ENERGY-EFFICIENT DESIGN OF REAL-TIME STREAM MINING SYSTEMS

Shaolei Ren[‡] *Cuiling Lan*[⋆] *Mihaela van der Schaar*[†]

[‡]Florida International University [⋆]Xidian University, China [†]University of California, Los Angeles

ABSTRACT

In this paper, we propose an efficient solution for supporting real-time stream mining applications on heterogeneous systems operating at various processing speeds. Unlike the existing solutions that (1) rely on accurate knowledge or prediction of the service demand of each *individual* service request and (2) only consider a *single* type of delay constraint (e.g., typically, average or maximum delay), we propose an optimal algorithm, MinEnergy-MD, which determines the processing speeds for all classifiers based on the probability distribution of the service demand to minimize the average energy consumption while simultaneously satisfying multiple delay constraints. We conduct an extensive study to quantify the performance of MinEnergy-MD.

1. INTRODUCTION

The past few years has witnessed the emergence of a plethora of real-time stream mining systems, such as video surveillance, online patient monitoring, disaster information management, video search etc. These applications require processing in real-time a large amount of data, and, they must be optimized for energy efficiency.

Recent trends in hardware and system architectures have highlighted the usage of heterogeneous systems for energy-efficient computing (e.g., [6]), and we have witnessed the emergence of deploying large-scale distributed stream mining system over heterogeneous processing nodes [1]. Our research mainly answers the following question: how to optimally choose the processing speeds for classifiers? Nevertheless, we face the following two challenges that cannot be addressed by the existing literature:

Unknown service demand: The service demand of each individual service request (i.e., how many classifiers the stream will go through) is unknown *a priori* in stream mining systems, whereas the existing solutions rely on accurate knowledge or prediction of the service demand (e.g., [5]).

Multiple delay constraints: In the existing literature (e.g., [1][3]), two commonly-used delay constraints are the maximum and average delays, which are inadequate to characterize the actual delay requirement of real-time stream mining systems:[1] bounding the maximum delay may lead to an intolerable average delay, whereas only considering the average delay may cause delay outliers.

For the first challenge, we propose an algorithm, called MinEnergy-MD, which only requires the probability distribution of resource demand. To capture the delay requirement, we use multiple L_p norms as the performance metric which, encapsulating the average and maximum processing delay as special cases, provide a unified view towards the sensitivity with respect to the processing delay [11]. MinEnergy-MD minimizes the expected energy consumption subject to various L_p norm delay constraints. Our solution makes use of heterogeneous hardware systems that can be harnessed for energy-efficient and delay-sensitive stream mining. To evaluate MinEnergy-MD, we build a real-time video stream mining system consisting of binary classifiers which extract features from video sequences and classify the video sequences using the support vector machine (SVM) algorithm. The results show that given the same delay requirement, MinEnergy-MD reduces the average energy consumption by more than 10% compared to the best homogeneous system as well as the best-known existing algorithm that only takes into account the maximum delay constraint.

2. MODEL

In the basic model as illustrated in Fig. 1, we focus on a classifier *chain* while noting that classification tree models can be dealt with similarly (as shown in Section 3.2.3).

Classifier. Conceptually, any stream mining system can be constructed using a set of binary classifiers that are deployed on (possibly heterogeneous) processing nodes/servers [1]. An input data is filtered sequentially along the cascaded classifiers and data labeled as "Positive" is forwarded to the next classifier if any, while data labeled as "Negative" is dropped.[2] We focus on a classifier chain which consists of N binary classifiers indexed by $1, 2, \cdots, N$, respectively. The resource demand of classifier i is quantified in terms of

This work is supported in part by NSF under Grant #1016081.

[1]Many applications define multiple delay constraints (e.g., both maximum and average delays) as service level agreements. Our focus is to satisfy multiple delay constraints *simultaneously* for one system, rather than supporting different delay performances to multiple systems (e.g., differentiated QoS work [13]).

[2]Compared to running multiple classifiers in parallel, sequential classification is more efficient as classifiers are activated only when necessary [1].

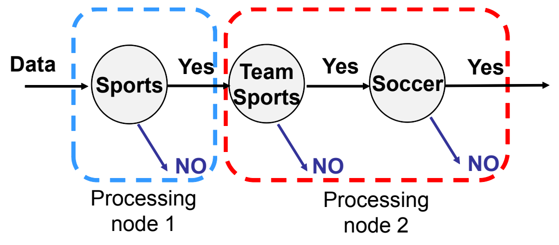


Fig. 1. Illustration of chain of classifiers.

the processing time w_i that is required if classifier i is deployed over a reference server. We denote by $f_i > 0$ the probability that the input data is processed by i classifiers, for $i = 1, 2, \cdots, N$. Equivalently, $f_1, f_2, \cdots, f_N$ can be interpreted as the probability mass distribution of service demand measured in terms of the number of classifiers. We denote the cumulative distribution function (CDF) by $F_i = \sum_{j=1}^{i} f_i$ (with $F_0 = 0$ for completeness of definition). The actual number of classifiers that a particular input data will go through is unknown in advance, whereas the statistical information (i.e., the values of $f_1, f_2, \cdots, f_N$) is readily available by, e.g., online/offline profiling (e.g., [1][3]) or based on the a priori selectivity of input streams [1].

Server. We use the general term "server i" to represent the processing node, over which classifier i is deployed and executed. In practice, a server may refer to: a physical server in data centers; a single core on a heterogeneous multicore processor; a virtual machine in a virtualized system [10]. We use the term "processing speed" as an indicator of the effective server performance. We denote the processing speed of server i by $x_i \in [0, s_{\max}]$ and the power consumption per unit work by $z_i = z(x_i)$, where $s_{\max}$ is maximum server performance due to hardware constraints. Thus, the processing time of classifier i is given by w_i/x_i. The energy function of server i for executing classifier i is $e_i = e(x_i) = z(x_i)/x_i$.

Assumption: The energy function $e(x)$ is a finite, continuously differentiable, and strictly convex function in terms of $x \in [0, s_{\max}]$. ■

The assumption been widely considered and validated extensively by both analytical models and practical measurement studies [3][7]. We use the vectorial expression $\mathbf{x} = (x_1, x_2, \cdots, x_N)$ wherever applicable, and express the average energy consumption of the stream mining system as

$$\bar{e}(\mathbf{x}) = \sum_{n=1}^{N} \left[\sum_{j=1}^{n} w_j \frac{z(x_j)}{x_j} \right] \cdot f_n = \sum_{n=1}^{N} [1 - F_{n-1}]\, w_n e(x_n).$$

The term "$\sum_{j=1}^{n} w_j \cdot \frac{z(x_j)}{x_j}$" represents the energy consumption of an input data that is processed by the first n classifiers (with a probability of f_n, and hence with simple mathematical manipulation we obtain the average energy consumption. Note that we only take into account the energy that is actually consumed by active servers and do not consider the idle server energy consumption, which may be reduced using various techniques (such as "power gating" [14]) beyond the scope of our study.

Job. Each service request carries an input data to be classified and is referred to as a *job*. As in [1], we concentrate on the processing delay: there is no resource contention or queueing delay. This model has been adopted by various studies such as such as [3] and captures a lightly-loaded system and/or a periodic system (e.g., video stream mining system). To characterize the delay performance, we introduce the L_p norm [11] expressed as

$$D(p) = \left[\sum_{n=1}^{N} (t_n)^p f_n \right]^{\frac{1}{p}} = \left\{ \sum_{n=1}^{N} \left[\sum_{j=1}^{n} \frac{w_j}{x_j} \right]^p f_n \right\}^{\frac{1}{p}}, \quad (1)$$

where $p \geq 1$ and $t_n = \sum_{j=1}^{n} \frac{w_j}{x_j}$ is the delay of a job that goes through the first n classifiers. Note that (1) reduces to the average delay when $p = 1$, and to the maximum delay when p approaches ∞. Furthermore, L_p norm is more closely related to several practical delay constraints. For example, the delay variance, which determines the *predictability* of a scheduling algorithm [12], highly dependent on the L_2 norm through the simple expression $\text{var}(t) = D^2(2) - D^2(1)$, where $D(2)$ and $D(1)$ are obtained by using (1). In addition, if the high-percentile delay is of interest, we can use various techniques, such as Chebyshev inequality, to bound the high-percentile delay performance. Thus, simultaneously considering multiple L_p norm delay constraints is a more appropriate approach to capturing the actual delay requirement.

3. MINIMIZING ENERGY ALGORITHM

In this section, we present our algorithm, MinEnergy-MD, and also extend our study in various directions to incorporate additional practical constraints.

3.1. Problem Formulation

We formulate the energy minimization problem subject to multiple delay constraints as

$$\textbf{P1}: \quad \min_{\mathbf{x}} \sum_{n=1}^{N} \{[1 - F_{n-1}] \cdot e(x_n) \cdot w_n\} \quad (2)$$

$$s.t., \quad \left\{ \sum_{n=1}^{N} \left[\sum_{j=1}^{n} \frac{w_j}{x_j} \right]^{p_k} f_n \right\}^{\frac{1}{p_k}} \leq \tilde{D}(p_k), \quad (3)$$

$$\text{for } k = 1, 2, \cdots, K,$$

$$\mathbf{0} \preceq \mathbf{x} \preceq \mathbf{s}_{\max}, \quad (4)$$

where "$\succeq$" is an element-wise operator and the processing delay constraints are imposed in the form of K different norms with $1 \le p_1 < p_2 < \cdots < p_K \le \infty$. It can be easily shown that the problem **P1** belongs to convex programming. We propose a primal-dual method, combined with Gauss-Seidel method, to derive the optimal solution. First, given fixed dual variables, we use Gauss-Seidel method to iterative find the optimal processing speeds. Then, we use sub-gradient method to update the dual variables until convergence. Due to space limit, we omit the algorithm description.

3.2. Extension

Next, we provide a brief discussion of how to incorporate additional constraints: (1) limited server speeds; (2) migration overheads; and (3) classification tree.

3.2.1. Limited server speeds

For some systems (e.g., heterogeneous multicore processor, DVFS), processing speeds may only be selected from $\{s_1, s_2, \cdots, s_M\}$, i.e., $\mathbf{x} \in \{s_1, s_2, \cdots, s_M\}^N$. One (heuristic) solution is to round the optimal x_i^* obtained in MinEnergy-MD to a value in $\{s_1, s_2, \cdots, s_M\}$. While there are various methods of rounding, a simple yet effective method is rounding the obtained continuous x_i^* to the closest value in $\{s_1, s_2, \cdots, s_N\}$ that is no less than x_i^*. Doing so will automatically ensure that the processing delay constraints will not be violated. Other more sophisticated algorithms, such as branch-and-bound technique, are also applicable to improve the simple rounding technique.

3.2.2. Migration overheads

Here, we briefly describe how to address the migration overhead, e.g., a certain amount of time during which no servers can process the input stream. Let τ_n^o be the migration overhead of migrating a job from server n to server $n+1$, for $n = 1, 2, \cdots, N-1$, and the delay constraint becomes

$$\left\{\sum_{n=1}^{N}\left[\sum_{i=1}^{n}\frac{w_i}{x_i}+\sum_{i=1}^{n-1}\tau_i^o\right]^{p_k}\cdot f_n\right\}^{\frac{1}{p_k}} \le \tilde{D}(p_k). \tag{5}$$

Therefore, we can reformulate the energy minimization problem by replacing the delay constraint (3) with (5) to account for the migration overheads. The solution can be found in a similar way following our preceding analysis.

3.2.3. Classification tree

In practice, stream mining involves classification trees where each leaf node represents a label. As in Section 2, we still index the classifiers from $i = 1, 2, \cdots, N$, although they may not follow a single chain. There are L different class labels indexed by $l = 1, 2, \cdots, L$. There is a unique path from the data entry point to each label l (i.e., from the parent node of the tree to each leaf node). We denote by $\mathcal{C}_l$ the set of classifiers along the path from the data entry point to label l, and by f_l the probability that an input data is classified into label l.[3] Thus, if an input data is classified into label l, the energy consumption is $\sum_{i\in\mathcal{C}_l} w_i e(x_i)$ and the processing delay is $\sum_{i\in\mathcal{C}_l}\frac{w_i}{x_i}$, where x_i is the processing speed for the node executing classifier i. Therefore, we can formulate the energy minimization problem for a classification tree as

$$\textbf{P2}: \quad \min_{\mathbf{x}} \sum_{l=1}^{L}\left[\sum_{i\in\mathcal{C}_l} w_i e(x_i)\right] f_l \tag{6}$$

$$s.t., \quad \left\{\sum_{l=1}^{L}\left[\sum_{i\in\mathcal{C}_l}\frac{w_i}{x_i}\right]^{p_k}\cdot f_l\right\}^{\frac{1}{p_k}} \le \tilde{D}(p_k), \tag{7}$$

$$\text{for } k = 1, 2, \cdots, K,$$

$$\mathbf{0} \preceq \mathbf{x} \preceq \mathbf{s}_{\max}, \tag{8}$$

which is convex optimization and can be solved similarly using a primal-dual approach as described in Algorithm 1. Note that more complex stream mining systems (e.g., multiple parallel classification trees) can be decomposed into multiple classification trees, each of which can be optimized for energy efficiency by solving **P2**.

4. PERFORMANCE EVALUATION

In this section, we validate our analysis by performing a simulation experiment over a chain of five binary classifiers for human action recognition based on the dataset [15].

4.1. Simulation setup

Name	**Processor**	**Avg. Power Per Core**	**Effective Speed**
A	Intel i7-2600	21.00W	1.0000
B	Intel i5-3210M	6.26W	0.5840

Table 1. Server configuration with measured power and performances.

Due to space limit, we only provide our measurement while noting that the detailed simulation setup is available in [16].

Server. In our experiment, a "server" refers to an actual physical server. Neglecting the almost constant power consumption of other components (e.g., memory, harddisk), we only consider the processor power consumption. We use two physical servers (shown in Table 1), based on which we quantify the resource demand of each classifier and fit the energy function in terms of the processing speed to emulate a heterogeneous system: (1) we randomly generate input streams from our data set and classify them using our physical servers;

[3] A single classifier chain is a special case of classification tree.

(2) we measure the delays and energy consumption; and (3) we substitute the measured delays and energy consumption with projected delays and energy consumption, as though the input streams are processed by other heterogeneous servers which we desire to have.

Classifiers. We measure the time complexity of each of each classifier by running them on the reference server (i.e., server A) and the results are: 0.6416s, 0.6332s, 0.6332s, 0.6431s, 0.6523s.

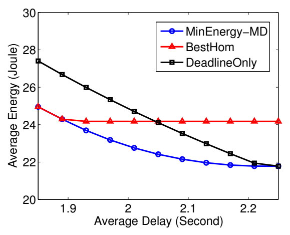
4.2. Experimental Results

For the ease of illustration, we consider two commonly used delays: maximum delay ("MaxD") and average delay ("AvgD"). We fix the default maximum delay as 4s (which is also the period of video sequence arrivals in our experiment) and vary the average delay.

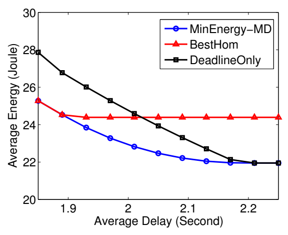
We first compare a heterogeneous system using MinEnergy-MD against a homogeneous one. For the homogeneous system, we choose to use the minimum speed such that the delay constraints are satisfied and the average energy consumption is minimized. This algorithm is referred to as "BestHom". We plot in Fig. 2 the energy consumption of the algorithms with different maximum delay constraints and by varying the average delay constraint. The result shows that we achieve an energy saving of up to 12% by using a heterogeneous system with MinEnergy-MD in comparison with the best homogeneous system. The benefit of MinEnergy-MD vanishes when the average delay constraint is dominant, while it becomes more significant as the average delay constraint loosens. The reason is that, as shown in Section IV, a homogeneous system (and BestHom) is optimal if and only if the average delay constraint is dominant.

To the best of our knowledge, no prior studies have considered determining the optimal server speeds on a heterogeneous system. Here, we compare MinEnergy-MD against the most relevant existing algorithm, which was first proposed in the context DVFS [3]. The algorithm, referred to as "DeadlineOnly", only considers the maximum delay constraint to minimize the energy consumption. If other delay constraints cannot be satisfied by the solution, then a reduced maximum delay constraint is considered until all other delay constraints are satisfied. Fig. 2 shows that MinEnergy-MD can save the average energy consumption by up to 10% compared to DeadlineOnly. The energy saving becomes more significant when the average delay constraint is more stringent: DeadlineOnly is optimal if and only if the maximum delay is dominant.

We also conduct sensitivity studies on different service demand distributions, delay constraints and other power-performance profiles. The results show that MinEnergy-MD still outperforms BestHom and DeadlineOnly in terms of the average energy consumption. The details are omitted here but can be found in [16].



(a) $z(x) = 21 \cdot x^{2.25}$, MaxD=4s.



(b) $z(x) = 21 \cdot x^{2.5} + 2.1$, MaxD=4s.

Fig. 2. Comparing MinEnergy-MD against BestHom and DeadlineOnly.

5. RELATED WORKS

Heterogeneous hardware systems, such as heterogeneous multicore processor [6], heterogeneous cluster [2] and even heterogeneous data center [9], have been shown as an appealing architecture for energy saving. In such a system, a key technique to realize the benefit of energy efficiency is of "job mapping": "hard" jobs are processed by high-performance components for performance improvement whereas "easy" jobs are processed by low-performance components for energy saving. For example, with predicted service demand, incoming jobs are scheduled to the most "appropriate" core based on the Euclidean distance in a multi-dimension performance metric space [5], and similarly an appropriate number of servers are turned on/off to minimize the energy while satisfying the delay requirement [8][9]. Nevertheless, "job mapping" is not applicable to stream mining systems, since the service demand (e.g., measured in terms of the number of classifiers an input data passes through) is unknown until the classification has been completed. Some prior studies, such as [3][4], achieve energy saving based on the probability distribution of service demand. However, these studies are insufficient for our research either because they only consider the maximum delay constraint.

6. CONCLUSION

We investigated energy-efficient scheduling for delay-sensitive stream mining systems under multiple delay constraints expressed in the L_p norms. We proposed MinEnergy-MD to optimally determine the processing speed for each classifier. We conducted experimental studies to validate MinEnergy-MD. The results showed that compared to the best homogeneous system and the best-known existing algorithm, MinEnergy-MD reduces the average energy consumption by more than 10% given the same delay requirement.

7. REFERENCES

[1] R. Ducasse, D. S. Turaga, and M. van der Schaar, “Adaptive topologic optimization for large-scale stream mining,” *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 3, pp. 620-636, June 2010.

[2] N. Yigitbasi, K. Datta, N. Jain, and T. Willke, “Energy efficient scheduling of MapReduce workloads on heterogeneous clusters,” *Green Computing Middleware*, 2011.

[3] J. R. Lorch and A. J. Smith, “Improving dynamic voltage scaling algorithms with PACE,” *ACM Sigmetrics*, 2001.

[4] R. Xu, C. Xi, R. Melhem, and D. Moss, “Practical PACE for embedded systems,” *EMSOFT*, 2004.

[5] J. Chen and L. K. John, “Efficient program scheduling for heterogeneous multi-core processors,” *DAC*, 2009.

[6] P. Greenhalgh, “Big.little processing with arm cortex.-a15 & cortex-a7,” *ARM Whitepaper*, 2011.

[7] D. Brooks, P. Bose, S. Schuster, H. Jacobson, P. Kudva, A. Buyuktosunoglu, J. Wellman, V. Zyuban, M. Gupta, and P. Cook, “Power-aware microarchitecture: design and modeling challenges for next generation microprocessors,” *IEEE Micro*, 2000.

[8] B. Guenter, N. Jain, and C. Williams, “Managing cost, performance and reliability tradeoffs for energy-aware server provisioning,” *IEEE Infocom*, 2011.

[9] M. Liu, Z. Liu, A. Wierman, and L. L. H. Andrew, “Online algorithms for geographical load balancing,” *IGCC*, 2012.

[10] S. Kundu, R. Rangaswami, A. Gulati, K. Dutta, and M. Zhao, “Modeling virtualized applications using machine learning techniques,” *VEE*, 2012.

[11] N. Bansal and K. Pruhs, “Server scheduling in the L_p norm: a rising tide lifts all boat,” *ACM STOC*, 2003.

[12] A. Wierman and M. Harchol-Balter, “Classifying scheduling policies with respect to higher moments of conditional response time,” *ACM Sigmetrics*, 2005.

[13] Y. Xie and T. Yang, “Cell discarding policies supporting multiple delay and loss requirements in ATM networks,” *IEEE Globecom*, 1997.

[14] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, “Power management of datacenter workloads using per-core power gating,” *IEEE Comput. Archit. Lett.* vol. 8, no. 2, pp. 48-51, July 2009.

[15] `http://www.nada.kth.se/cvap/actions/`

[16] S. Ren, C. Lan, and M. van der Schaar, Online supplementary materials for “Energy-Efficient Design of Real-Time Stream Mining Systems”, `https://www.dropbox.com/s/hau6e05z5hml3su/journal.pdf`.