

Visualizing and Managing Cloud Computing Networks

M.C. de Klerk and A.E. Krzesinski

Department of Mathematical Sciences

University of Stellenbosch

Stellenbosch 7600, South Africa

Email: deklerkmc@gmail.com, aek1@cs.sun.ac.za.

Abstract—This paper describes the methods used to visualize and identify possible problems within a cloud computing network. A technique, rendering the entire cloud network in Hyperbolic space, for visualizing the physical layout of the network is discussed. Issues concerning scalability with regards to effectively identifying problems within the cloud are discussed. The *Sun Cloud Manager* (Sun – Stellenbosch University) was created to demonstrate the techniques and data were generated to simulate real-time performance data obtained from the network.

Index Terms—Cloud computing, visualization

I. INTRODUCTION

Cloud computing [7] is a relatively new style of computing in which dynamically scalable and virtualized resources are provided as a service over the Internet. The term cloud is used as a metaphor for the Internet, based on how the Internet is depicted in computer network diagrams, and is an abstraction for the complex infrastructure it conceals.

Due to the large number of computers (the terms computers and nodes are used interchangeably) in the network, administrators potentially face an overwhelming task of managing this network. Furthermore, the cloud is often not fixed in one physical location, but is distributed over several continents in an attempt to decrease potential bottle-necking and service latency. Issues of scalability and physical location prevent the use of conventional administration tools and methods as it is not possible for an administrator to manually monitor thousands of computers at once, let alone over several locations.

The *Sun Cloud Manager* was developed to demonstrate the ideas presented in this paper. A simulator was used to synthesise cloud data. The data were stored in a database where each row of the database represents an incoming message from a node in the network. The *Sun Cloud Manager* reads each row without a time delay, simulating the maximum message input rate.

The remainder of this paper is organized as follows. Section II introduces the visualization software and describes the features that assist an administrator to manage a very large number of nodes. Policies and terms used in the paper are discussed in this section. Section III describes previous work done on visualizing large networks and describes how

this work was adapted to visualize the structure of the cloud network. Section IV reviews the progress made in identifying performance problems within a simulated cloud network using clustering algorithms and describes a visual metaphor for drawing attention to these problems. Section V indicates future work and presents conclusions.

II. BACKGROUND

The display of the changing status of a large number of nodes is a non-trivial problem. Furthermore, the speed of diagnosis by the administrator is of the utmost importance in solving problems timeously and keeping the system in check. A matter of concern to the administrator is the ability to rapidly inspect the network and be re-assured that all is well. The administrator should not have to concern himself/herself with nodes which are deemed to be *well-behaved* by some definition. On the other hand, nodes which are deemed to be *badly-behaved* should be the highlight of the information displayed. However, each visualized node displays several dimensions of data. Inspecting this information is trivial in the case of a few nodes, but scaling to a very large number of nodes may lead to information or cognitive overload. The displayed information therefore needs to be aggregated and filtered at some level and then presented.

The effective exploration of large graphs is a difficult problem. For example, the minimum spanning tree of a graph may be displayed as a tree rooted at some node. However, the problem of tree layout in 3D Euclidean space is that the number of children typically grows exponentially at each level, while the physical domain where they are placed grows polynomially.

The usual approach for avoiding collisions when placing nodes involves either placing child nodes far away from their parents or drawing the children smaller than their parents. However, such solutions suffer from various drawbacks including the fact that an overview of the tree is difficult to obtain. If the space allotted to leaf nodes decreases, zooming out to view the entire tree results in the leaf nodes becoming invisible. Likewise, if leaf nodes are sparsely laid out, changing the viewing perspective results in another part of the graph becoming invisible as the distance from the current viewing point is too great.

AEK is supported by grant numbers 2054027 and 2677 from the South African National Research Foundation, Nokia-Siemens Networks and Telkom SA Limited.

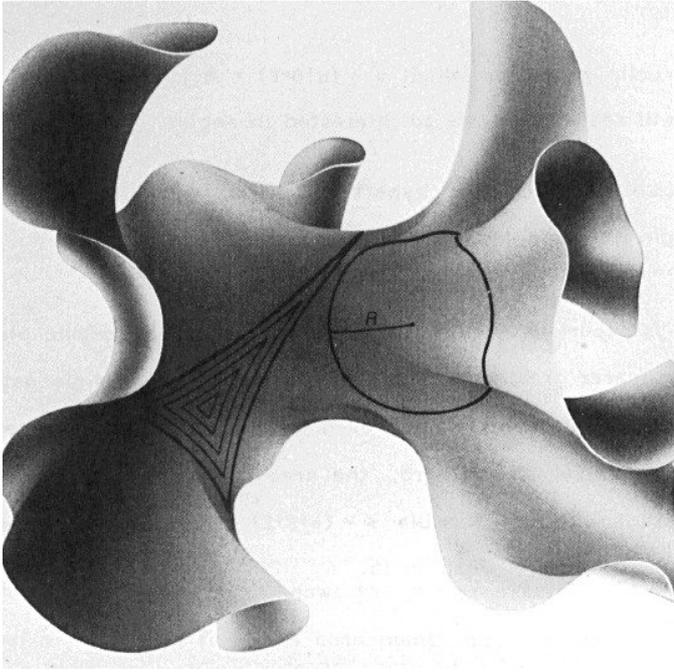


Fig. 1. There is more room in Hyperbolic space than in Euclidean space, as shown in this embedding of a hyperbolic plane in Euclidean 3D-space. A circle on a hyperbolic surface has an exponentially increasing circumference, as opposed to the Euclidean space where the circle circumference grows quadratically with respect to its radius [5]. Image courtesy of [9].

A. Hyperbolic Geometry

1) *Brief History and Benefits:* Hyperbolic geometry [2, 8] is a non-Euclidean geometry that can effectively be used for visualizing large graphs for three reasons.

- Area and circumference increase exponentially with respect to radius. Fig. 1 provides a sense of the exponentially increasing space provided by hyperbolic geometry.
- Elegant methods exist to draw focus and context projections that map an infinite 3D Hyperbolic space to a 3D Euclidean unit sphere.
- Various sections of the graph may be focused upon by translating the network structure in 3D Hyperbolic space and re-projecting.

Some distortion is introduced when projecting an infinite space to a finite space. However, the methods used to do such projections produce results which are readily comprehensible. The two projections (models) for mapping the infinite Hyperbolic space to finite Euclidean space are the projective model (known as the Kleinor-Klein-Beltrami model [13]) and the conformal model (known as the Poincare disc or ball model [1]).

The conformal model preserves Euclidean angle measurements but maps straight lines into arcs. The projective model preserves straight lines but distorts angles. While the conformal model may be aesthetically more pleasing, it is computationally more expensive to map straight lines to arcs. We therefore use the projective model. We construct a unit sphere in 3D Euclidean space. The centre of the unit sphere is the

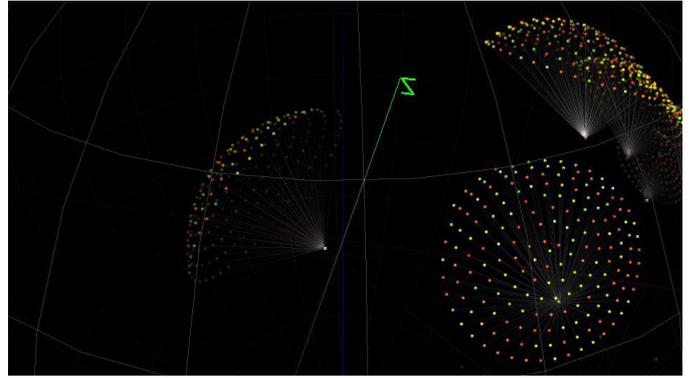


Fig. 2. The layout algorithm places the child nodes evenly and spaciouly. The result is an aesthetically pleasing visualization resembling sprinkles on an ice-cream cone.

origin of the coordinate system and distant objects effectively shrink as if they were compressed against the interior of the unit sphere. This provides a distorted fish-eye view and presents the nodes of interest in detail and maintains a sense of context which is discussed in the next Section.

2) *Graph Layout Algorithm:* Graph layout using hyperbolic geometry was implemented independently at Xerox PARC [10] and at the Geometry Center [6]. It requires as input a minimum spanning tree of the graph to be displayed. The two-pass (bottom-up and top-down) graph layout algorithm places the subtrees which consist of a parent node subtended by its children on the surface of a hemisphere as shown in Fig. 2.

3) *Focus and Context Technique:* It is not possible to view a large graph in its entirety without discarding some information. The idea behind focus and context visualization is therefore to present the objects of interest to the user in detail while retaining an overview or a context of the remaining objects which are of less interest to the user. The projection allows objects near the origin of the unit sphere to appear larger while those further away appear smaller as if they were increasingly compressed against the interior of the unit sphere.

The focus and context technique is accomplished as follows. The network structure is initially projected onto 3D Euclidean space. To focus upon another section, a point in Hyperbolic space must be chosen. The entire structure is translated in Hyperbolic space such that the selected point is at the origin. Once the structure has been translated it is re-projected onto the interior of the unit sphere in 3D Euclidean space, but with a different point as the origin of the sphere. This allows a different section of the network to be focused on. Although this may seem to be a time consuming task, the transformations are done using 4×4 matrices, which are implemented relatively efficiently in the Java virtual machine. Fig. 3 presents several focus and context visualizations of the same network.

4) *Optimizations:* Translating, projecting and rendering hundreds of thousands of nodes is a task that cannot be serially executed in a reasonable amount of time on commodity hardware. Optimizations have therefore been incorporated to

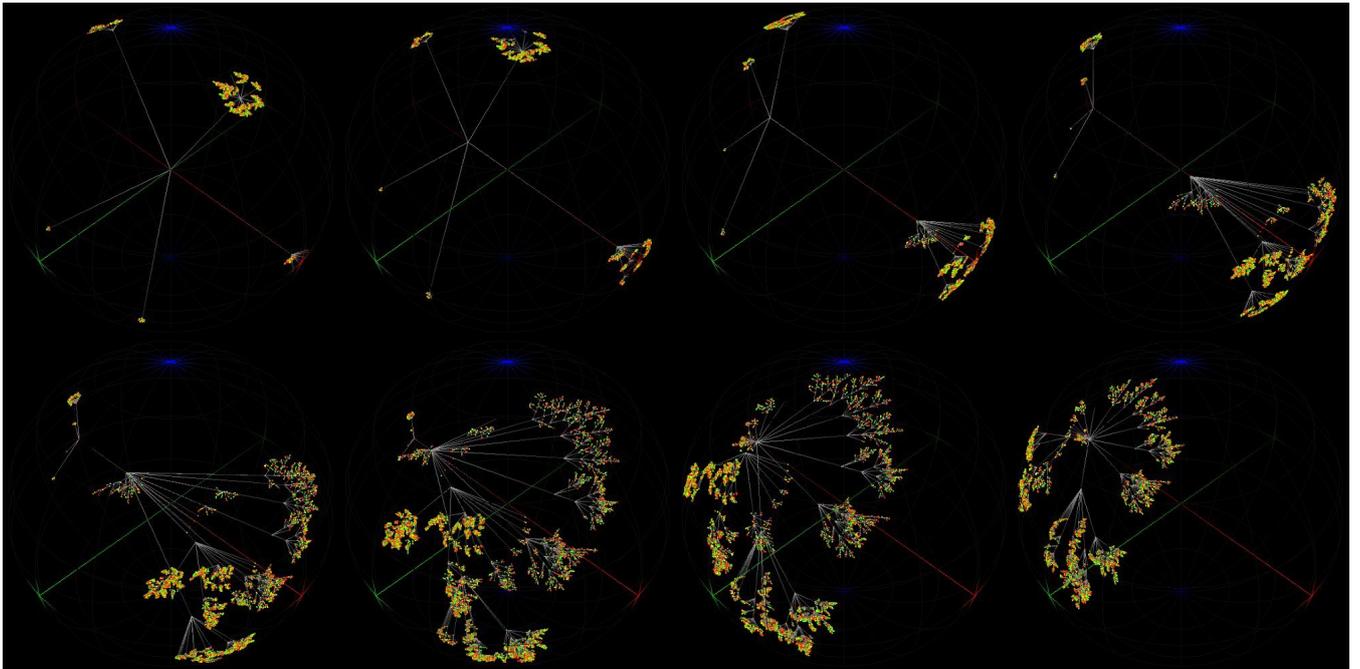


Fig. 3. Several screen shots taken while translating the network structure. This brings a different section of the network into focus while maintaining the surrounding context.

allow the user to interact with the system effectively.

First, when a node and its child subtrees are near the surface of the sphere, they are out of focus and typically project to less than one pixel. In these situations the children are not considered for processing and the parent node is substituted for the average of its children. This eliminates a large amount of processing.

The other technique ensures a constant frame rate when the visualization is rendered. The basis for this technique is the Adaptive Rendering algorithm [5]. Nodes closest to the origin (these nodes are in focus) are drawn until the allotted time for rendering that frame has elapsed. If the frame is idle and no user interaction is taking place, the allotted time for the frame is extended to allow more nodes to be drawn.

III. THE *Sun Cloud Manager*

The previous sections motivated the use of hyperbolic geometry to visualize the nodes which make up the cloud network. This forms the basis for the *Sun Cloud Manager* which provides a framework for the real-time analysis and management of the cloud network.

The solution devised is to cluster badly-behaved nodes together in an attempt to recognize a pattern within their characteristics and thereby identify the underlying cause of the problem. This provides information to be used when investigating potential problems within the cloud network.

A. Policies and Definitions

Each node has two sets of characteristics namely *static* characteristics and *dynamic* characteristics which can be used for clustering purposes [3]. The static characteristics of a

TABLE I
COLOUR CODING

Colour	Description
Green	Node dimension not utilized
Yellow	Node dimension utilization satisfactory
Red	Node dimension under stress
Purple	Node currently selected
Grey	Node disconnected or disabled

node include the operating system, hardware configuration and other fixed attributes. The dynamic characteristics indicate resource utilization, including CPU, memory and network usage. A node's colour indicates its overall resource utilization by mapping the average of the dynamic characteristics to a colour ranging from green to yellow to red. A well-behaved node has a low resource utilization and is thus green-yellow in colour. The definition of a well-behaved node (and hence its colour) is at the discretion of the administrator. Table I illustrates a typical choice of colour codes.

The graph layout is an abstraction of the cloud network. Although the layout does not correspond to the physical locations of the nodes, it provides a sense of how the nodes are connected. Not all connections are displayed in the visualization of the cloud network: for example connections between peers are not displayed. However, edges that are displayed between two nodes represent a physical network connection. The two-pass algorithm may be modified to include additional information e.g. more edges between nodes. On the other hand, extra information which is not of importance may overwhelm the administrator.

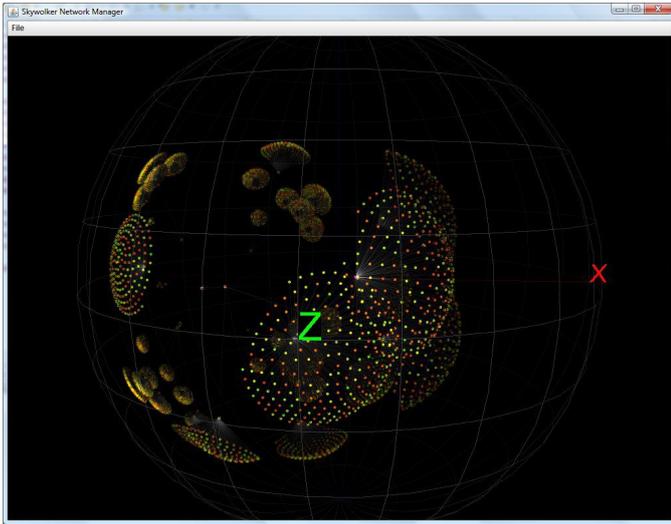


Fig. 4. The canvas window. Node clusters near the origin are in focus and appear to occupy more space and are in more detail, while nodes near the surface of the unit sphere appear compressed. In the extreme case a single node represents an entire subtree.

B. Cloud Network Database

A database containing information of the cloud structure must be provided to the *Sun Cloud Manager*. This database contains a row for each node with the following information

- the IP address
- the IP address of the parent node
- a specific shape to represent the node (i.e. sphere, cube)
- a list of static and dynamic characteristics.

Since the parent of each node is known in the database, a minimum spanning tree of the cloud network may be generated.

Each leaf node represents a physical machine in the cloud network. The root node represents an aggregated view of the entire cloud. The remaining nodes represent aggregating routers in the cloud network and they report aggregated values for their subtrees.

In practise, each node in the cloud network should run a daemon which reads the current dynamic characteristics of the nodes and sends a status update to a predetermined server where the *Sun Cloud Manager* will receive and process it. The protocol used when sending status updates is beyond the scope of this paper.

C. Software

The *Sun Cloud Manager* software package was developed to demonstrate the ideas presented in this paper. The package was initially created using C, OpenGL [12] and the window drawing toolkit QT. This combination of software resources allowed for portability. The software was later ported to Java using the native OpenGL wrapper JOGL. Java3D was not used in order to retain the ease of re-reporting the *Sun Cloud Manager* software. SQLite [11] was used for the database.

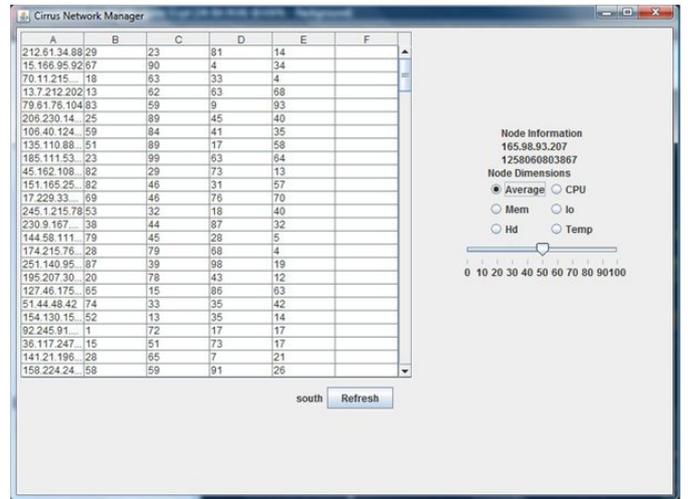


Fig. 5. The filter window. This window provides textual information displaying additional node details. Tools to filter nodes are also provided.

User Interface and Interaction: The *Sun Cloud Manager* displays two windows which are best presented on separate monitors. Fig. 4 shows the *canvas window* which displays the cloud network of nodes using an implementation of the hyperbolic viewer discussed in the previous section. Fig. 5 shows the *filter window* which displays textual and simple graphical information in the form of tables and graphs. The filter window also presents the user with notifications about the system.

The canvas window is constantly updated allowing the system to be monitored in real-time. Working with large graphs often requires focusing on another part of the graph. Immediately re-focusing on another part of the graph results in an instantaneous transition from one context to another which leaves the user confused as to where in the graph he is. Using short animations to ‘fly’ to another section of the graph provides an effective solution to this problem. As mentioned in Section II-A4, the adaptive rendering algorithm achieves a constant frame rate during animations, which allows for a fluid user interaction in the canvas window.

The unit sphere is illuminated from an angle to provide a sense of depth to the scene. The axes are coloured and remain visible as the nodes change colour. Without the axes the user becomes disorientated.

Navigation through the nodes is done by using the keyboard and mouse. Manipulating the currently projected section of the cloud network is done by using the w, a, s and d keys to move through 3D Euclidean space, effectively zooming in and out. In addition, the unit sphere may be rotated using the left mouse button.

Clicking on a node with the right mouse button selects the node. Once a node is selected, the enter key brings the node into focus by bringing it to the origin of the unit sphere via a short animation. The arrow keys change the node currently selected to either its parent, child or left or right peer. Similarly the space key brings the selected node into focus in the filter

window displaying its static and dynamic characteristics.

The filter window lists all the nodes in the subtree rooted by the currently selected node along with their accompanying details in a node table. Each cell of the table displays one dynamic characteristic of the node: the cell is coloured in the same fashion according to the formula previous discussed.

Navigating the nodes in 3D Euclidean space can be tedious, so the user is assisted with notifications, a filter box and other tools such as a threshold slider bar in the filter window. Clicking on a node in the node table causes the canvas window to 'fly' to the selected node. By default the canvas window colours the nodes using the average of their dynamic characteristics. This can be changed to colour the nodes based on any one particular dynamic characteristic.

The details of the currently selected node are displayed. The details include

- the IP address
- the last updated time
- the dynamic characteristics
- the static characteristics
- A pie chart of all the nodes in the subtree rooted by the currently selected node. The chart incorporates a standard deviation view, thus providing an extra dimension of information.

A threshold slider bar is provided to filter nodes in the canvas window. Only nodes whose dynamic characteristic is greater than a set threshold will be displayed. Well-behaved nodes are less important to an administrator and can in this way be de-emphasized in the visual display. Finally, a notification system is built into the filter window which alerts the administrator to potential problems within the cloud network which should be investigated.

D. Identifying Problems and Warning System

A linked list and a hash map are used to maintain a list of all nodes sorted by their last update time. The IP address of a node is used as an index into the hash table to locate the node and to update its dynamic characteristics. All nodes on the path to the root are similarly updated. This path typically contains very few parent and grandparent etc. nodes which contributes towards the scalability of the *Sun Cloud Manager*.

The *Sun Cloud Manager* is a multi-threaded application. One thread continually processes the queue of status updates received from the nodes in the cloud. A second thread, the *node analysis thread*, processes a list of badly-behaved nodes in an attempt to recognize a pattern amongst them so that the underlying cause of the problem can be identified.

Two methods for identifying problems are currently being implemented. The *Sun Cloud Manager* can recognize:

- 1) if a node and its subtree become disconnected.
- 2) potential problems in the cloud network by localizing the underlying problem to a set of static and dynamic characteristics.

1) *Disconnected Nodes*: Nodes for which a status update has not recently been received are reported and marked as disabled or disconnected in the canvas window.

Given a set of disconnected nodes, the algorithm finds the parent nodes which are common to the disconnected nodes. A parent is only considered to be disconnected if all its children are disconnected. The algorithm is thus recursive with the maximum number of iterations equal to the depth of the cloud network graph.

2) *Cluster analysis*: Cluster analysis is performed on nodes using their static and dynamic characteristics as the clustering criteria. A list of nodes sorted by the average of their dynamic characteristics is maintained. A number of the worst badly-behaved nodes are examined during cluster analysis. The well-behaved nodes are not subject to cluster analysis.

IV. EVALUATION

A simulation framework [4] was modified to generate a synthetic cloud network to test the system. The simulator generates a stream of update messages from a random cloud network according to the following specifications

- the maximum depth of the cloud network (the number of aggregating routers)
- the number of aggregating routers on each level of the cloud network
- the number of physical machines in the cloud network.

Data are extracted from the update messages and are inserted into the database which are subsequently read by the *Sun Cloud Manager*. The dynamic characteristics of each simulated physical machine vary constantly. During the simulation each physical machine periodically reports its dynamic characteristics to the server on which the *Sun Cloud Manager* is hosted.

A back-door was introduced into the simulation in order to simulate when a section of the network was disconnected. This allows the user to enter the IP address of the node to be disconnected. After a set period has been reached, all nodes in the disconnected subtree time out and become disconnected. The display window renders disconnected subtrees and nodes in grey. The filter window notifies the user that there are nodes which are disconnected. The user may then click on the notification, bringing that section of the graph into view.

V. CONCLUSIONS

This paper describes the methods used to visualize and identify possible problems within a cloud computing network. A technique, rendering the entire cloud network in Hyperbolic space, for visualizing the physical layout of the network is discussed. Issues concerning scalability with regards to effectively identifying problems within the cloud are discussed. The *Sun Cloud Manager* was created to demonstrate the techniques and data were generated to simulate real-time performance data obtained from the network.

The *Sun Cloud Manager* is a work in progress. The next step will be to investigate various cluster analysis techniques in order to recognize a pattern within the static and dynamic characteristics of the nodes and to identify the underlying cause of problems within the cloud network.

REFERENCES

- [1] J.W. Anderson. "The Poincare Disc Model." Hyperbolic Geometry. New York: Springer-Verlag, 1999.
- [2] J.W. Cannon, W.J. Floyd, R. Kenyon and W.R. Parry. "Hyperbolic Geometry". Flavors of Geometry, MSRI Publications, Vol. 31. 1997.
- [3] G. Fung. "A Comprehensive Overview of Basic Clustering Algorithms". Department of Computer Science, University of Wisconsin, America, June 2001.
- [4] N.F. Huysamen and A.E. Krzesinski. "A Scalable Network Monitoring and Bandwidth Throttling System for Cloud Computing". Proceedings SATNAC 2010 Southern African Telecommunication Networks and Applications Conference, Stellenbosch, South Africa, September 2010.
- [5] T. Munzner. "Interactive Visualization of Large Graphs and Networks". Department of Computer Science, University of British Columbia, Vancouver, Canada, June 2000.
- [6] M. Phillips and C. Gunn. "Visualizing hyperbolic space: unusual uses of 4x4 matrices". The Geometry Center, December 1991.
- [7] G. Reese. "Cloud Application Architectures: Building Applications and Infrastructure in the Cloud". O'Reilly Media (April 2009).
- [8] W. Stother. Department of Mathematics, University of Glasgow, Scotland.
<http://www.maths.gla.ac.uk/wws/cabripages/hyperbolic/hyperbolic0.html>
- [9] W.P. Thurston and J.R. Weeks. "The Mathematics of Three-dimensional Manifolds". Scientific American, pp 108–120, July 1984.
- [10] Corporate Association for Internet Data, <http://www.caida.org>
- [11] SQLite, <http://www.sqlite.org>
- [12] OPENGL, <http://www.opengl.org>
- [13] E. Weisstein. "Klein-Beltrami Model." From MathWorld—A Wolfram Web Resource.
<http://mathworld.wolfram.com/Klein-BeltramiModel.html>.

Marc de Klerk obtained the BSc (Hons) degree in Computer Science from the University of Stellenbosch, South Africa.

Anthony Krzesinski is a Professor of Computer Science at the University of Stellenbosch, South Africa.