LiveRAC: Interactive Visual Exploration of System Management Time-Series Data

Peter McLachlan, Tamara Munzner
University of British Columbia
Vancouver, BC V6K 3Z9
Canada
{spark343,tmm}@cs.ubc.ca

Eleftherios Koutsofios, Stephen North
AT&T Labs - Research
Florham Park, NJ 07932
USA
{ek,north}@research.att.com

ABSTRACT
We present LiveRAC, a visualization system that supports the analysis of large collections of system management time-series data consisting of hundreds of parameters across thousands of network devices. LiveRAC provides high information density using a reorderable matrix of charts, with semantic zooming adapting each chart's visual representation to the available space. LiveRAC allows side-by-side visual comparison of arbitrary groupings of devices and parameters at multiple levels of detail. A staged design and development process culminated in the deployment of LiveRAC in a production environment. We conducted an informal longitudinal evaluation of LiveRAC to better understand which proposed visualization techniques were most useful in the target environment.

ACM Classification Keywords
H.5.m Information Interfaces and Presentation: *Miscellaneous*

General Terms
Design, Human Factors

Author Keywords
Visualization, Time-series data, Longitudinal studies

INTRODUCTION
Time-series data is pervasive in science, engineering and business. Visualization helps people interpret data by exploiting human perception to reduce cognitive load. Statistical graphics, most notably line charts of time-value pairs, are heavily used for inspecting individual or small sets of time-series. However, understanding large collections of time-series data remains difficult. We selected large-scale system management as a domain where people need to understand large sets of time-series data at multiple levels of detail, and with respect to frequently changing groupings.

Data warehouses for managed hosting services can store details about tens of thousands of physical and virtual servers. For each system, parameters such as CPU load and memory usage are regularly logged. This data may be archived for multiple years. System management staff must be able to query detailed data to attend to the needs of individual customers, while maintaining awareness of the managed environment's global state.

Unfortunately, there is a gap between the capabilities of tools for gathering and storing such data, and the capabilities of tools to help systems managers to comprehend and act on it. A heavily used analysis method is to inspect charts showing the state of one parameter on one server. Inspecting such charts one at a time yields only a fragmented, low-level view of the environment. The limitations of human working memory make it difficult to synthesize a useful overview of the entire collection.

Many commercial systems management tools provide high-level dashboards with summary statistics, but these hide the true complexity of the system state. For instance, most use thresholds that rate the monitored state as healthy if some percentage of systems are available and responding. However, a data center may be unhealthy if just one critical system is down, or it may still be healthy if many systems are down for routine maintenance. Moreover, these tools do not allow sufficient drill-down exploration to develop and confirm hypotheses about root causes and their remedies. To move beyond this, visualization tools should help system managers to quickly analyze logged data at scales between single low level charts and high level summary dashboards.

Our primary contribution is the design methodology and implementation of LiveRAC, a visualization system for browsing and correlating time-series data with high information density. It scales to dozens of parameter channels across thousands of devices. LiveRAC allows side-by-side visual comparison at multiple levels of detail with respect to groupings of devices and parameters, using a reorderable matrix of charts. The charts support semantic zooming, providing visual representations adapted to the available space. Designing and deploying a research visualization system to operations staff in a large corporation, who must deliver reliable production service, is an interesting and ambitious methodological challenge. We adapted the standard user-centred design process in a staged model, with requirements gathering and prototype development in each phase. We demonstrated the utility of our ideas with increasingly high-fidelity prototypes, gaining buy-in from individuals within the organization that controlled access to production data and end users. We discuss the concomitant challenges and insights gained by moving from a proof-of-concept implementation of visualization techniques, to a deployed system for highly proprietary data in a large organization.

We describe two secondary contributions. The first is a novel time-series visualization system that can render a visual representation for millions of data points. The second is an informal field study of LiveRAC in a production setting with operational systems management staff, to determine whether our design choices provided the intended functionality in a production setting.

RELATED WORK

We review the previous work on visualization techniques, system management visualization systems, and studies of visualization systems deployed in the field.

LiveRAC is one of many visualization systems that use matrix layouts [20, 22]. Interactive re-orderable matrix visualizations were introduced by Bertin [2].

Semantic zooming [13] is a technique where objects adapt their visual representation to the available screen space. For example, a calendar object, when limited to a small area, might display only a condensed text list of high priority events, but if the region is enlarged, it might show the familiar calendar month view. LiveRAC is the first system to integrate semantic zooming with the accordion drawing technique described below.

Stretch and squish navigation [17] is an interaction metaphor where the user manipulates the display as though it was a rubber sheet tacked down at the borders. Expanding one region compresses the rest of the view. Accordion drawing [11] is a scalable approach to stretch and squish navigation that supports the guaranteed visibility of marked items, even within dense regions. LiveRAC uses the PRISAD infrastructure for efficient generic rendering and navigation in data sets with millions of items [21].

Time-series data is an area of extensive study in information visualization research. Aigner [1] examined the diversity of methods for visualizing time-oriented data. Several systems for visualizing time-series data use clustering for exploratory analysis, where the goal is to find interesting groupings of the data considered as a large unordered collection [6, 8, 25]. These approaches do not solve the system management tasks of inspecting individual time-series curves and finding correlations across specific sets of devices, parameters, and time. Similarly, TimeSearcher [5] supports time-based pattern finding, but does not scale sufficiently.

The SWIFT system [7] is a set of data storage, aggregation and visualization tools that provide an integrated interface to system management data from many distinct sources, with a single self-describing data format. SWIFT collects data continuously, and performs processing such as computing weighted rolling averages and aggregates. SWIFT visualizations include geographic, node-link, and table layouts. However, large sets of correlated time-series displays are not well supported by any of these visualizations. LiveRAC provides a new visualization front-end for SWIFT data, showing several levels of structure between high-level dashboards and low-level tables.

Although many visualization systems have been studied in laboratory settings, far fewer have been studied with a longitudinal approach in the field. Multi-dimensional In-depth Long-term Case Studies (MILCs) [19] are an emerging visualization evaluation research method proposed by Shneiderman et al. Our approach with LiveRAC is a variant of a MILC with a focus on gaining organizational credibility through prototype iteration. The work of González and Kobsa is illuminating: the promising results of an initial study [4] were followed by a longer longitudinal study, showing that achieving adoption was a difficult challenge [3]. Our goal with LiveRAC was to gain more insight about the use of deployed visualization systems in the field, moving us to continue design and development well beyond the initial technique demonstration phase.

SYSTEM MANAGEMENT

We describe system management professional roles and activities, and the limitations of existing tools in supporting this user population.

Roles and activities

The role of a system management professional in managed hosting services is to meet service level agreements that have been established between the hosting provider and its customers. These agreements specify all significant aspects of the services to be provided, including: what services will be provided, how they will be delivered, how service delivery will be measured, and the consequences if the service agreement is not met. Fulfilling these agreements is the main task of system management professionals. Delivering reliable production service requires a combination of business, analytical and system management skills.

To manage complexity, system management professionals are divided into multiple response tiers. The lower tiers work on very tight deadlines (in some cases, only minutes) and in highly structured environments. The tools they use are tightly constrained in functionality and process. In contrast, our target users, Life Cycle Engineers (LCEs), are senior operations staff at the topmost tier who cover several customer accounts. LCEs have a more analytical, less reactive role than lower tiers, and they seemed more likely to need the exploratory capabilities of a visualization tool such as LiveRAC.

Interpreting network environment status is the fundamental system management activity. It involves understanding the state of individual devices, and how these states affect overall end-to-end services. LCEs examine time-series data such as alarm records, statistics and log files. The LCEs in our study use SWIFT for this activity. More generally, systems management professionals employ various Network Management System (NMS) tools such as OpenNMS [1], HP OpenView [2], and BMC Patrol [3]. Most NMS tools integrate alarms, statistics and logs from multiple managed systems into a central database, and provide views of this data.

Report generation requires creating a sharable document showing relevant time-series values. The LCEs we studied obtain these reports from SWIFT by making a screen shot of charts of interest, or by copying HTML tables of individual time-series. Most NMS tools can run pre-built queries, such as reports of the top ten most heavily loaded systems over the past week. They may also support a custom query language or a form-based query builder.

Capacity planning involves forecasting future system and infrastructure needs. An LCE participant summarized its three phases this way: *"Phase 1 documents what [customers] have today, phase 2 provides short-term recommendations (one to three months) for things like adding memory, changing the way [customers] do their backups, etc. and phase 3 is long-term where we provide recommendations for things such as swapping out hardware, changing application versions to [ones] that are certified, maybe going to blade servers to reduce their footprint to fit more servers in their existing space."* Capacity planning depends on having an accurate and complete understanding of the network environment state captured by the NMS, and external knowledge about a customer's activities and projects. For example, knowing that a major upcoming event will increase server load, a life cycle engineer will decide if upgrades are required, or if the current system should be adequate.

Event investigation concerns specific events such as a service outage or network security breach. Events that have been escalated to the level of LCE scrutiny often have significant business impact. There are three distinct phases of event investigation. The first is notification of the event, usually by an automatically generated alarm, or a ticket submitted by a customer. In the second phase, the LCE uses the NMS to study time-series data parameters of the devices in question, and possibly to see if similar conditions occurred with other devices. In the third phase, the LCE directs a response, often the deployment of further forensic tools such as network sniffers or rootkit detectors.

Coordination between customers, engineering and operations involves communicating findings, change requests and recommendations. An LCE participant described this as "working as a conduit" to facilitate service delivery and incident response.

[1]`www.opennms.org`
[2]`www.managementsoftware.hp.com`
[3]`www.bmc.com`

Limitations of current tools

Existing NMS tools for system management professionals have significant limitations in the analysis of logged time-series data. The most critical limitation is the lack of mid-level views of environment status. Numeric summaries, such as counts of devices in various groups that are up or down, overall network utilization, and high priority alarms, do provide useful information to administrators and managers. However, these views are incomplete, and hide many important details, such as how close these systems are to specific thresholds, which particular systems are offline, or which systems are experiencing problems. Finding details such as the identities of the offline systems requires drilling down to another page, or scrolling down within one very large page. These tools give poor support to tasks requiring direct comparisons between many systems, for example, determining whether behavioral events such as load spikes or alarms affected several systems simultaneously.

Many NMS tools use rule-based and machine learning approaches to infer if there is a problem and send alarm notifications. In practice, these approaches yield an overwhelming number of non-actionable alarms. For example, in the environment we studied, the monitoring system generates around 10,000 alarms daily, with only one or two per day representing actionable events. Blanket suppression rules are used to automatically delete large volumes of alarms. These suppression rules suggest that system managers place low confidence in individual alarms or metrics, relying instead on their own analyses of logged data.

Current report generation tools that use pre-built or custom queries from the NMS also have limitations. Results from pre-built and custom reports are usually in a table or a scrollable page of charts, with no summaries or aggregation, so it is difficult to make comparisons or acquire a complete view of the environment state. Pre-built queries, such as "show the 10 systems with the highest CPU in the last 6 days" may not match task requirements. Even customized queries require that users *know in advance what they are looking for*, and thus are ineffective for discovering unexpected patterns.

LiveRAC was designed to address these three limitations in an interactive visualization system. It gives system managers views of data at multiple levels, allowing them to draw inferences from the complex reality of the data, and to communicate their interpretations to others.

ITERATIVE DESIGN

We describe our design methodology, the participants we recruited, and the four phases of our design process.

Methodology

Our project followed many aspects of a standard user-centred design process. We gathered requirements, then built and obtained feedback on a series of prototypes. We started with paper prototypes, continued to a proof-of-concept interactive software prototype using synthetic data, then to a high-fidelity prototype running on real data, and finally to a deployable system. We found that recruiting senior system management participants who have operational duties is a challenge in a corporate setting where involvement in a research project requires higher approval, and the routine workload is very demanding. We modified the conventional protocol by increasing our participant pool in stages as the project evolved, generating interest by means of interim results. Each working prototype increased credibility for the project, leading to buy-in from the next group within the organization that had closer access to production data and real users. In each succeeding phase, we were able to work with a larger pool of participants closer to the target user group, culminating in direct contact with system management practitioners, the LCEs. We gathered additional requirements at each stage. Our staged design process had four discrete phases of requirements gathering and prototype refinement.

We collected data about our target users by performing interviews with senior system management professionals, attending management-level meetings, recording sessions of users interacting with LiveRAC via a desktop sharing tool, and by collecting logs and surveys. Our meetings and interviews focused on the roles and activities of system management professionals, how ticketing and problem resolution processes work, how capacity planning is done, what types of insights were enabled by their current system, and how they communicate them.

Participants

Our design process involved 14 participants in three groups. The external group consisted of a single participant outside the target organization, who had senior system management experience as the CTO of a small company (C1).

The internal group consisted of senior technical personnel in our target organization: a member of a tools engineering team (T1), one executive director (E1), and four senior technical directors (D1, D2, D3, D4). All were intimately familiar with our target users; the directors managed them, and the engineer developed tools for them. This group participated in design meetings and gave feedback on prototypes. All members of this group except for E1 did use the LiveRAC system, but because they were not our target user population we did not perform individual interviews with them.

The LCE group consisted of seven Life Cycle Engineers (L1 to L7), our target user group. They participated in design meetings, responded to surveys, and tested the deployed LiveRAC system in logged sessions. Four participants, L1 to L4, took part in additional training, interviews, and email communication.

Many participants work at geographically distributed locations, so meetings were facilitated by screen sharing and teleconferencing.

Design phases

We began the first phase with initial requirements gathering. We interviewed C1 to gather information on the challenges of system management, and the tools commonly used. Also, we monitored our own local network with the freely available enterprise system management platform OpenNMS to to gain direct experience with such tools and data. We were able to draw on our own experience in this domain: one author has several years of professional experience in system management and architecture, and two others have built and deployed data integration and visualization systems in close collaboration with system management practitioners [7].

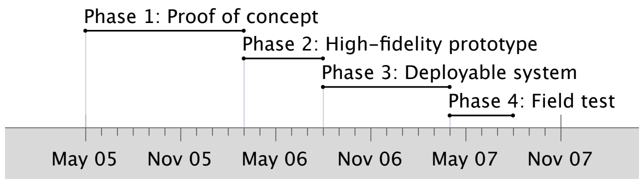


Figure 1. LiveRAC design timeline

We designed a prototype based on these initial requirements, choosing a combination of techniques supported by several visualization principles. After refining our ideas with paper prototypes, we built a proof-of-concept software prototype running on a synthetic data set. This prototype was presented to T1, a designer of software tools that our target LCE population uses daily. T1 confirmed the prototype was suitable, and was sufficiently encouraged by the potential of its interactive visual queries that he agreed to participate in the next design phase.

In the second phase, we refined the requirements through meetings with T1. We developed a high-fidelity prototype running on the production database, and collected feedback from T1 regularly through the iterative development process. This second prototype was presented to E1 and D1, who were enthusiastic about the ability to see the system management environment at a level of detail not previously possible, and became participants in the next phase.

In the third phase, we gathered requirements in meetings with E1 and D1, in addition to T1. We developed a third high-fidelity prototype, with the goal of realizing a flexible and robust deployable system for use by LCEs. We presented the resulting system to D2, D3, and D4, in addition to T1, E1, and D1. Their approval gave us direct access to the target users, LCEs L1 through L7, for the study's final phase.

In the fourth phase, we gathered requirements from L1-L7 through meetings and surveys, and further refined our system to incorporate their requirements and feedback. We then deployed the system in a production environment, continuing to gather feedback from all fourteen participants. We carried out an informal longitudinal evaluation over three months. Although managers E1 and D1 encouraged their employees to try the system, participation was not mandatory. We collected logs of 38 sessions from a total of 13 LiveRAC system users. Of these, four were directors, and the others worked in technical roles. LCEs L1-L4 were our most active users, with a total of 24 logged sessions between the four of them.

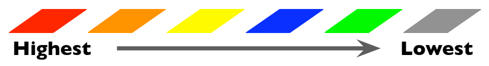


Figure 2. LiveRAC default threshold legend. LiveRAC cells are colored according to where the cell's computed value falls on the threshold scale.

DESIGN REQUIREMENTS

We identified and validated design requirements in discrete stages through an iterative design process.

In the first stage, for the proof-of-concept prototype, we distilled an overarching high-level requirement based on our discussions with C1 and our experience with using the OpenNMS software: Users needed to browse and correlate many instances of parameter, device, and time values. The data set contained dozens of parameters for thousands of devices, collected at 5 minute intervals over a period of years. The data was to be viewed in temporal windows at multiple levels: hours, days, weeks, months, and years. The specific sets of devices and parameters requiring investigation were highly variable. In some cases, investigation started with some grouping of devices; in others by finding critical values for one or many parameters. A particular challenge was to help users correlate alarms with other parameters.

In the second stage, for the high-fidelity prototype, we identified and addressed these requirements:

- search for specific devices by name or metadata
- provide a client which can run on corporate standard desktop hardware, handle large data volume by using a back-end server which manages the collection and storage of many years' worth of data
- dynamically change the time window
- map from parameter and device to visual representation at startup time
- provide shortcuts for resizing standard and dynamically changeable groupings to ease navigation

In the last stage, for the deployable system, the additional requirements were:

- dynamically map device parameters to visual representations, for example, for CPU usage: colored box, sparkline, low-detail line chart, high-detail line chart
- sort devices by parameter
- explicitly filter to device subsets, support selection based on an existing hierarchical organizational structure
- dynamically customize thresholds where interesting values are visually distinguished
- integrate results into the workflow, by exporting detailed information about selected parameters and devices in spreadsheet format
- support the familiar interaction of dragging a single line to resize in the style of spreadsheets

VISUALIZATION SOLUTIONS

We present the motivating visualization principles behind our design, describe the interface and capabilities of LiveRAC, and discuss its implementation.

Design Principles

The LiveRAC interface synthesizes several techniques to address the requirements stated above. In many cases, a choice of technique was guided by specific visualization principles, whose provenance we cite below. We make one assertion: that showing several levels of detail simultaneously provides useful high information density in context. The list below reflects our final design, after several stages of requirements gathering, iterative development, and validation by study participants.

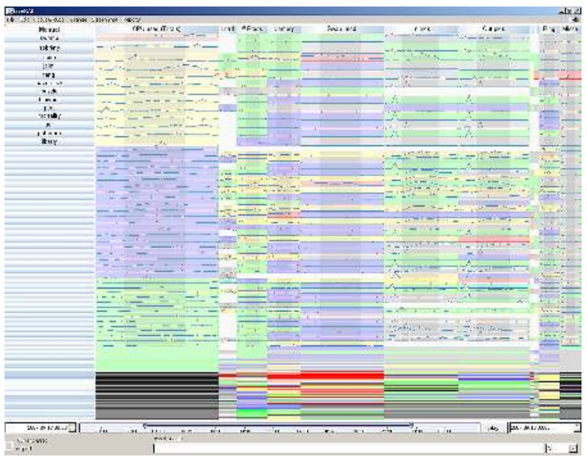
We show the LiveRAC interface in Figure 3 with data from a production server, anonymized by randomly mapping customer device IDs to nouns from a dictionary. The accompanying video shows the look and feel of the interactive interface.

LiveRAC Interface

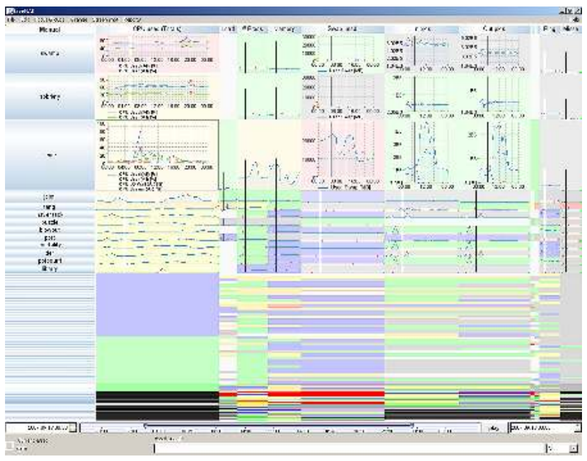
Principle: familiar visual representations should be preserved when appropriate. This approach exploits user intuition and experience for faster learning. The base visual data representations in LiveRAC are familiar line charts and bar graphs. These charts appear as cells in a spreadsheet-like matrix. By default, LiveRAC uses the same color coding conventions as the other internal software tools used by the LCEs, shown in Figure 2, with optional reassignment of the color palette on demand. Although the default colors are not maximally discriminable, they are adequate, so we preserved them for familiarity. We also provide VCR-like controls to play through the data at variable speed. Live data is viewed in real time, while archived data is often viewed under accelerated playback.

Principle: side-by-side comparison of small multiple views is easier than remembering previously seen views. The principle of small multiples [23], in which many small instances of different data sets are shown with the same representation, allows fast side-by-side visual comparison of many dozens of items. The alternative of inspecting charts one at a time and comparing them to previously seen charts is much less effective and does not scale well. The alternative of overplotting many curves on a single set of axes only scales up to a few dozen curves before visual clutter becomes overwhelming. The main LiveRAC frame, shown in Figure 3, presents a matrix where each cell contains an area-aware chart showing time on the horizontal axis and device parameters on the vertical axis. The same time period is shown for all charts. The time period can be changed with a double-edged slider or by entering explicit start and end times in the text fields at the bottom of the screen.

Principle: spatial position is the strongest perceptual cue. A core principle of information visualization is that encoding relationships by spatial ordering is more accurately perceived than other encodings such as color, size, or orientation [9]. This principle underlies the technique of reorderable matrices [2, 20], which allows relationships between cells to be detected visually during exploratory data analysis. In LiveRAC, each matrix row represents a monitored network device, and each matrix column represents a group



(a)



(b)

Figure 3. LiveRAC shows a full day of system management time-series data using a reorderable matrix of area-aware charts. Over 4000 devices are shown in rows, with 11 columns representing groups of monitored parameters. (a): The user has sorted by the maximum value in the *CPU* column. The first several dozen rows have been stretched to show sparklines for the devices, with the top 13 enlarged enough to display text labels. The time period of business hours has been selected, showing the increase in the *In pkts* parameter for many devices. (b): The top three rows have been further enlarged to show fully detailed charts in the *CPU* column and partially detailed ones in *Swap* and two other columns. The time marker (vertical black line on each chart) indicates the start of anomalous activity in several of *spire*'s parameters. Below the labeled rows, we see many blocks at the lowest semantic zoom level, and further below we see a compressed region of highly saturated blocks that aggregate information from many charts.

of one or more monitored parameters and can be sorted according to its values. For example, sorting by *Load* orders the device rows by load average, with the highest at the top. Columns can be sorted according to parameter values, such as the minimum, maximum, or average of the time-series. Rows can be sorted by device names or metadata such as location, customer, or other groupings. Columns can also be reordered by the user.

Principle: multiple views are most effective when coordinated through explicit linking. The principle of linked views [15] is that explicit coordination between views enhances their value. In LiveRAC, as the user moves the cursor within a chart, the same point in time is marked in all charts with a vertical line. Similarly, selecting a time segment in one chart shows a mark in all of them. This technique allows direct comparison between parameter values at the same time on different charts. In addition, people can easily correlate times between large charts with detailed axis labels, and smaller, more concise charts.

Assertion: showing several levels of detail simultaneously provides useful high information density in context. Several technique choices are based on this assertion. First, LiveRAC uses stretch and squish navigation, where expanding one or many regions compresses the rest of the view [11, 17]. The accompanying video shows the look and feel of this navigation technique. The stretching and squishing operates on rectangular regions, so expanding a single chart also magnifies the entire row for the device it represents, and the entire column for the parameters that it shows. The edges of the display are fixed so that all cells remain within the visible area, as opposed to conventional zooming where some regions are pushed off-screen. There are rapid navigation shortcuts to zoom a single cell, a column, an aggregated group of devices, the results of a search, or to zoom out to an overview. Users can also directly drag grid lines or resize freely drawn on-screen rectangles. Navigation shortcuts can also be created for any arbitrary grouping, whose cells do not need to be contiguous. This interaction mechanism affords multiple focus regions, supporting multiple levels of detail.

Second, charts in LiveRAC dynamically adapt to show visual representations adapted in each cell to the available screen space. This technique, called semantic zooming [13], allows a hierarchy of representations for a group of device-parameter time-series. In Figure 3, the largest charts have multiple overlaid curves and detailed axis and legend labels. Smaller charts show fewer curves and less labeling, and at smaller sizes only one curve is shown as a sparkline [24]. On each curve, the maximum value over the displayed time period is indicated with a red dot, the minimum with a blue dot, and the current value with a green one. All representation levels color code the background rectangle according to dynamically changeable thresholds of the minimum, maximum, or average values of the parameters within the current time window. The smallest view is a simple block, where this color coding is the only information shown.

Third, aggregation techniques achieve visual scalability by ensuring dense regions show meaningful visual representations. Given our target scale of dozens of parameters and thousands of devices, the size of the matrix could easily surpass 100,000 cells. Stretch and squish navigation allows users to quickly create a mosaic with cells of many different sizes, with many regions where the number of cells is greater than the number of available pixels needed, to draw even a single-pixel block for each cell. In these regions, an aggregate block is drawn representing all cells within the screen-space region. A naive approach might be to overdraw cells so the user sees either a blend of colors or the last cell drawn. This approach is inefficient and unlikely to show the most relevant information. Instead, LiveRAC computes the display for a given block using one of four possible aggregation functions for the time-series values in the cells it represents: minimum, maximum, mean and cardinality.

We increase the color saturation of aggregate blocks to show density, in proportion to the the number of cells represented, starting from the base saturation level of 25% for a non-aggregated block. Conversely, the saturation of the chart background color decreases as more details are shown, following the guidelines of Ware [26] and Tufte [23] that large areas should use desaturated color and that high contrast between foreground and background improves readability.

Given the target scale of thousands of devices, there is usually not enough space to draw legible labels for a horizontal row. When the devices are sorted by a shared descriptive parameter such as logical grouping a single label is drawn for this higher level structure, ending with a number indicating the number of aggregated devices it represents.

Fourth, the technique of guaranteed visibility [11] ensures that important information is always visible, even in highly compressed and aggregated regions. While the concept is simple, the challenge is to implement a rendering architecture that delivers interactive frame rates when the number of cells is huge. LiveRAC is built on the PRISAD infrastructure [21], which supports guaranteed-visible marks within a stretch and squish navigation framework. LiveRAC uses guaranteed-visible marks to show results during progressive search of device names and metadata. LiveRAC also marks key items with guaranteed visibility, including alarms of category *critical* and values above a critical threshold on several parameters. These parameters were adapted according to user requirements: the events that senior system management staff check are considerably above the day-to-day operational level.

Principle: overview first, zoom and filter, details on demand. This widely followed principle, articulated by Shneiderman [18], has been recognized as effective for coping with scale and complexity. The combination of techniques described above allow interactive exploration from overviews at many levels of detail within the same visual metaphor. We provide filtering with dynamic control over which parameters are shown in the matrix. Although we integrate alarms into the time-series framework by treating them as time-based events, it would be awkward to draw the full text of alarms within matrix cells. Instead, we present a traditional dialog box that pops up on demand. The user can export details of the time-series data and alarms for any selected set of devices in spreadsheet format, to integrate the visual queries supported by LiveRAC into the workflow of the LCEs. The requirement of exporting these details was not recognized until the final design phase, when we had direct access to the LCE target audience.

Principle: abrupt visual change should be avoided. This principle arises from the perceptual theories of object constancy [16] and change blindness [14]. In LiveRAC, all transitions are animated, for instance, growing or shrinking regions using navigation shortcuts. When the user changes the time window or first expands a cell from a block to a chart, the server query to obtain more data may take many seconds. During this time, we continue to show the old visual representation, but with a yellow dot in the upper right corner of the cell to indicate that an update is pending. When the new data is available, we draw its representation directly over the old one, avoiding flicker that would increase the risk of change blindness.

Principle: user actions should receive immediate visual feedback. The final design principle was adopted to enable high-interaction data exploration [16]. We achieve guaranteed frame rate rendering even with large data sets by building on the PRISAD infrastructure [21]. Also, LiveRAC performs rendering in a separate thread from server updates to ensure consistent interactive response even during long-running database queries.

Implementation

LiveRAC is a client that connects to a back-end database, in this case, the SWIFT server. LiveRAC is written on the PRISAD libraries [21], which provide a generic, efficient rendering infrastructure for guaranteed frame-rate accordion drawing, providing stretch and squish navigation with guaranteed visibility of marked items. Rendering and server updates occur in separate threads, as mentioned above.

LiveRAC has been shown to maintain interactive frame rates on data sets of 4000 network devices and 20 input channels, displaying six months of data collected at 5 minute intervals. In total, the raw data set contains billions of raw data points. The core structure for rendering and picking nodes supports O(*n log n*) insert, remove and search operations. Technical details of LiveRAC's implementation, as of phase two, are available in thesis [10]. To demonstrate interaction with the LiveRAC visualization system a short video is available[4].

LONGITUDINAL EVALUATION

We discuss the methodology of our informal longitudinal study, summarize the implications for design and present LCE usage scenarios.

Informal longitudinal study methodology

The objective of our longitudinal study was to better understand the strengths and weaknesses of the visualization techniques used in our design in a production environment.

[4] www.cs.ubc.ca/labs/imager/video/2007/liverac/liverac.mov

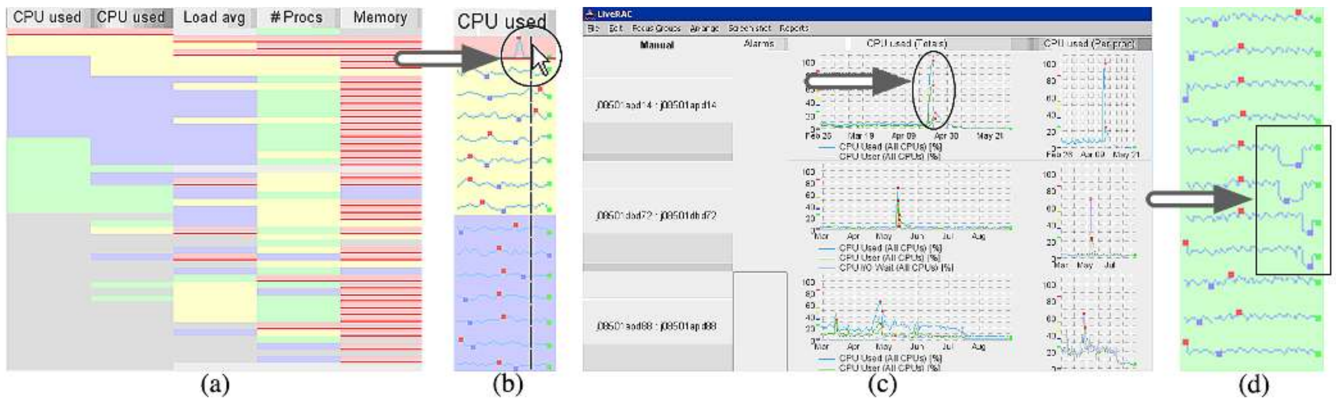


Figure 4. Usage scenarios from the longitudinal study. Scenario 1: In (a), the LCE examines an overview of 50 devices, after sorting by CPU usage. Column headers are truncated: the first column breaks down CPU usage by categories such as *system* and *IO*, while the second shows per-CPU data. Many of the systems reporting critical (red) values in the Load Average column did not have critical CPU usage levels, suggesting that the system load is I/O bound. (b) shows detail of the CPU usage column when expanded by the LCE to see the time-series data as sparkline charts. In the top row (circled), a critical threshold was reached. The highly saturated red, blue and green dots for each sparkline indicate the high-water mark, low-water mark, and final value, respectively. Scenario 2: In (c) the LCE has expanded the *CPU used (Totals)* and *CPU used (Per-proc)* columns and the first few device rows for about 3 months of data. The topmost device shows a sharp escalation in CPU load over a few days. After peaking at 100% (circled), data collection for the device abruptly terminates, where the dotted lines, representing weighted rolling averages, fall to 0. Scenario 3: Detail of the *CPU load* column from 11 web servers in the same pool over six months is shown in (d). Servers 5 and 6 show unusual behavior, where load drops to 0 for several weeks. Subsequently, servers 7 and 8 show a similar anomaly for a shorter duration.

Data collected for our longitudinal study includes: notes, audio, desktop sharing screen capture movies of interactive sessions, and log data from LiveRAC. Because our target population works in multiple remote locations, many interviews and meetings were conducted over the phone. We recorded interviews with our participants when possible, relying on hand-written notes for some large meetings where confidentiality considerations precluded making recordings.

Our participants had severe time constraints and were in disparate physical locations, presenting challenges for effective training. The two training tactics that proved essential to adoption were real-time presentations using desktop sharing software to describe and demonstrate the features of LiveRAC, and distributing a five-minute Flash training video demonstrating its basic functionality. Two LCEs stated that they had watched the video several times to fully absorb the information. We also provided written documentation and an interaction shortcut sheet. Questions and concerns from users were answered using email.

We analyzed the data by reviewing all notes, audio, and video logs and building an internal wiki which identified capabilities requested by participants, bugs, and notes on observed user behaviors.

Implications for Design

We summarize the key findings of our informal study.

Visual, interactive sorting offers significant benefits. LCEs are most interested in systems they consider to be top offenders in certain categories. Their current reporting tools were able to provide this capability only for fixed time periods, and without being able to show a value for all of the monitored parameters for every device on a single screen. We observed the LCEs using LiveRAC as a visual query tool. By sorting on each column, they were able to see not only the top offenders from the parameter associated with the column, but could also see values from the corresponding columns. While using the tool in this manner and discovering an anomaly in the behavior of two web servers, one participant commented, *"It's great and all to have the data, but if you can't visually see it, it's almost worthless! ... I can look at that Excel spreadsheet all day and never see what I can see just visually here ... It's such a night and day difference you know, the visualization part of it."*. Our findings indicate that data reordering is a key feature for deployed visualization systems.

Viewing large numbers of charts side by side was critical in serendipitous pattern discovery. For example, a participant discovered an unusual anomaly in CPU load in one device, scanned down the row, and saw that another device experienced identical behavior immediately after the first device returned to normal. The participant commented, *"Lasted a good week and a half, 2 weeks, ... another dip in [redacted] also ... the load balancer [is] sending traffic to one and not to the other. Wow, that's interesting."* Such a pattern could be detected by automated approaches, but only if users knew to expect it. The high information density of our display supported the ability of our participants to identify new, interesting patterns. Our findings suggest that visual, side-by-side comparison is important for serendipitous discoveries, and should be supported in future deployed visualization systems.

Manipulating the time window, and seeing overviews of data over long time periods was novel and exciting. The LCEs use long-term trend information when forecasting future load requirements. When Participant A introduced the tool to his colleague Participant B by demonstrating that he could shift from viewing a few hours of data to six months of data, the latter reacted with surprise and excitement. Our findings indicate that providing a dynamically changeable time window with support for large time ranges is a critical component of time-series visualization design.

Linked views aided correlation. For example, participants L1 and L2 identified several correlating patterns during interviews where we observed their interaction through desktop sharing software. In many cases, the correlation did not occur in systems that were side-by-side. In one instance, L1 observed that a web server in a load-balanced group stopped accepting traffic but remained online. L2 pointed out that almost immediately after it recovered, another system in the pool had identical behavior. This kind of problem is precisely the type that might not be detected by an automated approach, since neither device was offline or unresponsive to ping tests. L3 mentioned, *"What I liked about [LiveRAC was] where, if there was a particular alarm, you could put that vertical line and look at all the other parameters, where if CPU spiked you could look at all the other parameters and see where they are, or what we found was there was critical alarms on [the] ping test, but when you looked at the CPU utilization was extremely low, you could get a sense of the health of the asset."* Our finding suggests that tight linking and high information density support this type of pattern finding.

Stretch and squish navigation was not a barrier to adoption. Although a prior laboratory study found a performance penalty for stretch and squish navigation [12], we found that this interaction mechanism did not present as much of a challenge to our participants as we expected, despite its novelty and limited training. This difference may be due to the extensive use of structure-based navigation in LiveRAC, where users often employed shortcuts to enlarge the cell or column under the cursor. We conjecture that such navigation may require less cognitive load than dragging out an arbitrary rectangle on the screen and freely resizing it.

Adding report generation capability was critical for LCE acceptance. Our finding echoes that of González and Kobsa [4], who propose that tightly integrating visualization systems into the user's standard analysis tool chain facilitates adoption. Our LCE participants requested the ability to extract sorted, user-selected data from the visualization tool to transfer information acquired during exploration into their downstream workflow. We identified this requirement in the fourth stage of the design process, after we gained access to the LCEs. Our participants indicated that if we had not added this functionality, they would have lost interest in using the system. A major complaint of the LCEs about previous system management tools was the inability to generate customized reports that they could manipulate and share.

Iterative design can be successful even when full participation of the target user population is not available at project conception. Not having direct access to the entire participant pool from the beginning of the project was a significant challenge. A more conventional user-centred design process would have allowed us to iterate our design more rapidly. For example, we devoted significant development time to alarm processing based on our initial assessment of the requirements, only to discover that LCEs find alarms less interesting over long periods of time. Although the challenge of gaining access to participants within the political constraints of a large organization has not been explicitly discussed in the visualization research literature, many have noted the difficulty of transferring visualization techniques proposed by researchers into real-world settings. We suggest that our staged approach, where software prototypes demonstrate the value of visualization methods to potentially skeptical individuals in the management chain, may be emulated in other research contexts.

Usage Scenarios

We present three scenarios showing the use of LiveRAC by the LCE participants in our informal longitudinal study, as logged during recorded desktop sharing sessions. (The desktop sharing application reduces images to 256 colors and introduces other compression artifacts.) Images have been cropped to highlight key data.

Figure 4a shows Scenario 1. The LCE started with an overview of 50 devices at the lowest semantic zoom level, where colored blocks show the maximum values of each parameter for each device. Obtaining the same information using the tabular and chart views of previous software tools would require manually scanning 50 charts to determine where the highest values occurred. Correlating the impact of one parameter on others would be a tedious activity. Participants were most interested in sorting the systems by CPU load; memory usage, load average and network traffic utilization were also of interest. We observed LCEs use the overview-and-sort technique with as many as 4000 devices, as shown in the accompanying video and in Figure 3.

The LCE continued to explore the same data set, using sorting, stretch and squish navigation, and temporal navigation. After performing a sort operation, the LCE stretched the cells for systems with the highest CPU load to see trends, as shown in Figure 4b. The LCE was immediately interested by the unusual spike in utilization from the top CPU-use offender, stating it was a feature of interest. To check whether it was a frequently recurring trend that might affect capacity planning, or a less important, isolated event, he click-dragged the time slider to enlarge the time window. Discovering the device having the load spike with existing visualization tools would have required looking at one device at a time, or at an unsorted view that showed only the

CPU parameter. In LiveRAC, all the other parameters are available for simultaneous comparison.

Figure 4c shows Scenario 2, where an LCE was showing a colleague sorted trend information over a period of months. When he stretched the cells for devices showing the top CPU usage, he discovered that a system had stopped reporting data shortly after hitting 100%. Both LCEs agreed the behavior indicated a problem.

Figure 4d shows Scenario 3. An LCE was inspecting a group of web servers in the same load-balanced pool. These servers are intended to have similar behavior. After expanding those cells, he saw that on some devices, the load had unexpectedly dropped almost to 0 for weeks, yet they had not crashed and continued to report all their monitored parameters. The LCE said he would follow up with a colleague who was responsible for the systems.

We provide three more usage scenarios in the accompanying video, made by direct video capture of LiveRAC interactive sessions, instead of from desktop sharing logs.

CONCLUSION

We presented LiveRAC, a novel visualization system that supports browsing and correlating logged time-series data for system management. We employed a staged design process, in which we demonstrated working prototypes to gain access to additional participants in subsequent phases. We met the challenging goal of deploying LiveRAC in a full production environment. We conducted an informal longitudinal field study, and our findings support that many aspects of the final design were helpful for the intended tasks.

LiveRAC provides a high information density interface that conveys more information than dashboard approaches, and supports details on demand. The large number of devices and parameters that may be interactively monitored and explored was surprising and exciting to the users in our study. Active study participants, as well as managers who participated mainly in design reviews, were enthusiastic about the insights that could be gained from the system. LiveRAC has ongoing support from its users, with the prospect of wider deployment in other network management and planning applications.

ACKNOWLEDGMENTS

We thank AT&T Research and NSERC for funding the work carried out at UBC.

REFERENCES

1. W. Aigner et al. Visualizing time-oriented data: A systematic view. *Computers and Graphics*, 31(3):401–409, 2007.
2. J. Bertin. *Graphics and graphic information processing*. Walter de Gruyter, Berlin, Germany, 1981.
3. V. González and A. Kobsa. Benefits of information visualization systems for administrative data analysts. In *IV '03: Proc. Intl. Conf. on Information Visualization*, pages 331–336, 2003.
4. V. González and A. Kobsa. A workplace study of the adoption of information visualization systems. *Proc. I-KNOW '03: Intl. Conf. on Knowledge Management*, pages 92–102, 2003.
5. H. Hochheiser and B. Shneiderman. Interactive exploration of time series data. In *DS '01: Proc. Intl. Conf. on Discovery Science*, pages 441–446. Springer-Verlag, 2001.
6. R. Kincaid and H. Lam. Line Graph Explorer: scalable display of line graphs using Focus+Context. In *AVI '06: Proc. Adv. Visual Interfaces*, pages 404–411. ACM Press, 2006.
7. E. E. Koutsofios et al. Visualizing large-scale telecommunication networks and services (case study). In *Vis '99: Proc. IEEE Conf. Visualization*, pages 457–461, 1999.
8. J. Lin et al. Visually mining and monitoring massive time series. In *Proc. KDD '04*, pages 460–469. ACM Press, 2004.
9. J. D. Mackinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Trans. Graphics*, 5(2):111–141, 1986.
10. P. McLachlan. LiveRAC – Live reorderable accordion drawing. Master's thesis, Univ. British Columbia CS Dept, 2006.
11. T. Munzner et al. TreeJuxtaposer: scalable tree comparison using Focus+Context with guaranteed visibility. *ACM Trans. Graphics (Proc. SIGGRAPH '03)*, 22(3):453–462, 2003.
12. D. Nekrasovski et al. An evaluation of pan & zoom and rubber sheet navigation with and without an overview. In *Proc. CHI '06*, pages 11–20. ACM Press, 2006.
13. K. Perlin and D. Fox. Pad: an alternative approach to the computer interface. In *Proc. SIGGRAPH '93*, pages 57–64. ACM Press, 1993.
14. R. A. Rensink, J. K. O'Regan, and J. J. Clark. On the failure to detect changes in scenes across brief interruptions. In D. J. Simons, editor, *Change Blindness and Visual Memory*, pages 127–145. Psychology Press, London, 2000.
15. J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Proc. CMV '07*, pages 61–71, 2007.
16. G. Robertson, S. Card, and J. Mackinlay. The cognitive coprocessor architecture for interactive user interfaces. In *Proc. UIST '89*, pages 10–18, 1989.
17. M. Sarkar et al. Stretching the rubber sheet: a metaphor for viewing large layouts on small screens. In *Proc. UIST '93*, pages 81–91, 1993.
18. B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proc. IEEE Symp. Visual Languages*, pages 336–343, 1996.
19. B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proc. BELIV '06*, pages 1–7. ACM Press, 2006.
20. H. Siirtola. Interaction with the reorderable matrix. In *IV '99: Proc. Intl. Conf. on Information Visualisation*, pages 272–277, 1999.
21. J. Slack, K. Hildebrand, and T. Munzner. PRISAD: A partitioned rendering infrastructure for scalable accordion drawing (extended version). *Information Visualization*, 5(2):137–151, 2006.
22. C. Stolte, D. Tang, and P. Hanrahan. Query, analysis, and visualization of hierarchically structured data using Polaris. In *Proc. KDD '02*, pages 112–122, 2002.
23. E. Tufte. *Envisioning Information*. Graphics Press, 1990.
24. E. Tufte. *Beautiful Evidence*. Graphics Press, 2006.
25. J. J. van Wijk and E. R. van Selow. Cluster and calendar based visualization of time series data. In *Proc. InfoVis '99*, pages 4–9, 1999.
26. C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2nd edition, 2004.