International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 4 | Issue 2 | ISSN : 2456-3307

Reducing the Energy Consumption Energy-Efficient Query Processing Node In Web Search Engines

K.Hari Krishna [*1], Kosuru Anusha Rani [2]
[1*]harikanagala@gmail.com,anuranikosuru501@gmail.com

ABSTRACT

Web search engines are made by thousands out of question handling hubs, i.e., servers devoted to process client inquiries. Such numerous servers expend a lot of energy, for the most part responsible to their CPUs, however they are important to guarantee low latencies, since clients expect sub-second reaction times (e.g., 500 ms). In any case, clients can scarcely see reaction times that are quicker than their desires. Henceforth, we propose the Predictive Energy Saving Online Scheduling Algorithm (PESOS) to choose the most proper CPU recurrence to process an inquiry on a for every center premise. PESOS goes for process questions by their due dates, and use abnormal state scheduling data to decrease the CPU energy utilization of a question handling hub. PESOS constructs its choice in light of inquiry effectiveness indicators, evaluating the preparing volume and handling time of a question. We tentatively assess PESOS upon the TREC ClueWeb09B gathering and the MSN2006 inquiry log. Results demonstrate that PESOS can decrease the CPU energy utilization of a question preparing hub up to 48% contrasted with a framework running at most extreme CPU center recurrence. PESOS beats moreover the best in class contender with a 20% energy saving, while the contender requires a fine parameter tuning and it might brings about in wild inertness infringement.

Keywords: Energy consumption, CPU Dynamic Voltage and Frequency Scaling, Web search engines.

I. INTRODUCTION

Web search engines continuously crawl and index an im-mense number of Web pages to return fresh and relevant results to the users' queries. Users' queries are processed by query processing nodes, i.e., physical servers dedicated to this task. Web search engines are typically composed by thousands of these nodes, hosted in large datacenters which also include infrastructures for telecommunication, thermal cooling, fire suppression, power supply, etc [1]. This complex infrastructure is necessary to have low tail latencies (e.g., 95-th percentile) to guarantee that most users will receive results in sub-second times (e.g., 500 ms), in line with their expec-tations [2]. At the same time, such many servers consume a significant amount of energy, hindering the profitability of the search engines and raising environmental concerns. In fact, datacenters can consume tens of megawatts of electric power [1] and the related expenditure can exceed the original investment cost for a datacenter [3]. Because of their energy consumption, datacenters are responsible for the 14% of the ICT sector carbon dioxide emissions [4], which are the main cause of global warming. For this reason, governments are promoting codes of conduct and best practices [5], [6] to reduce the environmental impact of datacenters.

Since energy consumption has an important role on the profitability and environmental impact of Web search engines, improving their energy efficiency is an important aspect. Noticeably, users can hardly notice response times that are faster than their expectations [2]. Therefore, to reduce energy consumption, Web search engines should answer queries no faster than user expectations. In this work, we focus onreducing the energy consumption of

servers' CPUs, which are the most energy consuming components in search systems [1]. To this end, Dynamic Frequency and Voltage Scaling (DVFS) technologies [7] can be exploited. DVFS technologies allow to vary the frequency and voltage of the CPU cores of a server, trading off performance (i.e., longer response times) for lower energy consumptions. Several power management policies leverage DVFS technologies to scale the frequency of CPU cores accordingly to their utilization [8], [9]. However, core utilization-based policies have no mean to impose a required tail latency on a query processing node. As a result, the query processing node can consume more energy than necessary in providing query results faster than required, with no benefit for the users.

In this work we propose the Predictive Energy Saving On-line Scheduling algorithm (PESOS), which considers the tail latency requirement of queries as an explicit parameter. Via the DVFS technology, PESOS selects the most appropriate CPU frequency to process a query on a per-core basis, so that the CPU energy consumption is reduced while respecting a required tail latency. The algorithm bases its decision on query efficiency predictors rather than core utilization. Queryefficiency predictors are techniques to estimate the processing time of a query before its processing. They have been proposed to improve the performance of a search engine, for instance to take decision about query scheduling [10] or query processing parallelization [11], [12]. However, to the best of our knowl-edge, query efficiency predictor have not been considered for reducing the energy consumption of query processing nodes.

We build upon the approach described in [10] and propose two novel query efficiency predictor techniques: one to esti-mate the number of postings that must be scored to process a query, and one to estimate the response time of a query under a particular core frequency given the number of postings to score. PESOS exploits these two predictors to determine which is the lowest possible core frequency that can be used to process a query, so that the CPU energy consumption is reduced while satisfying the required tail latency. As predictors can be inaccurate, in this work we also propose and investigate a way to compensate prediction errors using the root mean square error of the predictors.
+
We experimentally evaluate PESOS upon the TREC ClueWeb09 corpus and the query stream from the MSN2006 query log. We compare the performance of our approach with those of three baselines: perf [8], which always uses the maximum CPU core frequency, power [8], which throttles CPU core frequencies according to the core utilizations, and cons [13], which performs frequency throttling according tothe query server utilization. PESOS, with predictors correc-tion, is able to meet the tail latency requirements while re-ducing the CPU energy consumption from ~24% up to ~44% with respect to perf and up to ~20% with respect to cons, which however incurs in uncontrollable latency violations. Moreover, the experiments show that energy consumption can be further reduced by PESOS when prediction correction is not used, but with higher tail latencies.

The rest of the paper is structured as follows: Section 2 provides background information about the energy consump-tion of Web search engine datacenters, the query processing activity, and the query efficiency predictors.

II. BACKGROUND

In this section we will discuss the energy-related issues in-curred by Web search engines (Sec. 2.1). Then, we will explain how query processing works and some techniques to reduce query response times (Sec. 2.2). Finally, we will discuss about query efficiency predictors, which we exploit to reduce theenergy consumption of a Web search engine while maintaining low tail latencies (Sec. 2.3).

2.1 Web search engine and energy consumption

In the past, a large part of a datacenter energy consumption was accounted to inefficiencies in its cooling and power supply systems. However, Barroso et al. [1] report that modern datacenters have largely reduced the energy wastage of those infrastructures, leaving little room for further improvement. On the contrary, opportunities exist to reduce the energy consumption of the servers hosted in a datacenter. In par-ticular, our work focuses on the CPU power management of query processing nodes, since the CPUs dominate the energy consumption of physical servers dedicated to search tasks. In fact, CPUs can use up to 66% of the whole energy consumed by a query processing node at peak utilization [1].

Modern CPUs usually expose two energy saving mecha-nism, namely C-states and P-states. C-states represent CPU cores idle states and they are typically anaged by the operating system [14]. C0 is the operative state in which a CPU core can perform computing tasks. When idle periodsoccur, i.e., when there are no computing tasks to perform, the core can enter one of the other deeper C-states and become inoperative. However, Web search engines process a large and continuous stream of queries. As a result, query processing nodes are rarely inactive and experience particularly short idle times. Consequently, there are little opportunities to exploit deep C-states, reducing the energy savings provided by the C-states in a Web search engine system [15], [16].

When a CPU core is in the active C0 state, it can op-erate at different frequencies (e.g., 800 MHz, 1.6 GHz, 2.1 GHz, . . .). This is possible thanks to the Dynamic Frequency and Voltage Scaling (DVFS) technology [7] which permits to adjust the frequency and voltage of a core to vary its perfor-mance and power consumption. In fact, higher core frequen-cies mean faster computations but higher power consumption. Vice versa, lower frequencies lead to slower computations and reduced power consumption. The various configurations of voltage and frequency available to the CPU cores are mapped to different P-states, and are managed by the operating system. For instance, the intelpstate driver controls the P-states on Linux systems1 and can operate accordingly to two different policies, namely perf and power. The perf policy simply uses the highest frequency to process computing tasks. Instead, power selects the frequency for a core accord-ing to its utilization. When a core is highly utilized, power selects an high frequency. Conversely, it will select a lower frequency when the core is lowly utilized. However, Lo et. al [15] argue that core utilization is a poor choice for managing the cores frequencies of query processing nodes. In fact, the authors report an increase of query response times when core utilization-based policies are used in a Web search engine. For such reason, Catena et al.

propose to control the frequency of CPU cores based on the utilization of the query processing node rather than on the utilization of the cores. The utilization of a node is computed as the ratio between the query arrival rate and service rate. Then, they propose the cons policy which throttles the frequency of the CPU coreswhen the utilization of the node is above or below certain thresholds (e.g., 80% and 20%, respectively). The frequency is selected so to produce a desirable utilization level (e.g., 70%). Similarly, in our work we control the CPU cores frequencies of a query processing node using information related to the query processing activity rather than to the CPU cores utilization (see Sec. 4). To this end, we build our approach on top of the acpicpufreqdriver [9]. This driver allows applications to directly manage the CPU cores frequency, instead of relying on the operative systems.

2.2 Query processing and dynamic pruning

Web search engines continuously crawl a large amount of Web pages. This collection of documents

is then indexed to produce an inverted index [17]. The inverted index is a data structure that maps each term in the document collection to a posting list, i.e., a list of postings which indicates the occurrence of a term in a document. A posting contains at least the identifier (i.e., a natural number) of the document where the term appears and its term frequency, i.e., the number of occurrences of the term in that particular document. The inverted index is usually compressed [18] and kept in main memory to increase the performance of the search engine [19].

When a query is submitted to a Web search engine, it is dispatched to a query processing node. This retrieves a ranked list of documents that are relevant for the query, i.e., the top K documents relevant to a user query, sorted in decreasingorder of relevance score (e.g., by using the popular BM25 weighting model [20]). To generate the top K results list, the processing node exhaustively traverses all the posting lists relative to the query terms. This is computationally expensive, since the inverted index can easily measure tens of gigabytes, so dynamic pruning techniques are adopted [21], [22]. Such techniques avoid to evaluate irrelevant documents, skipping over portions of the posting lists. This reduces the response time as the systems avoid to access and decompress portion of the inverted index. At the same time, these dynamic pruning techniques are safe-up-to-K, i.e., they produce the same top K results list returned by an exhaustive traversal of the posting lists. For such reasons, in this work we apply dynamic pruning strategies to the processing of queries.

2.3 Query efficiency predictors

Query efficiency predictors (QEPs) are techniques that es-timate the execution time of a query before it is actually processed. Knowing in advance the execution time of queries permits to improve the performance of a search engine. Most QEPs exploit the characteristics of the query and the inverted index to pre-compute features to be exploited to estimate the query processing times. For instance, Macdonald et al. [10] propose to use term-based features (e.g., the inverse docu-ment frequency of the term, its maximum relevance score among others) to predict the execution time of a query. They exploit their QEPs to implement on-line algorithms to schedule queries across processing node, in order to reduce the average query waiting and completion times. The works [11], [12], instead, address the problem to whether parallelize or not the processing of a query. In fact, parallel processing can reduce the execution time of long-running queries but provides limited benefits when dealing with short-running ones. Both the works propose QEPs to detect long-running queries. The processing of the query is parallelized only if their QEPs detect the query as a long-running one. Rather then combining term-based features, Wu et al. [23] propose to analytically model the query processing stages and to use such model to predict the execution time of queries.

In our work, we modify the QEPs described in [10] to develop our algorithm for reducing the energy consumption of a processing node while maintaining low tail latencies.

III. EXISTING SYSTEM

In the following, we introduce the operative scenario of a query processing node (Sec. 3.1), we formalize the general minimum-energy scheduling problem and we shortly present the state-of-the-art algorithm to solve it offline (Sec. 3.2), and we discuss the issues of this offline algorithm in our scenario (Sec. 3.3).

3.1 Operative scenario

A query processing node is a physical server composed by several multi-core processors/CPUs with a shared memory which holds the inverted index. The inverted index can be partitioned into shards and distributed across multiple query processing nodes. In this work, we focus on reducing the CPU energy consumption of single query processing nodes, independently of the adopted

partition strategy. In the following, we assume that each query processing node holds an identical replica of the inverted index [24].

A query server process is executed on top of each of the CPU core of the processing node (see Figure 1). All query servers access a shared inverted index held in main memory to process queries. Each query server manages a queue, where the incoming queries are stored. The first query in the queue is processed as soon as the corresponding CPU core is idle. The queued queries are processed following the first-come firstserved policy. The number of queries in a query server's queue represents the server load. Queries arrive to the processing node as a stream S = {q1, . . .

1. qn}. When a query reaches the processing node it is dispatched to a query server by a query router. The query router dispatches an incoming query to the least loaded query server, i.e., to the server with the smallest number of enqueued queries. Alternatively, the query processing node could have a single query queue and dispatch queries from the queue to idle query servers. In this work, we use a queue for each query servers since a single queue will not permit to take local decisions about the CPU core frequency to use for the relative query server. A similar queue-per-core architecture is assumed in [25], to schedule jobs across CPU cores to minimize the CPU energy consumption, and in [10]

IV. PROPOSED WORK

We propose the Predictive Energy Saving Online Scheduling algorithm (PESOS), which considers the tail latency requirement of queries as an express parameter. Via the Dynamic Frequency and Voltage Scaling (DVFS) era, PESOS selects the most suitable CPU frequency to method a query on a in keeping with-middle foundation, in order that the CPU power consumption is reduced even as respecting required tail latency. The algorithm bases its choice on query efficiency predictors in place of center usage. Query performance predictors are strategies to estimate the processing time of a query before its processing. In this paper we attention on lowering the CPU strength consumption of single query processing nodes, independently of the followed partition method. A query processing node is a physical server composed by numerous multi-core processors/CPUs with a shared memory. A query server system is achieved on pinnacle of each of the CPU middle of the processing node. All query servers get admission to a shared inverted index held in major reminiscence to method queries.

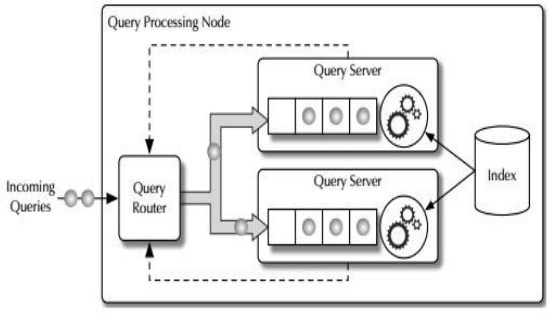

Figure 1. Structure of the Query Processing Node.

Each query server manages a queue, wherein the incoming queries are stored. The first query in the queue is processed as soon because the corresponding CPU center is idle. The queued queries are processed following the primary-come first served coverage. The number of queries in a query server's queue represents the server load. Queries arrive to the processing node as a circulation S = q_1. . . q_n. When a query reaches the processing node it's far dispatched to a query server by a query router. The query router dispatches an incoming query to the least loaded query server, i.e., to the server with the smallest variety of enqueued queries as shown in Fig.1. Alternatively, the query processing node could have a single query queue and dispatch queries from the queue to idle query servers. In this work, we use a queue for each query servers considering that a single queue will no longer permit to take nearby selections approximately the CPU middle frequency to apply for the relative query server.

C. Root Mean Square Error (RMSE)

The Root Mean Square Error (RMSE) is an often used measure of the differentiation among values predicted by a model & the values actually captured from the environment that is being modeled. These individual differentiations are also referred residuals, & the RMSE provides to aggregate them into a single measure of predictive power. The RMSE of a representation prediction with value to the predictable variable X_{model} is distincted as the square root of the mean squared error:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (X_{obs,i} - X_{model,i})^2}{n}} \quad (1)$$

where X_{obs} is practical values & X_{model} is modeled values at time/place i.In this proposed Predictive Energy Saving Online Scheduling algorithm, we can record this error for query efficiency predictors.

D. Applying Query Efficiency Prediction to Query Scheduling

While in popular the primary retrieval performance degree is the common time required to manner the queries (common response time), when a move of queries is acquired by using a search engine, it might not be possible to start processing a brand new question as soon because it arrives. Instead, when the system is busy processing a query, next queries are queued. Therefore, the real time delay experienced with the aid of a user whilst waiting for seek consequences (finishing touch time) is given through the execution time (response time) of the question, plus the time the question spent ready to be processed (ready time). Classically, queued queries were processed in a FIFO manner. However these only consequences in minimizing queuing time if every query has an equal response time. Instead, we recommend that queues of queries can be scheduled to execute out of arrival order, by way of deploying unique scheduling algorithms. In this manner, for example, short queries may be scheduled among longer queries, to lessen the mean time put off skilled through the user populace of the quest engine.

V. CONCLUSIONS

In this paper, we proposed the Predictive Energy Saving Online Scheduling (PESOS) algorithm. With regards to Web search engines, PESOS intends to decrease the CPU en-ergy utilization of an inquiry handling hub while forcing a required tail inactivity on the question reaction times. For each question, PESOS chooses the most minimal conceivable CPU center recurrence with the end goal that the energy utilization is diminished and the due dates are regarded. PESOS chooses the correct CPU center recurrence abusing two different sorts of question efficiency indicators (QEPs). The main QEP gauges the preparing volume of questions. The second QEP gauges the question handling times under different center frequencies, given the quantity of postings to score. Since QEPs can be wrong, amid their preparation we recorded the root mean square blunder (RMSE) of the forecasts. In this work, we proposed to total the RMSE to the genuine forecasts to repay expectation blunders. We at that point characterized two conceivable setups for PESOS: time traditionalist, where forecast rectification is authorized, and energy preservationist, where QEPs are left unmodified

VI. REFERENCES

[1]. L. A. Barroso, J. Clidaras, and U. H¨olzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 2nd ed. Morgan & Claypool Publishers, 2013.

[2]. I. Arapakis, X. Bai, and B. B. Cambazoglu, "Impact of response latency on user behavior in web search," in Proc. SIGIR, 2014, pp. 103–112.

[3]. The Climate Group for the Global e-Sustainability Initiative, "Smart 2020: Enabling the low carbon economy in the information age," 2008.

[4]. U.S. Department of Energy, "Best Practices Guide for Energy-Efficient Data Center Design." [Online].

[5]. D. C. Snowdon, S. Ruocco, and G. Heiser, "Power Management and Dynamic Voltage Scaling: Myths and Facts," in Proc. of Workshop on Power Aware Real-time Computing, 2005.

[6]. D. Brodowski, "CPU frequency and voltage scaling code in the Linux kernel."

[7]. C. Macdonald, N. Tonellotto, and I. Ounis, "Learning to predict response times for online query scheduling," in Proc. SIGIR, 2012, pp. 621–630.

[8]. M. Jeon, S. Kim, S.-w. Hwang, Y. He, S. Elnikety, A. L. Cox, and S. Rixner, "Predictive parallelization: Taming tail latencies in web search," in Proc. SIGIR, 2014, pp. 253–262.

[9]. S. Kim, Y. He, S.-w. Hwang, S. Elnikety, and S. Choi, "Delayeddynamic-selective (dds) prediction for reducing extreme tail latency in web search," in Proc. WSDM, 2015, pp. 7–16

[10]. M. Catena, C. Macdonald, and N. Tonellotto, "Load-sensitive cpu power management for web search engines," in Proc. SIGIR, 2015, pp. 751–754

AUTHOR DETAILAS

K.HARI KRISHNA his working various Engennring colleges Having 10years of experience in the teaching .he is working as an assistant professor in VIGNAN'S LARA INSTITUTE OF TECHNOLOGY &SCIENCE Vadlamudi, Guntur Dist. His interested in research areas are Datamining, C,Java Expert, Hadoop, Dbms, Cobol

Currently pursuing MCA in MCA Department, Vignan's Lara Institute Of Technology & Science, Vadlamudi, Guntur, Andhra Pradesh, India. she received his Bachelor of science from KRISHAN university

KOSURU ANUSHA RANI she