BVG Trust

J. Sci. Trans. Environ. Technov. 2018, 11(4) : 188-195

Scientific Transactions in Environment and Technovation

Framework for Big Data applications on Server Schema

https://doi.org/10.56343/STET.116.011.004.004
http://stetjournals.com

R.Anitha*, N.Subhalakshmi and K.Savima

Department of Computer Science, STET Women's College, Sundarakkottai, Mannargudi, 614016, Tamil Nadu, India.

Abstract

Big data needs to be considered in terms of how the data will be manipulated. The size of the data set will impact data capture, movement, storage, processing, presentation, analytics, reporting, and latency. Traditional tools quickly can become overwhelmed by the large volume of big data. Latency—the time it takes to access the data—is an important consideration as volume. Suppose to run an ad hoc query against the large dataset or a predefined report, a large data storage system is not a data warehouse, however, and it may not respond to queries in a few seconds. It is, rather, the organization-wide repository that stores all of its data and is the system that feeds into the data warehouses for management reporting. One solution to the problems presented by very large data sets might be to discard parts of the data so as to reduce data volume, but this is not always practical. Regulations might require that data be stored for a number of years, or competitive pressure could force to save everything. The characterization results vary with the Choice of the type of servers we choose, big vs little core-based server for energy-efficiency is significantly influenced by the size of data, performance constraints and presence of an accelerator. Furthermore, the micro architecture-level analysis gives us a clear picture of the analysis part that is much needed on the server architectures.

Key words: Performance, Power, Characterization, Big Data, High-Performance server, Low-Power server, Accelerator

Received : June 2017 Revised and Accepted : May 2018

INTRODUCTION

Advances in various branches of technology, data sensing, data communication, data computation and data storage are driving an era of unprecedented innovation for information retrieval. The world of Big Data is constantly changing and producing huge amounts of data that creates challenges to process the applications using existing solutions. Big data applications require computing resources and storage subsystems that can scale to manage massive amounts of diverse data. Individuals, businesses, governments and society as a whole now have access to enormous collections of big data, empowering them to build their own analytics. Data centers are therefore required to introduce more nodes to their infrastructure or replace their existing hardware with more powerful systems to respond to this growing demand. This trend increases the infrastructure cost and power consumption. We believe this is the right time to identify the right computing platform for Big Data analytics processing that can provide a balance between processing capacity and power efficiency. Emerging data applications, in particular from web service domain, share many inherent characteristics that are fundamentally different from traditional desktop, parallel and scale-out applications (Gao *et al.,* 2013). Big data analytics applications in these domains heavily rely on big-data-specific deep machine learning and data mining algorithms, and are running complex database software stack with significant interaction with I/O and OS, and exhibit high computational intensity, memory intensity, I/O intensity and control intensity. In addition, unlike conventional CPU applications, big data applications combine a high data rate requirement with high computational power requirement, in particular for real-time and near-time performance constraints. This new set of characteristics is necessitating a change in the direction of server-class micro architecture to improve their computational efficiency. However, while demand for data center computational resources continues to grow as the size of data grows; the semiconductor industry has reached its physical scaling limits and is no longer able to reduce power consumption in new chips. Physical design constraints, such as power and density, have therefore become the dominant limiting factor for scaling out datacenters (Ferdman *et al.,* 2012; Ghazal *et al.,*2013; Kontorinis *et al.,* 2012; Guiterrez *et al.,*2014). Current server designs, based on commodity homogeneous processors, will therefore not be the most efficient in terms of performance / watt

*Corresponding Author :
email: *anithaocsqa@gmail.com*

P - ISSN 0973 - 9157
E - ISSN 2393 - 9249

to process big data applications (Guiterrez *et al.,* 2014;Reddi *et al.,* 2010). In this work we show that while high performance big cores are optimized for traditional CPU applications, for big data they are very inefficient and are not satisfying their computational efficiency requirements. In exploring the choice of server architecture for big data, in this paper, we present a comprehensive analysis of the measurement of power and performance of big data applications on two very distinct micro architectures; a high performance big Xeon core and another a low power embedded-like little Atom core. These two types of servers represent two schools of thought on server architecture design: using big core like Xeon, which is a conventional approach to designing a high-performance server, and the Atom, which is a new trajectory in server design that advocates the use of a low-power core to address the dark silicon challenge facing servers (Reddi *et al.,* 2010; Homayoun *et al.,* 2012; Andersen David *et al.,* 2009; Hardvells *et al.,* 2011; Kontorinis *et al.,* 2014).In addition to power and performance study, we have also performed the Energy-DelayX Product (EDXP) analysis to evaluate the trade-off between power and performance to understand how near real-time performance constraints for big data analytics affects the choice of big vs. little core server as a more efficient architecture. As for big data applications, achieving a higher processing rate of large amount of data is a prime target and hence we have evaluated the processing capability under different data size (per node) by using two metrics – Data Processed Per Second (DPS) and Data Processed Per Joule (DPJ). The analysis helps us to understand as to how the choice of big vs. little cores introduces significant tradeoff in performance, power, energy-delay, and processing capacity for efficient and near-time processing of big data applications. The results demonstrate that while in most applications, server with little cores is more efficient in terms of EDP and DPJ, with constraints for near real-time performance the most efficient server architecture depends on the data size and the type of application. As chips are hitting power limits, computing systems are moving away from general-purpose designs and toward greater specialization. Hardware acceleration through specialization has received renewed interest in recent years, mainly due to the dark silicon challenge. To find out the right architecture for big data processing, it is important to understand ways for deploying an accelerator, such as FPGA, would necessitate adapting the choice of big vs. little cores. The post acceleration code characteristics are important to find the right architecture for efficient processing of big data applications. For this purpose, we analyze the choice of big vs. little core-based servers for the code that remains for the CPU after assuming the hotspots are off loaded to an accelerator, compared with the choice of big vs. little before acceleration. Overall, our characterization results across a wide range of real-world big data applications and various software stacks demonstrate how the choice of big vs little core-based servers for energy-efficiency is significantly influenced by the size of data, performance constraints and presence of accelerator. To provide insight into whether current design of big and little core requires improvement in the micro architecture parameters for efficient big data processing we further perform a comprehensive micro architecture characterization. This study assists in determining whether big data workloads require innovation in microprocessor micro architecture design.

Background on Big Data applications

The "cloud" is a new platform that has been used to cost effectively deploy an increasingly wide variety of applications. Vast amount of data is now stored in a few places rather than distributed across a billion isolated computers, and hence creates opportunities to learn from the aggregated data. The rise of cloud computing and cloud data storage, therefore, has facilitated the emergence of big data applications. Big data applications are characterized by four critical features, referred as the four "Vs", volume, velocity, variety, and veracity. Big data is inherently large in volume. Velocity refers to how fast the data is coming in and to how fast it needs to be analyzed. In other words, velocity addresses the challenges related to processing data in real-time. Variety refers to the number and diversity of sources of data and databases, such as sensor data, social media, multimedia, text, and much more. Veracity refers to the level of trust, consistency and completeness of data. Traditionally, cloud servers mainly use high performance CPU cores such as Xeon. However, low-power embedded cores such as Atom are gradually entering the server market. Therefore, it is important to characterize emerging big data applications on these two different platforms to understand their computational need and architectural bottlenecks.

Dominant Big Data Workloads

Big data Workload

Hadoop Micro benchmark

Apache Hadoop is an open source Java-based framework of Map Reduce implementation. It assists the processing of large datasets in a distributed computing environment and stores data in highly fault-tolerant distributed file system, HDFS. Hadoop has numerous micro benchmarks from which we have

included a combination of I/O intensive and CPU intensive applications as follows: • Word Count reads text files and determines how often the words appear in a set of files. Word count is a CPU intensive application (Huang *et al.,* 2010).Sort uses the map/reduce framework to sort the input directory in the output directory. The actual sorting occurs in the internal shuffle and sort phase of Map Reduce. The data is transferred to reducer that is an identity function. Sort is an I/O intensive application (Huang *et al*., 2010). Grep extracts matching strings provided by user from text files and sorts matching strings by their frequency. Grep is a CPU intensive application.Tera Sort performs a scalable Map Reduce-based sort of input data. It first samples the input and computes the input distribution by calculating the quantities equal to the number of reduces that uses a sorted list of N-1 sampled keys to define the key range for each reduce. TeraGen command generates the large random data for Tera Sort (Huang *et al.,* 2010).Test DFSIO-write/read is a storage throughput test that is divided into two parts, Test DFSIO-Write and Test DFSIO Read to write and read data to/from HDFS, respectively.

Graph Mining

Graph construction can be very challenging because of complex iterative and data-dependent nature of the graph. Hadoop is well suited for this task, but requires expertise to handle graph complexities. Graph Builder addresses this challenge by providing a scalable graph construction software library for Hadoop. Graph Builder constructs graphs for Page Rank and LDA algorithms implemented on Power Graph (Willke *et al.,* 2012). Collaborative Filtering (CF) Recommendation is a technique used by many recommender systems to predict the preference of users based on their previous rating history. Clustering is one of the fundamental tasks in Data Mining. Clustering assembles data items into groups based on their similar features. We have analyzed mean shift clustering as it is a non-parametric clustering technique that does not require prior knowledge of the number of clusters. Association Rule Mining is a well-known approach for exploring association between various parameters in large databases. We have analyzed FP (Frequent Pattern)-Growth; a resource intensive program that aims to determine item sets in a group and identifies which items typically appear together. Sequential Pattern Mining Framework (SPMF) is an open-source data mining library written in Java. It offers numerous data mining algorithms for sequential pattern, rule mining and frequent item mining for which we have selected Equivalence Class Transformation (ECLAT), Rule Growth, Generalized Sequential Pattern (GSP) and Sequential Pattern Discovery using Equivalence classes (SPADE).

Scales-Out Workloads

Classification technique learns from the existing categorizations and groups the unclassified items to the best corresponding category (Ferdman *et al*., 2012).Graph-analysis is performed by implementing the Trunk Rank on Graph Lab (Ferdman *et al*.,2012). This application studies the impact of a Twitter user for graph analysis. Data Caching. Memcached is a high-performance, general-purpose distributed memory caching system. It uses in memory key-value storage mechanism for small chunks of arbitrary data API calls (Ferdman *et al.*,2012).

Traditional CPU Benchmarks

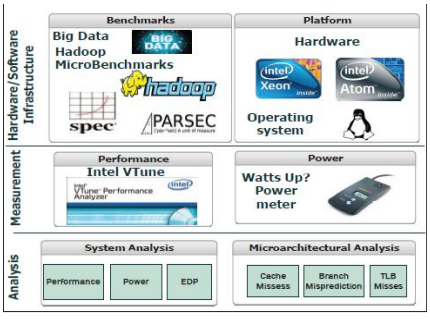
SPEC CPU2006 workloads are industry standard real life applications designed to stress the CPU, memory subsystem and compiler.

PARSEC 2.1 is an open-source parallel benchmark suite for evaluating multi-core and multiprocessor systems.

Measurement Tools and Methodology

The Figure 2 presents a methodology of our approach. We conduct our study on two state-of-the-art servers, Intel Xeon and Intel Atom. Intel Xeon E5 enclosed with two Intel E5-2420 processors that includes six aggressive processor cores per node with three-level of the cache hierarchy. Intel Atom C2758 has 8 processor cores per node and a two-level cache hierarchy. Table 2 summarizes the key architectural parameters of these two servers. The operating system used is Ubuntu 13.10 with Linux kernel 3.11. We analyze the architectural behavior using Intel V Tune, a performance-profiling tool that provides an interface to the processor performance counters. We have used Watts up PRO power meter to measure the power consumption of the servers. Watts up power meter produces the power consumption profile every one second of an application under test.

Fig.2. Methodology



The power reading is for the entire system, including core, cache, main memory, hard disks and on-chip communication buses. We have collected the average

power consumption of the studied applications and subtracted the system idle power to calculate the dynamic power dissipation of the entire energy analysis. The same methodology is used (Blem *et al.,* 2013) as well. We discuss the system-level - performance, power, EDP and micro architecture-level analysis- cache misses, branch mis prediction and TLB misses-for big data applications and Hadoop micro-benchmarks. In addition, Table 1 shows the datasets used to drive the studies applications.

Experimental results and analysis

In this section, we discuss the system-level and microarchitecture-level analysis of little and big cores, when running traditional CPU benchmarks, parallel benchmarks, scale-out, and big data applications. Due to space constraints, we are only reporting the average results for SPEC, PARSEC and Scale Out applications. Moreover, we have conducted the data size sensitivity analysis of Hadoop micro-benchmarks with the dataset of 10MB, 100MB, 1GB and 10GB per node to understand the impact of the size of data per processing node on system-level as well as microarchitecture-level parameters.

Performance Analysis

In this section, we analyze the performance measurements of big data applications in term of IPC and compare it with the traditional benchmarks. The average IPC of big data is 1.65 times lower than the traditional CPU benchmarks on big core and 1.21 times on little core. Therefore, noticeably more performance drop (37%, on average) is observed for big data applications compared to traditional CPU applications when running on big core server compared to little core server. In general, we observe lower IPC in big data applications compared with the traditional benchmarks. Furthermore, little core-based server is experiencing 1.43 times lower IPC in comparison to big core server as Xeon can process up to 4 instructions simultaneously while Atom is limited to 2 instructions per cycle. Figure 3.2 shows the IPC of Hadoop micro-benchmarks for different data sizes. The results are consistent with the results in Figure 3.1 showing lower IPC on little core compared to big core across all data sizes. We also observe that on little core, increasing the data size reduces the IPC since the cache misses increased (mainly Icache miss as will be described in section 5.5.1). Little core, due to its low processing capacity (issue width of 2), cannot hide cache miss penalty as effective as big core. However, on big core while for most cases, increase in data size per node reduces the IPC, there are few exceptions where increasing the data size from 100MB to 1000MB per node increases the IPC. This is mainly due to higher cache locality as a result of larger and more complex cache subsystem in big core, which results in reduction in cache miss rates

Power Consumption and Energy-Efficiency Analysis

In this section, we report the power consumption of big data applications and discuss the energy-efficiency analysis to evaluate the trade-off between power and performance.

Power Characterization

The results show the average dynamic power consumption of the studied applications on big and little core servers. The idle power of the servers is subtracted from the measured (runtime) power. Note that the power results reported are for the entire system, including core, cache, DRAM and on-chip communication buses. Big core consumes on average 35 Watts of dynamic power with the peak of 44 Watts in cluster application. Little core consumes much lower dynamic power as expected, ranging from 0.9 to 6 Watts with an average of 4.8 Watts. Figure 4.2 shows that the power consumption increases as the size of data per node increases in most cases across both big and little architectures. This is more noticeable in little core. While increasing in data size in little core reduces the IPC and therefore core power, it increases cache and off-chip traffic in DRAM and bus subsystem (see LLC MPKI reported in Figure 10). Therefore, for low-end little core where cache, DRAM and off-chip components are dominant power consumer (unlike high performance Xeon core), a clear rise in power consumption is observed as the size of data increases.

Energy-Efficiency Analysis with near Real–Time Processing Constraints

Based on the results of power consumption for both platforms, we have evaluated the trade-off between power and performance by investigating the EDP metric. Furthermore, we have explored the ED2 P and ED3 P to understand the impact of near real-time performance constraints on big data applications and how more constraints on performance affects the choice of most efficient server architecture.

Table.2. Architectural Parameters

Processor	Intel Atom C2758	Intel Xeon E5-2420
Operating Frequency	2.40GHz	1.9GHz
Micro-architecture	Silvermont	Sandy Bridge
L1i Cache	32 KB	32 KB
L1d Cache	24 KB	32 KB
L2 Cache	4*1024 KB	256KB
L3 Cache	-	15MB
PageTable	16972 KB	4260 KB
System Memory	8GB	32GB
TDP	20W	95W

DPS-DPJ Analysis

In this section, we evaluate the data processing capability of big and little core-based server for various data sizes in Hadoop micro-benchmarks. We report the data processed per second (DPS) and the data processed per joule (DPJ) metrics to compare the data processing capability and efficiency of the two server architectures. The peak DPS occurs in terasort and sort at only 1000MB of size, while in other applications occurs in at least an order of magnitude larger data size. The reason is that sort and terasort are I/O intensive applications and the rise in data size exacerbates the I/O cost to an extent that it diminishes the benefit of high processing capacity (Yu *et al*., 2004). The DPS difference between big and little cores-based server is becoming larger for CPU intensive applications such as grep and word count as the size of data increases. However, this is not the case for I/O intensive applications such as sort and DFSIO-read as the I/O cost becomes the dominant performance bottleneck and the processing capacity of the processor; i.e. big vs. little become a less important factor. Overall, for small data size, below 1000MB per node, the two architectures, big and little, have almost similar processing capacity in terms of DPS and it is only for large data sizes that the DPS gap between the two becomes clear. Similar to DPS, for DPJ, in all applications we observe a rise in data processing efficiency on big core as the size of data increases. However, in I/O intensive applications such as terasort, DFSIO-read and write, the rise in DPJ on Xeon is insignificant. For CPU intensive applications including word count and grep there is a significant rise in DPJ on Xeon as the size of data increases. Overall, for I/O intensive applications such as sort, terasort, DFSIO-write and DFSIO-read, Atom-based server is noticeably more efficient than big Xeon. However, in CPU intensive micro-benchmarks, Word Count and Grep, the DPJ gap between big and little core-based server reduces with the increase in data size. It is also interesting to observe that the DPJ of Xeon can exceed Atom in a number of applications and across a number of different data sizes.

Observation

The results illustrate that the choice of big vs little core-based servers in terms of DPS and DPJ analysis are closely decided by the application type and the size of data.

Performance Hotspot and Post-Acceleration CPU code Characterization

As chips are hitting power limits, computing systems are moving away from general-purpose designs to greater specialization. Hardware acceleration through specialization has received renewed interest in recent years, mainly due to the dark silicon challenge. In addition to big, medium and small cores, the integration of domain-specific accelerators, such as GPUs and FPGAs has become extensive. To find out the right server architecture for big data processing, it is important to understand how to deploy an accelerator, such as FPGA, would necessitate adapting the choice of CPU. The post acceleration code characteristics are important to find the right architecture for efficient processing of big data applications. In this section, we analyze the choice of big vs little core-based server for the code that remains for the CPU after acceleration, compared with the choice of big vs little before acceleration. A key research challenge for heterogeneous architecture that integrates CPU and accelerator such as FPGA is workload partitioning and mapping of a given application (which is alternatively referred to as scheduling) to CPU and FPGA for power, performance, and QoS (Neshatpour *et al*., 2015). This is commonly referred as hardware and software partitioning. A common method for HW/SW partitioning is to profile the application to find the performance hotspot region. These regions are candidates for FPGA acceleration, as long as the overhead of communication with CPU is not significant (Nilakantan *et al.,* 2013). To perform hotspot analysis on big data applications, we use Intel V tune to select the common hotspot modules of the applications running on big and little cores. First, we identify and analyze hotspot modules based on their execution time. Overall, Xeon provides a lower execution time, however, if speedup after acceleration is very small then considering the power consumption of Xeon, Atom-based will be a more efficient server to execute the post-accelerated code.

Observation

We studied as to how offloading hotspot map and reduce tasks to an accelerator such as FPGA affects the choice of big vs. little core-based server for processing. The results showed that the choice of big vs. little before and after accelerations is different. While most benchmarks clearly favor little core post acceleration, in several applications post accelerated code showed higher speed up on big core-base server over little core-base server compared to pre-acceleration.

Related Work

Recently, there have been a number of efforts to benchmark and characterize big data and cloud-scale applications, mainly on state-of-the-art high performance server platform. In general, there are two major approaches for benchmarking big data: A system benchmarking and a component benchmarking. A

system benchmark is an end-to-end benchmarking which includes the entire database and application software stack, including data preparation, data aggregation and data analytics. A component benchmark encloses only a portion of the entire end-to-end system (Arora Manish *et al.,* 2012). The most prominent big data benchmarks include Hi Bench, Scale-Out, Big Data Bench, Cloud Cmp and Link Bench. Hi Bench (Huang *et al.,* 2013) is a benchmark suite for Hadoop Map Reduce. Cloud Cmp (Li *et al., 2010*) use a systematic approach to benchmark various components of the cloud to compare cloud providers. Link Bench is a real-world database benchmark for social network applications (Armstrong *et al.,* 2013). The Transaction Procession Performance Council (TPC) has released a number of benchmark suites in recent years, including TPC-C, TPC-E, and TPC-DS for online transaction processing. Big Data Bench (Gao *et al*., 2013) was released very recently and includes online service and offline analytics for web service applications. Big Bench (Ghazal *et al*., 2013) is a new big data benchmark that adopts TPC-DS as its basis and expands it for offline analytics on Xeon high performance server. The Cloud Suite (Ferdman *et* al., 2012; Ghazal *et* al., 2013) benchmark was developed for Scale-Out cloud workloads and mainly includes small data sets, e.g., 4.5 GB for Naïve Bayes. Several prior researches have characterized traditional CPU and parallel applications such as SPEC2006, PARSEC, and NAS on high performance server-class processors (Prakash *et al*., 2008). It is also important to compare the characteristics of big data application with these traditional benchmark suites. We have included the SPEC CINT2006, SPEC CFP2006 and PARSEC 2.1 benchmarks for the comparison with Big Data Workloads. This work is different from all above benchmarking and characterization work as it perform a comprehensive system level (power, performance, EDXP, DPS and DPJ) and micro architecture-level(cache miss, TLB miss, branch misprediction) analysis of various big data applications and big data micro-benchmarks on two substantially different platforms one with high performance big core and another with low power little core to understand which of these two architectures is the choice for efficient big data processing. There have been also a number of researches into application-specific (Yu *et al., 2004*) and domain-specific accelerators. Using tightly integrated FPGA (Xi Luo *et al*., 2013) with CPU and GPU with CPU (Baru *et al*., 2013) to accelerate big data processing have been proposed in recent work. While deploying programmable accelerator is a new and hot research topic, there has been little attention paid to how CPU designs should be adapted to this change. To the best of our knowledge, the only work on this topic has been reported by (Arora Manish *et al.,* 2012), which studied the role of the CPU for a CPU+GPU architecture. They concluded that, in a CPU+GPU architecture, the CPU is running a code that is significantly different from a CPU-only code. They found that the post-GPU code has a lower ILP, higher branch miss prediction rate, and larger number of load and stores, and benefits less from multiple cores, as there is less TLP after GPU offloading. In this paper, we demonstrated as to how deploying accelerator such as FPGA for big data affects the choice of big vs. little core for efficient processing.

CONCLUSION

In this paper, we present a comprehensive system and micro architecture-level analysis of big data applications on two distinct server platforms; the conventional approach, a high performance big Xeon core; and the new trajectory in server design, a low power little Atom core, which advocates the use of a low-power core to address the power challenge. The characterization results show significantly larger performance drop (37%, on average) for big data applications compared to traditional CPU applications when running on big core server compared to little core server. Big core-based server provides a high performance, compared to little core, however, it is not as power efficient. Little core-based server is more efficient in terms of EDP for big data processing with small data sizes. However, as the size of data increases and with performance constraints, big core becomes an efficient choice. The analysis of data processing capability and efficiency of big data applications illustrates that the choice of big core vs. little core-based server in terms of data processing per second and data processing per joule is closely decided by the application type size of data, and computational and I/O intensity of the application. In addition, we performed the post-acceleration CPU code analysis to find out the most efficient server architecture to process the remaining code of big data applications. The results show that there is a difference between the choice of big vs. little core-based server before and after accelerations. While most benchmarks clearly favour little core post acceleration, several applications show higher speed up on big core over little core post acceleration compared to pre-acceleration. To provide insight on whether current server design based on big and little core architectures requires improvement in their micro architecture parameters for efficient big data processing, we perform a comprehensive micro architecture characterization and compare the results with traditional Spec, PARSEC, and scale out applications. Our analysis indicates that the size of data has a non-trivial impact on several micro

architecture parameters. Moreover, results show that while a small 4x1MB two-level data cache is sufficient for big data applications on little core the instruction cache hierarchy pipeline needs improvement. Also little core needs architectural improvement in instruction TLB miss overhead management as well as branch predictor. Furthermore, the analysis shows that the deep software stack of big data applications, along with the excessive non-loop branches, affects L1 cache hit rate and branch predictor accuracy in both big and little cores. Moreover, big data applications require efficient instruction pre fetchers to predict complex patterns and sophistication branch predictor to handle the unknown control flow.

REFERENCES

Andersen david, G., Jason Franklin and Michael Kaminsky, 2009. FAWN: A Fast Array of Wimpy Nodes. In : *Proc SOSP '09 ACM SIGOPS 22nd symposium on Operating systems principles Sky, Montana*, USA — P. 1–14. https://doi.org/10.1145/1629575.1629577

Armstrong, Timothy, G., Nagavamsi Ponnekanti, Dhruba Borthakur and Mark Callaghan, 2013. Link bench: A database benchmark based on the Facebook social graph. In : *Proc.ACM SIGMOD International Conference on Management of Data*, P. 1185-1196. https://doi.org/10.1145/2463676.2465296 PMid:23532973

Arora Manish, Siddhartha Nath, Subhra Mazumdar, Scott B., Baden , Dean, M. and Tullsen, 2012. Redefining the Role of the CPU in the Era of CPU-GPU Integration. In : Proc. *IEEE Micro,* 32 (6) : 4-16 https://doi.org/10.1109/MM.2012.57

Arora, Kiran Chandramohan, Nagaraju Pothineni and Anshul Kumar, 2010. Instruction selection in asip synthesis using functional matching. In : *Proc. 23rd International Conference on VLSI Design*, P. 3-7. https://doi.org/10.1109/VLSI.Design.2010.68

Baru, Milind Bhandarkar, Raghunath Nambiar, Meikel Poess and Tilmann Rabl, 2013. *Setting the Direction for Big Data benchmark Standards.* 7755 : 197-208. https://doi.org/10.1007/978-3-642-36727-4_14

Blem, Jaikrishnan Menon and Karthikeyan Sankaralingam, 2013. Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures. In : *Proc. IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, P.1-12. https://doi.org/10.1109/HPCA.2013.6522302

Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos and Babak Falsafi, 2012. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In : *Proc. 17th Conference on Architectural Support for Programming Languages and Operating Systems*, P.37-48. https://doi.org/10.1145/2150976.2150982

Gao, Yuqing Zhu, Zhen Jia, Chunjie Luo, Lei Wang, Zhiguo Li, Jianfeng Zhan, Yong Qi and Bizhu Qiu, 2013. Big Data Bench: a Big Data Benchmark Suite from Web Search Engines . In : Proc. *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, P.488-499.

Ghazal, Tilmnn Rabl, Minging Hu, Francois Raab, Meikel poess, Alain crolotte and Hans-Arno Jacobsan, 2013. Big bench: Towards an industry standard benchmark for big data analytics. In : *Proc. ACM SIGMOD International Conference on Management of Data*, P. 1197-1208.

Gutierrez, Michael Cieslak, Bharan Giridhar, Ronald G. Dreslinski, Luis Ceze and Trevor Mudge, 2014. Integrated 3D-stacked server designs for increasing physical density of key-value stores. In : *Proc. 19th international conference on Architectural support for programming languages and operating systems*, P.485-498 https://doi.org/10.1145/2541940.2541951

Hardavellas, Michael Ferdman, Babak Falsafi and Anastasia Ailamaki, July 2011. Toward dark silicon in servers. In : *Proc. IEEE Micro*, 31 : 6-15. https://doi.org/10.1109/MM.2011.77

Homayoun, Vasileios Kontorinis, Amirali Shayan, Ta-Wei Lin and Dean M. Tullsen, 2012. Dynamically heterogeneous cores through 3D resource pooling. In : *Proc. IEEE International Symposium on High-Performance Comp Architecture.* https://doi.org/10.1109/HPCA.2012.6169037

Huang, Jie Huang, Yan Liu and Jinquan Dai, 2010. The Hi Bench benchmark suite: Characterization of the Map Reduce-based data analysis. In : *Proc.IEEE 26th International Conference on Data.* https://doi.org/10.1109/ICDEW.2010.5452747

Khavari Tavana, Amey Kulkarni, Abbas Rahimi, Tinoosh Mohseni and Houman Homayoun, 2014. Energy-efficient mapping of biomedical applications on domain-specific accelerator under process variation. In : *Proc. IEEE/ACM International Symposium on Low Power Electronics and Design* (ISLPED). https://doi.org/10.1145/2627369.2627654

Kontorinis , Mohammad K., Tavana Mohammad, H., Hajkazemi Dean, M., Tullsen and Houman Homayoun, 2014. Enabling Dynamic Heterogeneity Through Coreon-Core Stacking. In : *Proc. 51st ACM/EDAC/IEEE Design Automation Conference (DAC).* https://doi.org/10.1109/DAC.2014.6881509

Li, Xiaowei Yang, Srikanth Kandula and Ming Zhang, 2010. CloudCmp: comparing public cloud providers. In : *Proc. 10th ACM SIGCOMM conference on Internet measurement*, P.1-14. https://doi.org/10.1145/1879141.1879143

Li, Zhigang Sun, Wu Jigang and Xicheng Lu, 2009. Fast enumeration of maximal valid subgraphs for custominstruction identification. In : *Proc. international conference on Compilers, architecture, and synthesis for embedded systems*, P. 29-36 https://doi.org/10.1145/1629395.1629402

Marnix Arnold and Henk Corporaal, April 2001. Designing domain-specific processors. In : *Proc. Ninth*

International Symposium on Hardware/Software Codesign, P.61-66.
https://doi.org/10.1145/371636.371677

Neshatpour, K., Malik M., Ghodrat, M.A., Sasan, A. and Homayoun, H., 2015. Energy-Efficient Acceleration of Big Data Analytics Applications Using FPGAs. In : *Proc. IEEE International Conference on Big Data (Big Data),* P. 115-123
https://doi.org/10.1109/BigData.2015.7363748

Nilakantan, S. and Mark Hempstead, 2013. Platform-independent analysis of function-level communication in workloads. In : *Proc. IEEE International Symposium on Workload Characterization* (IISWC).
https://doi.org/10.1109/IISWC.2013.6704685

Prakash,T.K., Lu and Peng, 2008. Performance Characterization of SPEC CPU 2006 Benchmarks on Intel Core 2 Duo Processor. In : *Proc. 17th annual international conference on Supercomputing* (ICS).

Reddi, V.J., Benjamin, C. Lee, Trishul Chilimb and Kushagra Vaid. 2010. Web search using mobile cores: quantifying and mitigating the price of efficiency, In : *Proc. 37th annual international symposium on Computer architecture* P. 314-325.

Vasileios Kontorinis, Liuyi Eric Zhang, Baris Aksanli, Jack Sampson, Houman Homayoun, Eddie Pettis, Dean M. Tullsen and Tajana Simunic Rosing, 2012. Managing distributed UPS energy for effective power capping in data centers. In : *Proc. 39th Annual International Symposium on Computer Architecture (ISCA).*
https://doi.org/10.1109/ISCA.2012.6237042

Willke, T.L., Nilesh Jain and Haijie Gu, 2012. GraphBuilder–A Scalable Graph Construction Library for Apache™ Hadoop™.

Xi Luo, W alid A., Najjar and Vagelis Hristidis, 2013. Efficient near-duplicate document detection using FPGAs. In : *Proc. IEEE International Conference on Big Data.*
https://doi.org/10.1109/BigData.2013.6691698

YU, P. and Tulika Mitra, 2004. Disjoint pattern enumeration for custom instructions identification. In : *Proc. International Conference on Field Programmable Logic and Applications*, P.273–278.

YU, P. and Tulika Mitra, 2004. Scalable custom instructions identification for instruction set extensible processors. In : *Proc. international conference on Compilers, architecture, and synthesis for embedded systems*, P. 69-78.
https://doi.org/10.1145/1023833.1023844