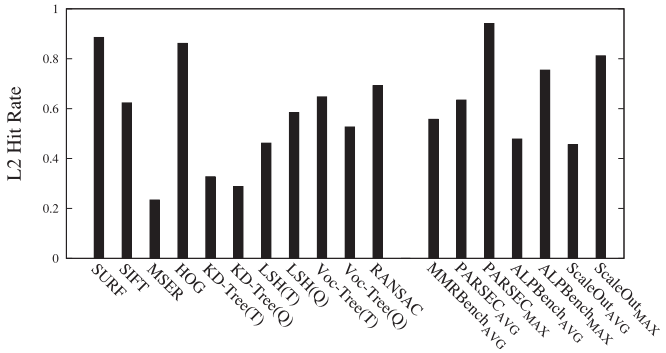


Fig. 15. L2 cache hit rate in program.

LLC. We first analyze computation intensity of these multimedia retrieval workloads. Moreover, we also analyze the cache behavior and the CoV of execution time among different threads when the multi-threaded versions are executed. Such analysis will provide some insights into what should be paid more attention to when designing a chip for those multimedia retrieval algorithms. Note that, we do not show the results of computation intensity and load balance for scale-out workloads here. The major reason is most of scale-out applications are service-oriented programs. It is difficult to define input size and compare the CoV of execution time of different threads for them when they process client requests.

6.3.1 Computation Intensity

To evaluate the computation intensity, We use IPB as the metric as discussed in Section 5. The higher IPB, the more computation resource an algorithm may need to process the input. We collect the IPB results of each algorithm in MMRBench and those in PARSEC and ALPBench. The data in Fig. 14 shows that the average IPB of MMRBench is higher, i.e., more computational intensive, than those of PARSEC and ALPBench.

The reason behind such results is because these multimedia retrieval algorithms are designed to guarantee certain retrieval accuracy. To achieve such a goal, some complex transformations, such as resizing or cropping of the input images or frames, are included in these algorithms. Moreover, they generally extract hundreds of feature points to represent an image or a video frame. Each feature point will be described as a multi-dimensional vector, e.g., 64 or 128. In the feature matching stage, each point needs to traverse to the backend database. The algorithms then determine whether there are matching pints through computing the distance (usually Euclidean distance).

Insights: These multimedia retrieval algorithms are facing a great challenge in real-time analysis especially with the explosion of image and video data in Cloud or Internet computing environment. For example, only about three images can be processed through the sequential version of SIFT or SURF on an Intel i7 processor according to our experiment. Thus, special accelerating scheme, such as more computational cores, should be explored for these multimedia retrieval algorithms.

6.3.2 Cache Sensitivity

Fig. 15 shows the L2 hit rate in these workloads. We observe that most of multimedia retrieval workloads have similar behavior in L2 cache data accesses (hit rate >= 50%) when compared to PARSEC and higher L2 hit rate than ALPBench and scale-out workloads. This means modestly-sized L2 cache mitigates most of the L1 cache miss. Fig. 16 shows the LLC misses per 1,000 instruction for these algorithms in different LLC sizes (from 1/32 to 32 MB). For most of these algorithms, while the LLC size is increasing, the number of LLC misses will start to decrease rapidly, and finally reach a relative stable level. Such a trend suggests the existence of working set. When the LLC changes from a size smaller than working set to a size larger than working set, the number of LLC misses will decrease a lot suddenly and will not decrease significantly with a even larger LLC, since the LLC has been already enough to accommodate the working set. We can see most of these algorithms have relative small working sets. When we shrink the LLC size to 1/8 MB, most of these workloads still have low LLC misses.

On the other hand, the LLC miss rate of KD-Tree(T/Q) and MSER keeps decreasing with the expansion of LLC for their dense and irregular memory accesses (as shown in Fig. 11). Therefore, for such algorithms, larger cache size will be beneficial. On the other hand, if the chip area is the major consideration and KD-tree matching is not an essential part, we can use the design with modest LLC cache size.

Insights: Modern processor architecture employs deep cache hierarchy to bring required data closer to core. Higher L2 hit rate in multimedia retrieval workloads indicates modestly-sized L2 cache (256 KB) plays a good role in enhancing overall performance. On the other hand, a larger LLC consumes more area (LLC is the largest structure on chip) and power, but gives little improvement to most multimedia retrieval workloads due to their small working set. Considering LLC is shared among cores, small LLC size requirement means more cores can share one LLC on a chip without side effect of high LLC miss. A small LLC size also brings the benefit of short data access latency.

6.3.3 Load Balance

Load balance is a very important issue for the performance of parallel applications executed on a multi-core architecture. To obtain insights into parallel optimization and system design of these multimedia retrieval algorithms, we evaluate their load balance characteristics based on the parallel version, which exploit the image-level parallelism in these algorithms. Input images are evenly distributed among threads during execution.

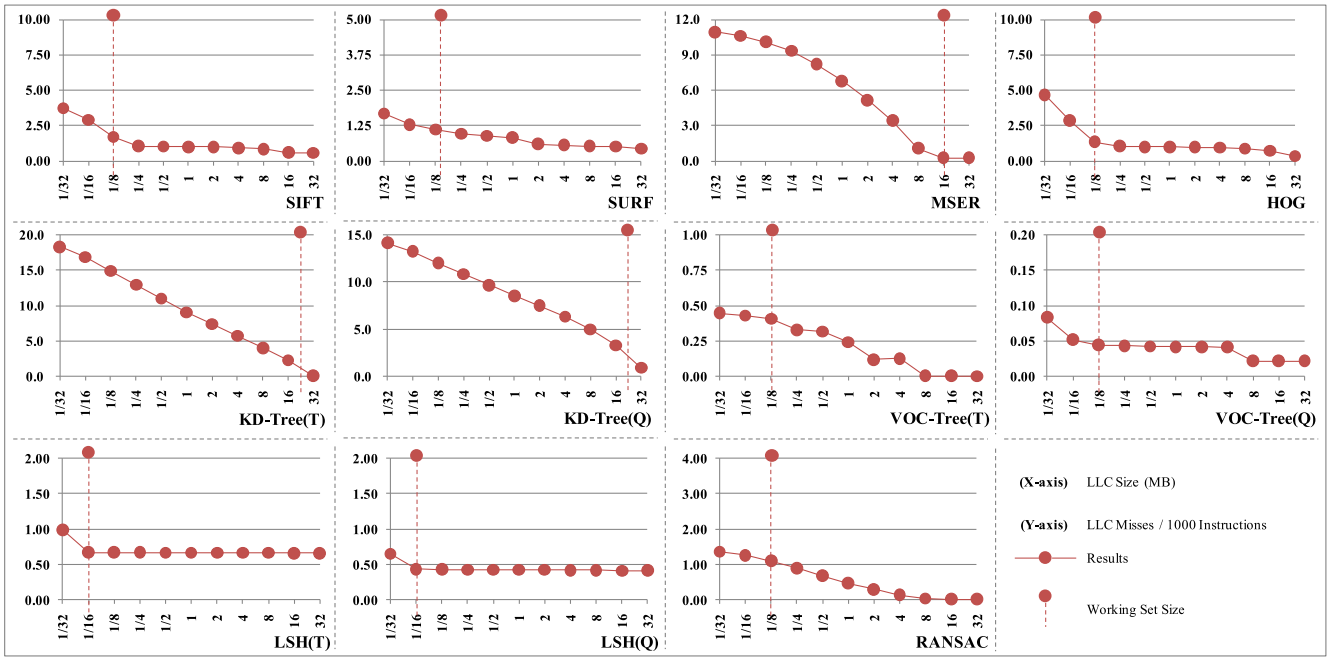


Fig. 16. Cache sensitivity.

We collect the execution time of each thread with different thread counts, e.g., 4, 8 and 16. Each thread is executed on a separate processor core to minimize the impact from intra-core resource contention. We use Coefficient of Variations (CoV) of the execution time from individual threads to indicate the level of load imbalance in the system. The results are shown in Fig. 17. It shows that, except MSER and HOG, all other algorithms show obvious imbalance (CoV greater than 10 percent) among threads as number of cores increases. As the data show, load imbalance in multimedia retrieval algorithms is more severe than that of PARSEC and ALPBench.

The reason behind such a result is that fundamental multimedia retrieval algorithms are all feature point based. In other words, the required computation in each thread is proportional to the number of feature points processed by it. Fig. 18 shows the number of feature points detected in each image in our small image set. It is obvious that different images have different point numbers. Some images have a larger number of feature points, while the others have a smaller one, which intuitively depends on the information in the image. It indicates that the various number of feature points at image level leads to different workloads for the threads to work on, hence different execution time. Moreover, the imbalance workload also exists at block level. To illustrate this problem, we evenly divide a 640 × 480 image into 4 × 4 blocks and collect the number of feature points detected in each block. The results are shown in Fig. 19. It turns out the number of feature points detected in each block is also different. Since the feature point count varies at both image and block level, it is challenging to achieve a statically balanced parallel design for these algorithms. Therefore, when mapping these algorithms to parallel hardware, it should pay more attention for dynamic load balance.

We expect approaches along the lines of Adaptive Mesh Refinement (AMR) can help reduce load imbalance in this context. System work and architectural support are also interesting to investigate, when extending AMR or others alike to multimedia retrieval applications. We sketch one preliminary approach in the following.

As described above, the workloads are imbalanced at both image and block level. Therefore, it is difficult to exploit coarse-grained parallelism at these two levels to achieve better parallel performance. Moreover, all the three stages in multimedia retrieval applications are feature points based and the required computation of each

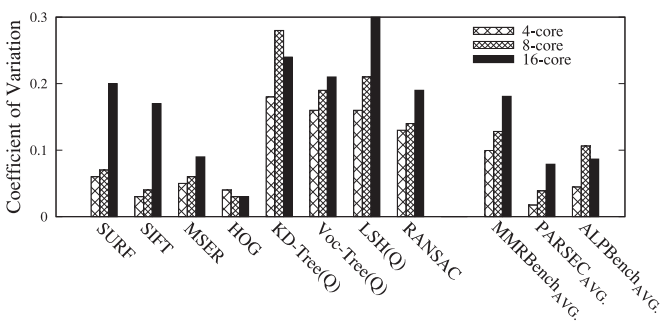


Fig. 17. Variations of per-thread execution time as an indicator of load imbalance.

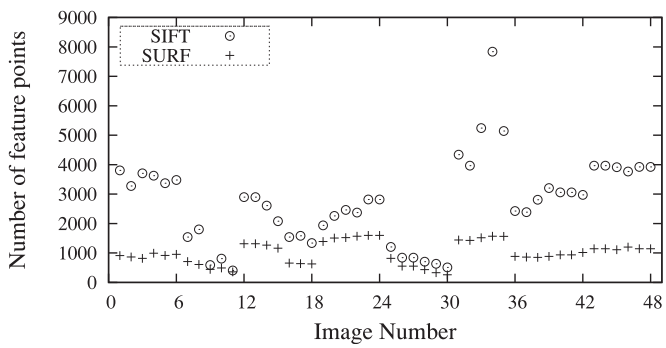


Fig. 18. Imbalance at image level.

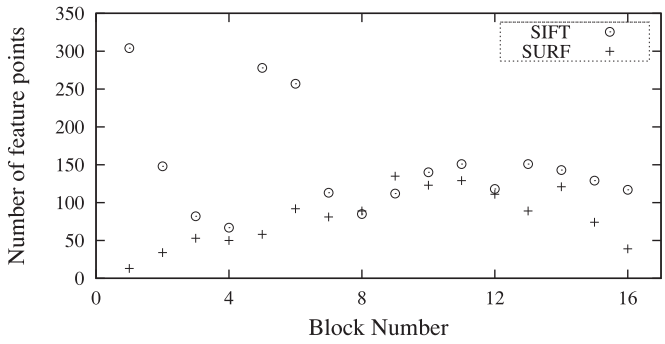


Fig. 19. Imbalance at block level.

algorithm is proportional to the number of feature points extracted from an input image. Therefore, we can exploit a fine-grained feature point level parallelism and design a scheduling scheme to allocate resource based on the number of feature points in input images. Based on this insight, we design a dynamic fine-grained pipeline parallelism for these multimedia retrieval algorithms. We use SURF and SIFT as a case study to verify the effectiveness of this approach.

We first partition the algorithms into two stages. The first stage is used to detect how many feature points in an input image. The second is to describe each point into a high-dimensional vector. After partitioning these two algorithms into a two stage pipeline, the description stage is further parallelized through exploiting the feature point level parallelism. The computation resource is then dynamically allocated to match the feature point counts in the images and achieve better load balance. To illustrate the effectiveness of such a design, we compare the results with those with image- and block-level parallelism on a 16-core machine. As the data in Fig. 20 show, such a fine-grained dynamic scheduler outperformance the other two ones.

Insights: These multimedia retrieval algorithms face a great challenge on load balance. Fine-grained parallelism at feature point level should be exploited in multimedia retrieval workloads. In addition, a certain amount of dynamic control should be fused to allocate the resource more efficiently when designing system and architecture, such as hardware thread scheduling mechanism. Considering such inherent thread-level parallelism in these workloads, they would be well-suited by architectures offering multiple cores on one chip.

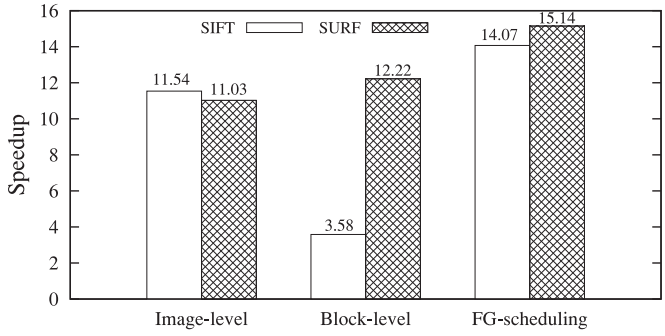


Fig. 20. Parallel performance on a 16-core machine with image-level parallelism (Image-level), block-level parallelism (Block-level) and our proposed fine-grained adaptive scheduling (FG-scheduling).

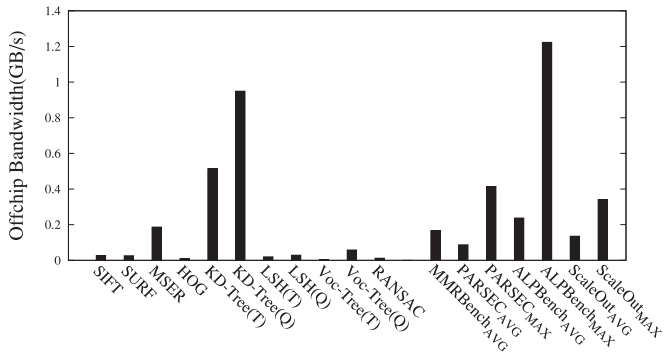


Fig. 21. Off-chip memory bandwidth.

6.4 Inter-Chip Level Analysis

Although the off-chip memory access latency has not been improved a lot, the off-chip memory bandwidth has a great progress. For example, the Intel core-i7 processor can achieve peak bandwidth at 21 GB/s with two independent memory channels. Architecture designers hope that such a large bandwidth can help improve programs which have great pressure on memory bus. We show the off-chip memory bandwidth of these workloads in Fig. 21. As the results show, most of multimedia retrieval workloads have very low off-chip bandwidth utilization compared to parallel, scale-out and traditional multimedia workloads. We note the main reason is due to its high computation intensity as mentioned in Section 6.3. Multimedia retrieval workloads require more computation on each data unit and also cause complex data dependencies.

Insights: Large memory bandwidth requires large area memory controllers and high power consumption by multiple memory bus. But our experiment shows that multimedia retrieval workloads do not need such a large off-chip bandwidth. Thus we can reduce the number of memory channel and make memory controllers simpler to make processor more cost-effective.

6.5 Architectural Insights

We have analyzed the characteristics of these representative multimedia retrieval workloads. Based on this analysis, we summarize some insights to architecture design.

- *Application level design:* (1) Feature extraction algorithms are different from traditional multimedia benchmarks in many significant architectural aspects; (2) A small input size is sufficient for feature extraction algorithms for evaluation while the size of input is still an important factor when evaluating other algorithms.
- *Core level design:* (1) A small instruction cache is sufficient for catching entire instruction working set and sophisticated branch prediction mechanism is necessary to deal with complex branch pattern in these workloads; (2) The ILP and MLP is comparatively low than traditional benchmarks, thus a simpler core design could save the area and energy; (3) Floating-point units can be partially replaced with fixed-point units to make more cores on a die without losing performance and accuracy.
- *Chip level design:* (1) A modestly sized L2 cache acts efficiently in data fetching and we can reduce

last-level cache size to reduce area and power consumptions; (2) Due to the abundant thread-level parallelism in these algorithms at image-level, sub-block-level and feature-point-level, feature-point-level parallelism is a critical optimization point for them; (3) load imbalance is also a key issue to pay attention to and fine-grained feature-point level parallelism can be exploited for great performance improvement, together with dynamic resource scheduling scheme or architectural support.

- *Inter-chip level design:* Off-chip memory bandwidth can be scaled down since low bandwidth utilization found in these workloads. This can further improve the cost effectiveness of processors designed for multimedia retrieval workloads.

7 Conclusions and Future Work

We have assembled and designed a multimedia retrieval benchmarking framework (MMRBench) for architecture design and system evaluation. In MMRBench, we provide multiple algorithm versions, supporting tools and a flexible framework. The design makes it easier for an end user to generate customized benchmark suites, or even a complete multimedia retrieval system, for various system requirements.

Furthermore, we analyze the architectural characteristics of these algorithms and offer insights to architecture design for multimedia retrieval applications, including core level, chip level and inter-chip level. We also sketch approaches to tackling some of the challenges.

Currently, many related algorithms are emerging, such as deep learning algorithms. We plan to survey and extend such algorithms in MMRBench in our future work. Moreover, we will also extend GPU version for these algorithms in MMRBench.

Acknowledgments

The authors are grateful to supports from the National High Technology Research and Development Program of China (No. 2015AA015303), and the National Natural Science Foundation of China (No. 61370081). They would also like to thank all their anonymous reviewers for valuable feedback on the paper. Weihua Zhang is the corresponding author.

References

[1] "Cisco visual networking index: Forecast and methodology, 2014-2019," 2015.
[2] (2015). 300+ hours of video uploaded to Youtube every minute. [Online]. Available: http://www.reelseo.com/youtube-300-hours/
[3] (2013). Facebook stores 240 billion photos and adds 350 million more a day. [Online]. Available: http://www.businessinsider.com/facebook-stores-240-billion-photos-2013-1
[4] B. Yang, T. Mei, X.-S. Hua, L. Yang, S.-Q. Yang, and M. Li, "Online video recommendation based on multimodal fusion and relevance feedback," in *Proc. ACM Int. Conf. Image Video Retrieval*, 2007, pp. 73–80.
[5] Y. Gao, J. Tang, R. Hong, Q. Dai, T.-S. Chua, and R. Jain, "W2GO: A travel guidance system by automatic landmark ranking," in *Proc. ACM Int. Conf. Multimedia*, 2010, pp. 123–132.
[6] A. Joly, C. Frelicot, and O. Buisson, "Robust content-based video copy identification in a large reference database," in *Proc. ACM Int. Conf. Image Video Retrieval*, 2003, pp. 511–516.
[7] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 404–417.
[8] J. Bauer, N. Sunderhauf, and P. Protzel, "Comparing several implementations of two recently published feature detectors," in *Proc. Int. Conf. Intell. Auton. Syst.*, 2007, pp. 143–148.
[9] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: A study of emerging scale-out workloads on modern hardware," in *Proc. 17th Int. Conf. Archit. Support Program. Languages Operating Syst.*, 2012, pp. 37–48.
[10] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis," in *Proc. IEEE Int. Conf. Data Eng.*, 2010, pp. 41–51.
[11] M. lap Li, R. Sasanka, S. V. Adve, Y. kuang Chen, and E. Debes, "The ALPBench benchmark suite for complex multimedia applications," in *Proc. IEEE Int. Symp. Workload Characterization*, 2005, pp. 34–45.
[12] 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 *Proc. IEEE Int. Workshop Workload Characterization*, 2001, pp. 3–14.
[13] V. J. Reddi, B. C. Lee, T. Chilimbi, and K. Vaid, "Web search using mobile cores: Quantifying and mitigating the price of efficiency," in *Proc. Int. Symp. Comput. Archit.*, 2010, pp. 314–325.
[14] Y. H. Wan, Q. L. Yuan, S. M. Ji, L. M. He, and Y. L. Wang, "A survey of the image copy detection," in *Proc. IEEE Conf. Cybern. Intell. Syst.*, 2008, pp. 738–743.
[15] (2016). The Standard Performance Evaluation Corporation. [Online]. Available: http://www.spec.org
[16] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proc. Int. Symp. Comput. Archit.*, 1995, pp. 24–36.
[17] C. Bienia and K. Li, "PARSEC 2.0: A new benchmark suite for chip-multiprocessors," in *Proc. Annu. Workshop Model., Benchmarking Simul.*, Jun. 2009.
[18] (2016). Illinois Microarchitecture. [Online]. Available: http://impact.crhc.illinois.edu/parboil.php
[19] S. K. Venkata, I. Ahn, D. Jeon, A. Gupta, C. Louie, S. Garcia, S. Belongie, and M. B. Taylor, "SD-VBS: The San Diego vision benchmark suite," in *Proc. IEEE Int. Symp. Workload Characterization*, 2009, pp. 55–64.
[20] J. Clemons, H. Zhu, S. Savarese, and T. Austin, "MEVBench: A mobile computer vision benchmarking suite," in *Proc. IEEE Int. Symp. Workload Characterization*, 2011, pp. 91–102.
[21] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Ferret: A toolkit for content-based similarity search of feature-rich data," in *Proc. 1st ACM SIGOPS/EuroSys Eur. Conf. Comput. Syst.*, 2006, pp. 317–330.
[22] N. Tuck and D. M. Tullsen, "Initial observations of the simultaneous multithreading pentium 4 processor," in *Proc. Parallel Archit. Compilation Tech.*, 2003, p. 26.
[23] N. Hardavellas, I. Pandis, R. Johnson, N. G. Mancheril, A. Ailamaki, and B. Falsafi, "Database servers on chip multiprocessors: Limitations and opportunities," in *Proc. Int. Conf. Innovation Database Res.*, 2007, pp. 79–87.
[24] D. Nan, X. Wei, J. Xu, X. Haoyu, and S. Zhenya, "CESMTuner: An auto-tuning framework for the community earth system model," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun.*, 2014, pp. 282–289.
[25] S. Xu, W. Xue, and H. X. Lin, "Performance modeling and optimization of sparse matrix-vector multiplication on NVIDIA CUDA platform," *J. Supercomput.*, vol. 63, no. 3, pp. 710–721, 2013.
[26] W. Xue, C. Yang, H. Fu, X. Wang, Y. Xu, L. Gan, Y. Lu, and X. Zhu, "Enabling and scaling a global shallow-water atmospheric model on tianhe-2," in *Proc. IEEE 28th Int. Parallel Distrib. Process. Symp.*, 2014, pp. 745–754.
[27] P. Ranganathan and N. Jouppi, "Enterprise IT trends and implications for architecture research," in *Proc. High-Perform. Comput. Archit.*, 2005, pp. 253–256.
[28] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward dark silicon in servers," in *Proc. Int. Symp. Microarchitecture*, 2011, 6–15.
[29] T. Kgil, A. Saidi, N. Binkert, R. Dreslinski, S. Reinhardt, K. Flautner, and T. Mudge, "PicoServer: Using 3D stacking technology to enable a compact energy efficient chip multiprocessor," in *Proc. Archit. Support Program. Languages Operating Syst.*, 2006, pp. 117–128.
[30] L. Zhao, R. Iyer, S. Makineni, J. Moses, R. Illikkal, and D. Newell, "Performance, area and bandwidth implications on large-scale CMP cache design," in *Proc. Workshop Chip Multiprocessor Memory Syst. Interconnect*, 2007.

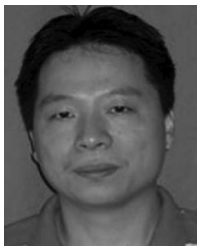
[31] C. Kozyrakis, A. Kansal, S. Sankar, and K. Vaid, "Server engineering insights for large-scale online services," in *Proc. Int. Symp. Microarchitecture*, 2010, pp. 8–19.
[32] P. Lotfi-Kamran, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, and B. Falsafi, "Scale-out processors," in *Proc. Int. Symp. Comput. Archit.*, 2012, pp. 500–511.
[33] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[34] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," in *Proc. ACM Int. Conf. Multimedia*, 2010, pp. 1469–1472.
[35] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2005, pp. 886–893.
[36] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2006, pp. 2161–2168.
[37] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.
[38] M. Datar and P. Indyk, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. ACM Symp. Comput. Geom.*, 2004, pp. 253–262.
[39] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[40] K. Mikolajczyk. (2007). Local feature evaluation dataset. [Online]. Available: http://www.robots.ox.ac.uk/vgg/research/affine/
[41] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2007, pp. 1–8.
[42] (2016). Intel VTune amplifier XE performance profiler. [Online]. Available: http://software.intel.com/en-us/intel-vtune-amplifier-xe
[43] (2015). Valgrind. [Online]. Available: http://valgrind.org/
[44] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in *Proc. Int. Conf. Archit. Support Program. Languages Operating Syst.*, 2002, pp. 45–57.
[45] (2016). INRIA Person Dataset. [Online]. Available: http://pascal.inrialpes.fr/data/human/
[46] A. Phansalkar, A. Joshi, and L. K. John, "Analysis of redundancy and application balance in the spec cpu2006 benchmark suite," in *Proc. 34th Int. Symp. Comput. Archit.*, 2007, pp. 412–423.

Yunping Lu is currently working toward the PhD degree in the School of Computer Science at Fudan University. Her research interests are in compilers, computer architecture, parallelization and systems software.

Xin Wang is currently working toward the graduate degree in the Software School of Fudan University and working in the Architecture Group of Parallel Processing Institute. His work is related to computer architecture, simulation, parallel optimization and so on.

Weihua Zhang received the PhD degree in computer science from Fudan University in 2007. He is currently an associate professor of Parallel Processing Institute, Fudan University. His research interests include compilers, computer architecture, parallelization and systems software.

Haibo Chen received the BSc and PhD degrees in computer science from Fudan University in 2004 and 2009, respectively. He is currently a professor in School of Software, Shanghai Jiao Tong University, doing research that improves the performance and dependability of computer systems. He is a senior member of the IEEE and the IEEE Computer Society.



Lu Peng received the PhD degree in computer engineering from the University of Florida in Spring 2005. He is currently an associate professor in the Electrical and Computer Engineering Department, Louisiana State University. His research focus on computer architecture, memory hierarchy system, reliability, power efficiency and other issues in processor design.

Wenyun Zhao received the master's degree from Fudan University in 1989. He is a full professor of the School of Computer Science, Fudan University. His current research interests include software reuse, software product line, software component, and architecture.