

Real Time Operations Management with a Stream Processing Engine

Chetan Gupta, Song Wang, Umeshwar Dayal, and Abhay Mehta
HP Labs

firstname.lastname@hp.com except for songw@hp.com

Abstract

Sustainability requires efficient management of resources. Management of large, complex systems such as a transportation network, massive data centers, etc., share a lot of common characteristics and requirements. In this paper, we identify these, point out the insufficiencies in current solutions in addressing the requirements and present some results that help meet the challenges.

1. Introduction

Real time operations management and control applications are increasingly relevant to the problems of sustainability. They have applications in domains such as managing transportation networks, managing large computer systems and massive data centers, managing electric grids, etc. These real time management problems all require a sophisticated stream management system that can process large amount of control data and take decisions based on the incoming data in real time.

In a typical operations management (OM) scenario, be it oil and gas drilling or IT operations, systems are fitted with sensors and other measuring devices which produce a constant stream of data. Since the data streams in from many sources, including raw data, and triggered events, the data is heterogeneous in nature. Human operators often man *operations centers* to monitor this streaming data. However, due to the sheer complexity and volume of the incoming data, operators often react to incidents *after* they have occurred. This leads to downtime and loss of efficiency. Moreover, due to the complexity, human operators are unable to optimize the normal running of the system itself causing an suboptimal use of resources. Since such large complex system will continue to grow and place an increasing burden on our resources (for instance, the US EPA estimates that the energy usage at data centers is expected to double every five years), it has become imperative, that we come up with techniques for efficient management of these systems. In such a scenario, systems that employ data mining techniques have a significant role to play. There is a need for systems that can detect patterns and outliers for quick automated action, filter and correlate data to reduce the volume of the incoming data, provide the operators with aggregated

view and visualizations for better insights, and provide access to historical data for root cause analysis, etc.

1.1. Life Cycle of Data

This class of applications can also be understood in terms of the life cycle of data. In Figure 1, we have

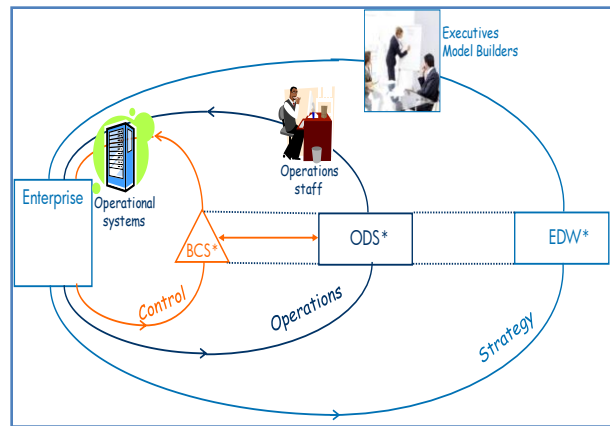


Figure 1: Control, Operations and Strategy Data Life Cycles

depicted the typical life cycle of data. The outside cycle is the *strategy cycle*, where the senior executives use historical data stored in a data warehouse (EDW) to take long term decisions, the middle cycle is *daily operations cycle* where the operations staff takes day to day decisions such as inventory management, and of our interest, there is the innermost cycle - the *control or the operations management (OM) cycle*, which is aimed at responding in real time to changes in the state of the enterprise for efficient management of resources. The requirements of all the three cycles are different and in this work we are focused on the innermost or the OM cycle.

In a typical OM scenario, data streams in from a large number of components comprising the system. These components are of course related but their interaction is not well understood and hence is difficult to model. Measurements or messages from these components are the *event instance tuples*. For instance, in oil and gas drilling where the drilling needs to be closely managed for avoiding large downtimes, data streams in from a large number of sensors measuring quantities such as temperature and pressure. These events then need to be correlated to form *complex events*. These complex events

can either be composed across the dimensions in data to get a *multidimensional event* or composed over time to get a *sequence event*. Of course, many events are *both* multidimensional and sequential in nature. For example, in the transportation domain multidimensional sequence events are a key challenge. From these inputs, the challenge is to *rapidly and appropriately respond in real time* to a complex event. These events can either be expected or unexpected, benign or harmful, slow to evolve or rapidly moving. Based on the nature of the event some control action needs to be taken. Such an action can vary from ignore, to a simple notification to a major alarm, etc.

1.2. Data Analysis and Mining

From an OM perspective, the common need in all these applications is a need for rapid analysis of the streaming data to compose complex events from raw events. The important and defining characteristics of the incoming streaming data are: input complexity, heterogeneity, volume and several abstraction levels. We need to do pattern matching and discover hidden concepts on this streaming data and suggest suitable management actions. The challenge for us data miners, is not only to come up with new data mining techniques (besides the ones that exist) that address the requirements and characteristics of such data but also to design systems in which such techniques are sufficiently integrated to arrive at a decision and a control action in real time.

1.3. Organization and Contribution

In this paper, we attempt a systematic study of the requirements for effective operations management. In Section 2, we discuss the characteristics and requirements of an OM scenario. In Section 3, we discuss existing relevant technologies and their shortcomings. In Section 4, we present the system design and a conceptual architecture for an OM application in terms of three layers, Detect, Analyze, and Respond. In Section 5 we discuss some of the algorithmic approaches we have been working on to address the challenges of a real time OM applications. Finally, we conclude in Section 6.

2. Characteristic and Requirements

Different OM scenarios share several characteristics across domains. These characteristic cause some of the key challenges to analytics required for a successful OM deployment. We define some of the most important characteristics:

- **Input Complexity:** By input complexity we mean that the system under study has hundreds if not thousands of real time inputs whose inter-relationships are not

well understood and hence difficult to model. Furthermore, several inputs are transient and erratic. In the IT OM context, an example of such an input is a human user, who chooses to test a new program causing several types of events to be published, such as memory overflow, cpu maximization, etc.

- **Heterogeneity:** The incoming data is derived from various sources, hence is heterogeneous in nature. This means that certain streams could be composed of data in real space while other could contain categorical variable. Moreover, many streams themselves are multidimensional and composed of heterogeneous data types. For example, in IT OM the data streaming in from an integration server is composed of 25 attributes comprising categorical, integer, character, and string data types.
- **Volume:** Another important characteristic is the large volume of data. An example is one of HP's smaller data centers, where every 30 seconds approximately 750 sensors send in temperature readings.
- **Hidden Concepts:** Since the streaming data is multivariate and heterogeneous, typically, a lot of concepts on which an action needs to be taken are hidden, meaning that they need to be discovered by computing some function over streaming data. This is a common characteristic of a lot of learning systems. The simplest example of such a concept in our context would be an outlier or an episode.
- **Various Abstraction levels:** Streaming data, like other data exists at various abstraction levels. These abstraction levels need to be exploited not only to discover hidden concepts but also to deal with the varying data rates in a streaming system. For instance, during high volumes of data, the data can be studied in aggregate form at a higher abstraction level, thus reducing data volumes.
- **Real Time Constraints:** In any OM system, a primary requirement is that of analyzing the streams of data and taking action in real time. Here real time would mean specifying a minimum time lag between the occurrence of an event and when an action is suggested. Once the data has been analyzed and a concept of interest has been discovered, there has to be a mechanism for taking some action in real time. The actions typically take the form of ECA (Event, Condition, Action) rules and are implemented through a rules engine.

These characteristics by themselves are not as challenging as when they exist in conjunction. For instance, the challenge is to discover the hidden concepts on large volumes of heterogeneous streaming data with high input complexity. Furthermore, the concepts need to be discovered in short amount of time so that some corrective action can be taken.

3. Current Technologies

In this section, we discuss some existing technology with reference to the characteristics and requirements discussed above. It places our work in the context of current approaches and underscores why in their basic forms they are insufficient for OM applications. Instead of reinventing the wheel, we aim to address the OM application space by extending and integrating some of these existing technologies.

We will discuss three broad areas: (i) Control Systems (ii) ETL (Extract, Transform, Load) Engines and EDWs (Enterprise Data Warehouse) (iii) DSMS (Data Stream Management Systems) and CEP (Complex Event Processing) Engine. In Figure 2, we have plotted the different system approaches on the axis of response time and stream data volume. Here the bubble size indicates the complexity of analysis possible with the technology. We explain further below:

3.1. Control Systems

A control system is usually single purpose but can serve large volumes of data efficiently. They typically model a few well understood variables such as temperature, pressure under some distributional assumptions. In our heterogeneous systems, these single variable control models are insufficient since they are not designed to handle the two important characteristics of relevance to us, *input complexity* and *heterogeneity*.

3.2. EDW and ETL

Database centric technologies such as EDW and ETL have had great success in arriving at generalized solutions across industries. These engines are designed to handle heterogeneous data from large complex systems. However, they are designed for static data. Active EDW systems conduct complex tasks through triggers for slow data updating frequency. ETL engine may serve various complexity tasks and different response requirement according to user specifications but they are not specifically designed for real time response. In other words EDWs and ETL engines cannot handle an important OM characteristic, *real time action*.

In response to applications such as financial transactions which need real time ETL, many researchers have started working on this problem [1]. However, with real time ETL the set of standard operations are a subset of what is required for a control application. For instance, traditional ETL operators are not designed for *learning concepts*. These engines will have to be enhanced with user-defined functions before they can be used as OM

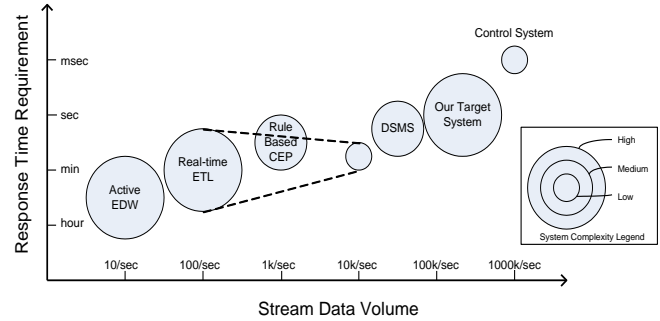


Figure 2: Existing Solution Comparison

engines for large amounts of streaming heterogeneous data.

3.3. CEP and DSMS

Complex Event Processing (CEP) [2] engines arose from expert systems and systems research. They are well designed for the responding in real time. Rule-based CEP engines can only provide medium complexity pattern matching for slower streams (several kilo tuples per second). In other words CEP engines cannot discover *hidden concepts* and exploit *abstraction levels*.

DSMS engines [3] can handle complex operations and are designed for fast streams but are still slower than a control system. Traditional DSMS are designed for continuous SQL query processing and like real time ETL do not provide the rich functionality needed for OM applications. Furthermore there is no direct mechanism for real time action.

Our best bet seems to be in enhancing DSMS with additional functionality and combining with a CEP to enable real time action. With this we introduce our target system.

4. System Solution

Referring to Figure 2, our target system needs to provide more complex service than DSMS and Control Systems, while still handling fast streams of data. To get our desired system, it is more intuitive to extend DSMS based engines rather real time ETL engines, which are primarily aimed at enabling real time DBMS transactions. This is also because, DSMS are designed for a streaming paradigm and it is relatively easier to extend them with the basic operators needed for analysis of the streaming data. Once the analysis and learning is done over streaming data, a CEP engine like functionality can be used for real time action. Hence, in our approach we extend a DSMS based engine [4] to include the required operators.

The solution to an OM application can be understood as composed of three primary components: *Detect, Analyze and Respond (DAR)*. An architecture depicting

this is shown in Figure 3. We discuss in more detail below:

4.1. Detect

By *detect* we mean an ability to process large amount of heterogeneous streaming data and identify the events in this streaming data. Detection requires both, *pattern based detection* where the patterns of interest are known *a priori* and *learning based* techniques such as outlier detection where interesting patterns are discovered in real time. The patterns detection is model based, where the models need to be aware of *concept drift*, meaning they need to evolve over time. These models can be of the form of simple threshold violations or could be sequence based. This points to three core techniques for detection: *maintaining appropriate moments* for stream for threshold violations, *sequence matching* over streams and *outlier detection*. There has been much work in moments over streams and some work on pattern matching and outlier detection. We will present solution outlines of latter two techniques, where we feel improvements are needed to achieve results suitable for OM applications.

4.2. Analyze

By *analyze* we mean an ability to process the detected events and make inference about them. The current events need to be understood in the context of historical events and other current events in the system. Consider a problem which can be stated as: “*If the number of high severity anomalies from a particular component, are increasing abnormally over time, then there’s a problem*”. To address problems such as these, we need *trend detection* whereas to contextualize the current events we need *correlation based* techniques. There has been a lot of work in statistics for trend detection over univariate data, but there is a need for techniques in multidimensional heterogeneous trend detection. We present the idea of *computational cube*, which helps in trend detection and correlating events.

4.3. Respond

The topmost layer is the *respond* layer, where by respond we mean taking appropriate action based on the events from the analyze layer. This is typically done with the help of rule based systems. There has been much work in this area with commercial products such as TIBCO and open source products such as Drools. However, the problem of managing alerts, i.e., reducing the number events on which rules are constructed from an exponentially large space of events is still an open problem. A common technique is through event correlation. Note, that this correlation is different from the one mentioned earlier, where the detected events were

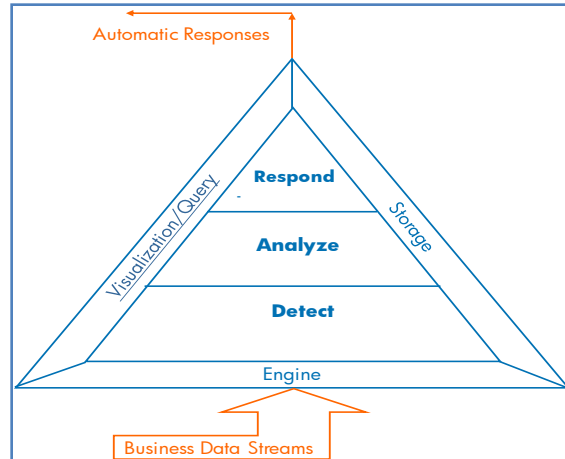


Figure 3: Solutions Components for a Control Application

being correlated for the purpose of discovering new more complex events, whereas in this layer the events are correlated to reduce the number of rules for action.

4.4. Other Layers

The other layers of interest are the infrastructure layer. They are comprised of the *storage*, *visualization* and *ad-hoc query* layer, and the *engine* layer. The visualization & ad-hoc stream query layer is used to facilitate end user control over the OM system and allows the user to ask intelligent queries (visual and SQL), which allows an operator to study a scenario and make judgments. There is a need for new techniques in visual analytics, such as correlation analysis over streaming data. The storage layer is needed for querying historical data, storing system states and *root cause analysis*. In OM scenarios, very often similar problems result in similar event types. If the events currently being seen already exist in the past, we can use similarity match with historical data to predict the root cause. In the historical data, every event sequence has a human assigned cause, which can be used to predict the behavior of current system state. The engine layer brings the different components, such as the DAR operators and the various infrastructural layers together. There are problems such as *scheduling* to be considered in this layer.

5. Analysis Solutions

We now briefly discuss three operators, which are important as analysis tools for an OM application: (i) Sequence Matching over Streams (ii) Outlier Detection and (iii) Computational Cube. Due to lack of space, there are several other data mining problems such episode detection through frequent item-set mining, synopsis construction through clustering, that we do not discuss.

As we design and introduce new operators, an important consideration is that the operators need to be designed keeping in mind the requirements and characteristics of OM system described before.

5.1. Sequence Matching over Streams

As mentioned in the requirements, in an OM application we might be seeking to match a large number of patterns over streaming heterogeneous data. Furthermore, these patterns could be hidden at different abstraction levels. For instance, in looking at traffic patterns, we could study them at the street level, at a zip code level, a city level and so on. We need an approach that can efficiently match the large number of patterns at different abstraction levels and over streaming data. In this context, given a workload of pattern queries, based on the interrelationships in terms of both concept and pattern refinement among the queries, we have constructed an approach to compose the queries into an integrated *event pattern query hierarchy*. To do this we (i) We define the concepts needed to build a meaningful event pattern hierarchy that integrates both pattern and concepts and study the interrelationships in terms of both concept and pattern refinement relationships among a workload of pattern queries, (ii) Based on the analysis of different pattern refinement relationships, we identify several forms of re-use of partial results from one pattern query to form the results of a related pattern query (iii) We propose and evaluate a family of pattern evaluation strategies for online operational decision making: Bottom up, Top Down and Hybrid.

5.2. Outlier Detection

In an OM application finding outliers is critical since they indicate a change in state. These outliers need to be detected efficiently over streaming heterogeneous data. Furthermore, the outliers may exist at an arbitrary set of dimensions over different abstraction levels. We have constructed a framework in which multiple users can query streaming data for outliers over an arbitrary set of dimensions at various abstraction levels. Each user creates a continuous query specifying the dimensions she is interested in, and the sliding window size over which the outliers need to be computed. The results of each query are a continuously updated list of outliers along with some measure of each outlier's "outlierliness". For example, in IT OM, where various hardware and software components are issuing events, some user might run a query primarily concerned with outlier alerts from hardware components, whereas some other user might be primarily interested in the software components. These could obviously, share dimensions such as severity, creation time, etc. Furthermore, for exploration on streaming data, such flexibility in composing various

dimensions is required, since unlike static data, a user does not have the opportunity to do pre-analysis and decide on the best set of dimensions on which to find outliers.

5.3. Computational Cube

Finally, for generic multidimensional, multi-abstraction analysis over streaming data, we have extended the OLAP cube for streaming analysis. This data cube forms the core of real-time control analysis and is called the *Computational Stream Analysis Cube* (computational cube for short). The proposed computational cube provides following important attributes for stream processing: 1. *Accuracy hierarchy*. Applications can require different levels of approximation of every snapshot of the stream data. 2. *Window-based aggregations*. On the time domain, aggregations over arbitrary window constraints are supported for sequential analysis. 3. *Multi-Dimensional data*. We also inherit the multi-dimensional data cube of traditional OLAP.

6. Conclusions

In this extended abstract, we have discussed a key problem towards achieving sustainability of large complex systems such as large compute installation, transportation networks, etc.: The problem of real time operations management. This means analyzing large amount of heterogeneous streaming data and taking real time action. This action could either be based on pre-defined concepts or discovered concepts. We have identified the general characteristics and requirements of such applications and discussed the state of the art and its shortcomings. Finally, we propose outlines of new solutions that can help overcome these shortcomings.

6. References

- [1] R. M. Bruckner, B. List, J. Schiefer, Striving towards Near Real-Time Data Integration for Data Warehouses, In Proc. 4th Int. Con. on Data Warehousing and Knowledge Discovery, Eds. LNCS, vol. 2454, 317-326.
- [2] B. Babcock, S. Babu, M. Datar, R. Motwani, Widom J., Models and issues in data stream systems, ACM PODS, 2002, June, pp 1-16.
- [3] R. S. Barga, J. Goldstein, M. H. Ali, M. Hong, Consistent Streaming Through Time: A Vision for Event Stream Processing. CIDR 2007, pp. 363-374.
- [4] C. Gupta, et. all, CHAOS: A Data Stream Analysis Architecture for Enterprise Applications, IEEE CEC 09, to appear.