UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

HONOURS PROJECT PROPOSAL

Design and Implementation of an Immensely Scalable Monitoring System

Duncan Clough 
duncanclough@gmail.com

Vincent Geddes 
vincenj.geddes@gmail.com

Stefano Rivera 
stefano@rivera.co.za

Supervisors:

Greg Kempe
Amazon.com
Amazon Web Services: EC2
gkempe@amazon.com

Dr. Michelle Kuttel
University of Cape Town
Department of Computer Science
mkuttel@cs.uct.ac.za

Dr. Patrick Marais
University of Cape Town
Department of Computer Science
patrick@cs.uct.ac.za

July 19, 2008
1 Project Description

There is no existing monitoring system that can scale to monitor large fleets of computers and provide effective visualisation.

The overall goal of the project is to develop a system that will allow effective monitoring of a large fleet of computers. The project will consist of 3 major parts: Collection, Storage & Retrieval, and Visualisation.

1.1 Visualisation

In a small scale system a user is able to mentally keep track of each node, and use this prior knowledge to identify patterns and perform diagnostics. However, as soon as this is scaled to upwards of 1 000 computers it becomes impossible to keep track of everything mentally. This increase in information to a level beyond that which a human can efficiently process is termed information overload. Our system will therefore aim to solve this problem.

Since multidimensional data is very difficult for humans to visualise, it is correspondingly difficult to find patterns within the data. One of the aims of the visualisation component is to design a system that will in some way facilitate this process by filtering out unwanted information.

Lastly, the visualisation needs to represent the change of metrics over time. In order to find patterns and debug a fleet of computers, a user would need to refer to what happened in the past. This has to be intuitive to allow the user to be able understand the changes that are displayed with the progress of time.

1.2 Storage & Retrieval

Storing metrics in a scalable fashion is non-trivial. The storage system needs to be able to give the visualisation front end all the metrics it needs on a machine or fleet of machines, in near-real-time.

The system needs to handle the rapid updating of metrics (via the collection system), and store these metrics for a useful forensic time period (i.e. a week). Data older than this time period, could be stored with reduced resolution, to determine long-term trends.

Missing data needs to be sensibly dealt with, as with a large fleet missing data will be the norm.

1.3 Collection

This module has the task the of monitoring each computer in the system and making the collected data available for the storage and visualisation modules. While monitoring an individual machine or a small LAN is an easy enough task, scaling the system to monitoring a large fleet of computers requires the implementation of a large distributed system.

2 Background / Related work

There are many existing monitoring systems, but they rarely scale beyond a couple of thousand nodes.

Most monitoring systems visualise with graphs from RRDTool by Tobias Oetiker [18]. RRDTool was the graphing and round-robin storage system from Oetiker’s MRTG project, [17] split out to make MRTG more modular. (MRTG provided simple performance metric graphs of data collected from routers by SNMP, but was extended by the community to support other metrics, due to its good graphing engine.) RRDTool is now the basis of most network visualisation tools, but its line-graphs aren’t capable of effectively visualising more than ∼ 20 hosts together.

CARD, by Anderson and Patterson [5] was written to collect metrics scalably. It used a hierarchical set of collection nodes, each running a SQL database. Data was aggregated to storage nodes (one for each metric), but visualisation was simply multiple graphs on one page. The CARD hierarchical approach has been expanded on in the other systems we evaluated.

Ganglia, by Massie et al. [3, 4] followed a similar design to CARD (they were both developed at U. C. Berkeley), but with multi-level collection. Data is shared at the lowest level by multicast among nodes, then collated up through a hierarchy of collation nodes (using XML). This collation doesn’t scale beyond 5,000 nodes [4]. At the top, it is stored and visualised, again by RRDTool.

Astrolabe [1, 2], was developed by Renesse et al. from Cornell at the same time as Ganglia. The design is similar, but storage is decentralised. Astrolabe’s lowest level of aggregate nodes store databases of their collected data. Queries are made to the head node, and passed down the tree, and results aggregated up. The system is designed for data mining, not visualisation. However, it scales very well, and has been calculated to scale to over 300,000 nodes, with the correct choice of branching factor [1].

2.1 Visualisation

In the context of this research project, we define visualisation to be the graphical presentation of information or clustering techniques to bring out patterns in the information. The field is often broken into data or scientific visualisation, where the former deals mainly with visualising in the general forms
of data and the latter focuses on representing data obtained through experiment or observation [9].

Since we will be monitoring physical data the project naturally has a connection with scientific visualisation. It however also deals with the problem of visualising large multidimensional data-sets, and therefore also needs to use ideas developed within the field of data visualisation.

Visualising large multidimensional data-sets is becoming an even more pressing problem within scientific and commercial computing as systems continue to scale, since it is impossible for a human to quickly and effectively assess huge amount of data. For this reason visualisation techniques are important in simplifying problems to make them easier for users to understand.

As previously discussed, implementations such as CARD and Ganglia use RRDTool to visually represent their data as a collection of graphs. In a large scale implementation there will simply be too many sets of values for them to be effectively represented using graphs. Therefore we will need to develop a new method of scalably visualising the multidimensional data-sets collected by the monitoring tool.

There are three aspects to visualising a large multidimensional data-set: the visualisation technique, interaction and distortion technique [11].

The visualisation technique deals with the question of how to represent the data. There are some methods which try represent all information at once such as a pixel-oriented approach (see Figure 1) [12], or less conventional approaches such as ShapeVis (see Figure 2) [14].

Interaction and distortion techniques deal with how the user can interact with the data. These methods are used to refine the display to show what is found to be interesting. This can be done by using a variety of methods:

**Filters:** Limiting the display to certain metrics [15]

**Zooming:** Scaling portions of the display

**Projections:** Transforming multidimensional data-points to lower dimensions [15] — see Figure 3

**Distortion:** Adjusting the perspective to enlarge certain part of an image, and reduce the space taken by other parts while maintaining context [14]

**Linking and Brushing:** Selecting various data-points in one view (brushing) and maintaining that selection while changing the view (linking) — see Figure 4 on the next page [16]

However the best methods use a mixture of visualisation and interaction/distortion techniques [11].
Marching Sphere [14] is an example of a package that combines ShapeVis with interaction/distortion techniques.

2.2 Storage & Retrieval

The problem is one of a Data Warehouse, specifically an On-Line Analytical Processing (OLAP) system.

Monitoring data only needs to be accurate for the length of time that is useful for forensic problem analysis. Beyond around a week, one is only interested in long term trends, not short term fluctuations.

For this reason RRDTool uses a set of round-robin databases of different resolutions. An aggregation function is specified for producing the lower-resolution data points.

RRDTool’s round-robin databases are efficient and well-suited to their task, but they are designed to be used as a single database for each node-metric set. In the scale of thousands, and even hundreds, this becomes inefficient. Visualising the current state of the network would require querying the latest data-point from every single RRD file.

CARD and Astrolabe both took the approach of a SQL database, which is probably the simplest implementation approach.

Astrolabe stores its data in a tree of storage nodes. When the head node is queried, the query passes down the tree, and the results are summarised on the way back up. While this system is highly scalable, the single-point summaries aren’t suitable for the required visualisation system. A visualisation system requires access to all the relevant points.

2.3 Collection

The collection system falls into the domain of distributed systems. One contemporary monitoring system, Ganglia [3] is used to monitor state from federations of clusters. Ganglia [3] uses a hierarchical approach to metric collection. Within a cluster, the gmond daemon is installed on each node. This daemon monitors various metrics on the node, and submits these measurements to all other nodes in the cluster. Each node thus has approximate information about the entire cluster. A second daemon, gmetad is used to aggregate state from a federation of clusters. A hierarchical organisation of gmetad servers can be used to collect data from over 2000 nodes. gmetad publishes collected state in XML form. Any application or visualisation tool can query this XML from gmetad using a special protocol.

3 System Architecture

Our architecture will have a modular design. Each component will have clearly defined boundaries, and well-defined APIs for communication. The architecture can thus be divided into three components, Visualisation, Storage & Retrieval and Collection. Figure 5 gives a visual overview.
Our proof of concept will be released under an Open Source license, probably the MIT/X11 license, and hosted on an Open Source Source Code Repository such as Sourceforge.net.

3.1 Collection
Our collection architecture will collect information from a hierarchy of resource domains. Following on from Renesse, et al. [1] we use the term zone to refer to a resource domain. They use the following definition:

A zone is recursively defined to be either a host or a set of non-overlapping zones.

The primary zone at the top of the hierarchy, will represent resources spread over a wide area network. At the next level, zones will represent resources in a local area network. Finally, the leaf zones in the hierarchy will represent host computers. A leaf zone will have a list of attributes representing the state of a host computer.

Within a LAN, a monitoring daemon will be installed on each host. These daemons will communicate with each other using a UDP broadcast protocol. Using this protocol, each host will routinely send its state to all other hosts in the network. In effect this means that each host can act as a source of information for the entire LAN. This ensures robustness when there is localised network failure. In addition, the protocol will also allow for automatic discovery of new hosts as they become visible to the network. To ensure a low processing overhead, the binary XDR format [6] will be used to encode state being replicated around the LAN.

An aggregation daemon will be used to collect state from a number of child sources using a TCP polling protocol. These sources can be monitoring daemons or other aggregation daemons. The collected state will then be aggregated and made available to aggregation daemons higher up in the hierarchy. The aggregated state will be encoded using the lightweight JSON [7] markup language.

3.2 Storage & Retrieval
Initially, the storage system will be built around a high-performance SQL database such as MySQL. This should provide a good starting point to test systems, and reasonable scalability. MySQL’s replication mode could allow separate collection and retrieval nodes [19].

Should MySQL not prove sufficiently scalable or the overhead too high, a custom storage system will be investigated.

We may investigate the possibility of temporal compressibility of monitoring metrics. This would highly complicate the SQL, but may be necessary to reduce storage requirements.

A daemon will be built that handles queries from the visualisation system, and returns relevant data. Another daemon will accept incoming data from the collection system.

There is a high storage overhead: data-points will be accessed by query, rather than address-offset indexing, and this metadata will have to be stored with each data-point.

An Astrolabe-style distributed database setup will be investigated. However, instead requiring aggregation of the data moving up the tree, full queried data-sets will be returned, as required by the visualisation subsystem.

3.3 Visualisation
The interface component retrieves data from the storage system for visualisation. The main purpose of this part of the project is to develop and effective means on visualisation, and therefore the interface will provide minimal control over the back-end systems.

The visualisation approach to be used will combine visualisation techniques with interaction and distortion techniques in an attempt to best represent information to the user so that it remains (i) in context, and (ii) easily understandable.

In order to make the visualisation tool useful in locating patterns, multiple views will need to be implemented so that data can be viewed in various formats simultaneously. These multiple view are also necessary in order to implement interaction methods such as linking and brushing.

In order to represent the multidimensional data effectively, it is likely that a 3D interface will be developed. This will be implemented using OpenGL.

Lastly, since there are no visualisation tools for this specific problem, a new system will need to be developed. Therefore part of the research project is to develop this new interface, making it difficult to give concrete specifications at this stage.

4 Outcomes
4.1 Questions to be Answered
The core questions to be tackled are:

Visualisation
- What is an effective way of visualising large sets of multidimensional data?
- How can information overload be minimised?

Monitoring and Collection
• Efficiently collecting data in a scalable manner without massive overhead

Storage and Retrieval
• Is it possible to effectively store massive amounts of temporal data?
• Can massive amounts of data in a way that is easily queryable?

Given enough time it would also be interesting to look at
• How to focus on parts of a visualisation without losing global context

4.2 Impact of Project
We aim to produce a monitoring system that is scalable, yet provides an effective visualisation for monitoring an entire fleet of computers.

Contemporary monitoring systems such as Astrolabe and Ganglia have focused more on the collection backend, while only providing simple and non-scalable visualization interfaces. Amazon.com, as proposers of this project, found that only Astrolabe provided sufficient scalability for their monitoring needs, but didn’t provide necessary visualisation or detail. Amazon has specific, known-problem monitoring systems, but need a system to quickly deal with unexpected situations.

4.3 Measurement of Success
1. The system is scalable, and is able to monitor a reasonably large collection of computers. This will be tested using computing resources at UCT, or by testing the system on a virtual fleet of computers.

2. The system effectively stores monitoring data in a way manner that successfully optimises use of space and response time.
   This will be tested by using simulated data from many (various orders upwards of 1,000) machines. The data should be queryable in a short space of time (under 5 minutes at the outside, for an average query).

3. The system effectively manages the information overload problem by visualising the information in a simple to grasp manner.
   This will be tested in a user experiment. See Section 5.5.1 on page 7.

5 Work Detail

5.1 Timeline
See Figure 6 on the following page.

5.2 Milestones

<table>
<thead>
<tr>
<th>Section</th>
<th>Date</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>30-31/7</td>
<td>Prototype Demonstration</td>
</tr>
<tr>
<td>Report</td>
<td>21/5</td>
<td>Definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Theory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Background</td>
</tr>
<tr>
<td></td>
<td>25/7</td>
<td>Literature Survey</td>
</tr>
<tr>
<td></td>
<td>17/10</td>
<td>Implementation Evaluation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Experiment” Write-Up</td>
</tr>
<tr>
<td></td>
<td>20/10</td>
<td>Implementation Testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Major Section Headings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Major Section Descriptions</td>
</tr>
<tr>
<td></td>
<td>27/10</td>
<td>First Draft Due</td>
</tr>
<tr>
<td></td>
<td>3/11</td>
<td>Final Report Due</td>
</tr>
<tr>
<td>Other</td>
<td>10/11</td>
<td>Poster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Web Page</td>
</tr>
<tr>
<td></td>
<td>12-13/11</td>
<td>Project Demonstration Dry-Run</td>
</tr>
<tr>
<td></td>
<td>19-20/11</td>
<td>Final Project Demonstration</td>
</tr>
</tbody>
</table>

5.3 Deliverables

5.3.1 Implementation

Storage Iteration 1 — Monitor 1 computer

Storage Iteration 2 — Monitor 5 computers

Full Iteration 1 — Integrate small scale monitoring with a naïve database and basic visualisation with graphs

Full Iteration 2 (Prototype) — Increase the scale for monitoring, make storage methods more efficient, and lastly have a basic multidimensional visualisation implemented

Full Iteration 3 — Solve issues that appear when applied to an extremely large scale. Visualisation needs to be scaled for larger data-sets.

Final Version — The complete integrated system that operates as a proof of concept

5.3.2 Report

Report — Outlines the development, discussion, results and conclusions of the project

Website — A website to make the project deliverables and descriptions available online

Poster — A brief summary of the project that communicates the key aspects, results and conclusions.
5.4 Risks

5.4.1 Scope

Even after doing basic background research we are relatively unfamiliar with the fields that this project encompasses. As a result methodologies that might seem straightforward could have rather difficult implementations.

As a result we can never be entirely sure that the defined scope is attainable within the specified time. Similarly small problems could escalate and cause the project to become too big.

In order to minimise this risk, the initial project specification has been discussed with our supervisors who are more knowledgeable. They have been able to provide us with rough guidance to limit the scope of the project.

In order to minimise this risk, the initial project specification has been discussed with our supervisors who are more knowledgeable. They have been able to provide us with rough guidance to limit the scope of the project.

Secondly, we plan to use an iterative development strategy. Therefore at each iteration there will be integration which will help prevent large problems from occurring early on.

5.4.2 Project Component Fails

Since there are 3 members in this project, there is always the chance that one of the components will not complete in time. This would cause problems with the final demonstration. In order to eliminate this risk the project has been divided into 3 separable components. Each component will have a well specified interface to allow easy integration, but also allow each component to operate independently.

5.5 Testing

We currently do not have a large fleet of computers at our disposal, which therefore makes testing non-trivial. Although Amazon has allowed use of EC2, this will not allow us to scale to an extremely large system.
In order to get around this problem, we will have the aggregation nodes of the system simulate data. Although this is not ideal, it can provide a rough idea of how the system works.

5.5.1 User Testing

In order to evaluate the visualisation interface we will need to run a user experiment in order to obtain qualitative results about the visualisation method. As a brief outline, we will require a control group to use the Ganglia-like method to extract some information, and another group to use the new visualisation to extract the same information. From this we will be able to obtain qualitative results about the system that can be compared to results from a Ganglia-like system.

5.6 Resources Required

5.6.1 People

Dr. Patrick Marias Patrick is a co-supervisor and will be involved with general guidance and ideas with respect to the academic components of the project. Dr. Marias’ areas of interest include computer graphics, visualisation, and HPC.

Dr. Michelle Kuttel Michelle is a co-supervisor and will be involved with general guidance and ideas with respect to the academic components of the project. Dr. Kuttel's areas of interest include scientific visualisation, parallel computing and HPC.

Greg Kempe Greg is a co-supervisor and the liaison with Amazon.com, who originally specified the project. He will be involved with general guidance, ideas, and providing access to Amazon web services, such as EC2 and SQS, which will be of use during the project. He will also be valuable in providing feedback on the effectiveness of the final product from a user’s perspective.

5.6.2 Software

At the collection level, we have made several software decisions. Our language of implementation will be C, using the C99 standard. To ensure portability across a wide range of UNIX systems, we will use the GTop and GLib libraries. The former provides programmatic access to various system metrics. The latter provides convenient abstractions for multi-threading, data-structures, and general application development. From experience, we have found that GLib reduces the amount of tedious involved in writing applications in C.

Storage systems will be prototyped in Python, and migrated to C if and when performance issues require.

5.6.3 Equipment

Development will be done on personal computers and our University-provided personal desktops. We will be using GNU/Linux systems for development.

Amazon.com have provided us with unlimited usage on up to 20 Elastic Compute Cloud (EC2) server instances. These should by sufficient for simulating data collection from large numbers of clients, and hosting the storage system. Should 20 instances not prove sufficient, we could request more.

UCT has a number of small clusters on campus. We could request to test our software on them. The Shuttleworth Lab has 84 desktop PCs running Ubuntu Linux, which we could use to test monitoring systems.

5.7 Work Allocation

Duncan Clough

1. User Interface
2. Data Visualisation

Stefano Rivera

1. Data Storage
2. Data Retrieval

Vincent Geddes

1. Computer Monitoring
2. Data Collection

References


A Image credits

Figure 1 on page 2 Image from [13]
Figure 2 on page 2 Image from [14]
Figure 3 on page 2 Image obtained from http://www.codeguru.com/dbfiles/get_image.php?id=10123&lbl=3DPROJ01_GIF&ds=20061023
Figure 4 on page 3 Modified version of image obtained from http://www.vrvis.at/via/resources/DA-RVoigt/images/ggobi/scatmat-ggobi-brushlink.png
Figure 5 on page 3 The embedded eyeglass image is courtesy of the Tango Project http://tango.freedesktop.org/. The entire diagram is therefore licensed under the Creative-Commons Share-Alike license.