Seeing It All at Once with Barry[†]

Neil J. Gunther[a] and Mario François Jauvin[b]
Performance Dynamcis Company, Castro Valley, California, USA[a]
MFJ Associates, Ottawa, Ontario, Canada[b]
{njgunther@perfdynamics.com, mario@mfjassociates.net}

Improving data visualization paradigms for performance management is an orphaned area of tool development. Tool vendors avoid investing in development if they see no demand, while capacity planners and performance analysts do not demand what they have not conceived. We attempt to cut this Gordian knot with 'Barry'; a 3D performance visualization suite based on barycentric coordinates. Potentially thousands of active processors, servers, network segments or applications can be viewed as a moving cloud of points that produces easily comprehended visual patterns due to correlations in the workload dynamics. Barry provides an optimal impedance match between the measured computer system and the cognitive computer system (your brain).

The purpose of computing is insight, not numbers. —Richard Hamming

1 INTRODUCTION

MacSpin [1] was a program written for the Macintosh computer circa 1986, which facilitated using the mouse as a means for orienting the apparent viewing angle of multivariate data in 3-dimensions. One of the example data sets was automobile engine attributes: MPG, displacement, horsepower, weight, etc., represented as a 3-dimensional scatter plot. Any three engine-attributes could be selected and their respective values defined on the (x, y, z) axes. Each statistical sample, of which there were perhaps 100, corresponded to a point in this volume. The initial view of these data was in the 2-dimensional (x, y)-plane and looked like a random scatter plot.

However, using the mouse to swivel the orientation of the axes or set them in constant rotation (hence the term "spin"), caused data points along the z-axis to come into view and one was surprised to discover that these otherwise random points actually lay in distinct parallel bands; all of which had been superimposed in the (x, y)-view. The meaning of the periodic bands notwithstanding, the ability to uncover such hidden structure in the data by the physical action of bringing the *third dimension* into view, was a watershed moment in personal computing.

MacSpin was inspired by an earlier program called *PRIM-9* (Position Rotation Isolation Masking in 9-dimensions) developed in 1973 on IBM 360 computers by a team led by John Tukey [2, Ch. 14]. Tukey, a mathematical statistician, is the father of "exploratory data analysis" or what has become known as "information visualization" (InfoViz). The concept of "spinning" data in 3D is now de rigueur for statistical packages like SAS, S-Plus and R, as well as mathematical software like *Mathematica*; which we use here. Physicists took up the gauntlet and developed "scientific visualization" (SciViz) during the 1990s (see Section 2.1). In our view, we are still waiting for the wholesale application of similar techniques to performance data and the purpose of this paper is to present a coherent set of paradigms for "performance visualization" (PerfViz) based on *barycentric coordinates* (defined in Sect. 4). It is our fervent hope that this small contribution might spark more interest in creating the next generation of PerfViz tools.

We take the position that the design of new PerfViz tools

should be firmly rooted in geometry. Based on our experience, certain visual properties emerge from the selected geometry which then appear useful for PerfViz. At this stage in our understanding it is only possible to speculate as to why this particular choice of geometry carries the attributes of good design, but even speculative statements about PerfViz design represent an advance over the informality of undirected experiment. With such design criteria in place, it then becomes possible to propose enhancements for our performance monitor which draw on more elaborate display technologies.

The concept of performance visualization is not new. Similar terms have become common currency in the domains of the performance analysis for parallel processing and high performance computing (HPC), as well as data networks [3]. Example project titles include: *HPVC: High Performance Visualization Center* [4], *Visualization for Parallel Performance Evaluation and Optimization* [5], *ParaGraph: A Performance Visualization Tool for MPI* [6]. The emphasis there, however, has been on creating tools to help identify where higher degrees of parallelism might be attained in scientific and engineering codes. Here, we are referring to the development and application of any type of visual aids for general purpose performance analysis and capacity planning. These might also include some of the ideas that have arisen out of the HPC community.

The reader might find the ideas presented here to be most useful when it comes to the procurement of new performance tools or the renewal of license fees for existing performance tools. It is then that a vendor is likely to be most receptive to new ideas like those presented here. In that way you might contribute to the improvement of PerfViz.

2 BEST IMPEDANCE MATCH

At the outset it is important to emphasize that although visualization techniques (some of which we are using to construct this presentation) have become ubiquitous with the advent of modern computing devices, knowing what constitutes *optimal* visualization is a very deep problem; primarily because we know so little about the workings of the human visual system (Fig. 1). Trial and error is the most common arbiter used to decide what constitutes good or bad visualization. The goal of PerfViz is easy to state but still difficult to implement. *We seek the best visual impedance match between the digital computer being measured and the cognitive computer (the visual system in our brain) which interprets those measurements.* A good impedance match is one that allows the brain to waste less energy deciphering a bad visualization and expend more energy on solving the performance problem at hand.

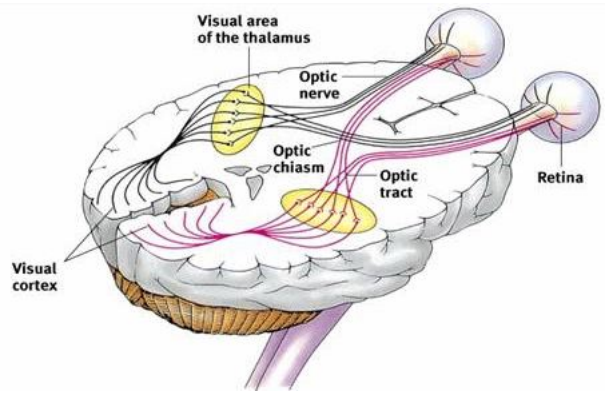


Figure 1: Human visual subsystems

Certainly we do not understand all the neural circuitry of the brain (which appears to be a very novel kind of non-Von Neumann parallel distributed-computer), but we do know quite a lot about certain pieces of the brain's neural circuitry and in particular the *visual* system. The most recent research suggests that the retina appears to form a sequence of movie-like frames (`http://blip.tv/file/175244`) containing data akin to colorized fourier transforms [7]. One dominant feature of the brain in general, and the visual cortex in particular, is that it is an excellent *differential analyzer*.

Figure 2: The apparent size of the moon is literally computed by the circuits in your retina and visual cortex and is therefore prone to errors known as *optical illusions*

For example, you may not realize it but, your brain actu-

ally computes things like size and color. Size computations you already know something about. Now that the weather is improving, go outside some night with your camera and wait for moonrise. The moon looks huge near the horizon (Fig. 2) but smaller at its zenith. Take a photo of the moon in the same angular positions and you will see that it's size remains invariant in the photograph. How can that be? The usual answer is that it is an *optical illusion*, but that doesn't explain why.

The "illusion" occurs because your brain *mis*-computes the size of the moon based on the size of objects within your visual field. Your brain does comparisons (or differential analysis) using the size of known objects near the horizon together with the knowledge that the moon is not nearby (you never touched the moon), and overestimates its size. Yes, your brain gets it wrong! At its zenith, however, there are no familiar objects to bias the sense of dimension, so your brain calculates the size of the moon correctly (i.e., consistent with the photograph).

The same thing applies to color. We do not *see* color, we compute it! Even if our eyes are receiving the same 650 nanometer red photons from a red flower, we don't see the same color red. The perceived color is determined by the color of the flowers surrounding the red flower we're looking at. The shade of red we see will be different if the surrounding flowers are blue rather than yellow. Moreover, things like *contrast* and *edges* are actually computed in the retina of your eye prior to being received by the visual cortex at the back of your head.

Wait a minute! We just pointed out that when it comes to the size of the moon or the color of a flower, our cognitive computer may get it wrong! And what about performance analysis? That's a more serious subject that is supposed to be rational and accurate. How can we live with cognitive miscalculations? Well, you live with it every waking moment, and you seem to survive. The possibility of getting it "wrong" stems from our cognitive computer calculating *differences* rather than *absolutes* [8]. Since we can't change that, such cognitive *relativity* is a very important aspect of perception to keep in mind when considering any PerfViz methods and tools. Just because a particular visualization looks fancy does not make it good because the brain is prone to illusion and error. So, when we said earlier that the brain is an excellent differential analyzer, we meant that it is much more efficient at relative computations than an electronic digital computer which operates on absolutes viz., numbers. What we need are PerfViz cues that present relative values to our cognitive computer with least error or illusion.

2.1 The Tyranny of Dimensions

When dealing with the complex set of tuning parameters in IBM's WLM (Workload Manager) (`www-03.ibm.com/servers/eserver/zseries/zos/wlm/`) in z/OS, it has been suggested that it is relatively straightforward for humans to visualize multiple dimensions[1], provided they are given the appropriate labels [9]. Conversely, some of us have trouble finding our shoes in the morning. Nonetheless, we certainly are bound physically by 3 spatial-dimensions and 1 time-dimension, yet we sometimes need to comprehend more abstract kinds of data that might become more comprehensible if we could appeal to higher dimensional visualizations. Like WLM tuning parameters, this need could hardly be more compelling than it is for comprehending computer performance data in general. Are there any role models that we might appeal to? Indeed, there is. Physicists have already attacked similar problems.

In the same way that physicists invented the World-Wide Web to solve the very real problem of exchanging massive amounts of data collected from "atom smashing" machines, they also recognized that all those desktop CPU cycles with graphical user-interfaces could be applied to the visualization of, not just the complex physical phenomena of particle physics, but also to comprehending the complex formation of tornadoes. This approach now belongs to the subject of *scientific visualization*.

Prior to scientific visualization, physicists were left with trying to comprehend the complex formation of tornadoes by numerically solving certain complicated differential equations. Without visualization tools, the solutions remain a morass of calculated numbers. A pile of computed numbers still has a very poor cognitive impedance match because it are indistinguishable from sampled statistical data or monitored performance data. With the advent of scientific visualization, the pile of calculated numbers can be rendered using colored animation such that the physicist sees the subtle evolution of the tornado meso-cloud and so on. The big-league version of this type of scientific visualization belongs to climate modeling, e.g., global warming, running on the *Earth Simulator* (`www.es.jamstec.go.jp/esc/eng/GC/index.html`). But why should the physicists, and the statisticians, for that matter, have all the fun? As practicing performance analysts, we could gladly use similar capabilities for visualizing a morass of performance metrics.

All non-relativistic physics (e.g., tornadoes) takes place

[1] One has to be careful to distinguish physical dimensions from degrees of freedom. See Sect. 6.

in $(3+1)$-dimensions: 3 space and 1 time. At high energies (e.g., cosmic rays), relativistic physics welds these dimensions together in the 4-dimensions of Minkowski space-time. There are some exotic proposals that all of physics might become unified in 11-dimensional space-time. But this is small potatoes when it comes to performance analysis, almost all of which takes place in n-dimensions where n is the number of independent performance metrics. A typical UNIX or Windows operating system samples on the order of $n = 500$ performance metrics every second. Even using Design-Of-Experiment techniques [10] to distill the most significant subset of these metrics, the number is likely to far exceed the $(3+1)$ or 11 of physics. In this sense, the physicists have lucked out. We performance analysts are stuck with trying to represent a large number of performance metrics on a 2-dimensional computer screen. This is the main problem we address in this paper. Essentially, SciViz has introduced *animation* (e.g., to encode time) and *color* (e.g., to encode temperature) as the key new visualization elements. In Sect. 6, we show how PerViz can also employ those same elements.

2.2 Visualization Criteria

It is possible to compile a set of criteria which ought to be met by any good PerfViz tools [11]:

1. Localized. Consumes the smallest possible area of 2-d screen real estate.
2. Dynamic. The ability to see temporal development e.g., through animation.
3. Multi-parameter. Allow multiple performance parameters to be viewed simultaneously.
4. Patterns. Facilitates the easy recognition of patterns and clustering in the data.
5. Cognition. Low cognitive overhead through the use of visual cues.
6. Semantics. Universally understood semantics (semiotics?) for the visual representation.
7. Generality. The ability to apply the same paradigm to different sets of performance parameters.
8. Journaling. Enables recording and redisplay of filtered performance effects.
9. Personalization. Perception is subjective so I need to be able to customize the visualization to suit my particular cognitive computer.

Table 2 (last page) applies these criteria to some of the PerfViz tools discussed subsequently in this paper. Certain paradigms, like parallel coordinates (Sect. 3.3), fail to meet this minimal set. Conversely, barycentric coordinates (Sect. 4) and treemaps (Sect. 3.4) fulfill more of them.

Personalization is very important aspect of PerfViz because everyone's brain is wired slightly differently. For example, one of us (NJG) is color blind to certain shades of green. We also need to distinguish clearly between what is *stylistic* and what is *principle*. Many people are fond of Edward Tufte's books and presentations. Tufte can show you what good graphics looks like (e.g., Napoleon's advance and retreat on Moscow in 1812 [12, p. 41]) and he can tell you what bad graphical habits are fostered by *PowerPoint* (`www.wired.com/wired/archive/11.09/ppt2.html`), but ultimately you need to understand basic principles. In general, Tufte tends to offer more examples than scientific principles. Conversely, the mathematical statisticians [2, 13, 14, 15] have spent more time and been more successful in developing principles of PerfViz.

Having presented the general motivation for our work, the remainder of this paper is organized as follows. The next Section 3 provides a brief historical overview of previous visualization schemes, some of which have also been applied to PerfViz. Section 4 introduces the *barycentric coordinates* which form the basis of our new *Barry-n* tools that contract the visual representation of n performance metrics into $d = n - 1$ spatial dimensions. We refer to this collection of tools as the *Barry007* suite. Section 5 presents various applications including, visualizing multiprocessor metrics and application response times using Barry-3, as well as network performance in Barry-4. In Section 6, we discuss some of the many potential visual enhancements that could be applied to our *Barry007* prototypes. Conclusions and future work are presented in Section 7.

3 PREVIOUS VISUALIZATION TOOLS

The n-dimensional performance metrics mentioned in Sect. 2.1 can be associated with degrees of freedom in the visual representation. The question becomes, how to project them onto a 2-dimensional surface such as an LCD display. From the earliest times, the 2-dimensional representation of choice has been a simple tabular format using simple ASCII characters. The limit of this type of 2-dimensional visualization is shown in Fig. 18, which presents performance information for a 32-way multipro-

cessor. The data for each processor appears on a different line. Graphical 2-dimensional representations are now more commonplace, and we now review some of them.

3.1 Chernoff Faces

One approach makes use of a well-known feature of our brains; our specialized circuits for recognizing human faces. Since we can process facial information very efficiently, it has been incorporated in *Chernoff faces* [14, 16] like those shown in Fig. 3. Chernoff faces can be used for any data having a large number of degrees of freedom. Each parameter or variable is mapped to a facial attribute such as the slope of an eyebrow, a smile or frown, and so on. Although Chernoff faces are obviously applicable to performance data, they do not seem to have been applied in PerfViz tools.

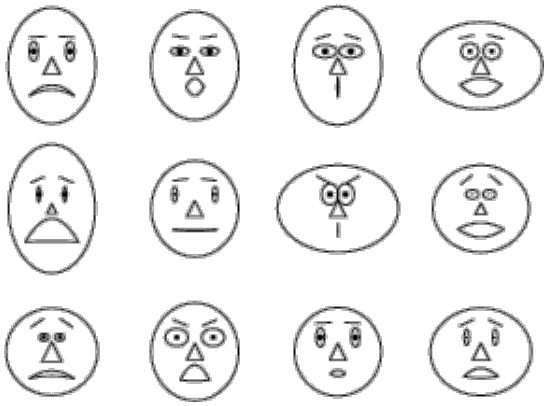


Figure 3: Chernoff faces

Nonetheless, Chernoff faces represent a good impedance match with our facial processing neurosystem and therefore helps us to spot visual anomalies very easily. One limitation, however, is that the semantics of the facial expressions is quite arbitrary and depends heavily on the context. This is not necessarily a bad thing, but one needs to have the context to know what the faces mean.

3.2 Kiviat Plots

A Kiviat plot (Fig. 4) is a graphical form based on polar coordinates, in which variables are plotted on radial axes at equal angular intervals [17]. The origin or zero lies at the center of the plot and 1 or 100% lies at the circumference. Kiviat graphs are also known as a "star plots" or "star graphs" [14, p. 360] in an InfoViz context.

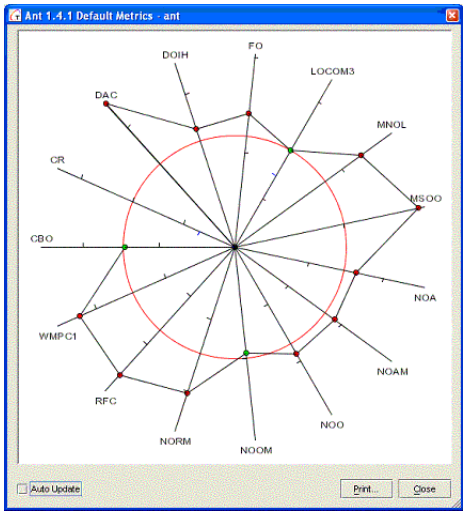


Figure 4: Kiviat plot

Broadly speaking, performance metrics can be classified according to whether they exhibit a *bigger is better* property, e.g., throughput, or a a *shorter is superior* property, e.g., response time. On a Kiviat plot, these metrics are arranged on alternating axes. Visual cueing comes from the appearance of loosely defined symmetries and arbitrary geometrical (polygonal) shapes. Ideal shapes have a high degree of symmetry, e.g., star shape, wedges, keels. Although these shapes can be dynamic [5], the assignment of metrics is rather arbitrary.

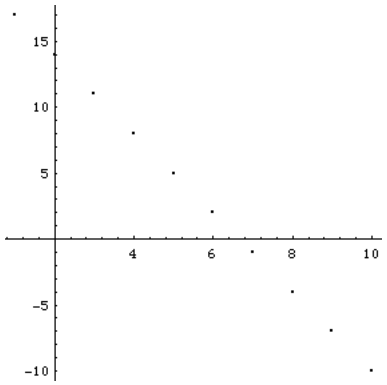
3.3 Parallel Coordinates

A completely different approach to compressing dimensions uses *Parallel Coordinates* [18]. Consider a straight line $y = mx+c$ with slope $m = -3$ and intercept $c = 20$, represented using standard orthogonal (x, y)-coordinates in Fig. 5.

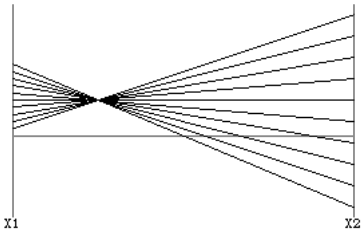
In order to represent $y = -3x + 20$ in parallel coordinates, we first treat the line as a set of points (Fig. 5(a)). The parallel coordinates labeled x_1 and x_2 in Fig. 5(b) are usually spaced equidistant from each other at positions $0, 1, 2, \ldots$. We see that each point on the original line now becomes a separate line in parallel coordinates. These lines intersect at the point:

$$\left(\frac{1}{1-m}, \frac{c}{1-m}\right)$$

or $(0.25, 5.0)$, in this case. Clearly, your usual geometric intuition goes out the window in a parallel coordinate system.



(a) A Cartesian line as points



(b) Line in 2-d parallel coordinates

Figure 5: A line in parallel coordinates

Interestingly, parallel coordinates have been applied to a 5-dimensional subset of the same automobile data mentioned in Sect. 1 (See §5 in `http://www.galaxy.gmu.edu/stats/syllabi/inft979/GeneralizedParallelCoordinates.pdf`). Figure 6 shows the points lying on the surface of a 3-dimensional sphere mapped onto 3 parallel coordinates (X_1, X_2, X_3). Unfortunately, this representation produces a rat's nest of lines that corresponds to cognitive noise. The poor visual impedance matching seems obvious. Color is often used as a discriminator [19, Chap. 11]. This probably explains why parallel coordinates have not found widespread application.

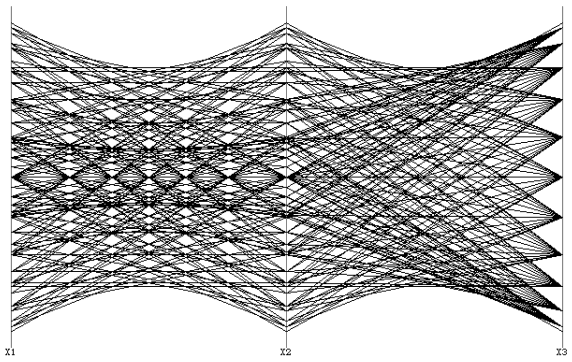


Figure 6: A sphere in parallel coordinates

3.4 Treemaps

Treemaps (`www.cs.umd.edu/hcil/treemap/`) are space-constrained visualization of hierarchies that were developed in 1991. Applications include the ability organize the *Google News* news aggregator (Fig. 7) by importance to the reader.

Figure 7: A treemap of Google News items

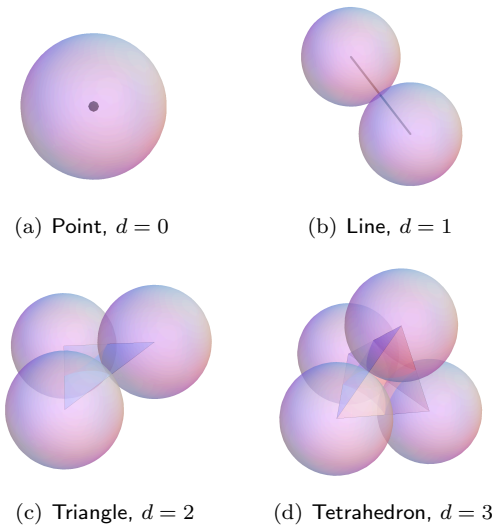
Several people have recognized treemaps as being useful for visually representing the state of a large server farm or an enterprise-wide collection of servers. For example, treemaps have been used to display a large number of servers or workloads as color-coded rectangles, where the size of each rectangle represents the capacity rating of the server and a traffic-light color (e.g., red, yellow, green) can represent its status or CPU utilization [20, 21]. Potentially, treemaps can be used to group and display all the servers in your enterprise. The relative merits of treemaps are compared with other tools in Table 2.

4 BARYCENTRIC COORDINATES

Our goal is to find ways to represent n-dimensional performance data (i.e., n performance metrics) in a d-dimensional space, such that those data are rendered with an optimal impedance match to the typical analyst's visual system. In this section, we show that barycentric coordinates can fulfill this role.

4.1 Simplexes

A few weeks after this CMG conference has ended, most of you will be busy wrapping Christmas gifts. The gifts are generally 3-dimensional objects and the wrapping paper is essentially 2-dimensional. To economize, you would like to use the least amount of wrapping paper to cover the gift. What you are doing, in fact, is computing the *convex hull*, i.e., the minimal surface that encloses all the vertices (points) of the gift you are wrapping.



(a) Point, $d = 0$ (b) Line, $d = 1$

(c) Triangle, $d = 2$ (d) Tetrahedron, $d = 3$

Figure 8: Progression of d-simplexes formed by connecting the centers of close-packed spheres with line segments. This progression tell us the appropriate geometry for barycentric coordinates in d-dimensions

Suppose now that you draw a set of dots on a piece of paper. You next locate the outermost dots of the set, and suppose further that there are just three of them. You stick toothpicks in those three outer dots and enclose a rubber-band around them. The rubber-band defines a triangle (with 3 sides of arbitrary length). The triangle forms another convex hull, this time in 2-dimensions. If we rearrange the outer dots so that they form an *equilateral triangle* (sides of equal length), then the convex hull is called a *simplex*, because it is the "simplest" polytope (many-sided figure) which encloses this 2-dimensional set of dots.

Using the handy symbol $\mathbb{R}^d$ for a d-dimensional Euclidean (flat) space, the 2-simplex in $\mathbb{R}^2$ for the rubber-band example is an *equilateral triangle*, the 3-simplex in $\mathbb{R}^3$ for the gift-wrapping example is a *tetrahedron*, the 4-simplex is a *pentatope* (whatever that looks like[2]), and so on. Notice that the degree of the simplex is identical to the Euclidean spatial dimension. This can be attributed to the connection between simplexes and packed spheres shown in Fig. 8.

How do simplexes help us with PerfViz? The answer is that a d-simplex contains $n = (d+1)$ degrees of freedom (i.e., n performance metrics) represented as *barycentric coordinates*; the simplex gives us the extra degree of freedom for free! Here is how it works. Consider a 2-simplex, the equilateral triangle of height $h = 1$ drawn in the (x, y)-plane (Fig. 9). The length of each side is $2/\sqrt{3}$. If we now draw a perpendicular line from each of the 3 faces or sides, they will intersect at some point p interior to the triangle. In Fig. 9(a) the distance to the point of intersection is given by the triple $(0.6, 0.3, 0.1)$. These lengths are the barycentric coordinates of that point. Note that the sum of their respective lengths is also 1; same as the height (h) of the triangle. Later, we shall refer to this as the *sum rule* (See eqn.(5)). The impatient reader, who is wondering where all this is going, is encouraged to skip ahead to Fig. 17 which shows $n = 4$ performance metrics for 1,000 items displayed in a 3-simplex. That is the destination, but we need to get there in small steps.

We generalize as follows. A d-simplex has $n = d + 1$ faces. The corresponding n barycentric coordinates are defined as the lines perpendicular to each face. If each vertex Vi, where $i = 1, 2, 3$, in Fig. 9(a) has Cartesian coordinates (Vi_x, Vi_y), and the corresponding barycentric coordinates have lengths a, b, c, then the Cartesian coordinates of the point p are given by:

$$\begin{aligned}(p_x, p_y) = (&aV1_x + bV2_x + cV3_x,\\ &aV1_y + bV2_y + cV3_y)\,.\end{aligned} \tag{1}$$

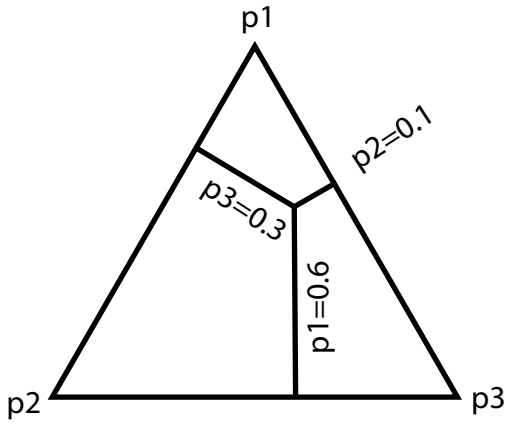
Consider the centroid in Fig. 9(b). Its barycentric coordinates are $(p_1 = \frac{1}{3}, p_2 = \frac{1}{3}, p_3 = \frac{1}{3})$, so its corresponding Cartesian coordinates are:

$$\begin{aligned}&(p_1V1_x + p_2V2_x + p_3V3_x,\ p_1V1_y + p_2V2_y + p_3V3_y)\\ &\quad = \left(\frac{1}{3}\frac{1}{\sqrt{3}} + 0 + \frac{1}{3}\frac{2}{\sqrt{3}},\ \ \frac{1}{3} + 0 + 0\right)\\ &\quad = \left(\frac{3}{3\sqrt{3}},\ \frac{1}{3}\right).\end{aligned}$$

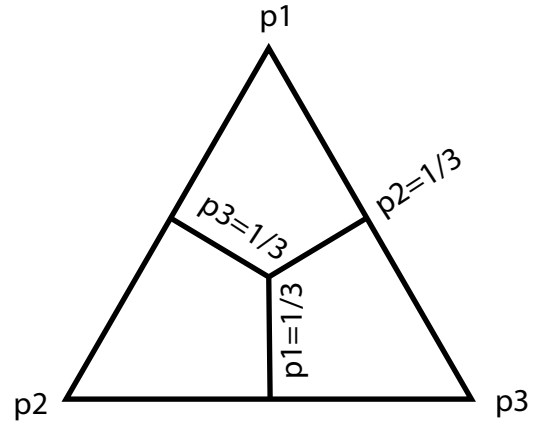
This simplifies to:

$$(p_x, p_y) = \left(\frac{1}{\sqrt{3}},\ \frac{1}{3}\right) \tag{2}$$

[2] It is not a triangular dipyramid i.e., 2 abutted tetrahedra with 6 faces. See `mathworld.wolfram.com/TriangularDipyramid.html` and Fig. 11.



(a) Arbitrary point inside the convex hull of a 2-simplex ($h = 1$) specified by three different barycentric coordinate lengths, which correspond to an uneven distribution of weights at each vertex



(b) The interior point located at the centroid, which corresponds to an even distribution of weights at each vertex making the barycentric coordinates each one-third the height $h = 1$

Figure 9: Barycentric coordinates in $\mathbb{R}^2$

in agreement with (1) and Fig. 9(b).

Barycentric coordinates were discovered and named by the German mathematician Franz Möbius in 1827. The name comes from the way these coordinates define the geometric *barycenter* or "center-of-mass" if weights were attached to each vertex, as described in Fig. 9. More recently, they have found application in fields as diverse as computer graphics, representing Markov states in queueing theory [22, p. 34] and Dalitz plots in high-energy particle physics [23, p. 301]

Although it is straightforward to define a simplex, it is not always easy to visualize them. Once again we find ourselves trapped in 3-space (cf. Sect. 2.1). For PerfViz applications, we require all the edges to be *equilateral* and *external* to the convex hull. As Fig. 10 demonstrates, this requirement cannot be met by any $d > 3$ simplex. Be careful not confuse faces with *edges*. The number of faces in a d-simplex is the same as the number of vertices. The number of edges, however, is given by:

$$^{(d+1)}C_2 = \frac{(d+1)!}{2!((d-1)!}$$

Only in the case of the 2-simplex are they the same. The third diagram from left in Fig. 10 shows a 4-simplex represented in $\mathbb{R}^2$ by a fully-connected graph. Clearly, this figure does not have a good visual impedance match for $n = 5$ performance metrics. What about a 4-simplex in higher-dimensional $\mathbb{R}^3$? Unfortunately, there is no such representation.

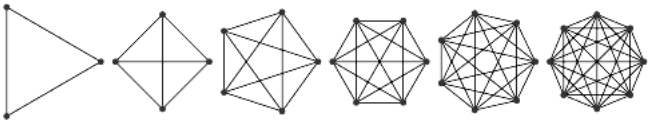


Figure 10: The projection of higher order d-simplexes ($d = 2, 3, 4, 5, 6, 7$) onto a 2-dimensional surface (e.g., this page or screen) demonstrates the increasingly poor visual impedance match

To verify this, consider extending the tetrahedron Fig. 8(d) as follows. We seek a fifth vertex such that it is equidistant each of the other tetrahedron vertices. Taking the three points at the base of the tetrahedron in Fig. 8(d), which are already equidistant to one another, we need to consider all possible locations in $\mathbb{R}^3$ where a fifth point could be located equidistant from those three vertices. There are only two possibilities, which are highlighted by the two large dots in Fig. 11. The lower dot cannot satisfy the equidistance constraint because it is further away from the top dot than any other vertex. The large top dot also fails to comply because it is not equidistant to the original top vertex; in fact, it is superimposed on that vertex. Therefore, there is no possible fifth vertex in $\mathbb{R}^3$ equidistant from all other vertices. Assuming that visualization in $\mathbb{R}^3$ has the best impedance match, this result gives us a clear hint that the visual impedance match of a 5-simplex is probably poor.

This discussion shows why a tetrahedron (3-simplex) is the *maximal* geometric representation for visualizing $n = 4$ performance metrics. As far as we know, this is the first time that limitation has been formally identified.

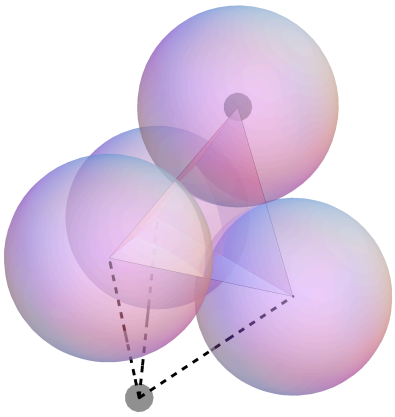


Figure 11: Attempt to represent a 4-simplex as a triangular dipyramid

4.2 Barry-2 Axes in $\mathbb{R}^1$

If two parameters, p_1 and p_2, represent two *independent* degrees of freedom, the appropriate geometrical representation is a 2-dimensional coordinate system (x, y) such that the location of a point in the (x, y)-plane is given by the values p_1 and p_2. If, however, the values of these parameters are restricted by barycentric the *sum rule*:

$$p_1 + p_2 = 1\,, \tag{3}$$

then one degree of freedom is removed and the range of parameter values can be represented on a 1-dimensional unit *line segment*.

4.3 Barry-3 Axes in $\mathbb{R}^2$

Next, we show that the locus of a point in $\mathbb{R}^2$, defined by barycentric coordinates, is bounded by an equilateral triangle. Hereafter, we assume *unit* height for such a triangle.

The location of any interior point is given by the distance along the three limbs that are perpendicular to each side. These limbs form the barycentric coordinates. Identifying the length of each limb with the parameters: p_1, p_2 and p_3, the centroid corresponds to: $p_1 = p_2 = p_3 = 1/3$. For all interior points, the sum of these three distances is equal to 1 such that:

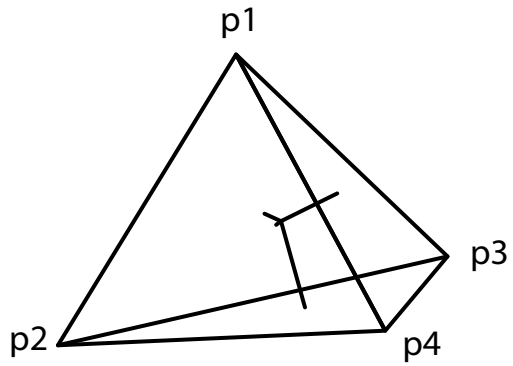
$$p_1 + p_2 + p_3 = 1 \tag{4}$$

is the corresponding 3-parameter sum rule. The invariant sum follows trivially from the way the three limbs partition the interior of the triangle in an area-invariant way.

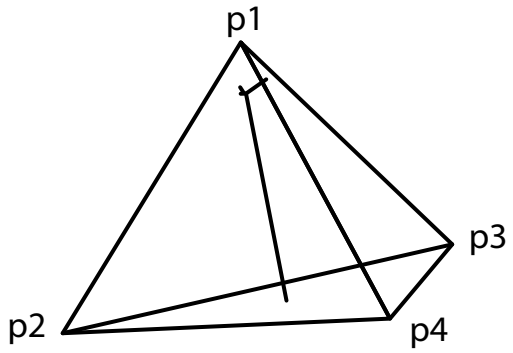
In an InfoViz context, a Barry-3 data representation is sometimes called a "triplot", but should not be confused with "trilinear graphs" [14, p. 423].

4.4 Barry-4 Axes in $\mathbb{R}^3$

The question arises, if a 3-parameter barycentric sum rule is represented by an equilateral triangle (2-simplex), what is appropriate geometrical representation for a 4-parameter sum rule?



(a) Single point located by the Barry-4 coordinates $p_1 = 0.4$, $p_2 = 0.25$, $p_3 = 0.1$, $p_4 = 0.25$, which corresponds to an uneven distribution of weights at each vertex



(b) Single point located by the Barry-4 coordinates $p_1 = 0.8$, $p_2 = 0.1$, $p_3 = 0.05$, $p_4 = 0.05$, which corresponds to a different uneven distribution of weights at each vertex

Figure 12: Barycentric coordinates in $\mathbb{R}^3$

One might be tempted, at first, to conclude that the appropriate choice is a *square*. But, from the discussion about dimensional compression in Sect. 4.1, we can know that $(d+1)$ performance parameters can be represented by a d-dimensional simplex viz., a tetrahedron like that

in Fig. 12. In general, for $n = (d + 1)$ parameters:

$$\sum_{i}^{n} p_i = 1 \tag{5}$$

In principle, a total of $(4 + 1) \times 12 = 60$ performance parameters could be displayed on a 2-dimensional screen. Attempting to go beyond this level of dimensional compression would us back to a state of visual overload, which it has been our goal to avoid.

We now apply this conceptual framework to multiprocessor performance analysis.

5 APPLICATIONS

The first paper about the application of barycentric coordinates to PerfViz [11] presented an `ncurses` implementation of Barry-3 along with several screen shots. In this section, we apply the definitions developed in Sect. 4 to present a suite of new Barry visualizations (Barry-1, Barry-2, Barry-3 and Barry-4) implemented in Java and Mathematica, along with several new PerfViz applications for multiprocessors and multicores (Sect. 5.2), application response times (Sect. 5.3) and network monitoring (Sect. 5.4).

5.1 Barry-2 for Resource Monitoring

Barry-2 is equivalent to a line segment in $\mathbb{R}^1$; the trivial 1-simplex in Fig. 8. The corresponding sum rule is $\rho + (1 - \rho) = 1$ on $\mathbb{R}^1$. An alternative generalization for including higher-dimensional data in $\mathbb{R}^2$ is an array of line segments; also known as a table (cf. Fig.18). Arrays of colored line segments have also been used e.g., MASF [24].

5.2 Barry-3 for Multiprocessor Metrics

Computer processor state can be expressed as three parameters:

1. Idle-time (i): The percentage of time the processor spends either not executing any code or waiting for something else to happen e.g., the completion of an 1/0 request.
2. User-time (u): The percentage of processor time spent executing application code.
3. System-time (s): The percentage of processor time spent executing code in the UNIX kernel (other than Idle).

These three parameters obey a sum rule:

$$u + i + s = 1 \tag{6}$$

so, it becomes possible to represent their values in barycentric coordinates with axes denoted IDL, USR, SYS in Fig. 13. There is also an additional degree of freedom viz., real-time, which although not directly associated with these geometrical considerations, can be incorporated by animating the barycentric display as noted in Sect. 2.2.

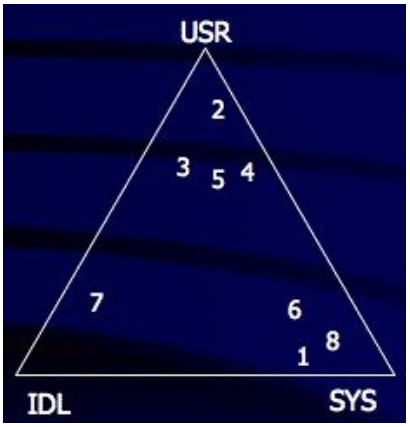


Figure 13: The *Barry1992* ASCII representation of CPU utilization for an 8-way multiprocessor based on a Barry-3 axes

Referring to the elementary trigonometry in Fig. 14, we have:

$$u_x = u/\sqrt{3} \quad \text{and} \quad s_x = 2s/\sqrt{3} \tag{7}$$

which are the ordinate-offset of the CPU user-rime and the abscissa-projection of CPU system-time, respectively. Compare this with the general barycentric equation (1).

An arbitrary point on the screen (x, y) is given by:

$$(u_x + s_x, u) \ . \tag{8}$$

Consequently, any algorithm for computing a screen position in barycentric coordinates only requires *two* of the three performance parameters: user-time, system-time, and idle-time. These two values, together with the appropriate scaling factors for the aspect ratio of the particular screen, provide all the necessary input.

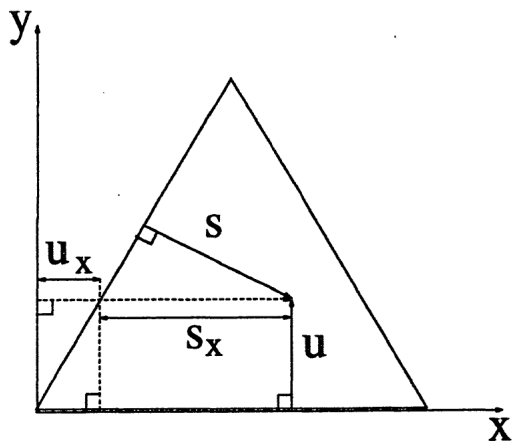


Figure 14: Mapping CPU-busy to Barry3 axes

All these performance attributes can be viewed simultaneously in 2D and more importantly, instantly *comprehended*. In addition to these three CPU performance metrics, the Barry-3 display can be *dynamic* in the sense of using animation to examine the temporal development of processor performance (e.g., transient effects). Time development can also be thought of as a sequence of triangular time-slices laid out like the segments of Toblerone chocolate bar.

5.3 Barry-3 for Application Response

Up to this point, we have considered only parametric performance data. In this section we show how barycentric coordinates can be applied to *categorical* performance data.

The Apdex Alliance (`www.apdex.org`) has defined an application performance index based on categorizing response time measurements from a user perspective (`www.cmg.org/conference/cmg2006/mkt/apdex.html`). The Apdex satisfaction metric (A_T) is a single number normalized so that $0 < A_T < 1$. This unit range can be further subdivided into a set of color-coded *Zones* defined in Table 1.

Table 1: Definition of Apdex Zones

Rating	Range
Excellent	$0.94 \leq A_T \leq 1.00$
Good	$0.85 \leq A_T < 0.94$
Fair	$0.70 \leq A_T < 0.85$
Poor	$0.50 \leq A_T < 0.70$

The usual limitations of expressing performance as a single notwithstanding, the Apdex metric also runs into the same scalability limitations as discussed in Sect. 3 for tabulated multiprocessor data in Fig. 18 (see last page). This limitation is likely to be manifest in most commercial data centers which typically support a large number of concurrent applications. We briefly indicate here how this scalability problem can be ameliorated by applying barycentric coordinates to the underpinnings of the Apdex index.

As part of the framework used to define the A_T index, the specification defines two threshold times $\mathcal{T}$ and $\mathcal{F} = 4\mathcal{T}$ to create three performance "zones" or categories:

Satisfied (S): Response times R that are short enough to satisfy the user by falling within the interval $R \in [0 \ \ldots \ \mathcal{T}]$.

Tolerating (T): Responses that fall in the interval $R \in (\mathcal{T} \ \ldots \ \mathcal{F}]$. Application responses in this zone are less than ideal but do not by themselves threaten the usability of the application.

Frustrated (F): Response times longer than the second threshold $R > \mathcal{F}$ such that a casual user is likely to abandon a course of action and a production user is likely to cancel a task.

The threshold $\mathcal{F}$ has been selected by the Apdex consortium on the basis that users often *perceive* response times to be far greater than the *measured* value. The Apdex specification gives the following example:

> *If users perceive response time as tolerable beginning at 4 seconds then they will be frustrated at greater than 16 seconds.*

This criterion should be compared with the well-documented device-independent limits (`www.useit.com/papers/responsetime.html`):

$\mathcal{T} \leq 0.1$ **seconds:** User feels that the system is reacting instantaneously, and no special feedback is necessary.

$\mathcal{T} \leq 1.0$ **second:** User's flow of thought is maintained, even though delay will be noticed.

$\mathcal{T} \leq 10$ **seconds:** User remains focused on the interactivity. Feedback during delays is especially important.

Figure 15: Apdex Zones (Table 1) superimposed onto Barry-3 coordinates. Apdex indexes for web applications measured at 5 different geographical locations are shown as black dots. The main cluster of points at the top-left each have high satisfaction values and near zero frustration components. They therefore reside in the *Good* to *Excellent* zones. Conversely, the outlier (center) with Apdex index $A_4 = 0.58$ and frustration component $F = 0.27$ resides in the *Poor* zone

The Apdex specification requires at least 100 samples to generate reasonable response time statistics. Denoting the sample counts for the Apdex intervals as C_s, C_t and C_f, we can normalize them as follows:

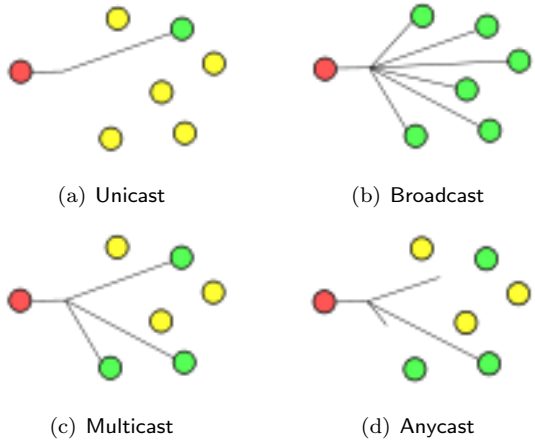
$$\begin{aligned} C &= C_s + C_t + C_f \\ 1 &= \frac{C_s}{C} + \frac{C_t}{C} + \frac{C_f}{C} \\ 1 &= s + t + f \end{aligned} \tag{9}$$

such that each of the ratios s, t, f are related by a sum rule identical to (6).

These normalized response time data can be rendered using the *Barry3* diagram in Fig. 15. This is similar to Fig. 13, but with the u, s, i axes relabeled by S, T, F. Not only does Fig. 15 provide a way to address the scaling problem of viewing possibly hundreds of applications simultaneously, but a moving cloud of such points can also show their time development. In other words, Barry-3 for Apdex offers a visual representation of application performance that makes it useful to both executive management (who want to know just one number, A_T) and to performance engineers (who want to know how A_T is comprised).

5.4 Barry-4 for Network Utilization

In the case of barry-4, any parametric performance variables where there is a sum rule relationship with four independent values is a great application for this 3D version of the barry visualization suite. This specific instance of the barry visualization arguably has the best impedance match due to its familiar relationship to the $\mathbb{R}^3$ space we live in.



(a) Unicast (b) Broadcast

(c) Multicast (d) Anycast

Figure 16: LAN segment addressing modes

A great application for this visualization is the different types of addressing when considering segment utilizations in an IEEE 802.3x or Ethernet LAN segment. The most common addressing modes in such environments are:

Unicast: Traffic in which the destination IP address refers to one specific host only. (Fig. 16(a))

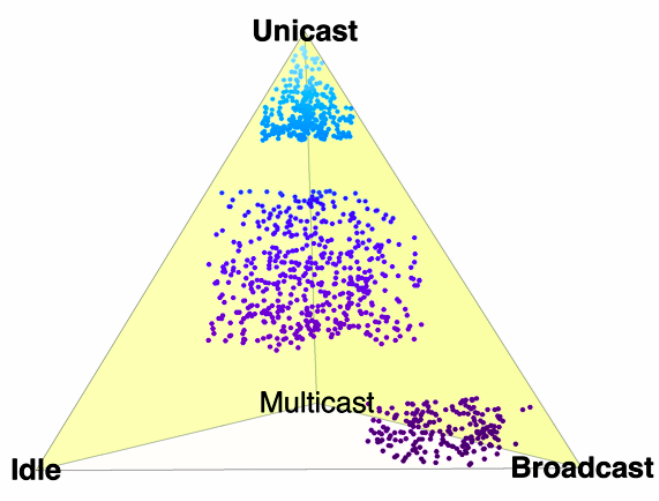
Broadcast: Traffic in which the destination IP address refers to a group of hosts in a network (network broadcast) or subnet (subnet broadcast). All hosts in the topology receive the traffic whether they want it or need it or not. (Fig. 16(b))

Multicast: Traffic in which the destination IP address refers to a group of hosts that have indicated an interest for the traffic. This could mean all hosts on a network or none. (Fig. 16(c))

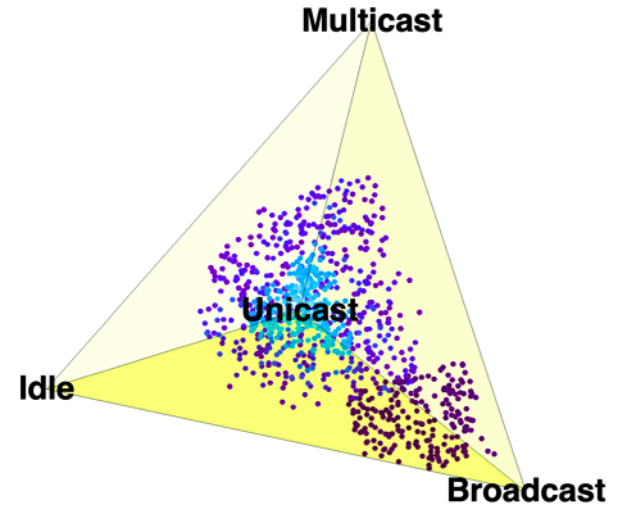
Idle: The LAN segment is not transmitting any of the above types of traffic.

Anycast (Fig. 16(d)) is a new type of addressing, with rather limited applications, that has emerged and is not usually of concern for the performance analyst. It is for

traffic in which the destination IP address refers to a group of hosts but only the nearest host will receive the data. The definition of nearest depends on the network topology, the protocols used and the associated administrative policies. (See RFC 3330 `www.rfc-editor.org/rfc/rfc3330.txt`). Such traffic is transmitted using unicast addresses that are under the control of the administrative group which has operational responsibilities. Although we mention it for completeness, it does not appear as an independent axis in Fig. 17.



(a) Default view looking "up" the *Multicast* axis



(b) Swiveled view looking "down" the *Unicast* axis

Figure 17: Two different Barry-4 views of 1,000 LAN segments displaying network utilization for the *unicast, multicast, broadcast* and *idle* modes defined in Fig. 16

Since the most common type of traffic is *Unicast*, we have chosen to orient the *default* view so that metric appears at the apex of the tetrahedron in Fig. 17; keeping in mind that the Barry-4 tetrahedron can be swiveled to any arbitrary orientation on the screen. Figure 17 depicts a snapshot of network utilization data across 1,000 LAN segments. As far as we are aware, there is nowhere else can you display this amount of data in such a compact form and *still make sense of it*; especially in currently available commercial tools.

Fig. 17(a) clearly shows three distinct "clouds" of performance data. The dark purple cloud (*lower right*) corresponds to a significant number of LAN segments experiencing extremely heavy levels of broadcast traffic which are consuming 60–85% of network bandwidth. This corresponds to a familiar horror story, affectionately referred to as a "broadcast storm", which used to occur frequently in the late 1980s or early 1990s but is less likely today because of better segmentation of networks and subnetworks. It is also clear that a lot of segments that are experiencing extremely high level (85–100%) of unicast traffic (*top apex*). For example, these segments could be involved with heavy video on-demand or file sharing activities. Fig. 17(b) corresponds to Fig. 17(a) rotated ("spun") by about 90 degrees so the the *Unicast* axis is now pointing out of the page towards the person viewing. This view enables the analyst to appreciate that the middle cloud (around 50% Unicast) is evenly distributed along the *Idle*, *Broadcast* and *Multicast* axes.

6 VISUAL ENHANCEMENTS

If you walk along the beach at Coronado Bay in San Diego, you kick up grains of sand as you go. Each grain can be specified by 3-location numbers (x, y, z) and 3-velocity numbers (v_x, v_y, v_z). The flying grain has 6 degrees of freedom in $\mathbb{R}^3$. The time dimension is implicit. If, instead of flying sand, you throw a beach ball, it rotates about it's own axis as it travels. Rotation or corkscrewing of the stripes on the beach-ball requires another 6 independent numbers. These 12 degrees of freedom can also be regarded as 12 independent dimensions in the statistical sense. We could also include other categorical degrees of freedom such as: color of the stripes, plastic or rubber material, etc. As noted in Sect. 2.1, since you (the reader) can easily visualize this flying beach-ball in color, humans can effectively visualize in more than 3-dimensions [9]. The problem for PerfViz is, how best to encode these higher dimensions.

Some proposed enhancements for the Barry007 suite include:

Swivel and Rotation: When viewing 3D tetrahedron on a 2D screen, a lot of information can be hidden or misinterpreted. Having the ability to swivel or spin the tetrahedron is a great help in gaining a deeper understanding of the data being viewed.

Animation: The ability to view points or dots moving in real time is very useful to gain an understanding of trends, transients or the periodicity of the data under observation. This has already been alluded to earlier in the paper.

Tooltips: Sometimes when there is a cloud of points in a Barry007 visualization, you would like to know what a given point in the cloud represents. For example, if you are looking at the CPU utilization for 1000 CPUs (whether separate distinct processors, cores on a system or separate servers) and you see an outlier, it would be useful to be able to move the mouse over that point and have a tooltip appear (a small window or bubble enclosing text) which states the name or number of the processor along with some identifying configuration data, e.g., location, purpose (server, desktop, mail server, DB-server etc.). Another variant could be that once the dot keeps on moving the tooltip follows the dot for a selected amount of time, e.g., 10 seconds or a minute or until another point is moused over.

Categorical highlighting: If you are looking at network utilization for LAN segments, one might have all the half-duplex segments highlighted in blue, whereas the full-duplex segments might be maroon. Similarly, one could have a color coding scheme for the speed of each segments, e.g., 10 Mbs, 100 Mbs or 1 Gbs. Again, one could have all head office segments with one color and the Ottawa branch office highlighted a different color. Another possible way of highlighting is using different symbols instead of the dots, e.g., you could have a symbol for file servers, desktops, DB servers etc. You could combine this with color highlighting for different granular effects, i.e., all DB servers at head office vs. all DB servers at a branch office.

Parametric highlighting: It is possible to have for example on a CPU Barry007 visualization all points close to the 100% idle utilization level highlighted as blue, all points close to the 100% user utilization level and all points close to the 100% system utilization as red. Also, the intensity of the color highlighting should be proportional to the proximity to the 100% level. The closer the point is to the 100% system utilization the brighter red it is.

Sound effects: Another possible enhancement to Barry007 is the use of sound effects to highlight certain conditions. For example, if a broadcast storm is an undesirable occurrence, a sound effect similar to the Geiger radioactive counter detector could be used where the intensity of the sound could be related to the seriousness of the broadcast storm.

7 CONCLUSION

Relatively little progress seems to have been made in developing new PerfViz paradigms since *Barry1992* [11]. Most commercial performance management tools still rely on 20 year-old GUI technology to display monitored data in relatively simple graphical styles.

According to statistician Michael Friendly's "Best and Worst of Statistical Graphics" (`www.math.yorku.ca/SCS/Gallery/milestone/sec9.html#`), more emphasis has been given to InfoViz applications such as: "Table Lens" (c.1994), cartographic data visualization (e.g., `http://earth.google.com/`) and "Sparklines" (`en.wikipedia.org/wiki/Sparkline`). Along with *Table Lens*, Stuart Card and colleagues at Xerox PARC also developed "Cone Trees" and the "Hyperbolic Browser" (see `http://www.ramanarao.com/papers/rao-infoviz-nextgen-workspace-1995.pdf`). Arguably, the most recent developments have occurred in 3D web visualizations, such as explaining the secret behind the construction of the Great Pyramid (`khufu.3ds.com/introduction/`) and dynamic Flash animations of blog statistics (`labs.digg.com/swarm/`).

In the PerfViz arena, a stalemate seems to exist between performance-tool vendors who do not want to invest in engineering development where they see no demand, and end users who are not aware of what they are missing in terms of alternative visualization techniques for solving performance management problems. Hence, an important motivation for writing this paper is to provide a catalyst for an ongoing dialog between CMG performance analysts, managers, tool vendors, statisticians, psychologists, and other experts, to formulate better visualization paradigms for the performance management tools of the future. The goal is not develop tools that merely look "cool" or present a novel "Gee-whiz!" factor (although that might not hurt sales), rather it is to find new paradigms that provide an optimal visual impedance match whereby performance management becomes more efficient in today's increasingly complex computer environments.

Like the categorical data in Sect. 5.3, we are looking at

methods to examine high-dimensional multivariate data by selecting any quartet of attributes for Barry-4 display. As with all visual representations, barycentric coordinates have strengths and weaknesses, and although we do not claim that one-size fits all, the *Barry007* suite of visualizations does rank well in Table 2. The screens shots and visual examples shown in this paper are not meant to imply the existence of any product with those capabilities. On the contrary, since no products exist today with these capabilities, all screen shots are necessarily from prototypes that give a hint of how a commercial tool might appear.

8 ACKNOWLEDGMENTS

One of us (NJG) wishes to thank James "Scott" Johnson for bringing the Apdex metric to our attention, as well as Peg McMahon and Justin Martin for making an early draft of their paper available. We are both grateful to Peter Sevcik of the Apdex Alliance for providing us with the data used in Sect. 5.3.

References

[1] M. Gasko A. W. Donoho, D. L. Donoho. "MacSpin: Dynamic graphics on a desktop computer". *IEEE Computer Graphics and Applications*, 8(4):51–58, July/August 1988.

[2] J. W. Tukey. Graphics 1965–1985. In W. S. Cleveland, editor, *The Collected Works of John W. Tukey*, volume V. Wadsworth & Brooks/Cole, Pacific Grove, California, 1988.

[3] J. A. Brown, A. J. McGregor, and H-W. Braun. "Network performance visualization: Insight through animation". In *PAM 2000: Passive and Active Network Measurement Workshop*, Hamilton, New Zealand, April 3–4 2000.

[4] "HPVC: High performance visualization center". `www.hpvc.chl.state.ms.us/about.htm`, June 2005.

[5] "Visualization for parallel performance evaluation and optimization". `www.cc.gatech.edu/computing/classes/cs7390_98_winter/reports/parallel/chap23.html`, March 1998.

[6] "ParaGraph: A performance visualization tool for MPI". `www.csar.uiuc.edu/software/paragraph`, August 2003.

[7] F. Werblin and B. Roska. "The movies in our eyes". *Scientific American*, pages 73–79, April 2007.

[8] W. Hendee and P. Wells (Eds.). *The Perception of Visual Information*. Springer-Verlag, 1993.

[9] R. Olcott. "The dimensions of service—Exploring WLM's solution space". In *Proc. CMG Conf.*, pages 15–82, Las Vegas, NV, 2002.

[10] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. Wiley, New York, 1978.

[11] N. J. Gunther. "On the application of barycentric coordinates to the prompt and visually efficient display of multiprocessor performance data". In R. Pooley and J. Hillston, editors, *Proceedings of Sixth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, volume Edinburgh, Scotland, pages 67–80. Antony Rowe Ltd., Wiltshire, U.K., September 1992.

[12] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[13] W. Cleveland. *The Elements of Graphing Data*. Wadsworth, 1985.

[14] R. Harris. *Information Graphics: A Comprehensive Illustrated Reference*. Oxford. Univ. Press, 1999.

[15] P. Keller and M. Keller. *Visual Cues: Practical Data Visualization*. IEEE Press, 1993.

[16] H. Chernoff. "The use of faces to represent points in k-dimensional space". *J. Amer. Stat. Assoc.*, 68:361–368, June 1973.

[17] K. Kolence and P. Kiviat. "Software unit profiles and Kiviat figures". *Performance Evaluation Review*, 2(3):2–12, September 1973.

[18] A. Inselberg. "The plane with parallel coordinates". *The Visual Computer*, 1:69–91, 1985.

[19] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, 4th edition, 2002.

[20] L. Merritt. "Seeing the forest and the trees: Capacity planning for a large number of servers". In *Proc. CMG Conf.*, Las Vegas, NV, 2004.

[21] P. McMahon and J. A. Martin. "Death to dashboards, and other thoughts on data visualization: Alarming, forecasting and performance management based on variance". In *Proc. CMG Conf.*, San Diego, CA, 2007.

[22] L. Kleinrock. *Queueing Systems. Volume I: Theory*. John Wiley, NYC, New York, 1976.

[23] Particle Physics Data Group. *Particle Physics Booklet*. Lawrence Berkeley Laboratories, Berkeley, CA, 2006.

[24] J. P. Buzen and A. W. Shum. "MASF—Multivariate adaptive statistical filtering". In *Proc. CMG Conf.*, pages 1–10, Nashville, TN, 1995.

Table 2: Comparison of performance monitor design attribute

Attribute	Barry007	Kiviat	mpstat	Chernoff
Localized	×	×	×	×
Animation	×	×	×	?
Multi-p	×	-	×	×
MP	×	-	×	-
Clustering	×	-	-	-
Patterns	×	-	-	-
Cognition	×	×	-	-
Semantics	×	×	-	-
Generality	×	-	-	-
Journaling	×	-	×	?

CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
0	2	0	4191	7150	6955	1392	93	374	573	14	1433	78	22	0	0
1	2	0	179	11081	10956	1180	132	302	1092	13	1043	79	21	0	0
2	1	0	159	9524	9388	1085	141	261	1249	14	897	79	21	0	0
3	0	0	3710	10540	10466	621	231	116	1753	2	215	70	29	0	0
4	5	0	28	355	1	2485	284	456	447	30	2263	77	23	0	0
5	5	0	25	350	1	2541	280	534	445	26	2315	78	22	0	0
6	3	0	26	331	0	2501	267	545	450	28	2319	78	22	0	0
7	2	0	30	292	1	2390	232	534	475	23	2244	77	22	0	0
8	4	0	22	265	1	2188	220	499	429	26	2118	75	25	0	0
9	2	0	28	319	1	2348	258	513	440	26	2161	76	24	0	0
10	4	0	23	308	0	2384	259	514	430	22	2220	76	24	0	0
11	4	0	27	292	0	2366	237	518	438	30	2209	77	23	0	0
12	11	0	31	314	0	2446	253	530	458	27	2290	78	22	0	0
13	4	0	31	273	1	2334	223	523	428	25	2261	79	21	0	0
14	12	0	29	298	1	2405	247	521	435	25	2286	78	22	0	0
15	4	0	32	330	1	2445	272	526	450	24	2248	77	22	0	0
16	5	0	28	271	0	2311	219	528	406	29	2188	76	23	0	0
17	4	0	23	309	1	2387	253	537	442	25	2234	78	22	0	0
18	3	0	25	312	1	2412	257	534	449	26	2216	78	22	0	0
19	3	0	29	321	1	2479	262	545	462	31	2287	78	22	0	0
20	14	0	29	347	0	2474	289	541	457	24	2253	78	22	0	0
21	4	0	29	315	1	2406	259	534	469	24	2240	77	22	0	0
22	4	0	27	290	1	2406	243	531	480	25	2258	77	22	0	0
23	4	0	27	286	1	2344	235	531	445	26	2240	77	22	0	0
24	3	0	30	279	0	2292	228	518	442	22	2160	77	23	0	0
25	3	0	26	275	1	2340	227	538	448	25	2224	76	23	0	0
26	4	0	22	294	1	2349	247	529	479	26	2197	77	23	0	0
27	4	0	27	324	1	2459	270	544	476	25	2256	77	23	0	0
28	4	0	25	300	1	2426	249	549	461	27	2253	77	23	0	0
29	5	0	27	323	1	2463	269	541	447	23	2277	77	22	0	0
30	2	0	27	289	1	2386	239	535	463	26	2222	77	23	0	0
31	3	0	29	363	1	2528	304	525	446	26	2251	76	23	0	0

Figure 18: One sample of *mpstat* output on a 32-way multiprocessor

TRADEMARKS