

Web Monitoring 2.0: Crossing Streams to Satisfy Complex Data Needs

Haggai Roitman¹, Avigdor Gal², Louiqa Raschid³

Technion - Israel Institute of Technology and the University of Maryland, College Park

¹haggair@tx.technion.ac.il

²avigal@ie.technion.ac.il

³louiqa@umiacs.umd.edu

Abstract—Web Monitoring 2.0 supports the complex information needs of clients who probe information and generate mashups by integrating across multiple volatile streams. A proxy that aims at capturing multiple client profiles that are customized to each client will face a scalability challenge in trying to maximize the number of clients served while at the same time fully satisfying complex client needs. In this paper, we introduce an abstraction of complex execution intervals, a combination of time intervals and information streams, to capture complex client needs. Given some budgetary constraints (*e.g.*, bandwidth), we present offline algorithmic solutions for the problem of maximizing completeness.

I. INTRODUCTION

Web Monitoring 2.0 extends Web 2.0 to create a personalized proxy based platform where users can satisfy their complex information monitoring and aggregation/mashup needs by polling multiple information-rich and volatile Web 2.0 data sources. Such a platform responds to the personalization needs of Web 2.0. Instead of simply filtering data streams as they are pushed from servers, a proxy actively decides when it needs to *cross* these streams using pull-based technology, to satisfy client customization. Example platforms include personalization portals that are available to users worldwide, *e.g.*, iGoogle¹ and MyYahoo!² They provide a single point of access for personalized information. Portals provide services for continuously refreshing user profiles and for integration via a *mashup*³ of data extracted from multiple heterogeneous data sources. Other examples include Web content analysis platforms (*e.g.*, [1], [2]) that require to collect data from multiple (possibly interrelated) Web sources such as RSS feeds.

Proxies cross multiple streams of events and probing must be performed in a timely manner to satisfy both the characteristics of the servers, *e.g.*, intensity of updates, as well as the customization needs of clients. Supporting complex information needs requires a proxy to recognize dependencies among probes across resources. The proxy will face a scalability challenge when trying to satisfy the customized complex needs of millions of clients.

Push-based solutions to satisfy complex user needs exist yet data collection requires a high cost [3], especially if such

push-based technology is not natural to the Web environment. Pull-based solutions have considered only simple monitoring solutions (*e.g.*, [4]) that cannot satisfy the complex user needs of Web 2.0 applications mashing up multiple continuously-updated streams.

Web 2.0 monitoring is also different from continuous query processing or adaptive processing on streams where the objective is to support complex query processing, *e.g.*, aggregation queries on data streams. To the best of our knowledge, our research is the first to support a new generation of solutions to address complex Web 2.0 monitoring in (primarily) pull-only settings. In prior work [5], we summarized the complexity of the Web 2.0 monitoring problem but did not provide solutions.

Complex user needs are expressed as a complex execution interval (CEI) [5] comprising of multiple simple execution intervals [6]. A proxy has to probe the corresponding resources during these intervals in order to satisfy the profile.

The rest of the paper is organized as follows. In Section II we summarize the model of [5] for complex profile monitoring, and formally define the problem, followed by the monitoring solution in Section III. Section IV concludes and provides directions for future work.

II. MODEL AND PROBLEM DEFINITION

Servers and *clients* share data through *proxies*. A server manages resources and can be *probed* (pull-based) by the proxy on behalf of its clients. Occasionally a server may *push* either an update or a notification of an update to the proxy. We assume that a proxy implements Web 2.0 monitoring by *crossing streams of data*; using a pull protocol (*e.g.*, *HTTP GET* queries). Our focus is on the scheduling of proxy pull tasks to satisfy complex client information needs. We discuss the three building blocks of our model, namely *client profiles*, *execution intervals*, and *schedules*.

A. Profiles and complex execution intervals

The complex information needs of a client is specified as a profile stored at the proxy. A client profile is associated with a set of resources and *complex execution intervals* (CEI) [5]. An execution interval (EI) [6] defines periods of time during which the profile must be synchronized with the state of the resource at the server. Crossing multiple data streams is represented by combining individual EIs to construct CEIs, possibly over a

¹<http://www.google.com/ig>

²<http://cm.my.yahoo.com/>

³See <http://www.programmableweb.com/> for mashup examples.

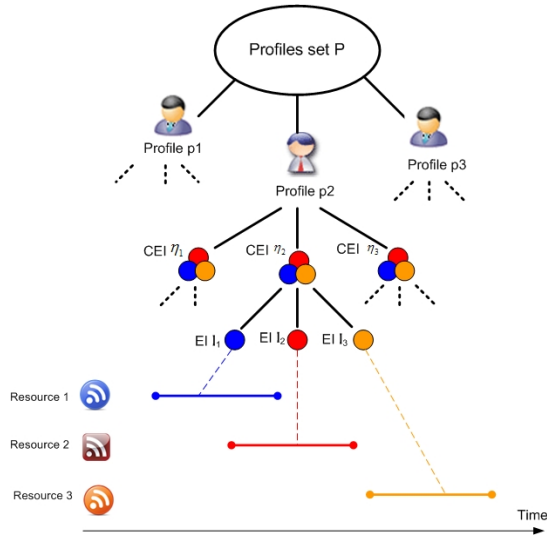


Fig. 1. A hierarchical illustration of the main concepts of our model.

set of resources. Each EI in a CEI should be monitored *once* for the CEI to be satisfied (or “captured”). We assume that the EIs of a CEI can be captured *in any order* and we consider *AND* semantics, i.e., all EIs of a CEI must be captured. This is equivalent to the conjunction joining operator in the Amit situation manager [7].

Formally, let $\mathcal{R} = \{r_1, r_2, \dots, r_n\}$ be a set of n resources and let $\mathcal{T} = (T_1, T_2, \dots, T_K)$ be an epoch with K chronons.⁴ We assume the proxy manages a set of client profiles $\mathcal{P} = \{p_1, p_2, \dots, p_m\}$. A client profile $p = \{\eta \mid \eta = \langle I_1, I_2, \dots, I_t \rangle\}$ is a collection of CEIs. A CEI η contains several EIs, where each EI I is associated with a resource $r \in \mathcal{R}$ and I contains a start and finish chronon $I = [T_s, T_f]; T_s, T_f \in \mathcal{T}; T_s \leq T_f$. We further denote by $|I|$ the number of chronons in EI I . Profiles, CEIs, and EIs construct a *hierarchy* (as illustrated in Figure 1), in which a profile is a *parent* of its CEIs, and a CEI is a *parent* of its EIs. Two CEIs within the same profile are *siblings*, and two EIs within the same CEI are also *siblings*. We use the number of EIs in a CEI to model *profile complexity*. Therefore, we denote by $rank(p)$ the maximal number of execution intervals in any CEI $\eta \in p$ ($rank(p) = \max_{\eta \in p} \{|\eta|\}$), where $|\eta|$ is the number of execution intervals in η . The definition is easily extended to a set of profiles \mathcal{P} as follows: $rank(\mathcal{P}) = \max_{p \in \mathcal{P}} \{rank(p)\}$.

The beginning of an interval is determined by an update event at a resource or a temporal event (e.g., every ten minutes). In the case that the server will *push* the update event, or for a temporal event, the beginning of the interval is deterministic. A proxy may need to predict an update event using an update model and stochastic modeling [8] and *pull* the update event. The window (length) of the interval is determined with respect to the stream of update events, (e.g., update = overwrite), or as a temporal event (e.g., within five minutes of the beginning of the interval).

⁴A chronon is an indivisible unit of time.

B. Schedules

A data delivery schedule $S = \{s_{i,j}\}_{i=1,\dots,n;j=1,\dots,K}$ (n resources and K chronons) assigns $s_{i,j} = 1$ if resource $r_i \in \mathcal{R}$ should be monitored (probed) by the proxy at chronon $T_j \in \mathcal{T}$, else $s_{i,j} = 0$. We denote by \mathbb{S} the set of all possible schedules.

We use an indicator $\mathbb{I}(I, S)$ to indicate whether a schedule S successfully probes and captures (the state of some resource r_i during) the EI I . Given a profile p , a CEI $\eta \in p$, and an EI $I \in \eta$ that refers to resource $r_i \in \mathcal{R}$, we have the following:

$$\mathbb{I}(I, S) = \begin{cases} 1, & \exists T_j \in I : s_{i,j} = 1 \\ 0, & \text{otherwise} \end{cases}$$

We extend the indicator to $\mathbb{I}(I, S)$ to describe capturing a CEI as follows: Given a profile p and a CEI $\eta \in p$, we say that η is *captured* by schedule $S \in \mathbb{S}$ if $\mathbb{I}(\eta, S) = \prod_{I \in \eta} \mathbb{I}(I, S) = 1$.

C. The monitoring problem

We assume that the proxy has a limited amount of resources that can be consumed for the monitoring task of client profiles. In this paper we consider a constraint similar to the one used in prior works of Web Monitoring [4] and Web Crawlers [9], where at each chronon $T_j \in \mathcal{T}$ the proxy can monitor up to C_j resources. This constraint is represented by a budget vector $\vec{C} = (C_1, C_2, \dots, C_K)$.

Given a set of client profiles $\mathcal{P} = \{p_1, p_2, \dots, p_m\}$, the proxy objective is to *maximize the gained completeness*, that is, to maximize the number of CEIs from \mathcal{P} that are captured given the budget \vec{C} . A CEI is successfully captured once *all* of its execution intervals are captured. Every CEI $\eta \in p$ that is successfully captured by the proxy schedule (indicated by $\mathbb{I}(\eta, S) = 1$) increases the gained completeness.

Given a schedule $S \in \mathbb{S}$, the *gained completeness* (denoted GC in short) from monitoring \mathcal{P} during \mathcal{T} according to S is calculated as follows (where $|p|$ denotes the number of CEIs in profile p):

$$GC(\mathcal{P}, \mathcal{T}, S) = \frac{\sum_{p \in \mathcal{P}} \sum_{\eta \in p} \mathbb{I}(\eta, S)}{\sum_{p \in \mathcal{P}} |p|} \quad (1)$$

Formally, the monitoring problem is defined by the following constrained optimization problem.

Problem 1 (Complex Monitoring): Given a set of profiles \mathcal{P} and an epoch \mathcal{T} :

$$\begin{aligned} & \text{maximize } GC(\mathcal{P}, \mathcal{T}, S) \\ & \text{s.t. } \sum_{i=1}^n s_{i,j} \leq C_j, \forall j = 1, 2, \dots, K \end{aligned}$$

Problem 1 is challenging, as we shall demonstrate in Section III. Nevertheless, it can be extended in several interesting directions. First, we do not consider the varying costs of probing resources. These variations may be due to computational costs, e.g., extracting a stock price may be cheaper than searching for a keyword in a blog, the bandwidth needed to download results of varying size, monetary charges at the servers, etc. In this case, each probe will not consume the same budget. We defer these extensions to future work.

III. OFFLINE MONITORING SOLUTIONS

We now present offline solutions to the complex monitoring problem. We see two motivations for this analysis. First, offline solutions can serve as a baseline and provide upper bounds to the optimal online performance. Secondly, by analyzing the complexity of an offline solution we can gain some understanding of the difficulty of designing solutions to the online case. We consider online solutions as future work.

In an offline setting, the proxy is provided with **all** CEIs in \mathcal{P} for K chronons **in advance** and has to determine the schedule S of probing resources in \mathcal{R} . Such a scenario cannot be achieved in practice in most cases. Recall that for all but very simple cases new requests for complex execution intervals arrive on the fly. We first discuss the complexity of a full enumeration of feasible schedules, yielding an optimal yet costly solution. Then, we describe an offline approximation algorithm and discuss its properties.

A. Schedule Enumeration

A *feasible solution* to Problem 1 is a schedule that satisfies the problem constraints. We denote by $\mathbb{S}(\vec{C}) \subseteq \mathbb{S}$ the set of all feasible schedules, given \vec{C} . Proposition 1 provides the cost of solving Problem 1 using full enumeration.

Proposition 1: Given n resources, K chronons, and a constraint \vec{C} on the number of probes per chronon, the cost of finding an optimal solution to Problem 1 by enumerating all feasible schedules is $O(Kn^{KC_{\max}+1})$ time, where $C_{\max} = \max_{j=1,2,\dots,K} (C_j)$.

Proof: [Sketch] From Lemma 1 in [5], the number of feasible schedules is given by $O(n^{KC_{\max}})$. The computation of a schedule cost is based on a constant number of accesses to each entry in an $n \times K$ matrix, which sums to $O(Kn)$. Therefore, the overall cost of finding an optimal solution by enumerating all feasible solutions is $O(Kn^{KC_{\max}+1})$. ■

It is worth noting that C_{\max} and K are known yet arbitrary constants and therefore the problem is polynomial in the number of resources. This serves as little consolation whenever C_{\max} or K are large (e.g., $K = 100$). We assume that $C_{\max} \neq O(n)$, otherwise scheduling would be easy since there will be sufficient budget to probe most resources at each chronon. Nevertheless, C_{\max} may still be of substantial size.

Clearly, enumerating all possible solutions may not be the most efficient way of solving this problem. However, to date we are unaware of any low-polynomial algorithm for solving Problem 1.

B. Offline Approximation

We now discuss how to achieve the best offline approximation for Problem 1.

We denote by $\mathcal{P}^{[1]}$ a set of profiles for which any EI I of any CEI has a width of *exactly one chronon*. Proposition 2 establishes the relationship between a solution to problems with input $\mathcal{P}^{[1]}$ to problems with a general set of profiles of the same complexity.

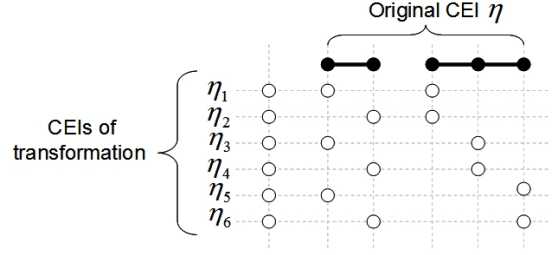


Fig. 2. Transformation illustration of a CEI with 2 EIs, the first with 2 chronons and the second with 3 chronons, producing 6 new CEIs each with 3 EIs

Proposition 2: Let \mathcal{P} be an arbitrary set of profiles with $rank(\mathcal{P}) = k$. Let \mathcal{A} be an algorithm that provides $\alpha(k)$ -approximation given an arbitrary $\mathcal{P}^{[1]}$ with $rank(\mathcal{P}^{[1]}) = k$, then \mathcal{A} can provide $\alpha(k+1)$ -approximation given \mathcal{P} .

Proof: [Sketch] Given \mathcal{P} we first transform it into a $\mathcal{P}^{[1]}$ set of profiles as follows. Let $\eta = \{I_1, I_2, \dots, I_k\}$ be a CEI with k execution intervals, where $\exists p \in \mathcal{P} : \eta \in p$. Further let n_q denote the number of chronons of $I_q \in \eta$. We generate $\prod_{q=1}^k n_q$ new CEIs η_j each has $k+1$ execution intervals of width of exactly 1 chronon (see Figure 2 for illustration). We position the execution intervals of each new CEI according to combination formed from the EIs chronons of the original CEI η . For example, the first new CEI η_1 has each execution interval positioned in a way that it collides with the corresponding EI of the original CEI η in the first chronon of that EI; the second new CEI η_2 has each EI positioned in a way that it collides with the corresponding EI of the original CEI in the first chronon of that EI, except for the first EI which collides with the first EI of the original CEI in the second chronon of its first EI; and so on. Obviously the output of the transformation is a problem with profiles $\mathcal{P}^{[1]}$, which further has $rank(\mathcal{P}^{[1]}) = k+1$. Thus, \mathcal{A} can provide $\alpha(k+1)$ -approximation to the new transformed problem. It is easy to show that any solution produced by \mathcal{A} to $\mathcal{P}^{[1]}$ is a solution to the original problem \mathcal{P} . Therefore, \mathcal{A} provides $\alpha(k+1)$ -approximation given \mathcal{P} . ■

Bar-Yehuda et al. have proposed in [10] an offline approximation to the problem of scheduling t -intervals, also termed *split intervals* in [10]. In their problem, t -intervals are composed of segments of arbitrary length⁵ and a segment represents an interval in which a resource is needed. Note that t -intervals are slightly different than our CEIs since we require that a resource should be probed only in a single chronon of each execution interval.

Equipped with Proposition 2, we utilize the *Local Ratio scheme* [10], which has the best approximation ratio given an arbitrary $\mathcal{P}^{[1]}$ with $rank(\mathcal{P}^{[1]}) = k$, and provides $2k$ -approximation for $C_{max} = 1$ and $(2k+1)$ -approximation for $C_{max} > 1$. Thus, for an arbitrary \mathcal{P} with the same rank, according to Proposition 2, applying the Local Ratio scheme of [10] we can achieve $(2k+2)$ -approximation when

⁵It is worth noting that segments in [10] correspond to execution intervals, and the segments of each t -interval refer to a single resource.

$C_{max} = 1$, and $(2k + 3)$ -approximation for $C_{max} > 1$. For example, assuming that we have as an input profiles of $\mathcal{P}^{[1]}$ with $rank(\mathcal{P}^{[1]}) = 2$ (each CEI has at most 2 segments), in the case of $C = 1$, we can produce a feasible solution to Problem 1 that guarantees at least $1/(2 \cdot 2) = 25\%$ of the optimal gained completeness.

C. Proof-of-Concept

As a proof-of-concept, we now compare the performance of the proposed offline approximation with WIC [4], a state-of-the-art Web monitoring solution. For this comparison we used *real-world* trace of 732 eBay 3-day auctions with a total of 11150 bids for Intel, IBM, and Dell laptop computers that was obtained from an RSS feed for a search query available on eBay⁶. We used an `AuctionWatch(k)` profile template, *i.e.*, the client requires to deliver tuples of k records, each record contains the latest bid values captured using *immediate* probing as soon as a bid is posted on every one of the k auctions specified in the profile. The $rank(\mathcal{P})$ varies from 1 to 5. For this experiment, if $rank(\mathcal{P}) = 3$, then all CEIs that were generated for that problem instance have exactly 3 EIs.

We further restrict our analysis to $\mathcal{P}^{[1]}$ profiles. To generate these profiles and CEIs, we use $C = 1$ and avoid intra-resource overlap [5] by ensuring that each EI of a CEI refers to a distinct resource. For this parameter setting, the offline approximation guarantees a $2k$ -approximation (for $rank(\mathcal{P}^{[1]}) = k$). This is the *best possible offline approximation* (see Section III-B). For this setting and $k = 1$, the WIC algorithm is optimal [4].

Figure 3 reports on WIC and the offline approximation performance. The Y axis reports on the percentage completeness of WIC and the offline approximation, calculated with respect to the worst case upper bound on the optimal completeness which was obtained using full enumeration as a baseline.

Our first observation is that as the rank increases, both solutions have to devote more probes to CEIs that may not be satisfied. Hence there is a general trend that the completeness decreases as the rank increases. We clearly observe that while for simple profiles (*i.e.*, $rank(\mathcal{P}) = 1$) both solutions reached the optimal performance, the offline approximation completely dominates WIC as the rank increases.

Such results were consistent using other parameter settings in our experiments. Our empirical analysis results suggest that simple state-of-the-art Web monitoring solutions do not fit well to the challenges of new Web 2.0 monitoring problems. This motivates us to seek more intelligent online solutions that can handle the complex requirements of Web 2.0 applications.

IV. CONCLUSIONS AND FUTURE WORK

In this work we presented a framework for satisfaction of complex data needs in pull based environments that involve volatile data. We then presented two offline solutions. Using empirical analysis we provided a proof-of-concept that demonstrates the need for new Web monitoring solutions that can handle the complex requirements of new emerging Web 2.0 applications.

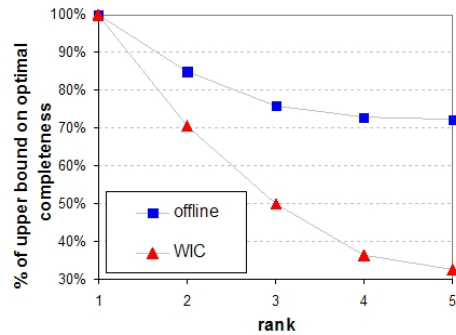


Fig. 3. Relative performance of online policies with the baseline with $W=0$

While the offline solutions can be used as baselines, they cannot offer a scalable alternative to online solutions. Therefore, we now consider new online solutions that can better handle the complex requirements. As another future extension of this work we shall consider more general profile satisfaction constraints given as client profile utilities. Such utilities can further help to construct better prioritized policies. In this paper we assumed that all execution intervals of any CEI are required to be captured. We further intend to extend the notion of CEIs to a more general construction which allow also alternatives (*e.g.*, capture of a subset of execution intervals).

REFERENCES

- [1] N. Bansal and N. Koudas, "Searching the blogosphere," in *WebDB*, 2007.
- [2] H. Roitman, D. Carmel, and E. Yom-Tov, "Maintaining dynamic channel profiles on the web," in *Proceedings of the 34th International Conference on Very Large Data Bases (VLDB 2008)*, Auckland, New Zealand, 2008.
- [3] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, Y. Xing, and S. B. Zdonik, "Scalable distributed stream processing," in *CIDR*, 2003.
- [4] S. Pandey, K. Dhamdhere, and C. Olston, "WIC: A general-purpose algorithm for monitoring web information sources," in *Proceedings of the International conference on Very Large Data Bases (VLDB)*, Toronto, ON, Canada, Sept. 2004, pp. 360–371.
- [5] H. Roitman, A. Gal, , and L. Raschid, "Satisfying complex data needs using pull-based online monitoring of volatile data sources," in *Proceedings of the IEEE CS International Conference on Data Engineering*, Cancun, Mexico, 2008.
- [6] —, "Capturing approximated data delivery tradeoffs," in *Proceedings of the IEEE CS International Conference on Data Engineering*, Cancun, Mexico, 2008.
- [7] A. Adi and O. Etzion, "Amit - the situation manager," *The International Journal on Very Large Data Bases*, vol. 13, no. 2, pp. 177–203, May 2004.
- [8] A. Gal and J. Eckstein, "Managing periodically updated data in relational databases: A stochastic modeling approach," *Journal of the ACM*, vol. 48, no. 6, pp. 1141–1183, 2001.
- [9] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen, "Optimal crawling strategies for web search engines," in *Proceedings International WWW Conference*. Honolulu, Hawaii, USA: ACM Press, 2002, pp. 136–147.
- [10] R. Bar-Yehuda, M. M. Halldórsson, J. Naor, H. Shachnai, and I. Shapira, "Scheduling split intervals," *SIAM J. Comput.*, vol. 36, no. 1, pp. 1–15, 2006.

⁶<http://search.ebay.com/ws/search/>