LA-UR-02-1929

Title: GRAPH VISUALIZATION FOR THE ANALYSIS OF THE STRUCTURE AND DYNAMICS OF EXTREME-SCALE SUPERCOMPUTERS

Author(s):
Kathryn Berkbigler
Brian Bush
Kei Davis
Adolfy Hoisie
Steve Smith

Submitted to:
InfoViz 2000
IEEE Symposium on Information Visualization
Boston, MA
28-29 October 2002

Los Alamos
NATIONAL LABORATORY

Graph Visualization for the Analysis of the Structure and Dynamics of Extreme-Scale Supercomputers

Kathryn Berkbigler, Brian Bush,
Kei Davis, Adolfy Hoisie, and
Steve Smith (sas@lanl.gov)
Los Alamos National Laboratory,
Los Alamos, NM 87545

Cheng Zhou, Kenneth L. Summers, and
Thomas P. Caudell (tpc@eece.unm.edu)
Albuquerque High Performance Computing Center
University of New Mexico
Albuquerque, NM 87131

Abstract

We are exploring the development and application of information visualization techniques for the analysis of new extreme-scale supercomputer architectures. Modern supercomputers typically comprise very large clusters of commodity SMPs interconnected by possibly dense and often nonstandard networks. The scale, complexity, and inherent nonlocality of the structure and dynamics of this hardware, and the systems and applications distributed over it, challenge traditional analysis methods. As part of the à la carte *team at Los Alamos National Laboratory, who are simulating these advanced architectures, we are exploring advanced visualization techniques and creating tools to provide intuitive exploration, discovery, and analysis of these simulations. This work complements existing and emerging algorithmic analysis tools. Here we gives background on the problem domain, a description of a prototypical computer architecture of interest (on the order of 10,000 processors connected by a quaternary fat-tree network), and presentations of several visualizations of the simulation data that make clear the flow of data in the interconnection network.*

1 Introduction

The magnitude of the scientific computations targeted by the US Department of Energy Accelerated Strategic Computing Initiative (ASCI) project requires as-yet unavailable computational power. To facilitate these computations ASCI plans to deploy massive computing platforms, possibly consisting of tens of thousands of processors, capable of achieving 10-100 teraOPS.

Better hardware design and lower development costs require performance evaluation, analysis, and modeling of parallel applications and architectures, and in particular a *predictive* capability.

The tools of the trade in performance modeling and analysis are typically categorized as algorithmic/analytical analysis, statistical analysis, analysis with queuing theory, and simulation. For systems of ASCI-proposed size and complexity *simulation* is the predictive tool of choice, though simulation may be considerably augmented by analytical and statistical analysis [1]. Because of the sheer volume of relevant data generated by a simulation run, *visualization* is an important, potentially the primary, method of practical data abstraction and comprehension.

1.1 Goals

The *à la carte* project seeks to develop a simulation-based analysis tool for evaluating massively-parallel computing platforms including current and future ASCI-scale systems. While developing a general framework for building these simulations and analyzing them, an intermediate goal is to model, and validate the model of, the ASCI Q machine [2] with a realistic ASCI workload.

It is clear that simulating systems of the size and complexity that we envision will require the use of parallel simulation [3]. DaSSF [4, 5], being developed at Dartmouth College, is the current choice for discrete-event simulation. Since this system does not include a visualization component to aid in the analysis of the complex time-varying interactions of the logical components, we are researching visualization methods independent of these tools. Because the network connecting the processors in the simulated machine is large and complex the visualization efforts have focused on representations of spatial/temporal graphs.

The following section describes the approach we have taken to visualize the results of these machine simulations, the Flatland visualization tool used for the 3D interactive environment, and the details of the models used to represent the machine switch fabric.

2 Visualization Approach

Our visualization efforts focus on viewing the execution of the simulation and on displaying the performance of the simulated system. Visualization also aids in debugging the simulation itself, in developing and evaluating the efficiency of load balancing of the simulation entities, and in understanding synchronization between simulation timelines. Visualizing the simulated system allows end-users to understand how varying workload or network interconnection architecture affects the overall performance of an advanced or novel architecture. They can also see communication patterns in the network, levels of network usage, and the presence of bottlenecks. Our visualization approach includes direct representations of the architecture as well as innovative abstractions of the architecture and dynamics of the system.

We assume each network switch has eight duplex I/O ports; the ports may be linked to computational nodes or to other switches. The network is organized into layers of switches that connect only to the layers above and below: for eight-port switches there are four upward connections and four downward connections yielding a quaternary fat-tree network: "quaternary" because each switch has four connections in each direction; "fat" because the number of switches per layer is the same for all layers.

Figure 1 illustrates the layout of such a network with 64 computational nodes and three layers of 16 switches each.

To aid in the following discussion, we introduce the some formalism describing these interconnection networks (or graphs). Let L be the number of layers in a complete quaternary fat-tree network. Number the layers $\ell = 1, \ldots, L$. Each layer has 4^{L-1} switches, so we label these with IDs $x = 0, \ldots, 4^{L-1} - 1$. This means there are $s_L = L \cdot 4^{L-1}$ switches in the network and $n_L = 4^L$ nodes connected to it. Each port p of switch x in layer ℓ connects to four switches

$$y_p = 4^\ell \left\lfloor \frac{x}{4^\ell} \right\rfloor + 4^{\ell-1} p + \left(x \bmod 4^{\ell-1}\right) \qquad (1)$$

in layer $\ell + 1$ at port $p + 4$, where $p = 0, 1, 2, 3$.

2.1 Flatland: An Immersive Visualization Development Framework

Flatland is an immersive visualization development framework developed at the University of New Mexico as part of the Homunculus Project [7]. It is used to facilitate rapid prototyping and research in scientific and information visualization, immersive environments and interfaces, and human factors engineering. The Flatland infrastructure aids in creating and managing complex scene graphs with OpenGL geometry, lighting, shadows, stereo rendering, and spatialized sound objects; dynamically loading applications without mutual interference; managing novel input and output devices; navigation of the resulting virtual spaces; and providing some basic, optional spatial reference objects such as a landscape, stars, or sun.

One of the tenets of our approach to visualization is that immersive 3D environments can offer unique advantages over non-immersive three-dimensional graphics, or two-dimensional plots, charts, or graphs. By placing the user of these tools within the same context as the objects being viewed and allowing the user to navigate around them, choosing their own point of view and using motion parallax to comprehend the three-dimensional relationships between objects, and allowing the objects to cast shadows and perhaps even to have behavior and emit sounds, a qualitative improvement in comprehension of the data is achieved. Figure 2 shows all of the representations referenced.

2.2 Direct Representation

To debug a simulation it is useful to have a visual representation that clearly and explicitly represents all of the simulated components. With the fat-tree connected massively parallel computing architectures, we started by laying out the processors, NICs, and switches in several relatively obvious direct representations. The first (Representation DL) is a simple distribution of processors and NICs along a line with the layers of switches above them, also along a line. The second (Representation DC) involves wrapping this line around into a circle, creating a sort of cone topped with a cylinder. The third (Representation DR) is a rectangular grid laydown of the processors, NICs and switches. The visualization is animated so that the user can see the progression of messages sent through the network.

Although the direct representations provide the highest level of detail of message properties and component connectivity, they suffer from the lack of scalability: the visual complexity of the display (i.e., number of overlapping entities) becomes overwhelming when large models—more than several hundred computational nodes—are displayed. The *ad hoc* attribute coloring capabilities make this representation extremely valuable for debugging purposes, however. The largest machine we have studied is a 4096 node machine with 6 layers of 1024 switches each. These representations were overwhelmed at that scale. For a more thorough description of these representations see the project website [6].

2.3 Layered Block Representation

To address the scalability and visual complexity issues of the direct representations we have developed a more abstract model, motivated by the graph connection matrix,

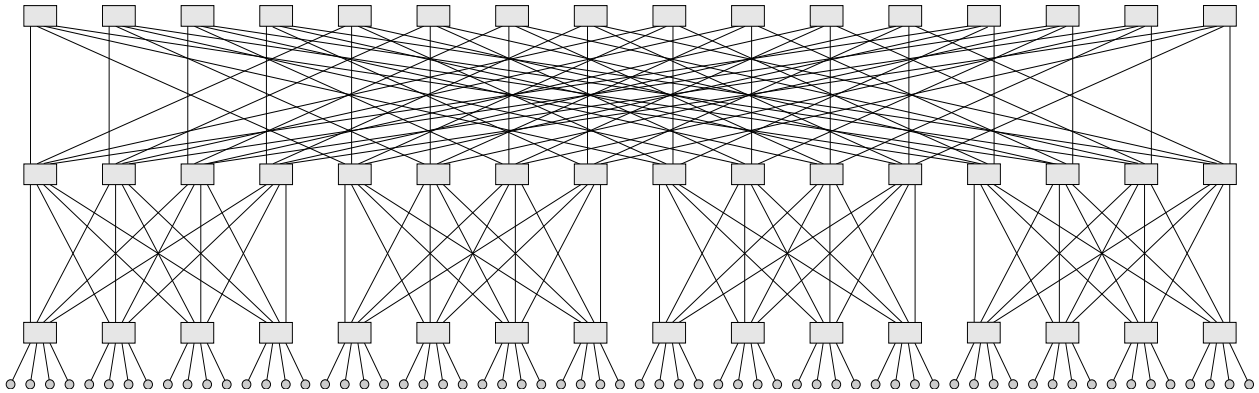

Figure 1. Representation QS: Quaternary fat-tree network with 64 computational nodes (small circles along the bottom) and three layers of 16 switches each (rectangles). Each switch has four duplex connections to the layer above and four duplex connections to the layer below.

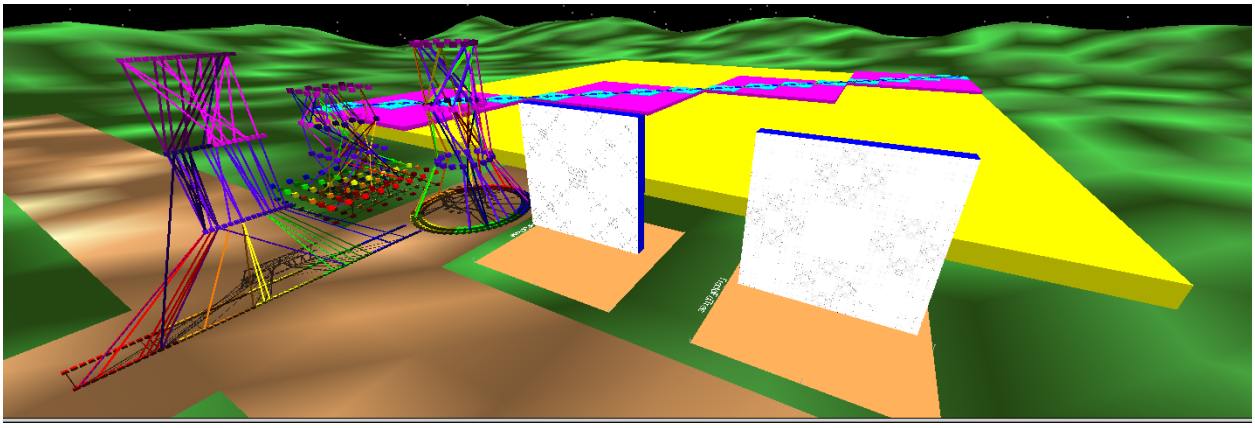

Figure 2. Six representations shown side-by-side in Flatland. Foreground, left to right: Representation DL: direct linear, Representation DR: direct rectangular, Representation DC: direct cone, Representation HT: H Tree, Representation F1: one fractal layout; Background: Representation LB: layered block.

where switch layers are grouped together into aggregate visual objects. The network is laid out on a square with equally-spaced pillars along the diagonal representing the computational nodes and their NICs. The various switch layers are grouped by fours on succeeding levels below. For example, the first level below the nodes consists of groups of four switches, the second level consists of groups of sixteen switches, etc. Figure 3 shows a view of this representation. When a processor sends a message to another processor through the switching fabric, a line or "pipe" leaves the processor and grows across the switching fabric until it comes to the point on the implied connectivity matrix where the two processors' connection would normally be indicated. At that point it makes a 90 degree turn and continues to grow until it reaches the destination processor. In one mode of use the layered switch blocks change color as they are involved in more and more communication traffic, allowing the viewer to recognize the relative level of utilization of each switch or switch group. It is also possible to subdivide a switch group vertically into individual switches.

The layered representation provides a more compact display than the direct representation and is somewhat more scalable, but it still suffers from similar problems once the number of nodes approaches one thousand. It does have the advantage of being able to show the activity in very large systems (e.g., 4096 computational nodes) if the pipes showing individual messages are suppressed. Unfortunately, the connectivity matrix scale is of order n_L, which naturally limits its scalability. At 4096 processors, individual processors, NICs, and even the first level of switches are smaller than a single pixel when viewed in their entirety, even on a high resolution screen (1600×1200). This is only a partial limitation since two dominant modes of use are likely to be a) macroscopic, attempting to understand the aggregate dynamics of the system which will be mostly differentiated at higher level switches, or b) microscopic, focusing on following single messages through the system. Nevertheless, in this representation, full microscopic detail and system wide macroscopic context cannot be apprehended simultaneously.

2.4 General Framework for Representations of Fat Trees

We now consider a general framework that allows us to express the layout of computational nodes and network switches in a two-dimensional compact scalable visualization of a fat-tree. These representations efficiently pack the switches into a planar image-like structure that utilizes the hierarchical nature of the fat-tree. With this representation, it is sufficient to consider only the switches in the network layout, as each quadruplet of computational nodes connects to only a single switch—the lowest-level switch in any layout can be replaced by that switch and its four connected nodes. The compact method maps switches in layers into the spatial coordinates of cells in a 2D plane.

A general technique to code the plethora of possible fat-tree compact 2D representations is a follows: Consider a pair of generating functions A_n and B_n which map the integers $0, 1, 2, 3$ to pixel coordinates; here n is a non-negative integer specifying the scale of the mapping. It is very important that the range of the functions A_n and B_n are disjoint—otherwise, switches on different layers will overlap on the same cell. Since the fat-tree is quaternary, it is useful to represent switch IDs in base four: namely, represent a number x as $x = \sum_{n=1}^{L-1} 4^{n-1} x_n$ where the x_n are its base-four digits. We can now write the cell coordinates of switch x in layer ℓ out of a total of L layers as:

$$F_{L,\ell}(x) = \begin{cases} (0,0) & \text{for } L = 1 \\ \sum_{n=\ell}^{L-1} A_n(x_n) & \text{for } \ell = 1 \\ \sum_{n=\ell}^{L-1} A_n(x_n) & \\ \quad + \sum_{n=1}^{\ell-1} B_n(x_n) & \text{for } 1 < \ell < L \\ \sum_{n=1}^{L-1} B_n(x_n) & \text{for } \ell = L \end{cases} \tag{2}$$

The layout is generated as the union of all the cell coordinates of the switches:

$$\bigcup_{\ell=1}^{L} \bigcup_{x=0}^{4^{L-1}-1} F_{L,\ell}(x). \tag{3}$$

Note that although we have considered two-dimensional layouts here, the formalism extends to the three-dimensional case.

2.5 Compact, Self-Similar, and Fractal Representations

Figure 4 shows two self-similar representations as implemented in Flatland.

The previous direct and connection matrix methods, which essentially scale with n_L, fail to scale up to large systems in several possible ways. While level of detail management would help make this *dynamic range* problem somewhat more graceful, it would not allow us to apprehend each individual node, NIC and switch, all at the same time, without improving the *compactness* of our layout. The *direct cone* representation described in Section 2.2 scales by approximately n_L, which is linear with n_L, but the *direct rectangular* layout scales by the $\sqrt{n_L}$. Thus the 4096 node system, for example, only requires an area on the order of 64×64 units to display. This seems very promising but by distributing the processors in a 64×64 array, the majority of the processors are in the middle of the area and similarly, the switching layers, laid out in 32×32 arrays tend to

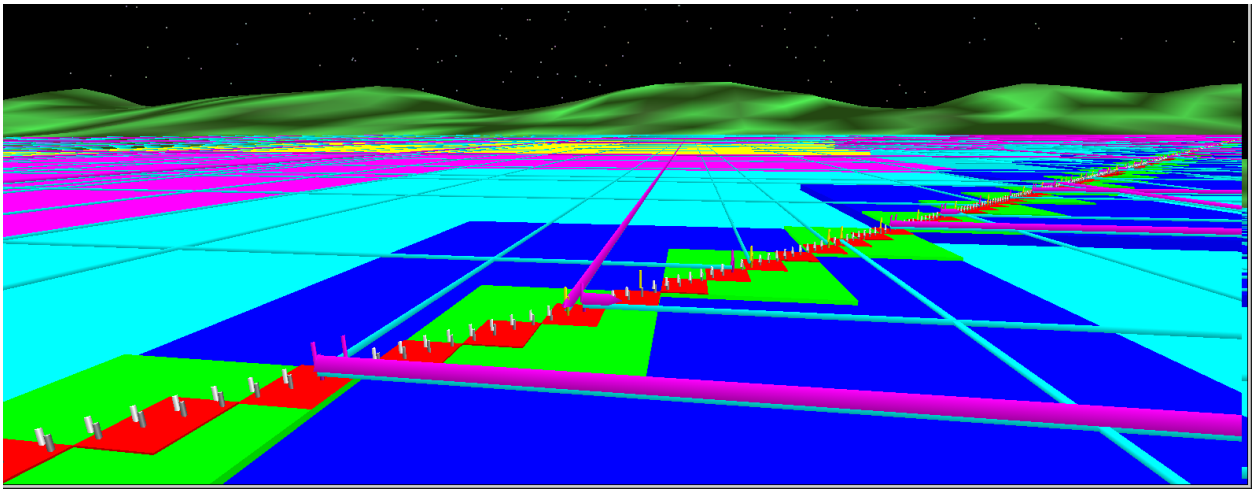

Figure 3. Representation LB: The network is laid out in a square with pillars representing nodes along the diagonal. The first layer of switches is below this in groups of four (shown in red), the second layer of switches is below that in groups of sixteen (shown in green)—subsequent layers are blue (3rd), cyan (4th), magenta (5th), and yellow (6th). The switch groups brighten in color when one of their switches is active, and the pipes represent individual messages passing through the switches below them.

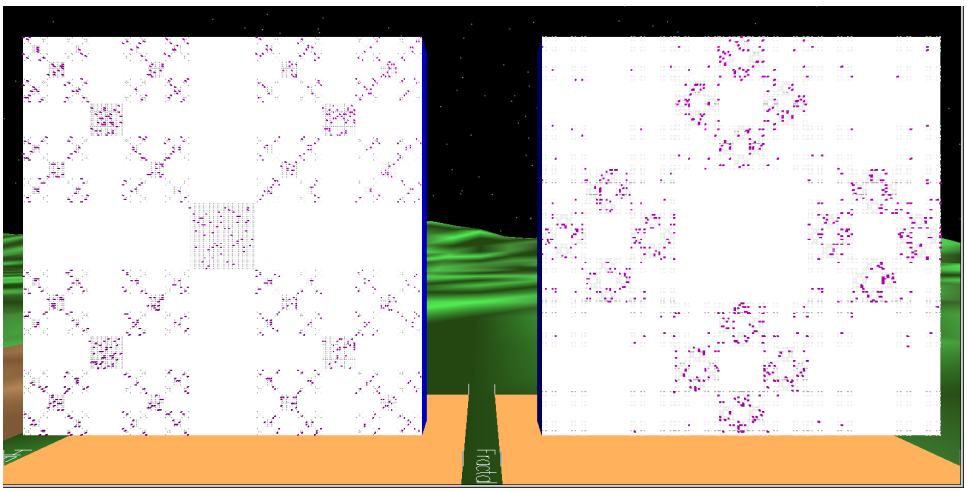

Figure 4. The H Tree representation (HT) is on the left, and a fractal representation F1 is on the right.

occlude each other nearly as badly. From this simple analysis, it appears that laying the processors, NICs, and each switch layer out in two dimensions is compact enough for our needs but leads immediately to occlusion problems.

Motivated by the somewhat self-similar nature of the fat-tree, we investigated two different compact 2D representations, one inspired by a simple pair of fractal generators as described in Section 2.5.1 and another inspired by *H-array radar antennae* as described in Section 2.5.2 below. The similarities between these two representations lead us to consider a more general representation of all layouts in two dimensions of fat-trees.

2.5.1 Fractal Representation

We start with a fractal-based representation defined by

$$A_n(k) = 3^{n-1}a(k) \text{ and } B_n(k) = 3^{n-1}b(k) \qquad (4)$$

where

$$a(k) = \begin{cases} (0,1) & \text{for } k = 0 \\ (1,0) & \text{for } k = 1 \\ (0,-1) & \text{for } k = 2 \\ (-1,0) & \text{for } k = 3 \end{cases} \qquad (5)$$

and

$$b(k) = \begin{cases} (-1,1) & \text{for } k = 0 \\ (1,1) & \text{for } k = 1 \\ (1,-1) & \text{for } k = 2 \\ (-1,-1) & \text{for } k = 3 \end{cases}. \qquad (6)$$

The function $a(k)$ places the lower-layer switches on the sides of a square, while the function $b(k)$ places the higher-layer ones on the corners of the same square. The 3^{n-1} coefficient in Eq. (4) ensures that subsequent squares are appropriately scaled to a larger size. Figure 5 shows the fractal for the first several L—one can plainly see the self-similarity between networks of different sizes.

This layout has the advantage that in scales well—one can represent the 6144 switches of a six-layer fat-tree in a 243×243 cell area, for instance. It has a disadvantage that the switches for different layers are interleaved, making it somewhat difficult to visually separate the activity in different network layers. We have used animated versions of these layouts to successfully distinguish the distribution of messages in two 4096-computational-node applications with different communication patterns, however.

2.5.2 "Fat H" Representation

We can address the problem of switch layers being interleaved in Representation F1 in a new representation defined by

$$A_n(k) = (n+2)2^{n-2}b(k) \text{ and } B_n(k) = 2^{n-2}b(k). \qquad (7)$$

In this case both the lower and upper level switches lie on the corners of a square, but the coefficient $(n+2)$ in Eq. (7) forces the lower level switches onto the corners of a larger square. Thus the more central groups of switches are in higher layers. Figure 6 illustrates this.

This representation has a fractal dimension $d = 2$, which "efficiently" fills two-dimensional space but is not technically fractal (since it does not have fractional dimension). One can see this in Figure 6 in that the highest, central layer of switches occupies a smaller region of the diagram relative to the layers below as L increases. Hence, the layout is not self-similar as a function of L. One could rectify this situation by altering the scaling factors in Eq. (7) at the expense of letting higher layers take relatively more area in the diagram.

2.5.3 Additional Representations

We can also reformulate other representations in terms of the formalism of Equation (2). In general, to create additional representations, one only needs to choose a pair $(A_n(k), B_n(k))$ whose ranges do not overlap.

2.5.4 Comparison

It is instructive to compare the advantages and disadvantages of some of the fat-tree representations we have created so far. Table 1 ranks all six representations in terms of usability metrics.

3 Conclusion

The representations developed here have been useful in helping both casual observers and researchers intimate with the topology of the architecture of the proposed ASCI Q machine get a better intuitive understanding of its structure. Animating the output from the simulations has been useful in troubleshooting the simulation. So far, the only simulations which have been run are of a uniform distribution of communication and a distribution which approximates the kinds of calculations done on volumetric grids where each cell in a grid communicates only with its nearest neighbors. We have been able to see, in the block structure and early compact, self-similar representations, the characteristic differences between these two distributions, most notably the localization of message traffic in the latter example and the attendant reduction in utilization of higher switch levels.

As we continue to investigate and refine the compact, self-similar representations, and begin to use all the representations to analyze the behaviour of more interesting data sets, we hope to see more subtle features in these data sets and most importantly in more realistic models of application mixes running on these machines.

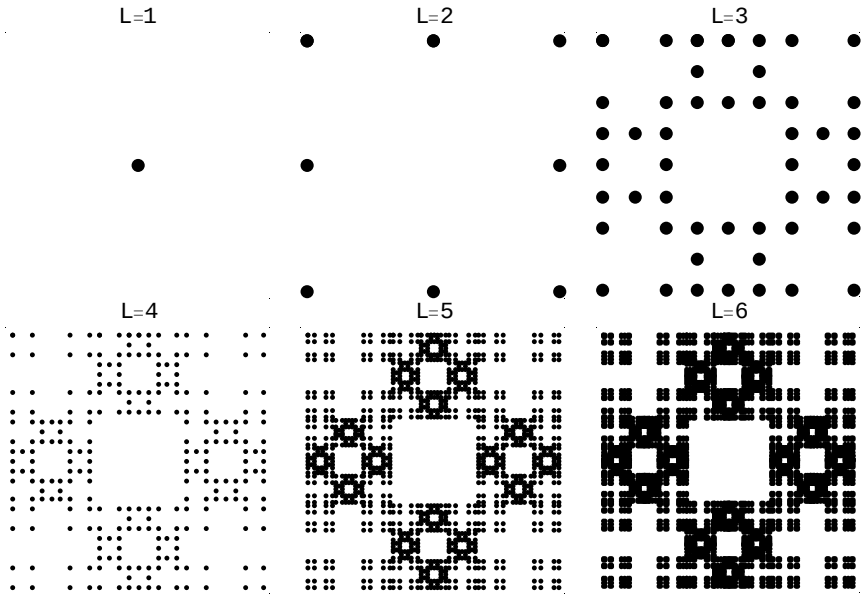

Figure 5. Representation F1: The six panels show placement of switches (circles) for the representation generated by Eq. (4) for $L = 1, \ldots, 6$.

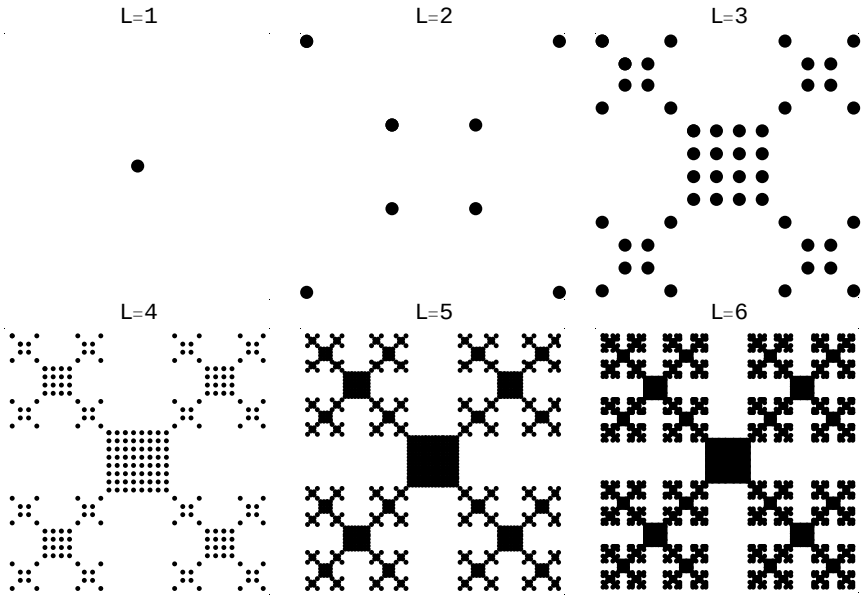

Figure 6. Representation HT: The six panels show placement of switches (circles) for the representation generated by Eq. (7) for $L = 1, \ldots, 6$.

Property	QS	DL	DC	DR	LB	F1	HT
Microscopic Detail	very	very	very	very	yes	somewhat	somewhat
Scalability	no	no	somewhat	yes	somewhat	yes	very
Visual Complexity	very high	high	moderate	high	moderate	moderate	low

Table 1. Advantages and disadvantages of various representations.

3.1 Future Work

Future work includes: Elaborating the H Tree and Fractal representations (HT and F1) with color, shape, and size coding; Extending the nearly two-dimensional, compact, self-similar representations into three dimensions; Exploring other self-similar representations; Adding optional, explicit representation of message traffic. We also plan to apply these same visualizations to real message traffic obtained from these systems as they are built and instrumented. We also anticipate that these tools can be used to analyze application level message traffic such as is generated by MPI or OpenMP based parallel programs. We are evaluating the data formats used by the Vampir performance analysis tools as a possible bridge to allow us to use those tools as well as to use our tools on Vampir trace data.

4 Acknowledgements

This work was carried out under the auspices of the Department of Energy at Los Alamos National Laboratory under ASCI DisCom2. We would like to thank LANL's DisCom2 project leader, Steven Turpin, for his support. We would also like to thank the Albuquerque High Performance Computing Center for their support.

References

[1] Francis J. Alexander, Kathryn Berkbigler, Graham booker, Brian Bush, Kei Davis, Adolfy Hoisie, and Steve Smith. *Design and Implementation of Low- and Medium-Fidelity Network Simulations of a 30-TeraOPS System*, Los Alamos National Laboratory, 2002.

[2] R. Kaufman, *The Q Supercomputer and Compaq.* High Performance Technical Computing News, Issue 18, November 2000, http://www.compaq.com/hpc/news/news_hpc_60171.html.

[3] Richard M. Fujimoto. *Parallel and Distributed Simulation Systems*, John Wiley & Sons, Inc., 2000.

[4] James H. Cowie, David M. Nicol, Andy T. Ogielski. *Modeling the Global Internet.* Computing in Science & Engineering 1(1):30-38, 1999.

[5] Jason Liu and David M. Nicol. *Dartmouth Scalable Simulation Framework User's Manual.* Dartmouth College Dept. of Computer Science, February 6, 2002.

[6] http://www.c3.lanl.gov/~parsim

[7] http://www.ahpcc.unm.edu/homunculus/indexold.html