Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 6 (2011) pp. 398–406

A Formal Temporal Log Data Model for the Global Synchronized Virtual Machine Environment

Sean Thorpe[1], Indrajit Ray[2], Indrakshi Ray[3], Tyrone Grandison[4]

[1] Faculty of Engineering and Computing, University of Technology
Kingston, Jamaica
sthorpe@utech.edu.jm

[2] Department of Computer Science, Colorado State University,
Fort Collins, USA
indrajit@cs.colostate.edu

[3] Department of Computer Science, Colorado State University,
Fort Collins, USA
iray@cs.colostate.edu

[4] IBM Research
Yorktown Heights, NY, USA,
tyroneg@us.ibm.com

***Abstract*: The use of timestamps is fundamental to the management of time varying information and arguably it may be even more important for the synchronization of the virtual machine (VM) log data sets. In the context of managing (VM) logs for transactional database activity, the consistency of its state can be evaluated by these timestamps. Temporal data models claim to be point based whereas other temporal models are interval based. Hence the premise for synchronization as a component of a time event has become critical to a distributed hybrid compute cloud. The contributions of this paper apply the use of formal temporal mechanisms to appreciate the behaviour of our case study deployment. In our study we design a software application called a global virtual machine log auditor. We use the auditor to synchronize virtual server log events across a suite of non native VM environments in distinct time-zones. This work is useful in managing cloud data migration and synchronization across these time zones. Our implementation uses a snapshot equivalent approach to monitor the synchronized log events on these VMs. In this context the paper precisely defines the notions of point based and interval based temporal data models as the application of the case scenario, thus providing a new and formal basis for characterizing such models within the cloud computing environment. This paper's motivation is an adoption of earlier work done [1, 4 15, 21].**

***Keywords*:** timestamps, interval logs, point, cloud, temporal

I. Introduction

Temporal data models include timestamp attributes in their relation schemas and give special semantics to the values of these attributes in their query languages. Virtually all data models intended for practical use employ some form of intervals for their timestamp values. Unfortunately, It is generally impractical to record individually all the time points for a distributed virtual machine database. For the purposes of our ongoing work [1] we manage and archive system event logs over periodic intervals as a function of the timestamps.

Intervals may simply be employed for reasons of practicality, i.e. as syntactical shorthands for time points [12]. Thus, referring to a data model as interval-based simply if it employs interval timestamps bears little real significance. It says little about the qualities of the data model. Rather for our synchronized VM log environment, the notion of point and interval based data model must be defined on a semantic level. The questions then what are the real defining properties of point and interval-based data models as a function of the synchronized temporal data model for the VM environment. This paper provides an answer to this question.

To get a real feel for the range of possible semantics of time data models, it is instructive to consider a simple example adopted from Bohlen et. al. [21]. We assume that the two tuple time-stamped relations r_1 and r_2, below are given and consider possible definitions of the temporal difference of these two relations, $r_1 -^T r_2$.

r_1 :

A	TS
a	[1,10]
a	[11,20]
a	[21,30]

r_2 :

A	TS
a	[5,15]

When we want to construct a difference operator between both relations, there are four possible definitions-: R_1 through to R_4.

R_1 :

A	TS
a	[1,4]
a	[16,30]

R_2 :

A	TS
a	[1,10]
a	[11,20]
a	[21,30]

R_3 :

A	TS
a	[21,30]

R_4 :

A	TS
a	[1,4]
a	[16,20]
a	[21,30]

The first result, R_1, contains the times associated with value a in r_1 that are not associated with value a in r_2. This result is consistent with the perception that intervals are abbreviations for time points, and nothing more. Thus, the first definition has a point based feel to it.

The first result may also be characterized as being coalesced. In coalescing, value equivalent tuples (tuples with identical non temporal attribute values) with adjacent or overlapping intervals are replaced by a single tuple with the same non temporal attribute values and an interval that is the union of the intervals of the original tuples.

In contrast, result R_2 is far from being point based in nature. This result contains all tuples in r_1 not in r_2. This definition of difference simply considers the intervals atomic values. Thus, it may be said to "respect" the actual intervals given to the tallest may even be questioned if the operator is temporal at all it is simply the standard set theoretical difference operator. Result R_3 returns tuples from r_1 with intervals that do not overlap with intervals of tuples in r_2. The utility of a temporal difference operator of this kind appears questionable.

The last result is similar to the first one: it also contains the times associated with value a in r_1 that are not associated with value a in r_2 . Put precisely, R_1 and R_4 are snapshot equivalent [12] . The difference is that the second tuple in R_1 is "represented" by two tuples in R_4. In other words R_1 is the coalesced version of R_4. The idea behind this definition is to be point-based while also trying to respect the intervals associated with the tuples in the argument relations.

It is our contention that R_1 and R_4 are results of point-based operations and that R_2 and R_4 are results of interval based operations. The operation yielding result R_3 is thus neither point based nor interval based.

This paper represents a first look at how we can use point-based and interval-based data models to formally represent timestamp associations for a synchronized VM environment. We seek to make such evaluations by applying these definitions to the occurrence of the VM System log events. Other than for traditional temporal databases [21], to the knowledge of the authors, no papers have previously been devoted to address the issues here as a part of the cloud computing environment. In our virtual machine compute cloud the system logs are relational database tables and form apart of the VM active database environment.

As more enterprises seek to capitalize on the economies of scale and efficiencies of virtual machine compute clouds, there will undoubtedly be an increase in malicious activity as enterprising people recognize the greater opportunities for the exploitation of the security risks inherent in trusting virtual data centres to third party providers that one has no physical jurisdiction over. This security challenge overlaps with the fact that the forensics community also shares its own concerns around auditing, searching, and providing traceable digital footprint analysis for victims of miscreant behaviour within this abstract domain. This is particularly true in that a VM object within a data centre may be subject to several eventualities through network distribution before reaching its final user(s).

On the premise of these eventualities, our work introduces the need to look at log event behaviour as an association of its timestamp information in order to enable effective auditing.

Snodgrass [12] advocates that "A temporal query language should have a canonical model, in which relations are identical if and only if all their snapshots are identical". Chomicki states that "It is important to see that the data model of TQuel is point-based, not interval-based. Intervals serve as a representational device. The truth value of facts are associated with points, not intervals" [5] and that a model is point-based if the facts are associated with single time points, and interval-based if they are associated with intervals (represented as pairs of points) [6, 18]. In this paper we establish the correspondence between the point-based and interval-based views as adopted from traditional temporal database theory and the corresponding first order temporal languages. This correspondence shows that all first order queries can be conveniently asked using point-based query language and then mechanically translated to an interval based query language. [17].

The relevant literature reveals that different researchers perceive the notions of point-based and interval-based data models quite differently as it relates to the creation of timestamp associations. In particular, the notion of interval-based data model remains to be given a formal definition for the synchronized time bound virtual machine.

In the next section we further motivate the topic and explore the problem space. Section III introduces the notions of VM temporal data models and time domains, providing the basis for formally defining the notions of point-based and interval-based temporal data models in Section IV and V. Section VI looks at an evaluative case study discussion, and Section VII provides the conclusion and future work.

II. Motivation and Problem Space

When asking queries on a temporal database, the results may vary depending on whether or not argument relations are coalesced. For example, this is the case for selections with

predicates that involve the arguments' timestamps. To see this, consider two relations in Table 1. The relation at the top is uncoalesced whereas the one at the bottom is the corresponding coalesced relation. The column value TS denotes the timestamp values.

Adopted from Bohlen et. al. [4, 21] we consider the uncoalesced Employment relation examples as an ideal set of baseline reference cases for understanding our own work on VM System Event Log relations. The Employment relation models job contracts in a company with temporary positions only. The query π Name, start (TS) $^{\text{Employment}}$ returns the start time of an interval. If the exact same query is evaluated over the coalesced instance, only two tuples are returned.

The example illustrates that there exists queries than can be asked over the uncoalesced instances, but not over the coalesced ones. For example, the coalesced instance of the Employment relation doesn't reveal that Lars signed two contracts, let alone when he signed the second one.

On the other hand, it is impossible to come up with a query that can be answered over the coalesced , but not over the uncoalesced instance. The answer for this is simply the fact that we can derive the coalesced relation instance from an uncoalesced one. e.g. using a regular SQL statement. [21].

Employment

Name	Position	TS
Lars	Programmer	92/01/01,94/12/31
Lars	Programmer	95/01/01,96/12/31
Niels	Accountant	92/01/01,96/12/31

Employment

Name	Position	TS
Lars	Programmer	92/01/01,96/12/31
Niels	Accountant	92/01/01, 96/12/31

Table 1. Uncoalesced and Corresponding Coalesced Relation Instance [21]

These considerations indicate that a model that is able to tell coalesced and uncoalesced relation instances apart (and in this sense is interval-based) is in some sense more powerful that a model that cannot tell them apart, i.e. a point-based model. In our own work of synchronizing the relational log tables for a virtual machine this is a critical concern. In the next subsection we explore this difference even further.

A. Data Modelling

The relative expressiveness of data models that do or do not differentiate between coalesced and uncoalesced relation instances may also involve data modelling. It may be argued that if the database schema is designed appropriately, it is possible to answer the same queries using a coalesced model as can be answered by an uncoalesced model even within the VM log database.

For example, if individual contracts are important, which is not likely, we can record their unique numbers in the Employment relation, as shown in Table 2.

Name	Position	ContrId	TS
Lars	Programmer	1091	[92/01/01,94/12/31]
Lars	Programmer	2154	[95/01/01,96/12/31]
Niels	Accountant	1095	[92/01/01,96/12/31]

Table 2. Alternative Modelling of Employment [21]

This way, it may be possible to "compensate" for the lack of uncoalesced relations in a point-based data model. It may be argued that it is quite natural that certain queries cannot be answered if they were not anticipated when the database was designed - this is true for any database.

Equally introducing additional attributes may sometimes have subtle drawbacks not experienced if the attributes could be omitted because the data model allowed uncoalesced relations. For example, we might introduce dependencies (contract numbers increase over time) or we might not be able to faithfully represent our mini-world ("new follow up contracts with the same contract identifier) .

B. Query Processing Optimization

The point versus interval basis of a query language also affects query processing and query optimization. For an interval-based language, care has to be taken that processing and optimization strategies respect the interval-based semantics, which can be quite complex. This severely restricts the possibilities to manipulate and transform intervals. In contrast, specific timestamps may be modified (as long as snapshot equivalence is preserved) in a point-based language, allowing the database system a choice of timestamps among several alternatives. This indicates that an interval-based language leaves less possibilities for query optimization and, thus, efficient evaluation strategies.

In favour of an interval-based language, it can be said that a point-based database system must guarantee that the result of queries do not depend on a specific choice of timestamp values. This guarantee is met by performing coalescing operations, which can be expensive [4]. While it is possible to sometimes eliminate coalescing during query optimization, there remains situations where coalescing has to be performed [13].

III. VM Temporal Log Data Models and Time Domains

For a Synchronized data model as a VM log compute cloud, we should not forget that these VMs must map to a physical machine and hence to a physical data set of structures. Against this background, we adopt (from [4, 5, 21]) that our VM relational data model, be a relational data Model M = (D, A). This is a relational composition of sets of data structures, D and a set of algebraic operations defined on the set of data structures. A VM temporal

relational data model is a relational data model that has temporal relations as the underlying data type, and whose operators are all temporal.

A VM log temporal relation includes a VM log temporal tuple attribute. The exact denotation of this attribute is not important for this paper, but for simplicity we assume that it denotes the synchronized VM log tuple's valid time i.e. when the information recorded by the remaining attributes of a tuple is true in the modelled reality. Synchronized VM log tuples of temporal relations may therefore be put under the form (x_1 ……x_n || ts) or as (x || ts) when the number of attributes is immaterial. We term x_1 ……x_n the non temporal (or explicit) attribute values, and ts is the tuple timestamp. A finite set of such relations may be referred to as the timestamp representation of a temporal database [2].

An operator is temporal iff it generates a temporal relation when applied to temporal relations. When designing a synchronized VM temporal data model, an important and central aspect is the choice of appropriate timestamps of database facts. Time points and time intervals, defined below; provide the most common choices [2, 4, 15]

Definition 1: (VM Time point and VM Time interval Domains)
Let T_v be an infinite VM log tuple set.

1. $T^p = (T_v , <)$ is a VM log time point domain over T_v iff $<$ defines the total order on T_v. Each element of T_v corresponds to a time point on T^p .
2. A time interval I of T^p is any connected subset of T^p , i.e. $(p_1 \in I \wedge p_2 \in I \wedge p_3 \in T_v \wedge p_1 < p_3 < p_2) \Rightarrow p_3 \in I$
3. $T^i = (\Gamma , C)$ is the VM time interval domain over T^p iff Γ is the set of all time intervals of T^p . Both T^p and T^i are time (or temporal) domains over T_v.
4. A timestamp over T^p is either a time point or a time interval of T^p .

A VM temporal log relation r whose tuples are all timestamped with either time points or time intervals of a time point domain T^p represents a temporal relation over T^p . When the timestamps are point-[intervals], r may be referred to as point - *[interval] timestamped (temporal)* relation over T^p .

If M = (D,A) is a VM temporal log data model such that D is a set of VM temporal log relations over T^p , then T^p is the time point domain of M.

In general, since time intervals are sets of VM log time points, it is not always clear in what sense the usage of timestamps differs from the usage of points. To exemplify this, assume that the integers with the $<$ order is our VM temporal log domain. Then it seems more than reasonable to claim that the relations r_1 = {(a ||2), (a||3),(a||4) } and r_2 = {(a || [2,4])} have the same information contents, i.e. that (a) is valid at instants 2,3,4, and nothing more. This assumption is nonetheless incorrect for r_2 because intervals in addition to being points also are uniquely delimited by start and end points, which may or may not be part of the interval. Hence we would timestamp a log tuple such as (a) with intervals if the end points bear some meaning, and use time points as timestamps if the notion of end points is meaningless.

Predicates and operations for points and intervals are described in almost all definitions of temporal data models [13]. Some interval predicates and operators apply just to interval data models; their properties would make them meaningless in a point-based framework. For example, the operators start and end that retrieve the initial and final instants of an interval could not be defined for a point-based database.

The point timestamp representation of a temporal database is infeasible from the storage viewpoint for all but the simplest temporal relations, so intervals are used as an abbreviation for sets of points for practical reasons. For example, relation r_1 above may be represented by r_2. Whether an interval is an abbreviation for a set of points or not depends on the operators of the data model. Only if the point contents of the output of a temporal operator remain invariant for sets of argument relations with the same point contents, it is possible to consider intervals as abbreviations for sets of points. We explore this notion of time point models below.

IV. VM Point Based Data Models

It would be easy to decide whether or not a data model is point-based or interval-based if this could always be determined by inspecting the data type of timestamps used. However, syntactic criteria are available only for simple point timestamped relations. The major difficulty concerns relations involving intervals as timestamps. This section defines the notion of a VM point based log data model.

In a point-based data model, two interval time-stamped relations that correspond to the same point timestamped relation are considered equivalent, in the sense that they record the same information. The notion of snapshot equivalence [7, 9, 21] formalizes this:

Definition 2: (Snapshot Equivalence)
Let $T^p = (T_v , <)$ be a VM log time point domain.

1. The time slice operator τ_p for a VM log time point p $\in T_v$ maps an interval timestamped relation over T^p to a non temporal one , and is defined as
   $\tau_p (r) = r^l$ iff
   $\forall x (\exists I ((x \| I) \in r \wedge p \in I) \Leftrightarrow (x) \in r^l)$
2. Two VM interval timestamped relations over T^p, r_1 and r_2 are snapshot equivalent i.e. $r_1 =^p r_2$, iff
   $\forall p (p \in T \Rightarrow \tau_p (r_1) = \tau_p (r_2))$

The notion of a VM snapshot log equivalence allows us to characterize operators that, when applied to snapshot equivalent relations, also yields snapshot results [9]. Arguably one could contend that such operators are faithful to the point based nature of timestamps of their argument relations, and we use them to define these point-based data models.

Definition 3: (**Point-based Operator**)
Let O be a n-ary operator on interval time stamped relations , and r_1 ……r_n) and ($r_1^{\ 1}$ ……$r_n^{\ 1}$)

Example 1 - The temporal log intersection natural join (∞^t) is a binary operator. Two argument tuples with identical explicit join attribute values contribute to the result if their timestamps overlap. Timestamps of result tuples are the intersections of the timestamps of argument tuples. Thus if r_1 = { (a||[2,5]), (a||[7,11]) } and r_2 = { (a||[3,9])} then $r_1 \infty^t$ r_2 = {(a||[3,5]), (a||[7,9]) }. It can be shown that this operator preserves snapshot equivalence, hence it is point -based.

Example 2 - The VM log coalescing operator (vlcoal) is a unary operator that merges value equivalent logtuples (i.e. VM log-tuples with mutually identical explicit attribute values) if the union of their timestamps is an interval. The merged log-tuple then has this union as its new timestamp. Thus, if r_1 = { (a||[3,9]), (a||[4,13]) } then vlcoal (r_1) = (a||[4,13]). Like temporal intersection natural join, this operation is point-based because snapshot equivalent arguments yield snapshot equivalent results; for snapshot equivalent arguments, the result will always be the exact same, which is a trivial case of snapshot equivalence. With the definition above, we are in a position to define point-based data models.

Definition 4: (Point-based VM temporal log data model)
A VM temporal data model M = (D, A) with time point domain T^p is point-based iff the following conditions are met.

1. D is entirely composed of either point or interval time-stamped relations over T^p , and
2. The operators of A are all point -based.

Lemma 1 A VM temporal log data model M = (D,A) is point-based iff , for every operator O of A. O(r_1 …….r_n) = ^{p} O(vlcoal(r_1) ..... vlcoal(r_n))
where r_1 …….r_n are log relations of D that satisfy the preconditions of O.

The lemma illustrates why the start and end functions mentioned in the previous section cannot be defined in a point-based model by considering individual intervals in isolation. The presence of the VM system event log tuple (x||[a,b]) in a point-based relation does not mean that a is the first time point associated with x, since the relation may contain other value - equivalent tuples that overlap with this interval. As a result, the computation of the above functions in a point-based data model requires that the argument relation first be coalesced. The definition of start could then be expressed as follows:

$$(x||I) \in r \wedge (x||\ I^1) \ \in \text{vlcoal}(r) \wedge I \subseteq I^I \ \Rightarrow \text{start}((x \ ||\ I), r) = \text{start}(I^1)$$

Finally, in a VM Log point-based data model, it holds true that intervals are nothing but abbreviations for sets of points. Hence, it is always possible to translate any interval time-stamped relation r into a corresponding point time-stamped relation r^p .

The following relationship holds between the two relations.

$(x \ ||y) \in r^p$ iff $\exists I(y \in I \ \wedge \ (x \ || \ I)$ in r by y a tuple (x||y) for exactly each time point $y \in I$.

V. Interval based VM Log data Models

It is well known that point-based and non point-based models are orthogonal. For example, suppose we assume an operator of an interval-based VM log data model needs not be point-based, but there are operators of such models that are point-based.

To define the notion of an interval-based VM log data model, we distinguish between the algebraic operators that are *timestamp Log preserving* and those that are *timestamp Log transforming*. The former operators are unproblematic and easily qualify as interval-based. The latter operators must satisfy additional properties to qualify for the interval-based status.

Specifically, when intervals are adopted as timestamps, there will normally be several ways of time-stamping resultant relations. In such cases, the argument interval timestamps should be preserved as best as possible. This is to suggest that whenever an operation requires the modification of an argument interval timestamp, the resulting interval should be the one that maximally takes the argument interval into consideration. Alternatively, this property could be stated as the largest possible fragments of the argument interval timestamps should be preserved in the result. In the next subsection we formalize these notions.

A. VM log Interval Based Requirements

The first step is to define the notion of the minimum requirements for an algebraic VM temporal operator. Informally, the minimum requirements define the set of time points that the timestamps of the result of a temporal operator must include. Explanations follow the formal definition.

Definition 5: (Minimum Requirements)
Let M = (D,A) be a VM temporal log data model with a time point domain T^p , where D is the set of interval time-stamped relations. The minimum requirements for n-ary temporal operator is a formula of the form ϕ (r_1 ……r_n , x,A) where

1. The log timestamp A associated with a result tuple that satisfies the requirements for the argument relations r_1 …… r_n $\in$ D is a set of (not necessarily connected) set of time points of T^p and
2. ϕ (r_1 ……r_n , x,A_1)$\wedge$ ϕ (r_1 ……r_n , x,A_2))$\Rightarrow$ A_1 = A_2

Clearly, ϕ must also include the preconditions for the specified operator. From the second condition of the definition, it follows that, for each sequence of explicit attribute values of x of the result, there is one and only one associated set of instances of A, since the minimum

requirements do not impose any partition on this set of instances (i.e. ϕ defines a partial , parameterized function fr_1 ……..fr_n such that fr_1 ……..fr_n (x) = A). Thus formula ϕ specifies a family of operators, in the sense that A may be (usually) split into a list of intervals in several distinct ways. We use A to emphasize that we are dealing with generic sets of instances, i.e., temporal elements , rather than with intervals only.

The next step towards defining interval-based data log models is to characterize the set of relevant argument tuples for each particular result tuple, as defined by the minimum requirements ϕ for an operator. A set of argument tuples S is relevant for a particular result tuple (x, A) iff both x and A can be entirely determined from S, but not from any proper subset.

Definition 6: (Relevant Argument Tuples)
Let M = (D, A) be a VM temporal log data model with a time point domain T^p , where D is the set of interval time-stamped relations. Let ϕ denote the minimum requirements for n-ary VM log temporal operator (r_1 ……r_n) be temporal relations of D, A be a set of time points of T^p , and x be a finite sequence of attribute values. S is a set of relevant argument tuples w.r.t. ϕ for the argument relations r_1 ……r_n and the result tuple (x,A) i.e. , relevant (x,A,S, ϕ, r_1 ……r_n) iff
$\varnothing(r_1 \ldots\ldots r_n\ x,A) \wedge \exists r^{l}_{1} \ldots\ldots\ldots r^{l}_{n}\ (r^{l}_{1} \subseteq r_1 \wedge \ldots\ldots\ r^{l}_{n} \subseteq r_n \wedge S = U^{n}_{i=1}\ r_i^{\ l} \wedge \phi\ r^{l}_{1} \ldots\ldots r^{l}_{n}\ x, A) \wedge \forall r^{ll}_{1} \ldots\ldots\ldots r^{ll}_{n}\ ((r^{ll}_{1} \subseteq r^{l}_{1} \wedge \ldots. r^{ll}_{n} \subseteq r^{l}_{n} \wedge U^{n}_{i=1}\ r_i^{\ ll} \subset S) \Rightarrow \neg\phi\ (r^{ll}_{1} \ldots\ldots\ldots r^{ll}_{n}\ , x, A))$.

Note that S does not necessarily correspond to a relation, since r_1 …….r_n may not be union compatible. Also, it is necessary to require that the result (x, A) satisfy the minimum requirements for both the original argument relations and their restricted forms because the one does not imply the other.

Example 3 - Lets assume a temporal difference operator. Let the integers with the < order be the underlying time point domain , and $\varnothing^D$ denote the minimum requirements for this operator. Assume r_1 = {(a||[2,10])} and r_2 = {(a||[1,4]), (a||[8,11]), (a||[12,17]) , (b||[2,6])}. If $r_1^{\ ll} = r_1^{\ l} = r_1$, r^{l}_{2} = {(a||[1,4]})and $r_2^{\ ll}$ = {(a||[1,4]}, (a||[8,11])} then

ϕ^D (r_1 , r_2 , (a),, {5,6,7})
ϕ^D (r^{l}_{1} , r^{l}_{2} , (a),, {5,6,7,8,9,10})
ϕ^D (r^{ll}_{1}, r^{ll}_{2} , (a),, {5,6,7})

The set of relevant argument tuples for the result tuple is $S^{ii} = r^{ii}_{1} \cup r^{ii}_{2}$. No proper subset of S^{ii} satisfies the minimum requirements.

The next example illustrates the set of relevant argument tuples is not uniquely defined.

Example 4 - The minimum requirements for a suggested VM Log temporal difference are given by the formula ϕ, union compatible (r_1, r) $\wedge$ $\forall p(\exists I_1((x\|\| I_1) \in r_1 \wedge p \in I_1 \wedge \forall I_2((x \| I_2) \in r_2 \Rightarrow p \notin I_2)) \Leftrightarrow p \in A)$

Assume r_1 = {(a||[4,8]), (a||[1,6]), (a||[7,10]) } and r_2 = {(a||[1,3]), (a||[9,10])}. Then A = { 4,5,6,7,8} satisfies the minimum requirements for temporal difference for the explicit attribute (a) and the input relations r_1 and r_2 . Concerning the relevant argument tuples of r_1 and r_2 for (a), there are two sets to satisfy the definition , S_1 = { (a||[4,8])} $\cup$ { } and S_2 = {(a||[1,6]), (a||[7,10]) $\cup$ (a||[1,3]) , (a||[9,10]) }. Note that $r_1 \cup r_2$ does not qualify as a set of relevant argument tuples, since there are sub-relations of r_1 and r_2 whose union also satisfies ϕ

A final auxiliary concept concerns the notion of maximal interval partition which determines the decomposition of a set of time points into the least possible number of non overlapping intervals.

Definition 7: (Maximal Interval Partition)
Let A be a (doubly bounded) set of time points. The maximal interval partition for A is a sequence of intervals I_1 ……. I_n such that
1. $I_1 \cup \ldots\ldots \cup I_n = A$
2. $I_i \cap I_j = \phi$ with $i \neq j$, $1 \leq i \geq n$, $1 \leq j \geq n$ and
3. Any other interval partition I^{l}_{1} ……. I^{l}_{P} for a satisfying the two above conditions is such that $p > n$.

Using the concepts developed so far, we can now define the notion of a VM log interval based operator.

Definition 8: (VM Log Interval based operator)
Let M = (D, A) be a VM temporal log data model with time point domain T^p ,where D is a set of log interval timestamped relations. Let ϕ denote the minimum requirements for a n-ary log temporal operator. A temporal operator O $\in$ A that satisfies ϕ is log interval based iff for any argument temporal relations r_1 ……$r_n \in D$, the following holds.

1. $\exists I((x\|\| I) \in O(r_1 \ldots\ldots r_n)) \Rightarrow \phi(r_1 \ldots\ldots\ldots\ldots r_n)) \Rightarrow \phi(r_1 \ldots\ldots\ldots\ldots r_n , x , U\pi_{ts}(\sigma_{expl=x}\ O(r_1 \ldots\ldots r_n)))$

2. If (a) $\phi(r_1 \ldots\ldots r_n , x , A)$, (b) $S = U^{P}_{i=1}\ S_i$, where relevant (x, A, S_i , ϕ , r_1 …….r_n), for all i, $1 \leq i \geq p$, (c) (y || I) $\in$ S , (d) A $\cap$ I $\neq \varnothing$ and (e) I_1 ……. I_m correspond to the maximal interval partition for A $\cap$ I, then (x||I_1) …….. (x||I_m) $\in O(r_1 \ldots\ldots r_n)$.

The first condition of Definition 8 ensures that for each group of synchronized log tuples of $O(r_1 \ldots\ldots r_n)$ whose explicit attributes values are x, the union of all timestamps of such tuples is identical to the set of time points identified by the minimum requirements for the same argument i.e. specified and resulting timestamps must be extensionally identical for x. The second condition ensures the preservation of the relevant input intervals in the result, whenever possible, under the form of overlapping fragments as in one (1) above.

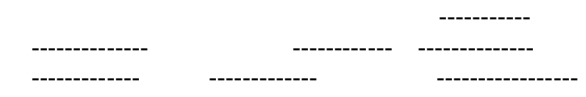


Figure 3. Hypothetical Interval Based Operator [21]

The preservation of argument timestamp fragments in an interval-based operator is illustrated in Figure 3. For the relevant argument intervals and the corresponding hypothetical set of output points A given in Figure 3, two sets of output intervals are given. The first one is built on top of a minimal decomposition strategy, where each interval of the result must be contained in one of the relevant input intervals, but no output intervals may overlap , even when there is overlapping at the input level. The second solution is the only one that satisfies all conditions of Definition 8: for each relevant input interval, its intersection with A, represented under the form of (maximal) intervals, is included in the output. In particular note that definition 8 does not allow intervals to be chopped or merged.

Definition 9: (**Interval-based VM temporal log Data Model**)
A temporal log data model M = (D,A) with time point domain T^p is interval based iff the following conditions are met.

1. D is entirely composed of interval time-stamped relations over T^p , and
2. The operators of A are all interval based.

VI. Discussion

In this section we discuss the properties of the point-based and interval-based data models in the context of our case study application within the University environment. We start by looking on the scope of our approach within the context of the case study. Then we look at mixed data models i.e. models that are neither point-based nor interval-based, and finally we evaluate representative temporal data models.

A. Scope of Our Approach

At the University of Technology (UTECH) we design a software application called a VM log auditor used to provide support to the system administration and access control of the virtual server environment. We attempt to achieve this by enabling the log auditor to synchronize the logs between the virtual machines (VM) and the physical hard disk on which these VMs are run. The log auditor maps the disk logs by transforming these logs to its Oracle 11g back end database. An ftp session is maintained between the production environment VM logs resident and the Storage Area Network (SAN) disks and the log auditor's database. The transformation mapping techniques are highlighted in separate work [19, 20].

Our current prototype uses ftp sessions at intervals over different points in time. We use these interval markers at the different time points as a snapshot equivalent of the dynamic environment on which to perform a typical log mining task of actual system events native to these VMs. The VM environment of our choice is VMware essx3i. When a virtual machine is first powered on, it sets the virtual machine's time (in the basic input/output system BIOS of the VM) to that of the time from the running ESX host. Assuming the virtual machine is part of a Windows domain; Windows will also attempt to synchronize the virtual machines clock with the domain so long as its current time is within the drift policy of your domains NTP settings.

Therefore, we find it best practice to synchronize each ESX host with the system's domain controllers and that your domain controllers are synchronized with a peer that synchronizes with an outside source. This will ensure not only accurate time throughout your domain, but that the VM's clock does not skew from the domains time, between the time they boot-up and the time they get logged into the domain. Not withstanding ongoing work extends the scope of the case study context to evaluate other cloud domains like Citrix's XenAppServers and Amazon's Elastic Clouds within different time zones.

In the above context, the scope of the definitions of point-based and interval- based operators are temporal extensions of relational algebra operators. i.e. temporal variants of σ, π. \, x, and their derivatives. These are basic operators of a temporal algebra, and they have been investigated in almost all temporal data models. Our definitions can be used to evaluate and classify these operators and models. However the definitions are applicable to all possible temporal operators. For example we have illustrated the application of coalescing earlier in this paper.

B. Mixed Data Model

With point-based and interval-based models being orthogonal we get four classes of operators. Specifically, coalescing is point-based but not interval-based, temporal selection is interval-based but not point-based, temporal intersection join is point-based and interval-based, and the regular time shift operator is neither point-based nor interval-based.

From definitions 4 and 9 it follows that there exists, for the VM environment, temporal data models that are neither point nor interval-based. In practice, we expect many models to have point-based and interval-based operations, which for the purposes of this discussion we will describe as mixed models.

C. An Evaluation of Temporal log data models

In this section we introduce a few popular and traditional temporal data models and incorporate them as an evaluation of our criteria for the VM environment.

Note that we only consider proper temporal algebraic operators i.e. operators that take temporal relations as arguments and return a temporal relation.

SQL-92 [11] extended with an interval data type is based on the relational algebra and treats intervals as atomic values without any special temporal semantics. This means that all operators are time fragment preserving. Therefore, SQL-92 is an interval-based data model. It also follows that SQL-92 is not point based.

IXQSL [10] operators are timestamp preserving because they inherit the standard SQL-92 semantics. In addition, IXSQL provides normalized and non-normalized operations in order to convert between time points and intervals. These special operations are point-based, but not interval-based, snapshot equivalence is preserved, but interval fragments are not. Thus IXSQL is a mixed data model.

TSQL2 [13], unlike the two previous models, employs a temporal algebra that gives a special meaning to timestamps. It was one of the design goals of TSQL2 to make the format of timestamps irrelevant. This is achieved by enforcing a canonical representation based on temporal elements. Thus, clearly TSQL2 is not interval-based. On the other hand, all operators preserve snapshot equivalence because they are defined over the canonical representation of a database. This makes TSQL2 a point-based data model.

ATSQL [3] introduces sequenced and non-sequenced statements together with corresponding algebras. Non-sequenced statements provide the power of regular SQL-92 statements and are, like SQL-92 and IXSQL statements, interval-based. Sequenced statements are also interval-based. In addition, most sequenced statements are point-based. Coalescing is available to enforce a canonical representation of snapshot equivalent relations. Thus, while clearly interval-based in nature, ATSQL has also a non interval-based operation (i.e. coalescing), which makes it a mixed data model. Temporal Logic [5] is point-based; as the temporal domain consists of points.

VII. Conclusions and Future Work

We have motivated the argument that point-based and interval-based operators are to be applied within a synchronized log event model on the virtual machine environment. Point-based operators are defined using the notion of snapshot equivalence as outlined by Bohlen's work. We did this as a basis of demonstrating how time delineates an important characteristic for synchronizing the VM logging requirements for your system administrator environment. We provided this characterization within the context of a proof of concept case study. Further work explores new experiments to perform temporal log mining for compute cloud forensic scenarios within our University environment.

Acknowledgment

The authors would like to express thanks to Debbie Bassaragh, Leon Stenneth and Kenville Thompson for their reviews.

References

[1] Thorpe, Ray, Ray. *Towards a Formal Parameterization Context for a VM Audited Compute Cloud*. (unpublished)

[2] S Abiteboul, L.Herr and J. Van den Bussche. Temporal Connectiveness versus Explicit Timestamps in Temporal Query Languages. *Recent Advances in temporal Databases*, pages 43-57, Springer Verlag, 1995.

[3] J. Van Benthem. The Logic of Time - A Model Theoretic Investigation into the varieties of Temporal Ontology and Temporal discourse. Kluwer 1991.

[4] M.H.Bohlen, R.T. Snodgrass, and M.D. Soo. Coalescing in Temporal Databases. *Proc. of the 22nd VLDB Conf. pg 180-191. Morgan Kaufmann, Sept. 1996.*

[5] J. Chocmicki. Temporal Query Languages: a Survey. *Proceedings of the First International Conference on Temporal Logic,* pages 506-534, 1994.

[6] J Chomicki. Temporal Query Languages: a Survey 1995. Submitted to IEEE TKDE.

[7] S.K. Gadia. A Homogenous Relational Model and Query Languages for Temporal Databases. ACM TODS, 13(4): 418-448, 1988.

[8] C. Jensen, M.Soo, and R.T.Snodgrass. Unifying Temporal Models via a Conceptual Model, Information Systems, 19 (7):513-547, 1994.

[9] N.A. Lorentzos and Y.G. Mitsopoulos. SQL Extension for Interval Data. IEEE TKDE,9(3):480-499,May 1997.

[10] J.Melton and R.Simon Understanding the new SQL. A Complete Guide: Morgan Kaufmann,1993.

[11] N.Gehani, H.V.Jagadish and O.Shmueli. Event Specification in an Active Object Oriented Database. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 81-90., San Diego, CA, June 1992. ACM Press.

[12] R.T. Snodgrass. The Temporal Query Language TQuel.ACM TODS, 12 (2): 247 -298, June 1987.

[13] R.T. Snodgrass, The TSQL2 Temporal Query Language Kluwer Academic 1995.

[14] Sean Thorpe and Kavan Farquharson. Design of an enterprise VM log auditor tool kit. *Proceedings of the 1st International Audit Conference and Journal of Arts and Technology*, University of Technology, Kingston, Jamaica. January 2011.

[15] Thorpe, Ray, Ray, Grandison, Barbir. Global Virtual Machine Auditor for a Federated Digital Identity. To Appear in the *Journal of Information Assurance and Security (Volume 5 issue March 2011).*

[16] Indrakshi Ray and Wei Huang, Event Detection in Multi level Secure Databases, *Proceedings of the 1st International Conference on Secure Systems*, Kolkata, India, and December 2005.

[17] A.Tansel, J.Clifford, S.Gadia, S.Jajodia, A. Segev and R.T. Snodgrass. Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummins, 1993.

[18] D.Toman, Point based vs. Interval based Temporal Query Languages. *Proceeding of the 15th ACM PODS Symposium*, pages 58-67, June 1996.

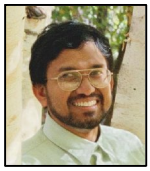
[19] Sean Thorpe, Indrajit Ray, Tyrone Grandison. Associative Schematic Mapping for a Synchronized Log Virtual Machine. Springer Verlag CSIS (to Appear June 2011).

[20] Sean Thorpe, Indrajit Ray, Tyrone Grandison. *Enforcing Data Quality Rules for Transformation Mapping of a Synchronized Log Virtual Machine. Springer Verlag CSIS* (to appear June 2011)

[21] M. H. Böhlen, Renato Busatto C. S. Jensen. Point based versus Interval Based Temporal Models *Proceedings from Computer Science Workshop, Aalborg University, 1998.*

Author Biographies

Sean Thorpe holds an M.S. and B.S. degrees in Information Security and Computer Science respectively from the University of Westminster, London, UK in November 2002 and from the University of the West Indies, Mona Campus Jamaica in November 2000. He joined the University of Technology (UTECH) as a Lecturer in January 2003 with responsibility for teaching System Security at the undergraduate level. Mr. Thorpe has worked extensively in the IT industry since 1995 as a System Programmer Analyst and Oracle DBA before joining academia. He is a 2009 recipient of the Fulbright Visiting faculty Scholarship award to Harvard University, where he explored collaborative research work in the area of Security Metrics. He is also the 2009 winner of the OOPSLA Educational Symposium Award for his innovative computer science teaching methods, and the recent 2011American National Science Foundation (NSF) awardee for Caribbean based research in the area of Cloud Computing. His specific research interest includes cloud forensics, and security policies. In 2010 he started his PhD in Computer Science.



Indrajit Ray is an Associate Professor at Colorado State University since 2002. Prior he was an Assistant Professor at the University of Michigan Melbourne from 1997 to 2001. He earned his PhD from George Mason University, Virginia in summer 1997. He obtained his B.S. Computer Science Degree from Bengal Institute in India in 1984 and then his M.S. from Jadvapur University in 1991, also in India. His primary research interest is digital forensics, security policies, access controls and intrusion detection.

Indrakshi Ray is an Associate Professor at Colorado State University since 2002. Her research interest includes access controls, UMLs and workflow security issues for the enterprise.

Tyrone Grandison is a Research Staff Member at the Thomas J. Watson Research Center. He received a B.S. degree in Computer Studies (Computer Science and Economics) from the University of the West Indies in 1997, a M.S. degree in Software Engineering in 1998 and a Ph.D. degree in Computer Science from the Imperial College of Science, Technology & Medicine in London. He then joined IBM at the Almaden Research Center in California, where he worked on data privacy and security. In 2010, he joined the Global Healthcare Transformation team as Program Manager for Core Healthcare Services. Dr. Grandison is a Distinguished Engineer of the Association of Computing Machinery (ACM), a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), a Fellow of the British Computer Society (BCS), has been recognized by the National Society of Black Engineers (as Pioneer of the Year in 2009), the Black Engineer of the Year Award Board (as Modern Day Technology Leader in 2009 and Minority in Science Trailblazer in 2010) and has received the IEEE Technical Achievement Award in 2010 for "pioneering contributions to Secure and Private Data Management". He has authored over 70 technical papers and co-invented over 20 patents.