Loom: Complex Large-Scale Visual Insight for Large Hybrid IT Infrastructure Management

James Brook[a], Felix Cuadrado[b], Eric Deliot[a], Julio Guijarro[a], Rycharde Hawkes[a], Marco Lotz[a], Romaric Pascal[a], Suksant Sae-Lor[a], Luis M. Vaquero[c,*], Joan Varvenne[a], Lawrence Wilcock[a]

[a]*Hewlett Packard Enterprise*
[b]*School of Electronic Engineering and Computer Science of Queen Mary University of London*
[c]*Department of Computer Science. University of Bristol*

Abstract

Interactive visual exploration techniques (IVET) such as those advocated by Shneiderman and extreme scale visual analytics have successfully increased our understanding of a variety of domains that produce huge amounts of complex data. In spite of their complexity, IT infrastructures have not benefited from the application of IVET techniques.

Loom is inspired in IVET techniques and builds on them to tame increasing complexity in IT infrastructure management systems guaranteeing interactive response times and integrating key elements for IT management: Relationships between managed entities coming from different IT management subsystems, alerts and actions (or reconfigurations) of the IT setup. The Loom system builds on two main pillars: 1) a multiplex graph spanning data from different ITIMs; and 2) a novel visualisation arrangement: the Loom "Thread" visualisation model.

We have tested this in a number of real-world applications, showing that Loom can handle million of entities without losing information, with minimum context switching, and offering better performance than other relational/graph-based systems. This ensures interactive response times (few seconds as $90th$ percentile). The value of the "Thread" visualisation model is shown in a qualitative analysis of users' experiences with Loom.

Keywords:
management, cloud, scale, visualization, complexity, extreme scale visual

*Corresponding author: luis.vaquero@hpe.com

analytics, visual analytics

Highlights

- Interactive insight extraction techniques can help to tame full stack IT management complexity
- Interactive insight extraction techniques need to be extended for manageability software. This is the role of the "Thread" visualisation model:
  - reduces context switches and minimises text on screen
  - minimises text on screen and avoids edge crossings, which tend to clutter the screen in densely connected graphs
  - creates IT infrastructure overviews for large IT installations by summarising IT entities status and highlighting relationships on managed entities of interest
  - deals with alerts as first-class citizens
  - enables IT operators to execute actions to change the current state of the IT infrastructure
  - works on touch-enabled displays with radically different form factors
- An extended multiplex graph powers the "Thread" visualisation model
- The extended multiplex graph delivers interactive response times (few seconds)
- Loom excels at aggregating data and on relational queries to these aggregations
- Loom builds on the advantages of interactive visual exploration techniques by enabling interactive spotting of relationships between aggregations of managed entities

1. Introduction

Information Technology (IT) infrastructures produce a myriad of messages and alerts that a human operator receives in order to make an assessment and execute some action that takes the IT infrastructure management (ITIM) system to a normal operating mode.

Many IT issues in an infrastructure are never analysed and resolved. This results in apparently benign early symptoms of failure (e.g. low priority alerts) being ignored until they have caused major problems. For example, an unusually slow drive tends to be ignored until the database running on it becomes unusable and transactions fail.

Most companies deal with this problem by continuously refining a set of thresholds on well-known metrics (Service Level Agreements, SLAs). This strategy has proven to be extremely efficient and effective when all services rely on a single ITIM in relatively modern IT setups [1].

Unfortunately, there are several factors that set most IT setups apart from this ideal scenario:

- the IT infrastructure is the reflection of the history of a corporation and it tends to include many ITIMs that have been added over decades
- the extreme abundance of open source ITIMs released by all sorts of companies in an attempt to reduce costs, make recruitment easier, commoditise competing solutions, and/or share some knowledge back with the community
- high heterogeneity of the software stack running on the IT infrastructure (tools for accounting, relationships with suppliers, provisioning, provenance, human resources, and many other) makes management more difficult
- the continued effort towards setting programmable infrastructure that was started by virtualisation/cloud technologies and is continued by "containerisation", software defined network and network function virtualisation
- miniaturisation of the user interfaces taking us from a world of "mobile first" to a situation closer to a "mobile only" where the ability to touch to manage the infrastructure is increasingly appealing

All these factors foster:

- the appearance of hidden interdependences (switching a host off with HPE iLO will kill the Kubernetes containers running on it but these two systems are not necessarily connected)

- the presence of partially overlapped functions so that an IT operator can exert the same effect via a different ITIM. One could, for instance, shutdown a container by directly using Docker or by invoking an appropriate Kubernetes command.

Hiring more administrators and increased automation (like testing, building and deploying [2]) have been the responses to cope the increased complexity of modern IT setups. In spite of these strategies, considerable human effort is still required to oversee and authorise semi-automated decisions, and full automation is often times only possible after careful human analysis of management data [3] .

Unfortunately, we have reached a point where increasing the number of administrators is simply not enough. Current projections estimate there will be 80 billion connected devices by 2020[1]. The number of servers a single sysadmin can manage range from 100 to 20000 depending on the complexity of the IT setup. This maps to 80-16000 times the number of U.S. majors in computer science in 2012[2].

Poorly managed infrastructures result in security problems (e.g. unpatched devices attacked on well-known vulnerabilities), operational issues (e.g. slow times to deploy or upgrade a running service) and, eventually, in poor user satisfaction and profit losses.

Visual analytics (VA) is the "science of analytical reasoning facilitated by interactive visual interfaces" [4]. Extreme-Scale Visual Analytics (ESVA) focuses on large scale complex and possible multi-sourced data [5]. ESVA establishes a close interplay between 1) a fast (often in-memory) analytics backend capable of merging data from several sources to create insightful summaries and 2) fine interactive visualisations that promote exploration of summarised data and quick extraction of details of interest. Thus, ESVA enables humans to focus their full perceptual and cognitive abilities on the analytical process and extract actionable insight from complex data [5, 6]. Similar guidelines to deliver interactive

[1]http://www.forbes.com/sites/gilpress/2014/08/22/internet-of-things-by-the-numbers-market-estimates-and-forecasts/

[2]http://archive2.cra.org/uploads/documents/resources/taulbee/CS_Degree_and_Enrollment_Trends_2010-11.pdf

exploratory tolls are also advocated by Shneiderman in his famous mantra [11]: ”Overview first, zoom and filter, then details-on-demand”. We refer to these as interactive visual exploration techniques (IVET).

While these techniques have been widely applied across disciplines [7, 5], IT infrastructure management has remained indifferent to their advancements except for a few position statement papers that claim that a new breed of systems (known as “Data Visualization Management Systems”) is needed to increase collaboration between visualisation and data infrastructures [8, 9].

IT management requires to build on interactive visual exploration techniques to cope with information overload. In addition, there are some peculiarities of IT management that have not been explored by current IVET solutions:

- alerts also need to be summarised and visualised together with the elements of the IT infrastructure they are associated with
- spotting sources of problems and filtering noise in IT management makes it essential to highlight relationships across managed entities in different ITIMs. For instance,“discovering a failure in network link A caused errors in application X via server 1”
- IT management is not just about extracting insight from management data: the ability to exert “Actions” to change the state of the many ITIMs in the infrastructure in a coordinated way becomes critical to reduce cognitive overload
- as many IT operators can be “on the go” and access the management interfaces from displays with different form factors, the ability to touch on representations of the managed entities becomes increasingly interesting

Our system, Loom[3], is the first system that:

- 1) enables interactive touch-enabled exploration. This is possible thanks to a new graph model that includes managed IT nodes and aggregations of interest, which result from queries by the user. Loom can manage data from several ITIMs, creating multiplex aggregated graphs.

[3]The name comes from the metaphor that summarising information can help users “weave” their own “tapestries” rendering a custom overview of the ITIMs in their IT setup.

- 2) a new interactivity model where elements in the frontend are mapped to entities in the backend but only an abstract representation and a summary of the managed IT infrastructure elements are sent to the frontend.

- 3) a set of internal optimisations that make Loom capable of coping with large heterogeneous scale IT installations in interactive response times (less than 30 seconds).

The rest of this paper is structured as follows: Section 2 presents how Loom builds on visual interactivity and exploration techniques for large full stack IT setups containing several ITIMs. After presenting the main functionalities and the fine orchestration required between frontend and backend, we present Loom design in Section 3. The advantages of Loom are exemplified with two use cases in Section 4. We evaluate Loom in terms of performance and user experience and present the results in Section 5. Loom is compared with state of the art systems in Section 6. The main conclusions are summarised in Section 7.

2. The Loom Visual Insight Extraction Model for Large-scale IT Management

Corporate IT users have been characterised in enterprise usability studies [10]. Loom caters for two of them. Advanced users (also referred to as "hackers"in [10]) are in charge of pulling data from several data sources and specifying how less tech-savvy users ("application users" as defined in [10]) would use the data reaching the user interface (UI). We will stick to the "hacker" / "application user" terminology for the remaining of this manuscript.

This section presents Loom from the point of view of these two types of users and highlights reveals how large-scale IT management is made exploratory and interactive by a closer interplay between UI and backend, as anticipated by recent position papers [8, 9].

2.1. The Application User: Visualising the IT Infrastructure

2.1.1. Summarisation + Homogeneisation

Loom enables the creation of summaries of unprecedented numbers of managed entities coming from different ITIMs.

Figure 1 shows the main building block of Loom, the "Thread"[4].

[4]We are aware the term is heavily overloaded in computer science, but in this paper we

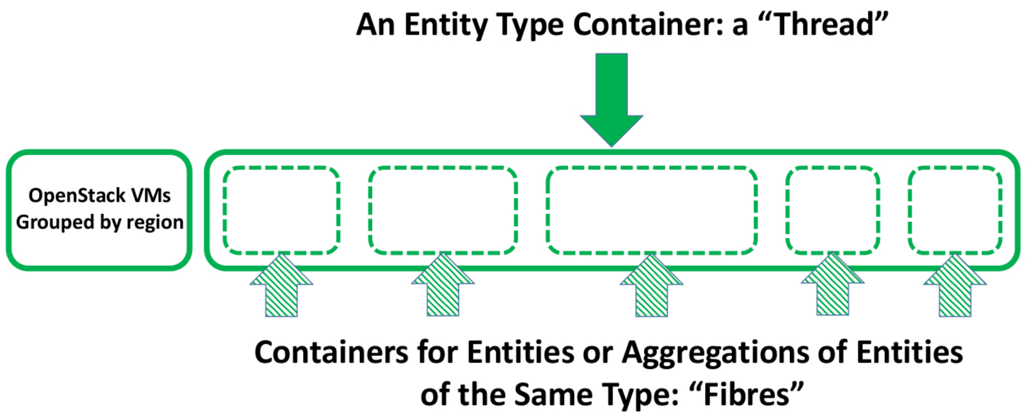

An Entity Type Container: a “Thread”
OpenStack VMs Grouped by region
Containers for Entities or Aggregations of Entities of the Same Type: “Fibres”

Figure 1: Example of “Thread” and its constituent elements: “Fibres”

On the UI side. A “Thread” is nothing more than a rectangle[5] containing visual representations of managed entities of the same type (e.g. OpenStack Virtual Machines, VMs), see Figure 1. When the number of managed entities is too large to fit in the screen, Loom performs aggregations on the managed entities so that a user can interact with them in a touch-enabled display. Loom also automatically rolls up the properties of the entities contained in an aggregation to create summaries (e.g. Loom extracts statistics on numeric properties of the entities in the same aggregation).

Each of the dashed-line rectangles inside the visual “Thread” in Figure 1 is referred to as a “Fibre”. “Fibres” can represent either an individual managed entity (e.g. a VM) or an aggregation of entities of the same type (a set of VMs). The length of a “Fibre” in a “Thread” is proportional to the number of managed entities it contains (and the resolution of the display).

“Threads” also include a “Thread” header, as shown on the left hand side of Figure 1. This header includes information about the type of entity contained in the “Thread” and if we are seeing an aggregation of entities or plain entities in the “Fibres” it contains.

metaphorically refer to it in the purest traditional sense of the word: one of the lengths of yarn forming the warp or weft of a woven fabric.

[5]Many other layout arrangements are possible, like a circle.

On the backend. Managed entities of the same type are kept in an indexed collection. When no aggregations are needed on the UI ("Fibres" correspond to single managed entities), this backend collection is uniquely mapped to its visual representation as a "Thread".

Whenever the number of entities is too large to remain touchable on the display employed by the user, Loom backend creates aggregated "Fibre" objects. A "Fibre" can also be the result of a specific query performed by a user to explore the state of the IT infrastructure. Each backend "Fibre" keeps a set of pointers to the managed entities it contains and a set of summary properties resulting from the roll-up of the entities contained in the "Fibre".

When the managed entities belong to one or more ITIMs, Loom merges them in a single "Thread" if it detects they have the same attributes. This is key to enable the display of overlapped functionality across ITIMs as a single pane of glass (e.g. VMs in different OpenStack installations are represented in the same visual "Thread").

2.1.2. Exploration

When confronted with a sea of options and false positive alerts, ITIMs operators often need to perform exploratory queries to find their way and distil the root cause of the incident they are handling.

On the UI side. Loom provides a "query customisation" menu that enables users to control the type of operators applied to the managed entity/ies.

By selecting the sequence of operators to be applied to one or more "Threads" users can visually construct a "Thread query" that returns a more informative arrangement of the managed entities in that "Thread".

By default, Loom provides a "NoOP" operation (Figure 2 top) that creates aggregations of managed entities in an arbitrary manner so that they are compliant with the form factor of the device used to visualise the results.

Query construction is as simple as selecting operations from a combo box on the UI and concatenating them as needed (see bottom part of Figure 2). In this example, an application user would filter OpenStack VMs by operating system when this equals to 'Ubuntu', and then group them by region. If after these two operations there were more resulting "Fibres" (aggregations of OpenStack VMs by region) than recommended to keep touchable "Fibres" on screen, then the "NoOP" would tell Loom backend to create arbitrary aggregations of OpenStack regions. In the example in the Figure, we assumed there were just 3 different regions in our OpenStack installation.

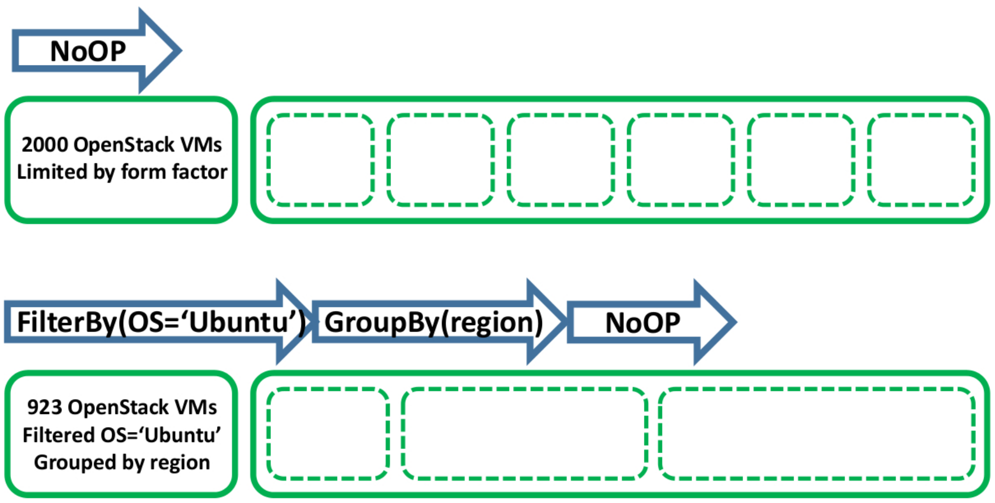


Figure 2: Example of "Thread" query customisation.

The operations at hand are those provided by Loom by default or a set of operations written by the "hacker" user:

- *Aggregation*: group by, string similarity clustering (e.g. K-means), hierarchical geographical clustering (e.g. based on polygons or 2D grid projections), deciles, percentiles and the like.
- *Visualisation*: "NoOp" (a hierarchical aggregation that limits maximum number of fibres (entities/aggregations) in a thread, and thereby on screen, to a client-specified number), limit drill down (same as "NoOP" but managed entities are placed in fibres so that the number of clicks to reach managed entities when drilling down is minimised).
- *Other*: free-text based search, regex-based search, predicate-based filtering, sort by, community detection, geographical analysis, etc.
- *"Hacker" defined*: custom operations needed to handle a particular type of data. For instance, the location of fleets of mobile devices of an organisation can be clustered by location or communities of devices can be found using custom clustering and community-detection techniques (see next section).

On the backend: internal abstractions. Loom dynamically creates an *extended graph* by recursively connecting new "Fibres" resulting from applying operations to previously created "Fibres".

Figure 3 (top) shows an example of how the graph is extended[6]. From a graph of OpenStack VMs, we want to get rid of non-Ubuntu VMs. Two new "Fibres" (which effectively become graph nodes themselves) are connected by containment relationships (dashed red lines) to all graph nodes that meet that predicate (OS = 'Ubuntu'). It can also be observed how these aggregations of managed entities would be displayed as "Threads" and "Fibres" on the UI.

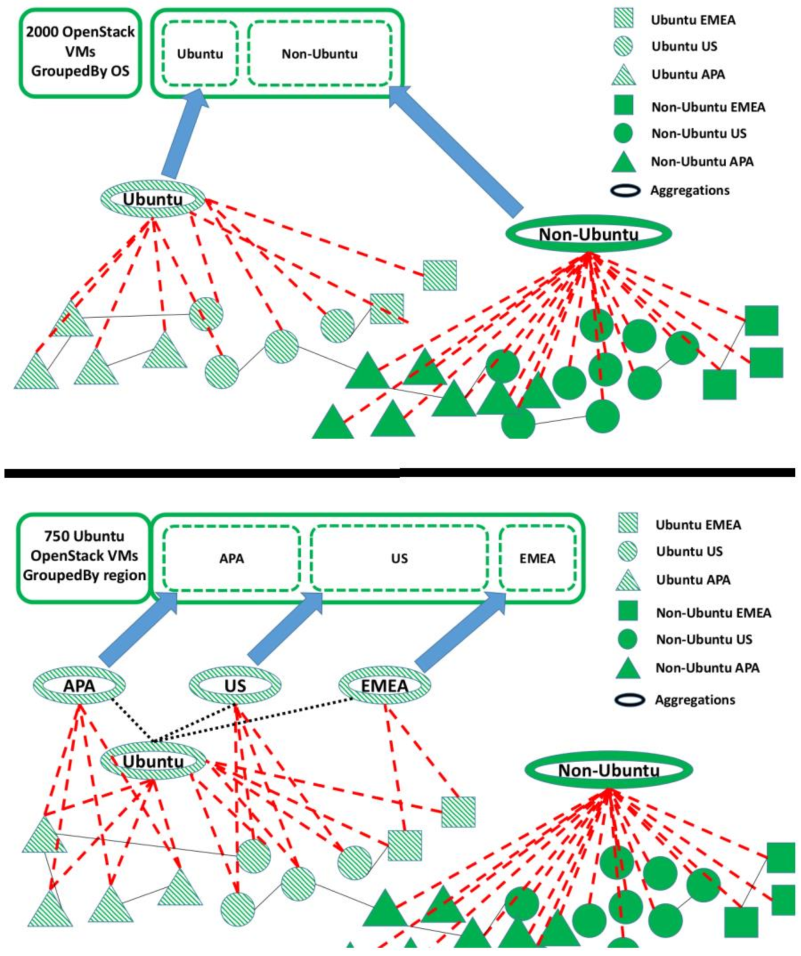


Figure 3: Example of "Thread" query customisation on the Loom graph: extensions.

[6]Note that graph node, graph extension, "Fibre" and aggregation indicate the same concept: an aggregation of managed entities or other aggregations (a "Fibre") that is added as a node to a graph of other "Fibres", effectively creating an extension of the graph.

The second part of the pipeline (bottom part of Figure 3) takes all the OpenStack VMs that are contained in the newly formed 'Ubuntu' graph node as an input and creates three new extensions (graph nodes/"Fibres") from it. These are based on the region the VM are located at. These three new graph nodes are, in turn, connected to the managed entities they contain (dashed red lines) and to the "parent graph node they originated from (OS = 'Ubuntu'); black dotted lines show parenthood relationships between the 'Ubuntu' "Fibre" and the 3 newly generated children "Fibres" (EMEA, US, APA). It can also be observed how these aggregations of managed entities would be displayed as "Threads" and "Fibres" on the UI.

The "Fibre" resulting from the last operation is the one sent back to the UI. All others (like the 'Ubuntu' one in the example) are memoised and are, thus, sharable across queries and users.

2.1.3. Overview

According to Shneiderman [11], a single "Thread" would not be sufficient to create overviews ("Tapestries") that can be used to explore data coming from different ITIMs.

When dealing with issues, IT operators often need to find relationships between different elements in the infrastructure. Finding relationships between individual managed entities becomes more complex as the scale and connectivity of the graph increase. The number of potential paths between entities grows exponentially.

This becomes more complex when the UI displays aggregations/summaries[7]. Keeping and maintaining links of aggregations of managed entities to the entities they contain needs to be well structured so that relationships can be found in interactive times and user queries resulting in new aggregations (or changes in the data from the backend ITIM) are easy to add/update.

Loom is unique in its ability to spot relationships between aggregations of managed entities in interactive response times. Figure 4 shows how related "Fibres" are highlighted when a user clicks or taps on a "Fibre" on the UI.

On the UI side. Overviews ("Tapestries") are created by lining together several visual "Threads". Showing relationships between thousands of managed entities is not possible in a single pane of glass. Thus, Loom displays relationships to

[7] As shown above, "Threads" consist of several "Fibres", which can contain individual managed entities or aggregations of managed entities.

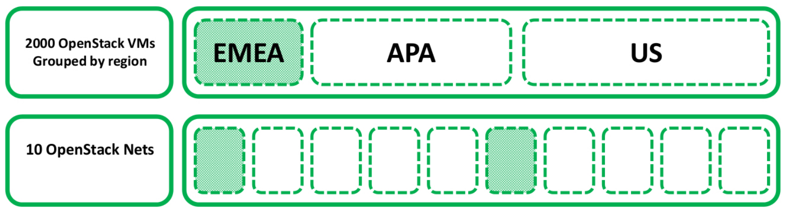


Figure 4: Application users can “weave” their own “Tapestries” by placing the “Threads” of interest on the screen and observing relations between “Fibres”

any “Fibre” of interest by clicking on that “Fibre” and showing related “Fibres” as highlighted in a different colour. The example in Figure 4 shows how when a user clicks on a “Fibre” containing OpenStack VMs in the same region, some networks highlight as well. This indicates that some of the VMs in that “Fibre” are attached to those networks.

On the backend. A “Tapestry” is merely a collection of collections of managed entities of different types. Loom backend takes care of navigating the extended graph and reporting whether or not any two “Fibres” on the screen are connected.

Loom takes all “Fibres” in the “Tapestry” and computes whether a path exists connecting any two nodes. The extended graph is navigated from each “Fibre” on the screen reaching the managed entities in that “Fibre” (‘descending’ the dashed red lines and finding entity to entity connections which are created by the “hacker” user, see below). This is represented as a thick dashed blue line in Figure 5.

Loom then tests whether a path has already been identified (cache path lookup) and if not, Loom tests if the entities are connected to any common “hub”[8]. If they do not share a path to a common hub, 1-hop neighbours are tested to see if their neighbours are connected to a common hub. If no path is found using hubs, a fully-blown path finding algorithm is used and the path cache is updated if one happens to exist. This accelerates the computation of relationships between aggregations on the screen.

Unless other relatedness functions are indicated as a configuration parameter to Loom by the “hacker” user, Loom assumes that two “Fibres” are connected if any of the managed entities they contain are themselves connected.

As shown above, “Fibres” are created by a single sequence of operations ap-

[8]Highly connected managed entities are identified when they are loaded into the system. See subsection on Relationship Handling below

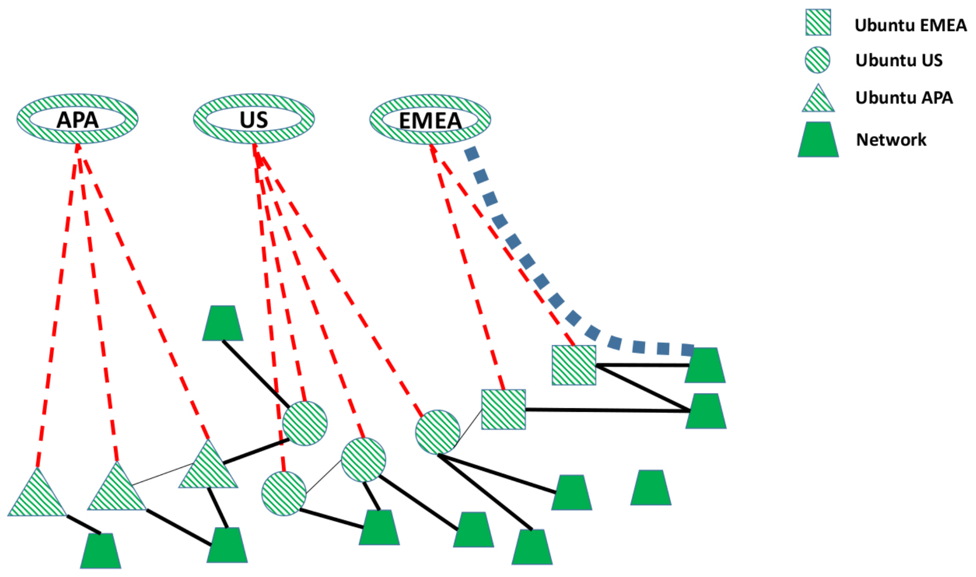


Figure 5: Calculation of Relationships of Managed Entities/Aggregations on Screen

plied to managed entities or aggregations of managed entities in a previously created "Fibre". This sequence of operations, its inputs, and its parameters are hashed to create a unique name. This name is used as key for a key-valued cache which contains the results of the query. The UI uses this unique name to get up-to-date results.

As mentioned above, the different types of entities in the Loom graph are kept in separate indexed collections. This can be seen as a multi-partite graph separating managed entities by their type. However, connections to entities of the same time are not prohibited, they are simply discouraged for they imply more object-level checks are needed at query time to prevent loops and filter possible relationships.

2.1.4. Details

Once an IT operator has explored the status of the IT infrastructure, she is directed to a small set of managed entities of interest to solve the issue at hand. For instance, alerts point to a single hard drive as the root cause of an unexpected performance degradation in mobile apps. Loom is tuned to deliver details on entities of interest without losing the overview of all other managed entities.

On the UI side. Application users select a “Fibre” of interest in a “parent Thread” and get details for it contents.

Say the application user is using a small factor device that supports a maximum of 5 touchable “Fibres”. If the user grouped OpenStack VMs by region and the result returned more than 5 regions, the “NoOP” operation in Loom would create additional aggregations.

In the case of Figure 6 (top “Thread”), there are 5 “Fibres” in the VMs Thread each resulting from applying the “NoOP” to the results of the “groupBy” operation.

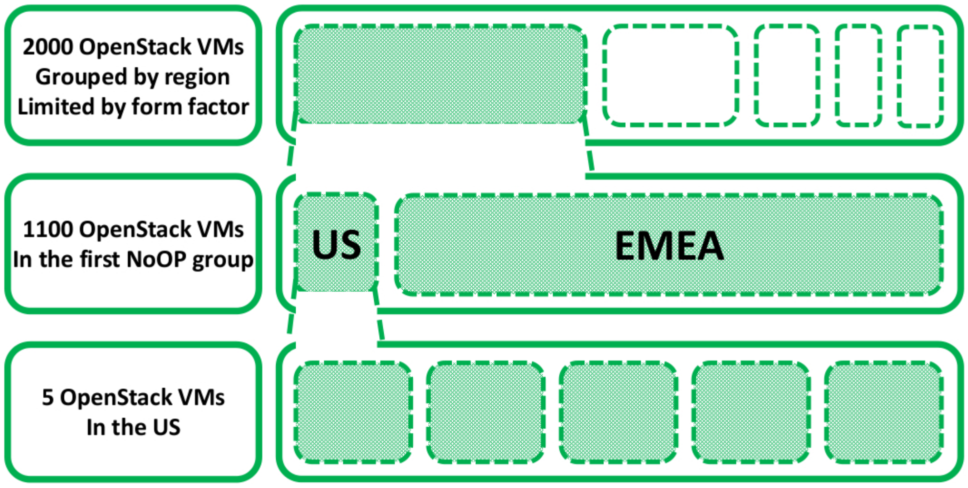


Figure 6: Obtaining Details from a “Fibre” containing aggregated data

When the user requests more details about the first arbitrary “Fibre” (e.g. one in the “Thread” of VMs), the UI sends the name of that “Fibre” to Loom backend and the response contains the names of two “Fibres” (the ones representing the aggregations of all VMs in the US and EMEA). This is represented as a “child Thread” in the UI. It is a “Thread” of the same type of entities, but containing just a subset of all the entities.

If for whatever reason the user is interested in finding out more about the VMs in the US, she can then request details about them. The UI sends the name of the US “Fibre” to Loom backend in order to get the contents of that “Fibre”. In this case, the result is just 5 OpenStack VMs and all the details about them (see bottom “Child Thread” in Figure 6).

On the backend. The name of the “Fibre” is used as “entry point” in the graph. When the user gets details, the UI sends a request of details to Loom backend. It simply follows graph extensions until the entities of interest are found in the indexed collections described above.

2.1.5. Manageability Specifics

The exploratory nature of the work an IT operator goes through when trying to solve an issue on the IT infrastructure is similar to other information visualisation-inspired domains [11]. The importance of dealing with relationships and showing them interactively is a differentiating aspect.

Also, manageability data is different from any other type of data in that alerts are one main driving force for operators and they deserve a special treatment. The same goes for Actions, since the IT operator will eventually tweak some levers to solve the issue he or she is dealing with.

On the UI side. Loom displays alerts coming from different systems as a tiny differently coloured square on the corner of the “Fibres” in a “Thread”. Since some “Fibres” are aggregations of managed entities, alerts are aggregated too.

Actions are represented as a button a user needs to click on in order to change the state of one or more ITIMs (see Figure 7).

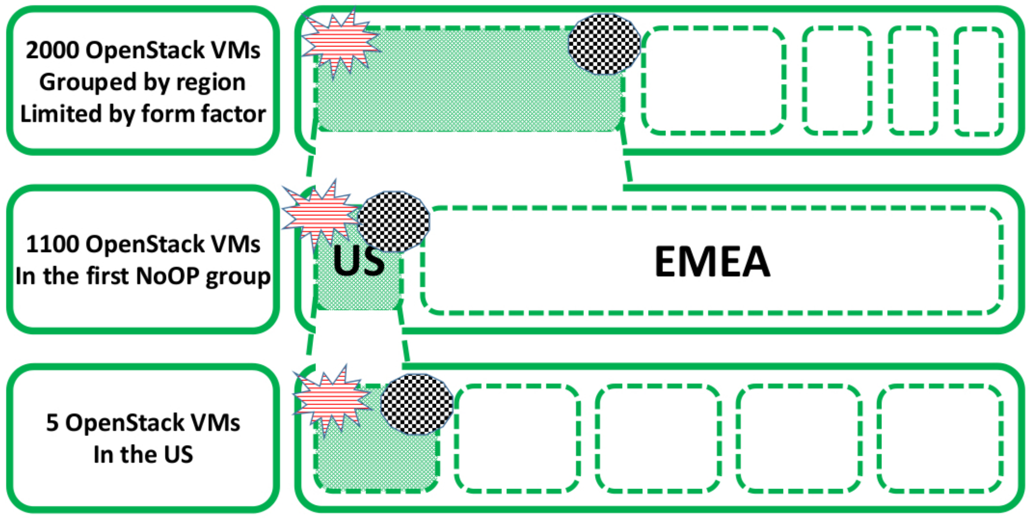


Figure 7: Overlaying buttons for Actions (checked board dodecagon) and Alerts (red horizontal line stars)

Alert/Action aggregation is possible since “Fibres” in a “Thread” contain managed entities of the same type (e.g. an alert on a VM failing unexpectedly in a public and a private OpenStack instance). Loom infers alerts and additional actions by inspecting the Java annotations from the entity adapters. Alerts will be displayed for any element contained in the “Fiber”. Aggregate views present the actions available for every individual instance belonging to the “Fiber”.

On the backend. The “hacker” user can define custom aggregation mechanisms. Loom provides a default aggregation for alerts: the maximum alert priority is shown for the whole aggregation (the alerts of all the managed entities in a “Fibre” are inspected simply by following containment relationships, e.g. dashed red lines in Figure 5). This prevents “median filter” effects that would obscure high priority alerts when aggregated with many low priority ones.

Action aggregation implies very little work on the backend: Actions available to a managed entity are made available to aggregations of that type of entity. More coordination is required when the user executes an Action on the UI. When a user executes an Action on a “Fibre” containing an aggregation of managed entities (e.g. power off whole OpenStack region containing thousands of VMs), the UI sends the name of the “Fibre”, an Action id, and any possible parameters for the Action (as populated by the user on the UI). Loom backend then finds all the managed entities in that “Fibre” and issues an individual call to execute the Action on the ITIM.

2.2. The “Hacker” User

While application users can benefit from all the advantages of Large-Scale IVET, there is some preparation work that needs to be done for them to be able to extract inisght from and operate complex large scale installations.

Modeling ITIM Data. “Hackers” are required to model the entities in the ITIM in a format Loom can digest (schema reconciliation). This is done by building an Adapter to construct a graph model each of the ITIMs.

In order to create an Adapter containing a graph model of the data of the ITIM of interest, a “hacker” user goes through three well-defined steps:

- *Data collector*: each Adapter taps into an ITIMs in their preferred way (Loom does not prescribe anything about the communication with the backend system).
- *Managed entity*: data structure representing each instance of a real world managed entity.

- *Managed entity type*: schema definition used by Loom to describe attributes and management Actions.

Loom provides an Adapter Manager that orchestrates several Adapters by invoking a "collect" method in each of the Adapters to trigger a collection of data from the original ITIMs. The collector will then pull data and map them to specific managed entity types. Managed entities types are Java classes containing attributes (string, numbers, enums, dates and arrays are supported data types) and relationships and Actions that can be executed on managed entities of that type.

Loom allows "hackers" to specify managed entities directly as Java classes where annotations are used to easily specify managed entity relationships or Actions[9]. For example, the following snippet shows how a managed entity (an OpenStack network) from an OpenStack Adapter is "Connected to" other types of managed entities (namely OpenStack Projects, Instances and Regions). At query time, Loom uses these annotations to filter possible connections at the class (entity type) level during a graph traversal. This is faster than inspecting individual entity attributes only to find a connection type is not allowed, which is why connections to managed entities of the same type are supported, but discouraged.

```
@ConnectedTo(toClass = OsProject.class)
@ConnectedTo(toClass = OsInstance.class)
@ConnectedTo(toClass = OsRegion.class)
```

Listing 1: Tagging an Entity Type to Define Connections Between Entities

Connecting Multiple Sources. In most IT infrastructures, the information available for a real-world managed entity is obtained from a variety of ITIMs that do not directly communicate with each other. Different assumptions (e.g. naming schemas) in each system may cause the same real world managed entity to be represented differently (name and attributes) in each of those systems. This makes finding relations between individual managed entities in different systems hard. Administrators have to switch from the UI or console of an ITIM screen to the next and miss the big picture in the very many hops they are forced to follow. Loom enables those "equivalent" managed entities to be "stitched" together by identifying potentially equivalent managed entities across data sources (graphs) and building links across them.

[9]These could also be given in a JSON format indicating entity properties that can be converted to a Java Bean object

This way, two or more otherwise disconnected ITIMs (and their respective multi-partite graphs each modelled by a different Adapter) are joined together forming a multiplex graph. As a result, relationships can be found and queries can be made across graphs (ITIMs) in a seamless manner.

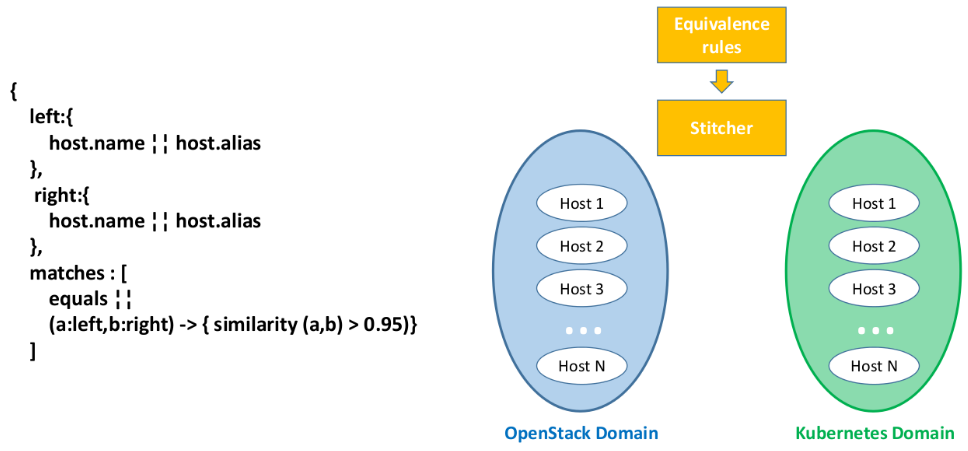
When several Adapters have been pre-loaded into the system, the person developing a new Adapter ("hacker") has the possibility of defining "stitching" functions that define equivalence relationships between managed entities across different Adapters (different ITIMs). This is shown in Figure 8.

Figure 8 shows the graphs corresponding to two different Adapters one pulling data from an ITIM dealing with HR information and other one from a second ITIM handling IT resources. Both ITIMs contain details about people. Loom enables connections across Adapters, forming a multiplex graph. A stitching rule takes the form of a matching function where a left-hand side is matched against a right-hand side of the rule (see left hand side of Figure 8). The matching function ("Ite Equivalence Rules in Figure 8) is really an array of user-provided lambdas that can be combined resulting in a number n $\in [0,1]$ indicating the probability that two managed entities are equivalent or not). "Hackers" wanting to create a homogeneous view on the myriad of systems they have to cope with (also called "hackers" in this document) can specify their own stitching rules when the Adapter is loaded and the Graph Manager re-evaluates them as needed on entity updates. More complicated functions like data mining techniques can be used.

Modeling Actions. The Actions that can be exerted on managed entities of that type are also indicated in the managed entity class. The following code snippet shows how to specify an Action on an ITIM like HPE's Insight CMU[10] node managed entity types.

An @ActionDefinition annotation describes a set of name, id and a set of parameters that the UI needs to capture from the user input in order to be able to execute the Action. In this case, the UI needs to show all possible values ("on", "os off", "boot", etc.) for a "power" operation. This enables the application user to click on the right option (@ActionRange) so that Loom knows what power operation to execute on CMU nodes. This description offers some information that can be useful for a UI (like an icon for visualisation in a UI or a human readable description), but it does not mandate how the information should be displayed (e.g a combo box, a list or any other UI elements could be used for the application user to select what power operation to perform). The most important parts are the type

[10]https://h20392.www2.hpe.com/portal/swdepot/displayProductInfo.do?productNumber=INSIGHTCMUSW



```
{
  left:{
    host.name ¦¦ host.alias
  },
   right:{
    host.name ¦¦ host.alias
  },
  matches : [
    equals ¦¦
    (a:left,b:right) -> { similarity (a,b) > 0.95)}
  ]
```

Figure 8: Breaking Silos with the Creation of Multiplex Graphs. Stitching

(if it applies to individual managed entities or aggregations of managed entities of the same type) and the parameters.

```
@ActionDefinition(
    id = "powerNodes",
    name = "powerNodes",
    type = Fibre.Type.Aggregation,
    description = "Power manage multiple Nodes",
    parameters = {
        @ActionParameter(
            type = ENUMERATED,
            id = "powerOptions",
            ranges = {
                @ActionRange(id = "OFF", name = "Off"),
                @ActionRange(id = "OS_OFF", name = "OS off"),
                @ActionRange(id = "BOOT", name = "Boot"),
                @ActionRange(id = "STATUS", name = "Status"),
                @ActionRange(id = "PRESS", name = "Press"),
                @ActionRange(id = "UID_OFF", name = "UID off"),
                @ActionRange(id = "UID_ON", name = "UID on")
            }
        )
   }
)
```

Listing 2: Annotating an Entity Type to Specify a Power Operation with Several Possible Parameters (for power on, off, boot, etc.) or parameters that tell Loom whether or not the Action needs to be applied to “Fibres” (type)

Custom Data Operations. “Hackers” can also extend Loom’s default operation set (see above) with custom operations that suit their specific entities. The Adapter Manager will invoke a method in the Adapter interface that imports all Adapter-specific operations. When a “hacker” writes these operations, they need to comply with a well-defined interface:

```
@FunctionalInterface
interface QuadFunction<Input1, Input2, Input3, Input4, Output>
```

Listing 3: Interface to Define New Operations.

where Input 1 is the dataset(s) that we want to perform an operation on, Input 2 the arguments of the operation, Input 3 a data structure to report execution errors, Input4 is the operation context (e.g. Adapter that is executing it, security credentials, etc.) and Output is the result of the execution.

This simple interface enables “hacker users” to easily register operations on their own data. “Hacker” users can add new operations that are taken by Loom when the Adapter is first loaded by complying to the interface in Listing 3 above. An example text representation of a UI query on a “Thread” of VMs may look as follows:

```
VMs.filterBy
(uptime > 10 days &&
    OS ==  Ubuntu14.04 &&
   lastUpdatedFirmware > 1 month)
   .clusterBySupportRegions
```

In the example above a “hacker user” would have pre-registered the ”clusterBySupportRegions” operation before an “application user” could put together such a query. This operation is totally dependent on the different regions offering IT support at a global level for a specific company (Loom does not have that knowledge).

Changes in ITIM Managed Entities. Loom is also in charge of automatically detecting changes between poll cycles to the original data sources. Loom keeps a buffer with the last update on data retrieved from the data source. This last update is compared with the current update. Basic hashing functions are used to spot changes, but Loom also lets “hackers” indicate whether an entity attribute is a constant or a rapidly varying attribute (e.g. CPU usage would change every ms and trigger lots of unnecessary updates in the Loom graph, see below). Changes in hashes are inspected further (ignoring constant and fast changing attributes).

If any of the entities in a “Fibre” is changed, Loom flags it as “dirty”. This dirty bit is propagated to all graph extensions that may result from previous queries on the graph. Dirtied “Fibres” are lazily pre-computed upon user request. Loom offers eventual consistency guarantees for concurrent updates. The “hacker” does not need to deal with any of this.

On every update, Loom is also in charge of checking the existence of equivalent managed entities across Adapters (across different ITIMs) and, if one exists, creating the right edges across graphs (“stitches”), which effectively create a multiplex graph across all Adapters.

Relationship Handling. To speed relationships search, Loom enables Reporter objects that define custom strategies to navigate the graph. A Reporter is simply a lambda function injected into a generic graph navigation process that defines which “Fibres” on-screen should be reported back to the UI. The default Reporter prevents loops from happening (e.g. visiting managed entities of the same type twice) and handles the handover in stitched graphs (a query spanning two or more graphs joined in a single multiplex).

Other Reporters can also be developed by a “hacker” user for specific customer needs (e.g. ignore some entities or enable/disable certain connections to be reported and visualised). The Reporter interface is customisable in that a Reporter object (a lambda function that filters graph nodes based on pre-defined predicates) is passed as an argument. The Reporter contains navigation rules that indicate Loom what edges it should follow and when to stop navigating.

```
Relationships calculateRelationships(Fibre src,Reporter reporter);
```

While expansions of the graph will change rapidly based on the ability of the user to ask queries to the graph of managed entities, the graph of managed entities will only change when data in the back-end ITIMs change as well. Thus, computations of path existence between managed entities can be pre-computed when the Adapter loads the data from the ITIMs into Loom. Computing all paths between entities in a dense graph is expensive. Highly connected managed entities are identified when they are loaded into the system (we call these “hubs”). For each hub, we precompute 2 hop paths from the hub and populate a path existence cache.

When using the default Reporter, the “hacker” does not need to deal with graph navigation and loop prevention.

3. Loom Architecture

In this section we show how all the functions outlined in the previous sections work together.

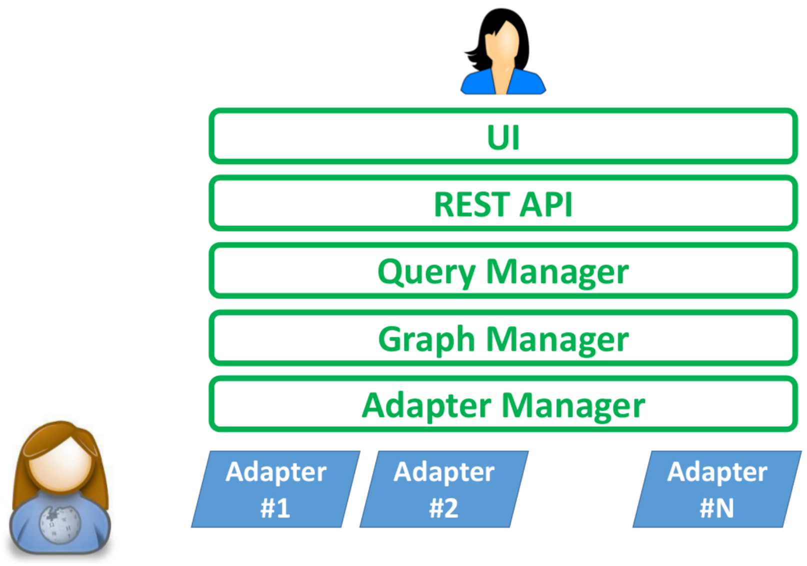


Figure 9: Overview of the Design of the System. The application user interfaces with the system via a UI or a REST API, while the "hacker" user deals with writing Adapters, interfacing mainly with the Adapter Manager.

When an application user defines a query (an operation pipeline) to be executed on an input "Thread", this is sent by the UI to Loom via a Query API supporting a single method (*query*) which receives the uuid of the input dataset and the operation pipeline. Loom uses a JSON version of a dataflow language on the wire to allow communication between the UI and the Loom backend.

The operation pipeline is deserialised by the API and converted to an ordered set of "Operation" objects that comply with the interface defined in Listing 3 above.

The Query Manager (see Figure 9) is in charge of receiving the Operation set and executing operations in the right order by keeping intermediate results and memoises these "Fibres" in the "Graph Manager". There is an optional query optimisation phase that detects if any two consecutive operations can be executed in parallel[11]. The Query Manager also caches results and checks dirty bits before computing results.

The Query Manager relies on the Graph Manager to perform graph extensions

[11]There is no translation of an operation pipeline into a query execution plan beyond this

and summarisations as explained above. The Query Manager also uses a Relationship Calculator' to indicate which "Fibres" on the screen are related. This calculator is in charge of controlling Reporter execution.

The Adapter Manager is the layer in charge of controlling Adapter behaviour and validates that the schema reconciliation performed by the "hacker" user is a correct one. It is also responsible for detecting changes in managed entities so that the Graph Manager can flag some "Fibres" as dirty for the Query Manager to trigger re-calculation of previous queries.

4. A Couple of IT Management Use Cases

4.1. Hybrid Cloud Management

Loom can bring together two graphs (Adapters) where the type of the managed entities is identical, as in the case if a hybrid cloud deployed on the same cloud management software (e.g. a public and a private OpenStack-based cloud installation).

Loom automatically detects that the two Adapters (for private and public cloud) refer to the same type of managed entities (e.g. OpenStack instances) and creates a unified collection that looks like using a single system in terms of user experience. "Fibres" in the UI are mapped to collections. In the case of dealing with two systems, Loom simply appends the entities of a cloud to the collection containing the entities of the other cloud.

Dynamic attributes (those that change rapidly, like IT infrastructure metrics) can optionally be added, contributing to the progressive disclosure of information predicated by Shneiderman [11]. See demo[12].

4.2. Software and Hardware Management: Stitching Separate Domains

Figure 10 shows data from two Adapters: a hardware control and configuration tool, Hewlett Packard Enterprise Insight CMU deployed on a set of Moonshot[13] chassis, and a homebrew software configuration tool (known internally as Slim, Service Lifecycle Manager).

Since there are not common entity types, two separate graphs are created to represent the information brought into Loom by each Adapter. These two ITIMs are disconnected unless Loom's stitching feature is enabled. A user may want to

[12]https://www.youtube.com/watch?v=epl78PBe_Ig

[13]https://www.hpe.com/us/en/servers/moonshot.html

know which Moonshot nodes are actually running the services he has deployed, but that is hard to know since nodes and hosts have not been stitched yet.

Stitching effectively binds the two graphs together based on a matching function provided by the user. In this case Slim's Adapter writer knows about the existence of CMU entities. A quick scan through the names suggests that CMU Node managed entity types can be equivalent to Slim Host managed entity types. A more detailed look at their schemas (managed entity type) shows that both have an attribute that could reveal they indeed correspond to the same real world entity (*host and node*: machines). The "Hacker" then creates a stitching function checking whether there is an exact match between the host attribute of any CMU host and the node attribute of any Slim node. Loom executes that function and creates multiplex edges connecting the Nodes and Hosts where there has been a match.

As shown in Figure 10, Loom highlights "Equivalence" relationships and spots equivalent managed entities. Also, the relationships calculator follows multiplex (equivalence) relationships[14] and as a result when a user clicks on a running service deployed with Slim, it is very easy to spot the networks and logical volumes associated to on Moonshot.

5. Evaluation

5.1. Quantitative Evaluation

In this section, we present some experiments that highlight Loom's ability to synthesise information and find data of interest in a rapid manner, preserving interactive response times. These are more difficult to achieve in connected datasets where relationship reporting is important.

All these experiments were performed on a HP DL 980 with 4TB RAM and 80 cores running Ubuntu14.04 and Java 8. None of the results obtained in this section make use of Loom's caching system. Unless otherwise indicated, the results come from running $N = 15$ experiments and we report 90^{th} percentiles.

For the experiment we used a synthetic dataset that models frequent large-scale OpenStack deployments: large number of hosts, high consolidation of VMs in each host, and long-tail distribution of OpenStack projects are some of the generational characteristics of the dataset. This allowed us to scale up the number of OpenStack entities and how they are connected, while preserving realistic constraints that can reflect actual customer workloads. The only difference between

[14]Other relation types are possible

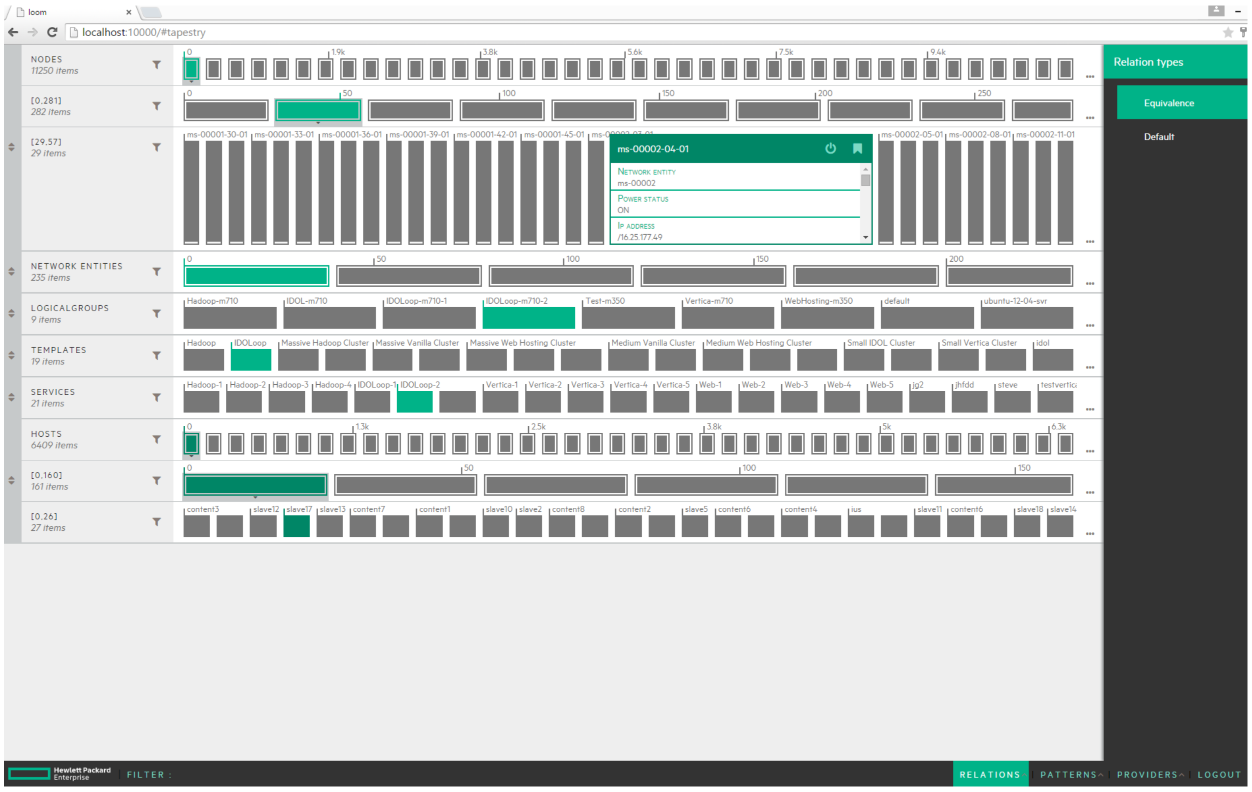


Figure 10: Managing software and hardware from 2 separate ITIMs handling tens of thousands of managed entities on a single pane of glass. This figure represents information about 235 Moonshot chassis (containing 45 nodes each) or a total of 11250 hosts. 6409 hosts are actually populated by a running service (Hadoop, HPE Autonomy's IDOL or HPE Vertica). This Figure needs COLOUR for proper visualisation.

synthetic and real data (from Loom's point of view) is the latency in fetching the data from the backend management systems. For instance, it can take minutes to tens of minutes to fetch VM monitoring data from some cloud management systems. Once the data has been loaded in Loom, there is no difference in terms of processing, memory consumption or scalability between synthetic and real data. Unless otherwise indicated, the default query for these experiments included a filtering operation, followed by a couple of grouping operations and a final "NoOP" (reducing number of "Fibres" to the display size).

Query Time vs # of Managed Entities. Most academic works focus on the number of managed entities in the graph in order to give an idea on how the graph analytics/systems scale. The queries we performed for these experiments are similar to the examples given above and consist of a fixed number (3) of aggregation operations (e.g. group by and cluster per geographical information) followed by a filter operation (e.g. OpenStack VMs whose operating system is Centos 7), and a final grouping (“NoOP”) operation to accommodate the result to small factor (e.g. 10 inches touch-enabled displays).

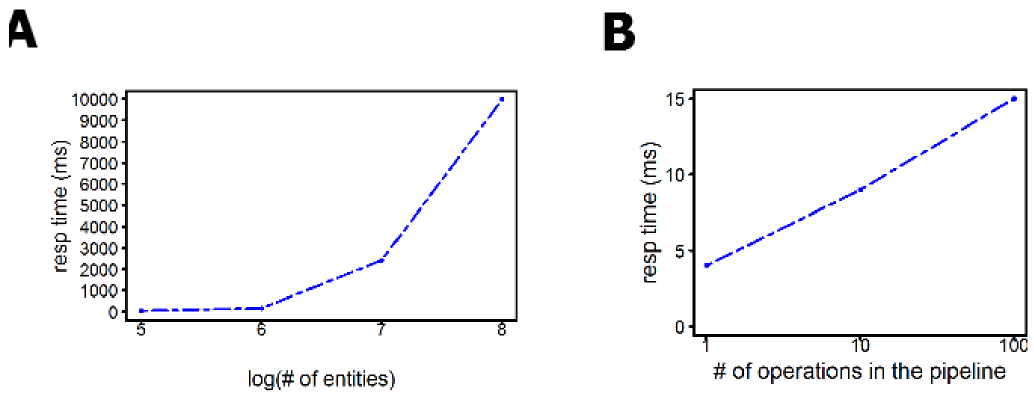


Figure 11: A. Response time vs number of entities. 90th percentiles are given. B. Response time vs # of operations in the query pipeline in a graph with 10M entities. 90th percentiles are given. In our set of use cases, the average density of graph modelling IT infrastructure was 3-5.

Figure 11A shows the modelling of an OpenStack deployment containing OpenStack VMs, volumes and networks. Each of these is modelled as a Loom ”thread”. 90% of the nodes in this graph have $\simeq$ 5 edges.

These times are 2 orders of magnitude faster than operating Loom without any pre-computation of paths between hubs or caching. The precomputations of the existence of paths between highly connected (“hub”) managed entities results in faster results when the full graph (including query-derived expansions) needs to be queried and works for large systems, without massive subsecond churn levels.

Query Time vs Pipeline Length. More complex queries would result in longer response times. In order to determine if Loom’s graph extension is effectively useful for speeding up the retrieval of details on demand, Figure 11B shows a set of experiments in which we added groupBy operations on random attributes of the entities (e.g. for VMs. groupBy(region). groupBy(IP). groupBy(OS). groupBy(kernelVersion)...). We fixed the length of the query pipeline and added as many groupBy operations as needed. If the result of a groupBy operation results in one or no groups, a new attribute is picked until the desired length is reached.

In this experiment, the results show how the response time scales sublinearly with the number of operations in the pipeline.

Query Time vs Connectedness. The number of nodes in the graph does not reflect the complexity that arises in traversal-heavy workloads (like the ones that are needed in order to spot how things in our IT infrastructure relate to each other). Edge density, defined as the average ratio of edges per node, is a better metric for how complex relationships in the graph can become. In our set of use cases, the average density of graph modelling IT infrastructure was 3-5.

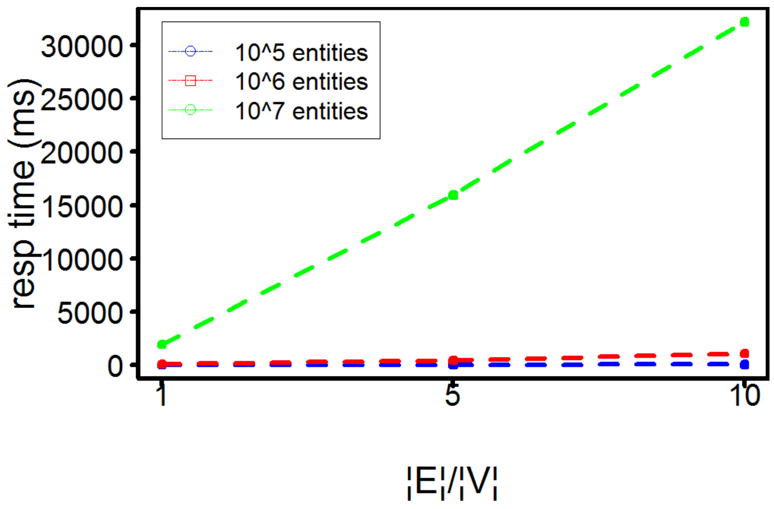


Figure 12: Response time vs. number of entities for different edge densities (1, 5 and 10 as indicated in the legend). 90th percentiles are given.

Figure 12 highlights how edge density is an important factor: as edge density increases, the number of potential paths to follow from every node increases. As most graph queries consist of traversals (e.g. finding paths), the complexity becomes exponential with the number of hops in the traversal. In our experiments, response times are kept $<$ 30s for small graphs (in the order of millions of nodes), increasing the number of connections between nodes results in huge increases in response times.

Figure 13 represents the time spent in calculating relationships vs the number of entities in the synthetic graph we used in these experiments. It shows how as edge density increases (1, 10, and 100), more time is spent on navigation (finding relations between aggregations becomes harder). This situation is worsened as the size of the graph increases. Some deep query pipelines can become non-interactive with 100 million entities with an edge density of 5, but lack of interactivity is more common with 1 billion nodes (for that same edge density) or increasing edge densities (100 million nodes and an edge density of 10 takes a 1-3 minutes to complete).

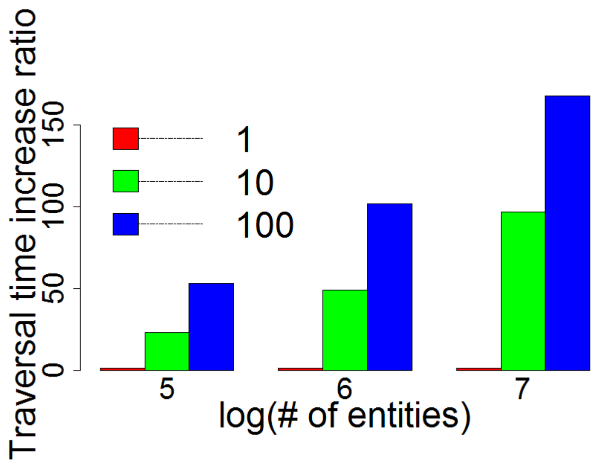


Figure 13: Time to calculate relationships (navigate the graph) (90th percentiles) vs. number of entities depending on edge density (E/V ratios of 1, 10 and 100 as indicated in the legend).

Stitching. We tested Loom's ability to find equivalent managed entities in the IT systems of a large public cloud. Depending on the backend system the data about these entities were coming from (monitoring, cloud management, etc.) physical nodes in the system we referred to by IP, hostname, hostname and region, fully qualified domain name, etc.). This created several aliases for the same real world entities in different systems. Loom was used to unveil the existence of such aliased entities and enable navigation across systems, following graph connections.

In general, our incremental stitching implementation is able to cope with changes in 10% of a 1 million managed entities graph in 100 ms. Related techniques around data deduplication in a single relational database management system take at least 150 ms for a few thousand managed entities [12]).

Obtaining Details. Finding information about an entity of interest is really quick in all the sizes of graph we tested (tens of milliseconds mainly due to network latency) since every UI "Fibre" (element on-screen) is mapped to a node in the expanded graph. This aggregation node in the expanded graph keeps a reference to the entities it contains, which makes obtaining entities of interest almost immediate.

Comparison to Other Systems. Since finding relationships calculation becomes a dominant problem as graph density increases, we performed a comparison with other systems that could perform the same type of queries. In particular, we performed a shortest path query in an extended graph.

In the relational database world, finding how any two managed entities in a multi-partite graph are related boils down to nesting several joins across the tables containing every entity type (e.g. a table for VMs, another for volumes, another for networks, etc.). We used VoltDB as an example of in-memory relational database.

We also loaded the data into Neo4J[15] (modelled as a property graph where each node had an attribute reflecting the type of managed entity).

Figure 14 shows how Loom performs faster than graph databases and in memory relational databases, especially for larger graph sizes. Loom takes $< 30s$ (90^{th} percentile) for all of these queries, while the other alternatives soon become non-interactive (5 and 25 times slower for larger graph sizes).

Memory Consumption. As described above, Loom keeps entities as objects with references to one another. The more attributes an entity contains, the higher the memory consumption (Figure 15). Using a scenario with 100 attributes per node and 5 edges we get an average memory consumption of $< 2KB$ per node.

Figure 16 represents memory consumption as the graph density increases.

5.2. Qualitative Evaluation

Interactive response times at scale on complex related data is not sufficient if the display receiving the results cannot accommodate them in a way that makes sense for the user. In this subsection, we present a lightweight analysis on the usability aspects around the Thread Visualisation Model.

[15]neo4j.com v2.3

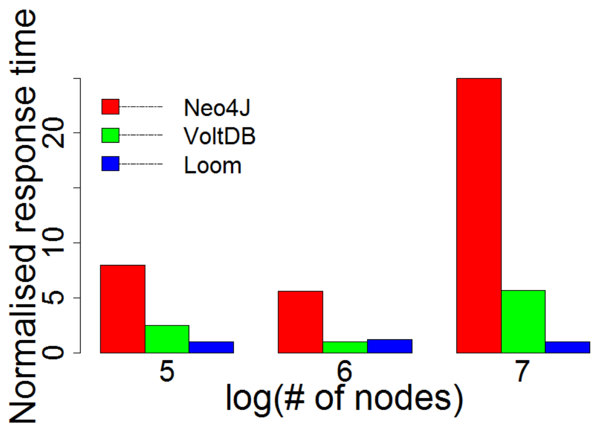


Figure 14: Comparison of performance between Neo4j, VoltDB and Loom when trying to spot relationships between N on-screen elements (N is usually $\leq$ 1000 since data need to be aggregated to fit conventional displays). Response times are normalised to the fastest system. Results represent 90^{th} percentiles

.

5.2.1. Methodology

We conducted semi-structured interviews with a dozen enterprise analysts to better understand their process and needs. We use the term “analyst” to refer to anyone whose primary job function includes working with data to answer questions that inform operational or business decisions. Our interviewees held a number of job titles, including “data analyst”, “data scientist”, “software engineer” and “consultant”.

The organisations they work for are mainly from the IT industry, with a large proportion of large corporations. The analysts ranged from Ph.D. graduates in their first year of work to consultants with 10-20 years of experience.

We recruited interviewees by emailing contacts at organisations within our personal and professional networks. In some cases, we emailed analysts directly. In others, we emailed individuals who forwarded us to analysts within their organisation. This recruitment strategy introduces potential bias in our results. For example, the majority of our interviewees were based in the U.S. and U.K. Our goal here was to get a primer on the reaction of enterprise users to the Loom Thread Model, not to quantify a broad population rating. Other methods and con-

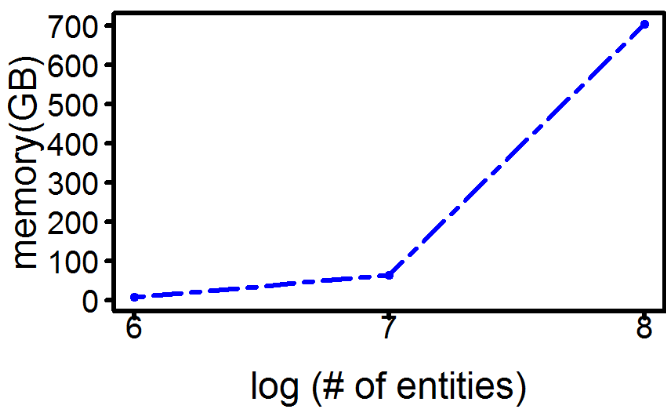


Figure 15: Memory consumption (GB) (90th percentiles) vs. number of entities in the graph for a density of 5.

tinuing studies with an increased number and variety or organisations would be better suited for quantifying our findings.

We began each interview with a quick introduction describing the purpose of the interview and a 5 min demo on what Loom does from the UI perspective. After that we also asked them more practical questions that explore their understanding of relation highlighting: "how is VM 1 related to host #342 in the OpenStack cloud and how many projects are using that project?". A second subset of questions was directed at testing their understanding of the operation pipelines, for instance: "how would you find which VMs run Ubuntu14.04 and visualise their CPU utilisation per geographical zone"?

Each interview lasted around 25 minutes. Whenever possible, we interviewed analysts on location at their job. For interviewees outside of the U.K. we conducted interviews over the phone or via Skype. During the interviews we took extensive notes.

As shown in Table 1, interviewees were asked to respond to a questionnaire containing 8 different assertions covering different UX aspects of the system towards the end of the interview. We posed these during interviews with several customers and partners).

Loom's ability to create aggregations and provide summaries and relationships across them was well-received. The ability to get an overview of the IT infrastructure as a whole ("stitching" across ITIMs) was especially welcome by those with

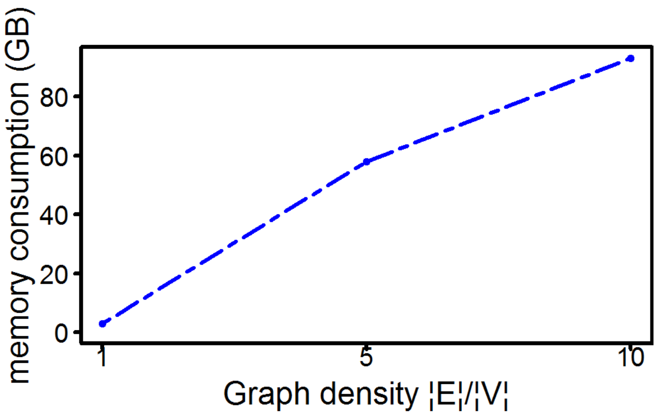


Figure 16: Memory consumption (GB) (90th percentiles) vs. edge density for a 10M entity graph.

Question	**Rating**
”Loom Thread model aggregates information in a useful and meaningful way”	4.1
”Highlighting relationships on aggregations help me identify dependencies in my system”	4.7
”The ability to bring data from different systems into a single screen facilitates my work ”	4.2
”Thread Bundles make it easy to track where an entity belongs in a group as I drill down”	3
”Propagation of alerts and actions to parent “Threads” helps my day to day tasks go faster”	3.6
”The Loom model makes it easy to build custom operations and pipelines”	4.2
”The Loom model eases creation of overviews and filters on the details I am interested in”	4.1
”The Loom thread model is intuitive and easy to use”	3

Table 1: Ratings for several UX features. Data represent median ratings. Ratings are: 1) Very Poor/highly disagree, 2) Poor/disagree, 3) Mild/ not agree nor disagree, 4) Good/Agree, 5) Excellent/Extremely agree.

larger/more complex IT infrastructures and even more so for some roles in network and security operation centres.

Site reliability engineers liked the idea of having alerts and Actions aggregated across ITIMs, as these tend to be their input/output points to solve problems in their infrastructures. Being able to interactively modify a query pipeline (as in Figure 2) so that changes could be applied to some of the entities in the infrastructure was deemed as extremely useful for things such as canary tests and releases.

Administrators and practitioners dealing with smaller scale setups or newer infrastructures found aggregations and relationships less useful and preferred to work on home-brewed solutions (e.g custom shell scripts) as opposed to rely commercial or mainstream open source tools.

Most partners liked the idea of a text-free UI that would let them keep everything under control. Our current UI is different to what administrators are used to. Thus, it comes with a learning curve that shows up in users taking a while to figure out by themselves that highlights were indicating relationships across "Fibres" in our UI. This was especially true when there were children "Threads" derived from a user request for more details on a specific "Fibre". We also found that the topology of the graph determines how intuitive the UI can be: graph topologies containing loops between entities of different types make relationships more difficult to explore and understand (even when our UI includes a "explain relationship" function aimed at showing the path followed to determine if two fibres on-screen are related or not.

However, as some industrial experts have claimed, in complex enterprise environments things can be made simpler, but not simple [13].

Historical queries are a common requirement for many of our partners. Complexity increases even further when time-varying graphs and queries are considered. Also, although aggregated Actions were useful for most of our partners, they mentioned the ability to operate semi-automated remediation Actions.

6. Related Work

6.1. Large-scale Analytics

Overview. SQL databases perform joins in a very efficient manner, but their efficiency decreases when too many nested joins across most or all of the tables in the database are required. Graph databases are designed to shine at this type of nested relational queries that help spot relations, but they are not as good as relational databases for roll-up operations. Loom's in-memory graph model delivers the advantage of relational databases for rolling-up and the speed of graph databases to explore relationships.

Several systems including Dremel [14], PowerDrill [15], Druid [16] or Scuba [17] are oriented towards efficient execution of aggregated queries and visualisation of large distributed databases.

Loom decouples the aggregation from the visualisation layer and enables manipulation of very rich datatypes and displays. Unlike Sketch [18], Loom does not deal with plotting data in charts, but it provides a graph abstraction model where relationships are first class citizens, which also limits potential rendering issues (the abstraction is dynamically adjusted to the size of the display).

Similar to Sketch's Partitioned Data Structures (and unlike Spark's RDDs [19]), the Loom graph is partially memoised [18]. Some other techniques try to accelerate MapReduce to make it work interactively for memory rendering intensive workloads [20]. Loom is optimised for spotting relationships across aggregated data sets keeping interactive response times in exploratory procedures.

There is a wealth of software devoted to gaining insight from IT infrastructures (e.g. HPE's Operation Analytics [21] or Splunk [22]). In many cases they rely on structured queries (often times based on a query language that is close to plain natural English). These solutions typically focus on spotting trends and outliers (as "standing queries") and heavy use of correlation to "explore" unknown patterns. However, they do not offer a tight integration with visualisation capabilities that could help create a customisable overview to reduce the cognitive burden and guide human operators in visual exploration of their data sets. Analytic tools sometimes offer the possibility of personalising a set of charts to create dashboards or associating data sets to pre-defined visualisations (e.g. Splunk Pivot [23]). They fall short at providing a true overview of the whole dataset and relationships at a high level. Moreover, users cannot customise their overview to explore data or arrange them in a more efficient way.

Loom makes heavy use of a variety of aggregations, but it also visualises individual entities and entity properties. This is unlike the idea of restricting visualisations to aggregates for large data that was proposed by imMens [24]; imMens precomputes cubes for faster rendering which is expensive and does not work if the pre-computed cubes are not what the user is asking for. VisReduce [25] also enables computing user-defined aggregation functions, as well as incremental renderings.

Unlike [26] and [27], Loom does not offer any support for incremental visualisation (trading off precision in IT management may lead to powering off the wrong machine or misconfiguring some software) or dynamic query re-writing to reduce query resultset. Since relationship calculation between aggregated elements is a dominant factor in the response times of manageability queries, Loom focuses on trying to resolve these faster (in time scales < 1 min). This is a unique feature, essential for building usable overviews.

Multiple ITIMs. Record linkage, entity reconciliation and entity duplication refer to the problem of identifying "equivalent"/duplicate entities in a dataset [12]. These can be seen as a special type of relationship where two managed entities represent the same real world entity. While advanced machine learning techniques (e.g. similarity or event correlation) are often used to identify duplicates, this is often not necessary for management entities, where introspecting and comparing a few fields may do for many scenarios. In addition, the stitching process needs to be revisited in near time in order to provide accurate information to the visualisation layer, which suggest to use simple "equivalence"-finding functions. Some graph-based configuration ITIMs, like HPE's UCMDB [28] detect duplicated managed entities in their graph during data ingestion by executing similarity functions similar to those used in record linkage systems. None of those systems works across separate datasets with the goal of building a multiplex graph, which is a necessary feature to work seamlessly with data from disparate ITIMs. Loom does build a multiplex graph.

6.2. Parallel Computation and Visualisation

As in Sketch [18], Loom's design decouples the parallel-execution framework from the actual rendering. Unlike Sketch, though, Loom is not a "sort-middle" rendering pipeline [29] since the Loom "Thread" model abstracts details away so that users do not have to deal with conventional chart representations of their data (like plots or barcharts), but it creates blobs that represent aggregations of data. This simplifies rendering on the client side.

Loom also follows the idea of responsive design, by which content is adapted to screen resolution (also followed by Zoomable User Interface works [30]).

Tableau is a commercial visualization platform for tabular data [31]. Sketch [18] expands Tableau capabilities to scenarios where there are many more data points than pixels on the screen, and data exceeds the resources available to a single machine. By providing visualisations on aggregated data, Loom expands the ability of Tableau and Sketch.

6.3. Graph Visualisation

In that regard, most visualisation tools still use the "node-link" diagram model [32, 33, 34, 35, 36, 37, 38, 39, 40, 41]. "Node-link" diagram representation, while most common, may not be the most user-friendly to the general public, nor is it the most pixel-efficient.

There are a myriad of works dealing with graph layouts in 2-D and 3-D representations. As the graph size increases, however, edge crossings increase and node occlusion becomes significant, hiding potential patterns that can be in existence. Graph abstraction methods apply graph visualisation simplification algorithms [42].

Graph visualisation simplifications tend to exploit the redundancy in graph topology relying on elements keeping a structural equivalence [43, 44]. Graph "compression" methods do not work well with small world graphs and this is where graph clustering algorithms (hierarchical and spectral clustering are examples of this) are well-studied for abstract representation of graphs (see [42] for review).

An alternative for graph visualization abstraction is semantic abstraction. It mainly depends on node/edge attributes to create a super graph to explain or complement the original large graph visualisation. For instance, PivotGraph enabled aggregation operations by selecting a value the graph nodes with the same value on one or two attributes into node aggregations [33]. Finally, some other approaches focus on edge/node filtering and bundling [42]. Like the work by van den Elzen and van Wijk [45], Loom enables dynamic exploration and understanding of multivariate networks based on node/edge attributes and network topology. Loom also supports filtering operations that help reduce the amount of information on the screen.

Loom builds on techniques for visual abstraction of large graphs. Like OnionGraph [46], Loom does not prescribe the abstraction techniques that can be applied (Loom supports redundancy, clustering or semantic abstraction on node or edge attributes). Unlike any of the previous approaches, Loom does not apply classic drawing algorithms but it presents a new visualisation paradigm for graph abstraction: the Loom "Thread" model, which relieves the UI client from the need of rendering complex edge layouts and reduces screen clutter by minimising text and edge crossover (relationships shown as highlights).

As large graphs are often abstracted into hierarchical structures for visualisation, the most relevant interactions are on the manipulation of graph hierarchies [42]. Elmqvist and Fekete [47] classified the hierarchical abstraction based visualisation into several types: above traversal, below traversal, level traversal, range traversal and unbalanced traversal. Like ASK-View [48], Loom lets users select an aggregation and expand with any traversal type, which is one of the most flexible approaches to hierarchical interaction (see [49] for review). Like Fisheye [32], Loom pre-computes the next level in the hierarchy (although the pre-computation is on the abstract representation of data for Loom, unlike Fisheye, which focuses on rendering pre-computation). Like in the work by Van Ham and Perer [32], Loom includes a method to start the graph analysis from free text searches.

6.4. Contemporary Management User Interfaces

Contemporary management interfaces let users operate on their infrastructure by exposing familiar tables (see OpenStack's[16] Horizon [50] or Amazon's AWS [51] Dahsboards), trees or graphs (like Cisco's UCS [52]) or cylinders offering a 3d representation of the machines (like HPE's Insight CMU [53]). Current management interfaces do not scale well beyond a few tens of managed entities and often do not deal with custom aggregation of information. Some systems (like Netflix's Atlas, InfluxDB or Vertica) excel when it comes to aggregating time series of numeric data.

Systems like Nutanix[17] are praised for their high UX quality, but they focus on predefined arrangements that do not encourage customisable overviews and visualisations and there are concerns about their ability to scale to millions of managed entities. Some visualisation systems, like Tableau[18], query relational databases, cubes, cloud databases, and spreadsheets and then generate a number of graph types that can be combined into dashboards. They do not provide users with embedded near-time analytics functions to reduce cognitive overload in their visual exploration of the data and it is complicated for a user to compose their own view of the world and spot relationships across managed entities in a holistic manner. Generally, information dashboards do not provide the capability to effect change on the underlying systems via the visualisations.

6.5. Schema Reconciliation

Industry leaders in retail or travel, like Amadeus [54], export a very well defined schema that partners wanting to use their platforms need to comply with. Loom uses a similar philosophy, but it relies on a very flexible graph-based schema so that users only need to create entity-relationship models of their data to create a multi-partite graph.

6.6. Graph Density

IT systems are not typically dense graphs since they tend to be part of a hierarchy (e.g. datacentre network architectures, hardware allocation in enclosures, racks, regions, data centres, etc.). In some cases there is not a single hierarchy, but several of them. For instance, microservices architectures can be modelled as a forest including several hierarchical trees.

[16]One of the most popular open source cloud management frameworks.
[17]http://www.nutanix.com/
[18]www.tableau.com

6.7. Usability

Kandel et al. [10] challenge most assumptions about current UI work. For instance, regarding the need for interactive queries: “future work to examine how low-latency query processing over data subsets of various resolutions impact both the quantity and quality of analysis”. Loom helps bridge the gap between “hacker” users and application users so that the latter can benefit from the tech knowledge and deep coding skills of the former, which result in data curation. Our data indicate that Loom’s enabled user experience improves the quantity of analysis and users can do more with less.

None of the above systems consider applying interactive visual exploration techniques to manageability data and they do not provide a comprehensive solution for critical elements for ITIMs, like Alerts and Actions.

7. Conclusions

To the best of our knowledge, Loom is the first example of a system that builds on and expands interactive visual exploration techniques to tame complexity in large IT infrastructures. Loom delivers in situ analysis capabilities on data from different systems and brings user-driven data summarisation and reduction, thus empowering users to find insights using interactive exploratory analyses, even on small factor touch-enabled displays.

Loom’s graph is dynamically extended to include query results and graphs from different ITIMs can be queried as if they were one, making it easy to spot relationships between aggregations of managed entities. The “Thread” visualisation model is capable of displaying overviews on complex IT infrastructures and details on entities of interest at the same time for large installations. Actions and Alerts are first-class citizens in Loom. Loom handles tens of millions of managed entities from different systems in interactive response times in touch-enabled screens.

Partners found the creation of custom multiplex overviews especially useful in situations other solutions cannot cope with in an interactive manner. Loom has proven its usefulness in large complex installations with many ITIMs.

We plan to expand Loom’s support for historical (time-based) queries and push the UX limits to smaller form factor devices, relying on heavier automation of some of the most common tasks administrators and site reliability engineers do.

8. Author Contribution

Authors were ordered alphabetically. Their specific contributions are: J.B., E.D., J.G., R.H, M.L, R.P., S.S.L., L.M.V., and L.W. contributed to the original idea, software design and implementation. L.M.V. and F.C. wrote the paper, analysed state of the art and designed experiments for this paper. L.M.V. tuned Loom and performed the experiments for this paper.

[1] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, J. Wilkes, Large-scale cluster management at Google with Borg, in: Proceedings of the European Conference on Computer Systems (EuroSys), Bordeaux, France, 2015.

[2] J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, 1st Edition, Addison-Wesley Professional, 2010.

[3] N. Ayachitula, M. Buco, Y. Diao, S. Maheswaran, R. Pavuluri, L. Shwartz, C. Ward, It service management automation - a hybrid methodology to integrate and orchestrate collaborative human centric and automation centric workflows, in: Services Computing, 2007. SCC 2007. IEEE International Conference on, 2007, pp. 574–581. doi:10.1109/SCC.2007.75.

[4] J. J. Thomas, K. A. Cook, Illuminating the Path: The Research and Development Agenda for Visual Analytics, National Visualization and Analytics Ctr, 2005.
URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0769523234

[5] P. C. Wong, H.-W. Shen, C. R. Johnson, C. Chen, R. B. Ross, The top 10 challenges in extreme-scale visual analytics, IEEE computer graphics and applications 32 (4) (2012) 63–67.

[6] P. C. Wong, H.-W. Shen, C. Chen, Top Ten Interaction Challenges in Extreme-Scale Visual Analytics, Springer London, London, 2012, pp. 197–207. doi:10.1007/978-1-4471-2804-5_12.
URL http://dx.doi.org/10.1007/978-1-4471-2804-5_12

[7] B. Craft, P. Cairns, Beyond guidelines: What can we learn from the visual information seeking mantra?, in: Proceedings of the Ninth International

Conference on Information Visualisation, IV '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 110–118. doi:10.1109/IV.2005.28.
URL http://dx.doi.org/10.1109/IV.2005.28

[8] E. Wu, L. Battle, S. R. Madden, The case for data visualization management systems: Vision paper, Proc. VLDB Endow. 7 (10) (2014) 903–906. doi:10.14778/2732951.2732964.
URL http://dx.doi.org/10.14778/2732951.2732964

[9] D. Fisher, Big data exploration requires collaboration between visualization and data infrastructures, in: Proceedings of the Workshop on Human-In-the-Loop Data Analytics, HILDA '16, ACM, New York, NY, USA, 2016, pp. 16:1–16:5. doi:10.1145/2939502.2939518.
URL http://doi.acm.org/10.1145/2939502.2939518

[10] S. Kandel, A. Paepcke, J. M. Hellerstein, J. Heer, Enterprise data analysis and visualization: An interview study, IEEE Transactions on Visualization and Computer Graphics 18 (12) (2012) 2917–2926. doi:http://doi.ieeecomputersociety.org/10.1109/TVCG.2012.219.

[11] B. Shneiderman, The eyes have it: A task by data type taxonomy for information visualizations, in: Proceedings of the 1996 IEEE Symposium on Visual Languages, VL '96, IEEE Computer Society, Washington, DC, USA, 1996, pp. 336–.
URL http://dl.acm.org/citation.cfm?id=832277.834354

[12] A. Gruenheid, X. L. Dong, D. Srivastava, Incremental record linkage, Proc. VLDB Endow. 7 (9) (2014) 697–708. doi:10.14778/2732939.2732943.
URL http://dx.doi.org/10.14778/2732939.2732943

[13] R. Hoekman Jr., Experience Required: how to become a UX leader regardless of your role, 1st Edition, New Riders, USA, 2016.

[14] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, T. Vassilakis, Dremel: Interactive analysis of web-scale datasets, Proc. VLDB Endow. 3 (1-2) (2010) 330–339. doi:10.14778/1920841.1920886.
URL http://dx.doi.org/10.14778/1920841.1920886

[15] A. Hall, O. Bachmann, R. Büssow, S. Gănceanu, M. Nunkesser, Processing a trillion cells per mouse click, Proc. VLDB Endow. 5 (11) (2012) 1436–1446. doi:10.14778/2350229.2350259.
URL http://dx.doi.org/10.14778/2350229.2350259

[16] F. Yang, E. Tschetter, X. Léauté, N. Ray, G. Merlino, D. Ganguli, Druid: A real-time analytical data store, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD '14, ACM, New York, NY, USA, 2014, pp. 157–168. doi:10.1145/2588555.2595631.
URL http://doi.acm.org/10.1145/2588555.2595631

[17] L. Abraham, J. Allen, O. Barykin, V. Borkar, B. Chopra, C. Gerea, D. Merl, J. Metzler, D. Reiss, S. Subramanian, J. L. Wiener, O. Zed, Scuba: Diving into data at facebook, Proc. VLDB Endow. 6 (11) (2013) 1057–1067. doi:10.14778/2536222.2536231.
URL http://dx.doi.org/10.14778/2536222.2536231

[18] M. Budiu, R. Isaacs, D. Murray, G. Plotkin, P. Barham, S. Al-Kiswany, Y. Boshmaf, Q. Luo, A. Andoni, Interacting with large distributed datasets using sketch, in: Proceedings of Eurographics Symposium on Parallel Graphics and Visualization (EGPGV16), 2016.

[19] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster computing with working sets, in: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 10–10.
URL http://dl.acm.org/citation.cfm?id=1863103.1863113

[20] H. T. Vo, J. Bronson, B. Summa, J. L. Comba, J. Freire, B. Howe, V. Pascucci, C. T. Silva, Parallel visualization on large clusters using mapreduce, in: Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on, IEEE, 2011, pp. 81–88.

[21] Hp operations analytics: a new analytics platform to support the transformation of it (2013).
URL http://www.hp.com/hpinfo/newsroom/press_kits/2013/HPDiscover2013/EMA_OpsAnalytics_Whitepaper_Final.PDF

[22] Splunk (2015).
URL http://www.splunk.com

[23] Pivot manual: Introduction to pivot (2015).
URL `http://docs.splunk.com/Documentation/Splunk/6.2.3/Pivot/IntroductiontoPivot`

[24] Z. Liu, B. Jiang, J. Heer, immens: Real-time visual querying of big data, in: Proceedings of the 15th Eurographics Conference on Visualization, EuroVis '13, The Eurographs Association &#38; John Wiley &#38; Sons, Ltd., Chichester, UK, 2013, pp. 421–430. `doi:10.1111/cgf.12129`.
URL `http://dx.doi.org/10.1111/cgf.12129`

[25] J. F. Im, F. G. Villegas, M. J. McGuffin, Visreduce: Fast and responsive incremental information visualization of large datasets, in: Big Data, 2013 IEEE International Conference on, 2013, pp. 25–32. `doi:10.1109/BigData.2013.6691710`.

[26] J.-D. Fekete, ProgressiVis: a Toolkit for Steerable Progressive Analytics and Visualization, in: 1st Workshop on Data Systems for Interactive Analysis, Chicago, United States, 2015, p. 5.
URL `https://hal.inria.fr/hal-01202901`

[27] L. Battle, M. Stonebraker, R. Chang, Dynamic reduction of query result sets for interactive visualizaton, in: Big Data, 2013 IEEE International Conference on, 2013, pp. 1–8. `doi:10.1109/BigData.2013.6691708`.

[28] Data sheet: HP Universal CMDB and HP UCMDB Configuration Manager (2015).
URL `http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA1-6156ENW.pdf`

[29] T. W. Crockett, An introduction to parallel rendering, Parallel Comput. 23 (7) (1997) 819–843. `doi:10.1016/S0167-8191(97)00028-8`.
URL `http://dx.doi.org/10.1016/S0167-8191(97)00028-8`

[30] K. Perlin, D. Fox, Pad: An alternative approach to the computer interface, in: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93, ACM, New York, NY, USA, 1993, pp. 57–64. `doi:10.1145/166117.166125`.
URL `http://doi.acm.org/10.1145/166117.166125`

[31] C. Stolte, P. Hanrahan, Polaris: A system for query, analysis and visualization of multi-dimensional relational databases, in: Proceedings of the IEEE

Symposium on Information Vizualization 2000, INFOVIS '00, IEEE Computer Society, Washington, DC, USA, 2000, pp. 5–.
URL `http://dl.acm.org/citation.cfm?id=857190.857686`

[32] E. R. Gansner, Y. Koren, S. C. North, Topological fisheye views for visualizing large graphs, IEEE Transactions on Visualization and Computer Graphics 11 (4) (2005) 457–468. `doi:10.1109/TVCG.2005.66`.
URL `http://dx.doi.org/10.1109/TVCG.2005.66`

[33] M. Wattenberg, Visual exploration of multivariate graphs, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '06, ACM, New York, NY, USA, 2006, pp. 811–819. `doi:10.1145/1124772.1124891`.
URL `http://doi.acm.org/10.1145/1124772.1124891`

[34] A.-S. Dadzie, M. Rowe, Approaches to visualising linked data: A survey, Semant. web 2 (2) (2011) 89–124. `doi:10.3233/SW-2011-0037`.
URL `http://dx.doi.org/10.3233/SW-2011-0037`

[35] V. Peysakhovich, C. Hurter, A. Telea, Attribute-driven edge bundling for general graphs with applications in trail analysis, in: 2015 IEEE Pacific Visualization Symposium (PacificVis), 2015, pp. 39–46. `doi:10.1109/PACIFICVIS.2015.7156354`.

[36] A. Graves, Techniques to reduce cluttering of rdf visualizations, Future Gener. Comput. Syst. 53 (C) (2015) 152–156. `doi:10.1016/j.future.2014.11.005`.
URL `http://dx.doi.org/10.1016/j.future.2014.11.005`

[37] W. Didimo, F. Giacchè, F. Montecchiani, Kojaph: Visual Definition and Exploration of Patterns in Graph Databases, Springer International Publishing, Cham, 2015, pp. 272–278. `doi:10.1007/978-3-319-27261-0_23`.
URL `http://dx.doi.org/10.1007/978-3-319-27261-0_23`

[38] R. Pienta, J. Abello, M. Kahng, D. H. Chau, Scalable graph exploration and visualization: Sensemaking challenges and opportunities, in: 2015 International Conference on Big Data and Smart Computing, BIGCOMP 2015, Jeju, South Korea, February 9-11, 2015, 2015, pp. 271–278. `doi:10.1109/35021BIGCOMP.2015.7072812`.
URL `http://dx.doi.org/10.1109/35021BIGCOMP.2015.7072812`

[39] N. Bikakis, T. K. Sellis, Exploration and visualization in the web of big linked data: A survey of the state of the art, CoRR abs/1601.08059.
URL `http://arxiv.org/abs/1601.08059`

[40] M. van der Zwan, V. Codreanu, A. Telea, Cubu: Universal real-time bundling for large graphs., IEEE Transactions on Visualization and Computer Graphics PP (99) (2016) 1–1. `doi:10.1109/TVCG.2016.2515611`.

[41] W. Q. Wang, Z. Cai, K. Zhang, Visualizing big graphs with labels through edge bundling, in: 2016 IEEE International Conference on Big Data Analysis (ICBDA), 2016, pp. 1–5. `doi:10.1109/ICBDA.2016.7509837`.

[42] Y. Hu, L. Shi, Visualizing large graphs, Wiley Interdisciplinary Reviews: Computational Statistics 7 (2) (2015) 115–136. `doi:10.1002/wics.1343`.
URL `http://dx.doi.org/10.1002/wics.1343`

[43] L. Shi, Q. Liac, X. Sun, Y. Chen, C. Lin, Scalable network traffic visualization using compressed graphs, in: Big Data, 2013 IEEE International Conference on, 2013, pp. 606–612. `doi:10.1109/BigData.2013.6691629`.

[44] C. Dunne, B. Shneiderman, Motif simplification: improving network visualization readability with fan, connector, and clique glyphs, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2013, pp. 3247–3256.

[45] S. van den Elzen, J. J. van Wijk, Multivariate network exploration and presentation: From detail to overview via selections and aggregations, IEEE Transactions on Visualization and Computer Graphics 20 (12) (2014) 2310–2319. `doi:10.1109/TVCG.2014.2346441`.

[46] L. Shi, Q. Liao, H. Tong, Y. Hu, Y. Zhao, C. Lin, Hierarchical focus+context heterogeneous network visualization, in: 2014 IEEE Pacific Visualization Symposium, 2014, pp. 89–96. `doi:10.1109/PacificVis.2014.44`.

[47] N. Elmqvist, J.-D. Fekete, Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines, IEEE Transactions on Visualization and Computer Graphics 16 (3) (2010) 439–454. `doi:10.1109/TVCG.2009.84`.
URL `http://dx.doi.org/10.1109/TVCG.2009.84`

[48] J. Abello, F. V. Ham, N. Krishnan, Ask-graphview: A large scale graph visualization system, IEEE Transactions on Visualization and Computer Graphics (2006) 2006.

[49] Visual Analysis of Large Graphs.
URL http://visualanalytics.de/sites/default/files/upload/publications/egstar10.pdf

[50] Openstack dashboard (2015).
URL https://wiki.openstack.org/wiki/Horizon

[51] Amazon web services management console (2015).
URL http://aws.amazon.com/console/

[52] Cisco unified computing system (2015).
URL https://en.wikipedia.org/wiki/Cisco_Unified_Computing_System

[53] White paper: HP Insight CMU (2015).
URL http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA2-5035ENW.pdf

[54] Amadeus (2015).
URL http://www.amadeus.com