A Generic Approach to Event Aware Prediction

Matthew Sladescu\textsuperscript{1,2}, Alan Fekete\textsuperscript{1,2}, Kevin Lee\textsuperscript{2,3}, and Anna Liu\textsuperscript{2,3}

\textsuperscript{1} The University of Sydney
\textsuperscript{2} National ICT Australia
\textsuperscript{3} The University of NSW
\{firstname.lastname\}@nicta.com.au

Abstract. The unavailability or poor service of web sites that are faced with surges in workload during important events, like product announcements, sporting events, and sales events, have frequently been the subject of news headlines. Companies that host such important events often do so with the aim of positively influencing their public profile or maximizing their revenue, whereby the negative news headlines that are often associated with such events, come into direct contradiction with these goals. The accurate prediction of event-associated workload bursts is key to more effectively managing important events, however many of the existing prediction methods fail to accurately predict the time and magnitude of such bursts. In this paper we extend on earlier work in event aware prediction to provide a generic method for predicting event-associated workload that can be re-used for multiple event types. We propose evaluation methods for this generic event aware prediction approach, which will be carried out in future work, for which a positive assessment can indicate the applicability of event-aware-prediction to a multitude of other domains beyond workload prediction.

Keywords: cloud computing, workload prediction, prediction, events, workload bursts

1 Introduction

Important events like product announcements, sporting events, and online sale events have frequently been responsible for inducing workload bursts for affected web-resources. These bursts have often overwhelmed the compute resources available to serve requests, consequently leading to increased web site response times and unavailability. Not only can these effects lead to lost revenue, but they can also negatively impact on a web site’s reputation.

One of the most attractive properties of cloud computing, known as elasticity, has the potential to ameliorate these negative effects. Elasticity allows IaaS customers to acquire and release compute resources on an as needed basis, hence potentially providing sufficient capacity to support the demand induced by flash crowd events. However, as discussed in earlier work \cite{11, 10}, the on-time availability of cloud computing resources to serve flash crowds depends on how
well resource provisioning policies make decisions about when and how many resources to acquire or release.

Recent literature [12, 6, 3, 7] has promoted the use of demand predictions in making resource-provisioning decisions. This has been driven by the need to pre-provision resources to overcome initialization lag [9, 13], as well as other financial benefits that knowledge of future demand can bring (e.g., making more informed decisions about when it would be financially beneficial to reserve resources based on future load).

Many of the current workload prediction approaches show promising results in predicting the mean workload, but have been found to be ineffective at predicting bursts in workload [8, 11]. When considering that workload bursts are one of the most opportune times to lease resources from an IaaS provider, rather than running a surplus of compute resources in-house which would at other times mostly remain idle, it becomes imperative to be able to predict when these workload bursts occur.

Earlier work [11, 10] introduced an event-aware approach for predicting event-associated workload. This method made use of available information about upcoming events, together with a model for how workload fluctuates during each event, to predict event-associated workload. Studies in [11] showed how a generic EAP framework can be used to predict auction bidding workload, where the burst prediction accuracy yielded by this approach was superior to all the prediction methods trialed, including a method that was dedicated workload burst prediction.

In this paper we extend on earlier work in EAP by contributing a generic EAP prediction module in Section 2 and illustrating how the EAP framework can be re-used across multiple domains. Further, we hypothesize that this approach brings benefits in terms of allowing you to more effectively predict workload bursts; providing more independence to prediction horizon than many existing methods; as well as allowing for re-use across multiple domains and event-types. In Section 3 we propose techniques for evaluating these hypotheses in future work.

## 2 Event Aware Prediction

In this section we describe a generic approach, subject to the assumptions discussed later, to predicting event-associated workload. This approach makes use of the EAP framework shown in Fig 1, as well as the prediction process shown in Fig 2. The EAP framework elements shown in Fig 1 include the main inputs used by EAP, as well as the EAP prediction module that uses these inputs to form a prediction for workload.

### 2.1 Event Repositories

Two of the main inputs used by EAP, shown in Figure 1, are the Past and Upcoming Event Repositories. These repositories each contain a list of events, where
each event can be defined as in [11] as “a scheduled or unscheduled occurrence, (like an online sale); which can be described by an arbitrary set of features, (like the event starting and ending times); and can induce load for associated resources (like the web pages or database entries associated with a popular online sale).” In this paper we introduce two types of features to describe an event: time features and secondary features. We label the important times during an eventful period, like the starting time and ending time of an auction event, as time features. These time features are often closely linked to when changes in the workload profile for an event occur. Examples include the surge of bids in an auction being close to the closing time of an auction, as well as surges in workload for a product announcement being close to the announcement time.

We label the features that don’t describe a time during an eventful period, but which could influence the workload profile associated with an event, as secondary features. Examples for time features and secondary features for three different types of events are shown in Table 1.

In order to allow for a generic event aware prediction approach, we permit the addition of any type of event, that can be described by at least one time feature, and any number of secondary features into the repositories of past and upcoming events. As shown in Fig 2, a repository loader is responsible for determining the types of features that are important to the type of event that they are loading. Examples of repository loaders can include an Auction Event Repository.
Table 1. Examples for Time Features and Secondary Features

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Time Features</th>
<th>Secondary Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auction Event</td>
<td>- Auction Starting Time</td>
<td>- Auction Category</td>
</tr>
<tr>
<td></td>
<td>- Auction Ending Time</td>
<td>- Seller Reputation</td>
</tr>
<tr>
<td>Product Announcement Event</td>
<td>- Announcement Time</td>
<td>- Product Name</td>
</tr>
<tr>
<td></td>
<td>Derived Time Features:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Event Associated Burst Starting Time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Event Associated Burst Ending Time</td>
<td></td>
</tr>
<tr>
<td>Sporting Event</td>
<td>- Sporting Event Starting Time</td>
<td>- Teams Playing</td>
</tr>
<tr>
<td></td>
<td>- Sporting Event Ending Time</td>
<td>- Game Round (eg: quarter final, semi final or final)</td>
</tr>
<tr>
<td></td>
<td>- Sporting Event Half Time Start</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Sporting Event Half Time End</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Derived Time Features:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Event Associated Burst Starting Time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Event Associated Burst Ending Time</td>
<td></td>
</tr>
</tbody>
</table>
Loader, a Product Announcement Event Repository Loader, and a Sporting Event Repository Loader, where these repository loaders correspond to the event types shown in Table 1.

We define an eventful period as the duration of event-associated workload. For the event type examples listed in Table 1, we notice that the eventful period for some events can be bound directly by time-features that are specified (e.g.: auction events don’t experience any bids before the auction starting time or after the auction closing time). Other events like product announcements and sporting events can’t be bounded directly by specified time-features. For instance, product Announcements and Sporting Events can exhibit event-associated workload before the earliest specified time feature (e.g.: workload generated in anticipation of a product announcement or the start of a sporting event), or after the latest specified time feature (e.g.: while interest in the announced product or sporting event is still high). For this type of event, we need to derive the time-features that are defined as the limits for the eventful period. For bursts associated with events like product announcements and sporting events, the time features that describe the limits of an eventful period can be derived by using the earliest and latest time features as inputs to the gradient descent technique used in [8] to identify the lower and upper bounds of a workload burst. This will result in the derived time features listed in Table 1 that describe the event associated burst starting and ending times.

Each event in an event repository is also linked to a list of resources that will be affected by event-associated workload (i.e.: resources like web-pages and database objects that will experience event-associated workload during the eventful period). For events in the past event repository, each of these resources is also linked to a time-series that represents the workload history for the resource during the eventful period. The resources linked to events in the upcoming event repository, where the event is active (i.e.: the event is currently taking place), also reference any workload history which has developed since the lower bound of an eventful period.

2.2 Generating the Workload Model

Each of the event types listed in Table 1 has a different associated workload profile. Instead of identifying a common workload model for all these event types, we define a common process that can learn the workload profile for the type of event under consideration, based on the event entries in the past event repository. This common process is shown as the “Train Event Workload Model” step in Fig 2, and it produces a workload model to fit the event type under consideration. We note that when discussing the workload model linked to an event, we are implicitly referencing a composite workload model that contains a separate model for each resource that is affected by the event. The rest of this section describes the process used to train a workload model from a repository of past events.

The common learning process for generating a workload model is described in Fig 6. This process creates a workload model, for each resource affected by
an event, that consists of a number of smaller phase models as described in CalcWorkloadModel. Each phase model is bound by two time features, which correspond to the start and end of the phase. As described in CreatePhaseModelBetweenTwoTimeFeatures, each phase model is composed of 100 separate hypotheses, where each hypothesis is responsible for predicting the number of %hits that will occur at a given percentile. The function UpdateHypothesisForPercentile illustrates how the number of %hits expected at a given percentile can vary based on the specified secondary features of an event.

Hence, all time features that are specified for an event will be used to define the starting and ending point for phases, where there are n phases for n+1 time features. We note that the n+1 time features may also include derived time features if there is event associated workload to the left of the earliest time feature, or to the right of the latest time feature. We make use of secondary time features to allow us to model variations in the workload model that may be influenced by such features. We use multiple phases bounded by time features since this allows important workload features like workload surges, (which are usually found close to these time features), to be more accurately located. We will illustrate this with a concrete example after showing the phases that compose the workload model for the three event types discussed in Table 1.

In Figures 3 to 5 we show the number of phases used to model workload bursts for auction, product announcement, and sporting events respectively, as well as the time features that are used to bind these phases. When present, the derived time features that represent the lower and upper burst bounds are shown in italics.

Fig. 2. EAP Process
The workload shown in Figure 4 represents the hit-rate of a Wikipedia page around the time of the announcement for a popular product, as retrieved from [5]. The lower and upper bounds of the announcement event associated spike are identified by running the gradient descent algorithm described in [8] to the left and right of the announcement time. We note that the tail to the left of the announcement time can be longer or shorter, based on how much anticipation there is for the product being announced. If only one phase was used to model this workload, the location of the burst, quantified by %Age in this phase, would vary based on how long the left tail is. Hence, using a single phase would not allow one to consistently and accurately locate the position of the spike. From the workload traces we've seen, the announcement time has always been relatively close to the workload burst, hence it would be beneficial to use this time feature to help in locating the burst more accurately. This is achieved when using multiple phase models that are bounded by time features, since this would locate the important workload characteristics close to the start or end of a phase model.

The process defined in Fig 6 makes use of the past event repository to generate a workload model that fits a given event type, based on the features in the repository of past events. Although the existence of past event instances can help in creating a more accurate model for how workload fluctuates during an eventful period, there are alternatives to acquiring a workload model for use with EAP, for example:
**Fig. 4.** Product Announcement Workload partitioned into the phases that will compose the workload model.

**Fig. 5.** Sporting Event Workload [2] partitioned into the phases that will compose the workload model.
CalcWorkloadModel:
(Calculates multi-phase workload model, where each phase is bounded by two time
features (where time features can be specified or derived))

1. While next pair of time features (features i and j) exists:
   a. CreatePhaseModelBetweenTwoTimeFeatures(i, j)
   b. Add Created Phase to Multi-Phase Workload Model

CreatePhaseModelBetweenTwoTimeFeatures:
(Calculates the phase model between two time features i and j)

1. For each event in the past event repository:
   a. For each resource k linked to event:
      i. For each of 100 hypotheses that are used to predict the
         A. UpdateHypothesisForPercentile(m, k, event features, workload his-
            tory for resource k, i, j)
   2. Return a workload model for each resource associated with this event. The
      workload model for each resource associated with this event consists of 100
      hypotheses. Each hypothesis can predict the

UpdateHypothesisForPercentile:
(Updates the hypothesis for a percentile m for a given resource k given an arbitrary
set of event secondary features, as well as the workload history for resource k
between time features i and j of a given event)

1. Calculate total number of hits (totalHits) occurring at percentile m based on
   the workload history for resource k.

2. Form a training example, which includes the arbitrary set of secondary features
   as the inputs, and totalHits as the output.

3. Update hypothesis for percentile m for the workload model corresponding to
   phase between time features i and j; and resource k.

Fig. 6. Generating a Workload Model
A number of generic burst models have been created for the purpose of load testing, including [4] and [1].

There are numerous event types like product announcements, that are a commonly required by many companies. If a company had access to a library of common workload models, it would not necessarily require any of their own events defined in a past events repository to begin using EAP.

2.3 Generating a Workload Prediction

Now that we’ve described how the workload model for an event type can be created from a repository of past events, we look at how to use this workload model with a repository of upcoming events to predict workload. Note that, in the explanation that follows, workload is predicted for a single resource associated with each event, however, the same process can be repeated to predict workload for any number of resources, since each resource has an associated workload model. Before prediction can begin, the phases for all upcoming events, and hence the lifetime of upcoming events must first be defined, as shown in Fig 7.

Once the phases are created for all upcoming events, prediction can take place. The process for generating a prediction for a given minute i is described in Fig 8.

A similar workload prediction algorithm was described in [11], where it was pointed out that a bid-bucket approach was required to more accurately predict workload bursts. We appropriate the bid-bucket approach here, renaming it as the hit-bucket approach, and show the workload prediction method updated to include this approach in Fig 9.

2.4 Assumptions for the Generic Workload Prediction Approach

In this paper we have presented a generic approach for predicting event-associated workload. The assumptions under which this approach works are listed below:

1. Each event in the event repositories includes at least one time feature
2. The workload profile exhibited by future events is similar to the workload profile exhibited by events of the same type in the past.

3 Proposed Hypotheses Evaluation

This section will outline the evaluation methods proposed for testing hypotheses made for the event aware prediction approach. These evaluation methods will make use of training and testing data sets from three different domains, for different event types, as shown in Table 2.
**DefinePhasesForEvent:**

1. Identify earliest and latest time features specified for the event

2. Run CalcAvgStartEndDist.

3. If avgStartDist != 0, add new earliest time feature that occurs avgStartDist time units earlier than before.

4. If avgEndDist != 0, add new latest time feature that occurs avgStartDist time units later than before.

5. Create phases for the event under consideration, that are bounded by each of the specified and derived time features.

**CalcAvgStartEndDist:**

(Calculates average distance from burst lower bound to earliest time feature and average distance from latest time feature to burst upper bound)

1. For each event in past event repository:
   
   (a) Find lower bound for event-associated burst by running gradient descent as in [8] from the earliest time feature.

   (b) Add distance from burst lower bound to earliest time feature to running total startDist.

   (c) Find upper bound for event-associated burst by running gradient descent as in [8] from the latest time feature.

   (d) Add distance from latest time feature to burst upper bound to running total endDist.

   (e) Calculate avgStartDist: average distance between burst lower bound and earliest time feature.

   (f) Calculate avgEndDist: average distance between latest time feature and burst upper bound

   (g) return avgStartDist and avgEndDist

---

**Fig. 7.** Defining Event Phases
PredictHitRateMinute i:

1. Find all active events in the upcoming event repository at minute i:

2. For each active event:
   (a) Identify the phase p for the active event that contains minute i.
   (b) Find the %Age (Ap) of phase p at minute i
   (c) Find the % hits expected at Ap of phase p using the phase workload model, and the minute interpolation equations defined in [11]
   (d) Find equivalent hit magnitude for 1% of hits in phase p based on history of the current event, or events in the past event repository.
   (e) Calculate the expected hits at minute I based on the expected % hits identified in 2 (c) and the equivalent hit magnitude per percentage discovered in 2 (d).
   (f) Add the estimated hit magnitude of 2 (e) to a running total for minute i.

Fig. 8. Predicting Workload
**PredictHitRateMinute i:**

1. Find all active events in the upcoming event repository at minute i:

2. For each active event:
   
   (a) Identify the phase p for the active event that contains minute i.

   (b) Find the %Age (Ap) of phase p at minute i

   (c) CalculateHitBuckets(i)

   (d) Identify the hit bucket that contains minute i for the current phase

   (e) Identify where on the identified bucket hits most frequently occurred based on the workload model.

   (f) If hits most frequently occurred on the current hit bucket at minute i, add a hit to the running counter for minute i. Note that more than one hit may be added to the running counter at this point, if the hit bucket only spans minute i, and the hit bucket contains more than the

**CalculateHitBuckets**

(Calculates the hit buckets at minute i for the phase of an event)

1. Assign %hits to each minute in phase lifetime, using the minute interpolation equations defined in [11] to identify how many %hits should be allocated to each minute

2. Based on past history of the current phase up to minute i, or based on the repository of past events, calculate the equivalent %hits for 1 hit magnitude.

3. Create hit buckets along the lifetime of the phase, where each bucket spans 1 magnitude hit, the first bucket starts at the first phase minute, and all subsequent buckets are contiguous. Note that a hit bucket may contain more than one hit if the bucket spans 1 minute, and contains more than the %hits equivalent to 1 hit magnitude.

---

**Fig. 9.** Predicting Workload Using the Hit-Bucket Approach
Table 2. Training and Testing Sets Used for Evaluation

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Training Set</th>
<th>Testing Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auctions</td>
<td>Bid Workload Trace (Bids/Minute) for an Auction Set including all auctions ending within a two-day period, described by an arbitrary set of keywords, which we label as K.</td>
<td>Bid Workload Trace (Bids/Minute) for an Auction set including all auctions ending within a two day period that is different to that used in the training set, and which are described by keywords K.</td>
</tr>
<tr>
<td>Product Announcements</td>
<td>Wikipedia hourly load traces (Hits/Hour) for the page of a popular tablet device, which we will call “X”, on months when new devices of type “X” were announced. When using leave-one-out cross validation, the month being predicted is left out of the training set.</td>
<td>Load (Hits/Hour), for each month when there is an announcement for a tablet “X” device, is predicted using leave-one-out cross validation.</td>
</tr>
<tr>
<td>Sporting Events</td>
<td>Soccer World Cup 1998 Load Traces (Hits/Minute) for soccer match events from June 7 1998 to July 18 1998. When using leave-one-out cross validation, we leave out 12 hours on either side of the starting time for a match for which load is predicted.</td>
<td>Load (Hits/Minute) for each 12 hours on either side of a soccer match’s starting time is predicted using leave-one-out cross-validation.</td>
</tr>
</tbody>
</table>
3.1 Reusable Event Aware Prediction Process

In our first hypothesis, we postulate that the event aware prediction process shown in Fig 2 can be re-used across multiple domains. This includes allowing for the re-use of the training and prediction phases described in Sections 2.2 and 2.3, respectively, where only a new plugin repository-loader, as described in Section 2.1 is required for each different domain.

We test this hypothesis by evaluating the performance of the generic EAP process, as described in Section 3.2 when used with three different repository loaders, for three different domains as shown in Table 2. If the burst prediction accuracy of EAP is comparable or superior across all three domains, then this hypothesis qualifies as true.

3.2 Superior Burst Prediction Accuracy of EAP

Secondly, we hypothesize that the burst prediction accuracy of the generic EAP process, outlined in Section 2, is higher than that of the most commonly and recently used competitor methods for predicting workload, as outlined in [11].

To test this hypothesis:

1. EAP, along with each of the competitor methods listed in [11] is trained using the training sets outlined in Table 2
2. The BLPAm accuracy metric [11] is reported for each method when predicting the testing sets outlined in Table 2. Leave one out cross validation is used for the product announcement and sporting event domains. A prediction horizon of 15 minutes is used for all prediction tests.

Our hypothesis qualifies as true if the reported BLPAm metric for EAP is higher than that of all other prediction methods assessed.

3.3 Superior Long Term Prediction Accuracy of EAP

In our final hypothesis we posit that the burst prediction accuracy of competitor methods discussed in [11] degrades much more quickly with an increasing prediction horizon than EAP.

To test this hypothesis:

1. We report the BLPAm metric using the process described in Section 3.2 for prediction horizons of 15 minutes, 1 hour, 24 hours, and 48 hours.
2. For each method, we report factors of accuracy degradation. A factor of accuracy degradation reflects the ratio between the prediction accuracy at 15 minutes and the prediction accuracy at any given higher prediction horizon.

This hypothesis will qualify as true if the factor of accuracy degradation of EAP is lower than that of all other methods when considering the degradation at the highest prediction horizon of 48 hours.
4 Conclusion

In this paper we have specified a generic method for event aware prediction, which is based on the EAP framework introduced in [11], and which exhibited superior burst prediction accuracy. Apart from superior burst prediction accuracy, we also hypothesise other benefits including increased prediction accuracy independence from prediction horizon when compared to other methods; as well the re-usability of EAP across multiple domains. We have outlined tests to prove each of these hypotheses, which we aim to run in future work to validate the benefits of EAP. Positive results for these tests, and agreement with existing positive results for event aware prediction [11] can mean that EAP can also be applied to a multitude of domains beyond workload prediction where event-associated burst prediction is important.

5 Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

