On the Application of Barycentric Coordinates to the Prompt and Visually Efficient Display of Multiprocessor Performance Data*

N. J. Gunther

Systems Technology Laboratory, Pyramid Technology Corporation,
3860 North First Street, San Jose, California, 95134, U.S.A.

Abstract

Processor utilization is the leading indicator of computer system performance. Monitoring the utilization of large multiprocessor (MP) systems (with $N = 10$ to 30 processors) rapidly becomes tiresome using conventional performance tools; particularly those where the interface amounts to scrolling a torrent of ASCII data across a terminal screen. Histograms and similar visual presentations are also cumbersome in an MP setting since the occupied screen real-estate grows with N. We introduce the notion of barycentric (center of mass) coordinates and show how it can be applied as a performance monitor to the efficient, dynamic, display of MP statistics. Screen occupancy is small and independent of N. The greatest benefit appears to be the emergence of simple, visual "signatures" which correlate with the workload to a degree where the performance analyst's peripheral attention is mostly drawn to significant *changes* in performance. Although reading these signatures is not intuitive, learning ss trivial once the semantics is explained. Some example barycentric signatures are presented.

1 INTRODUCTION

The results presented in this paper arise out of exploring a rather remote tributary of mainstream research in the area of visual pecfastmance tools or more colloquially, performance visualization for computers (PVC)—the whole subject being deeper and broader than any of us currently understands. This paper is concerned more with uncovering design principles (in so far as they can be identified) than introducing yet another PVC tool. To this end, we will not rely on window-bed displays, graphical workstations, novel pointing devices or any other of the paraphenalia [2] usually associated with PVC [3, 4].

Instead, we consider the design for a performance monitor with foundations firmly rooted in geometry. Out of this geometrical design emerge certain visual properties which, based on our experience, appear useful for PVC. 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 PVC 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.

*Reproduction of [1] by the author using an OCR scan made by M. Jauvin on April 7, 2007

2 TOOLS AND TECHNIQUES

2.1 Monitoring Tools

Performance monitors for commercial maintiame computer systems have generally tended to be terminal-based with ASCII numerical output. An example is shown in Fig. 1. This has been particularly true of standard UNIX systems [5].

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 1: One cycle of *mpstat* output for an $N = 32$ multiprocessor

Since the advent of bit mapped displays [6] and window-style user interfaces, however, there has existed the opportunity for software emulation of various types of devices for analyzing real-time data [7]. These devices are also capable of representing aspects of computer system state itself viz., performance indicators for such things as file fragmentation, processor loading, and paging rates [8]. This capability has resided mostly in research software environments and has only recently made its way into commercial software environments.

As the need has increased for better performance tuning of computer systems which perform complex tasks such as, supercomputing [9], and transaction processing [10], so has the need for merging performance tools with window systems. In its full embodiment, this has become the subject of PVC. Typically, such tools provide a more visual display of performance information. Common examples are: time-series plots, tachometer or dial indicators, Gantt charts, contour plots, strip recorders, and histograms.

The logical extension of this approach is to inevitably open so many windows as to consume significant portions of screen real-estate [7, 9, 11] and, worse yet, reach the point where the performance analyst is potentially suffering from visual overload!

2.2 Postmortem Tools

Other visual performance tools which are not usually integrated into any particular operating system or software environment include: Kiviat graphs [12–14] and Chernoff faces [15]. Typically, these tools provide static, post-run, data presentation by displaying some arbitrary number of performance parameters to produce equally arbitrary shape or pattern definitions, e.g., "keelboars", "arrows" and "smiles". A more recent approach to the static presentation of multidimensional performance data is the "multigraf". This tool provides a visualization of clustering within the context of multivariate statistical analysis (MANOVA) of performance-related variables [16].

There is a common drawback for most of these tools, however. The semantics of the patterns can differ dramatically across different systems or even on the same system under different workloads. From our point of view, the benefit of these postmortem tools lies in the way they enbrace *pattern recognition*, which in turn draws directly on the human brain's superior architecture for visual analysis. Compared with conventional tools, the patterns formed by these alternative representations offer a better match between the states of the *digital* computer being analyzed and the states of the *cognitive* computer of the analyst. Unfortunately, dynamic pattern recognition seems to have been adopted slowly at best in the context of PVC tools. The inertia is probably both technical and cultural; technical in that performance data (as will be revealed in the subsequent discussion) is difficult to represent as meaningful patterns; and cultural in that established data-presentation techniques are not available in conventional software environments.

We also note in passing, that PVC gets much of its current impetus from the emerging technology of "scientific visualization". It is our opinion, however, that there is a fundamental difference between these two activities. Scientific visualization techniques rely heavily on the implicit 3-dimensional structure of typical data, e.g., seismic waves. Computer performane data, on the other band. usually belongs to an undefined, multi-dimensional hyperspace [17]. Moreover, the subtleties associated with rendering single-processor performance are further exacerbated in the case of MP architectures. In fact, monitoring performance on an MP platform forces one to think anew about performance tools. In this paper, we focus on monitoring MP performance in general and processor utilization in particular.

2.3 Prompt Patterns

In the subsequent discussion, we will attempt to integrate pattern recognition with dynamic monitoring of performance data. We will confine ourselves to UNIX systems althomgh, the PVC tool we shall be describing cold be implemented in any operating system that allows non-invasive sampling of the appropriate processor statistics.

Of the PVC tools reviewed so far, our approach has little in common with any of them save perhaps for Kiviat graphs. The Kiviat graph is multidimensional and can be dynamic [9]. Visual cueing comes from the appearance of loosely defined symmetries and arbitrary geometrical (polygonal) shapes. Unlike the arbitrariness of the Kiviat representation, however, our approach produces consistent dynamic, visual "signatures" due to the choice of a highly constrained geometry for the multidimensional representation of processor performance data.

3 BARYCENTRIC REPRESENTATION

3.1 Dimensional Compression

If two parameters, p_1 and p_2, represent two *independent* degrees of freedom, the appropriate geometrical represention is a 2-dimensional coordinate yaem (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

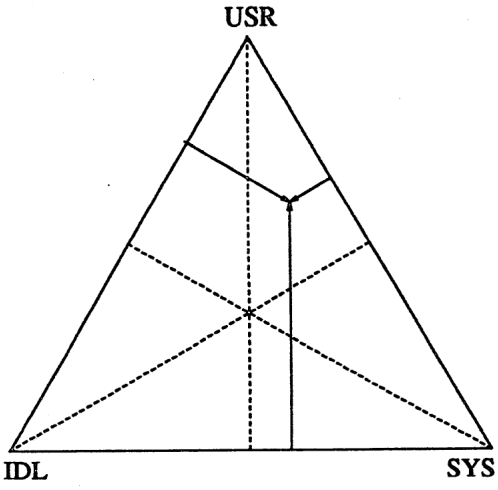

Figure 2: Barycentric coordinates for the centroid and an arbitrary point

these parameters are restricted by the *sum rule*:

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

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

Similarly, the locus of a point in three dimcasiotls, having only two degrees of freedom, can be shown (Fig. 2) to be bounded by an equilateral triangle[1]. 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:

$$p_1 + p_2 + p_3 = 1 \tag{2}$$

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 general, for n parameters:

$$\sum_{i}^{n} p_i = 1 \tag{3}$$

We now apply this conceptual framework to MP performance analysis.

3.2 Parametrizing Multiprocessor State

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.

[1]The marketing potential of the barycentric coordinate system invoking the image of a pyramid has not been entirely lost on the author. He, however, regards it as both coincidental and inconsequential.

3. System-time (s): The percentage of processor time spent executing code in the UNIX kernel (other than Idle).

These three parameters do obey a sum rule so, it becomes possible to represent the values of these performance parameters in the barycentric coordinate system (IDL, USR, SYS in Fig. 3). There is also an additional degree of freedom viz. real-time, which is not directly associated with these geometrical considerations.

Our approach provides for a visual display of up to 12 processors[2]. They are currently labeled: $0, 1, .., B$ in hexadecimal, so that each label only occupies one ASCII character on the screen. In addition in these three performance parameters, the display is dynamic and provides the capability of examining the temporal development of processor performance (e.g., the way in which steady-state transaction processing is achieved [18]). Hereafter, we refer to this as a $3 + 1$-dimensional representation.

In other words, at least $(3 + 1) \times 12 = 48$ performance parameters can be viewed simultaneously in 2-dimensions and more significantly, instantly *comprehended*. In the next section, we shall indicate how these dimensions can be compressed even further.

3.3 Coordinate Transformations

Referring to Fig. 3, and setting the constant $\rho = 1/\sqrt{3}$, we bave from elementary trigonometry:

$$u_x = \rho u \quad \text{and} \quad s_x = 2\rho s \tag{4}$$

These being respectively, the ordinate offset of the user-rime and the abscissa projection of system-time. Now, it can be seen that any location (x, y) on the terminal sceen is given by:

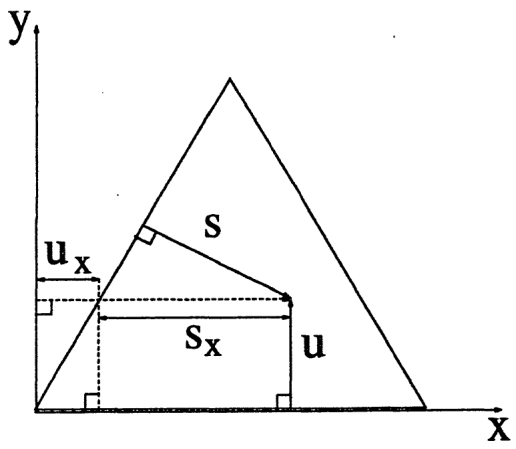

Figure 3: Transformation of barycentric to screen coordinates

$$(u_x + s_x, u)\ . \tag{5}$$

Consequently, any algorithm for computing a screen position in barycentric coordinates only requies *two* of the three performance parameters: user-time, system-time, and idle-time. The two values, together with the appropriate scaling factors (in character units) for

[2]The current generation of MIServer hardware supports 12 processors. The next generation will support up to 24 processors (2 per CPU board) and will therefore require a modification to the current labelling scheme. One possibility would be to drop the hexadecimal notation and label the "upper" processors (on the vertically mounted board) alphabetically A–L and the "lower" processors with lower case letters, a–l.

the aspect ratio of the particular terminal, provide all the necessary detail. The choice of barycentric coordinates in the previous section was motivated by the 3-parameter sum rule:

$$u + i + s = 100\% \tag{6}$$

Idle-time, however, can be decomposed further into the sum of percentage cpu-idle and cpu-waiting for I/O to complete (See Fig. 1).

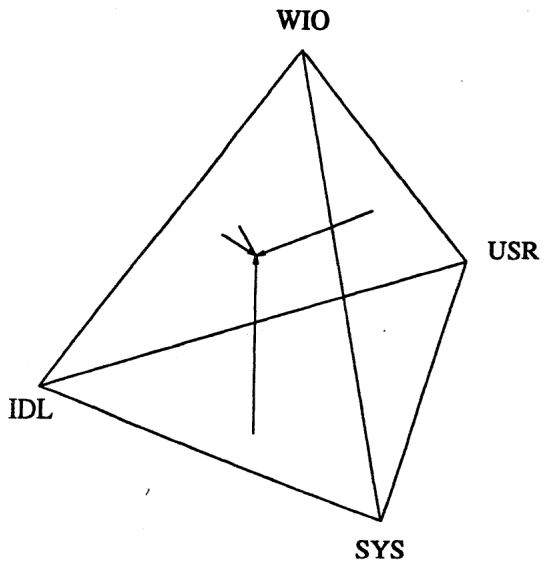

Figure 4: Tetrahedral barycentric coordinates for $p = 4$

As an example, the format which the DataCenter/OSx performance tool, *mpstat* uses for reporting processor statistics is shown in Table 1. This performance data can be repar-

Table 1: Format of *mpstat* output

id%	**us%**	**sy%**	**io%**
78	10	12	38

titioned as shown in Table 2. Therefore, processor utilization can actually be expressed as

Table 2: Repartitioned processor performance indices

cpu-idl%	**cpu-wio%**	**usr%**	**sys%**
40	38	10	12

a 4-parameter sum rule. The question then arises, if a 3-parameter sum rule is represented by an equilateral triangle, what is appropriate geometrical representation for a 4-parameter sum rule?

One might be tempted. at first, to conclude that the appropziate choice is a square but, from our earlier discussion of dimensional compression we can state that, in general, far n performance parameters we require d spatial dimensions to uniquely represent the state where d is given by.

$$d = n - 1 \tag{7}$$

So, from purely geometrical considerations a 4-parameter barycentric representation (e.g., usr, sys, idl, wio) would require a 3-dimensional coordinate system viz., a tetrahedron shown is Fig. 4. One can side-step the complexity of rendering such a 3-dimensional image on the screen by resorting to the obvious trick of looking "down" the WIO axis and "color" encoding it, but we will not pursue that topic here. 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.

4 UNIX IMPLEMENTATION

We now discuss the UNIX implementation of these fundamental concepts in a PVC tool affectionately christened *barry*[3]. The variant of UNIX used for the innplementaton is System V Release 4 and the discussion is presented in the context of a standard terminal screen[4] or *xterm* window [19] as shown in Fig. 5.

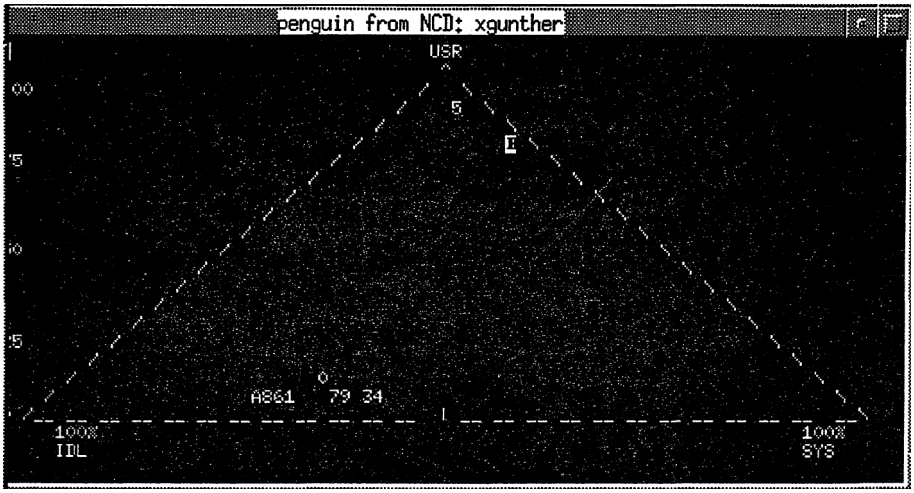

Figure 5: The appearance of *barry* in an *xterm* window

The program *barry* has a lightweight structure of about 500 lines on non-invasive user code that makes calls to kernel data structures. It has no more system-call overhead than the existing *mpstat* tool. Like the Berkeley tool *systat* it uses the *curses* package for updating the terminal display. The default update rate is once per second which is the finest time resolution permitted by the kernel code that collects processor statistics.

The entire package was designed and implemented in a matter of weeks by the author. It has been used for the analysis of several important benchmarks. In particular, it was employed for database and system tuning in the course of achieving a record-setting UNIX TPC-B transaction processing benchmark results [18, 20, 21]. The "desktop" appearance of barry running in a multiple xterm window environment is shown in Fig. 6. It should be noted that the window-based display is depicted here merely because it provides a convenient medium in which to capture the bitmap image of Fig. 6.

[3]The spelling was chosen primarily for friendliness and only secondarily for its mnemonic value.

[4]To be useful to Pyramid Sales Engineers in the field, where window-based displays are not yet commonplace, the author made the conscious design decision to first explore a simple terminal-based implementation.

5 DYNAMIC CLUSTERING

In Fig. 5 we see that only 11 processors are displayed (processor 2 is missing) but the MP system is known to have 12 processors instaleld. This effect arises as a consequence of the limited character positions on the terminal display resulting in occassional overlap of some processor points. We also see some clustering of processors executing multi-user applications. The interpretation is that pracessors 5 and B are better than 75% utilized with some system time and very little idle time. The remaining processors are mostly executing kernell code with a high percentage of idle time (presumably due to WIO).

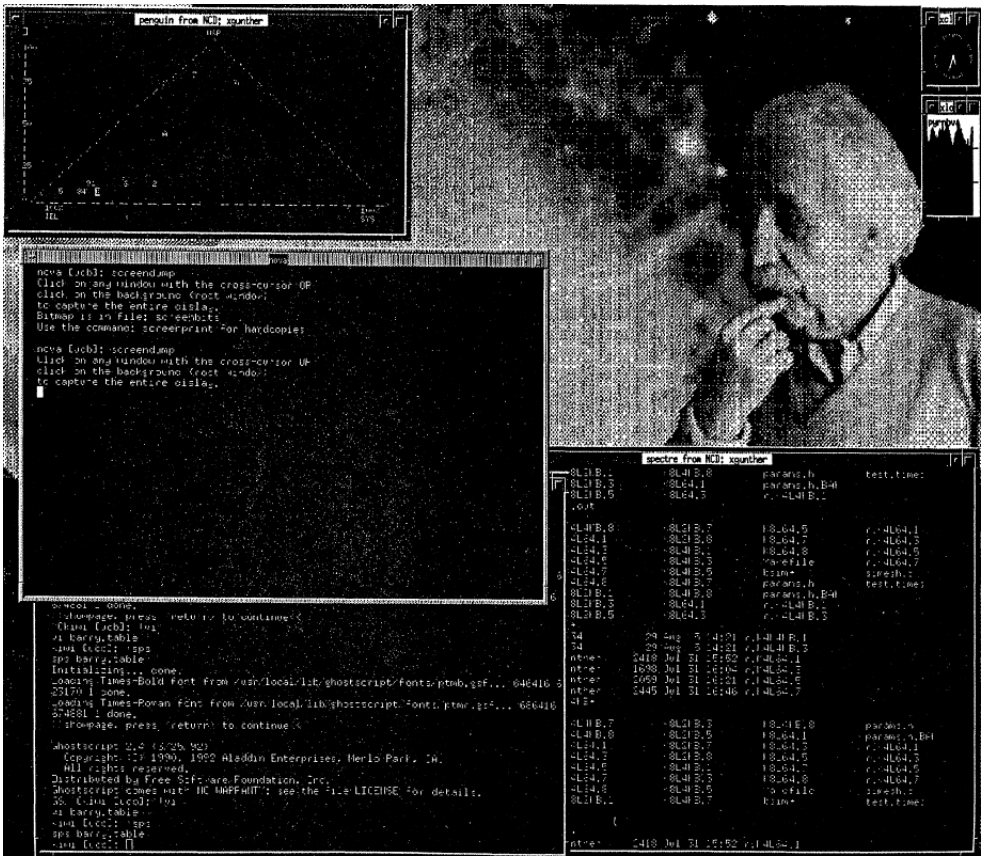

Figure 6: Desktop appearance of *barry* in the upper left corner of the screen

Empirically, one can detect the following important clustering properties:

1. Locadon of the cluster in the triangle
2. Intra-cluster distance or point dispersion
3. Inter-cluster distance and the presence of outliers
4. Distinct patterns or signatures

Each of these properties can be comprehended in a visually efficient way in this barycentric data representation.

Certain aggregations or clusters are often approximately repeated (as with hand-written signatures) and therefore take on the significance of *symbols* in their own right. Transaction

processing systems, for example, will often cycle through identifiable clusters according to whether the database management system is accessing data or performing a checkpoint. In that case, the analyst may tend to concentrate more on the *periodicity* of these cycling clusters as the important diagnostic.

6 CONCLUSION

PVC is a gray area in which colorful statements adhere all too easily. Certain qualitative features of *barry*, however, stand out enough to warrant some speculative remarks. In some respects the efficient visual semantics of *barry* seem akin to those of clock-hands. A standard clock-face is easier to read than one with discrete ("digital") numbers because the relative position of the hands forms an image that is, by virtue of edge-detection probably being an hereditary strength of the human visual system, more convenient for our visual system to process (compute?).

In particular, the visual efficiency attributable to *barry* may be due to the point-like patterns and their compact motion coupling strongly to the visual pre-processing that is known to be performed *within* the human retina [22]. This conclusion is supported by the observation that most of the time only a cursory awareness of the patterns in *barry* is necessary. Therefore, the performance analyst's attention is drawn only to those visual signatures that appear discrepant with expectations. This is to be contrasted with most conventional tools which require constant attention, laborious deciphering and interpretation, i.e., huge amounts of cognitive processing.

Interestingly, experience with *barry* suggests that this visual semantics is quite robust to distortions of the equilateral triangle due to incommensurate aspect ratios in terminals. Such distortions, of course, are only acceptable when one is not concerned with quantifying the location of the processor points.

6.1 Design Attributes

At this point it is worth reviewing the degree to which we have met our stated goal of uncovering "good" design principles for PVC tools. Table 3 compares a number of desirable design attributes across the selection of performance tools discussed in this paper.

A bullet indicates the tool possesses that attribute. A question mark (?) indicates uncertainty or ambiguity about the presence of that attribute. Included as desirable attributes are: the display of the performance tool should be localized on the screen as discussed in Sect. 2.1; dynamic display to view temporal development Sect. 2.2; universally understood semantics of the data representation; the ability to apply a data representation to arbitrary performance parameters; capability to record and redisplay performance data collected over a defined measurement period.

On this basis *barry* exhibits the major weakness that it is not generally applicable to a wide variety of performance data. This is a reflection of the choice of geometrical constraint expressed in the sum rule. Moreover, we see that the Kiviat graph comes in second as a tool which qualifies in terms of these particular attributes. This is consistent with our earlier qualitative expectations discussed in Sect. 2.3.

6.2 Future Work

Finally, we remark on short-comings observed while using barry and suggest ways in which these might be addressed in future work.

Barycentric location of individual processors often overlap: This effect can lead to ambiguous signatures. As example, consider the case where one processor is iocated at the USR apex and another at the IDL vertex. If there are more than two processors in the system then it remains unknown how the other processors are being

Table 3: Comparison of performance monitor design attribute

Attribute	barry	multigraf	mpstat	xload	xtacho	kiviat	chernoff
Localized	☺	-	☺	☺	☺	☺	☺
Dynamic	☺	☺	☺	☺	☺	☺	?
Multi-p	☺	-	☺	-	-	☺	☺
MP	☺	☺	☺	-	-	-	-
Clustering	☺	☺	-	-	-	☺	-
Patterns	☺	-	-	-	-	☺	-
Cognition	☺	-	-	☺	☺	-	-
Semantics	☺	-	-	☺	☺	☺	-
Generality	☺	☺	-	-	-	-	-
Journaling	☺	-	☺	-	-	?	?

instantaneously utilized. Fortunately, such a signature is unlikely to be very persistent and the immediate context may be enough to disambiguate events. Additionally, bit-mapped displays and windows offer a way to offset overlapping data by using their finer resolution.

Visual tracking of a particular processor is difficult: Generally, it is very difficult to visualy track a particular processor due to both overlap effects and the abrupt displacement of data points. The latter can be severe at one second sampling rates or when sampling is pre-empted by a higher priority process for some period. The solution here is to implement time-averaged motion as an option.

Barycentric coordinates are often distorted: Due to incommensurate aspect ratios, the appearance of the equilateral triangle is distorted on standard terminal screens. The aspect distortion on *xterms* is even worse (Fig. 5) . This effect would also be deleterious under the time-averaging option. As mentioned earlier, if accurate statistics are required then, a non-distorting display device must be used.

Barycentric axes are only implicit: Because the barycentric axes are only implicit. It is difficult to accurately assess percentage times for points that lie away from the vertices. a moment's reflection reveals the diagonal axes showing the centroid (allowing SYS and IDL measurement) cannot be drawn in terminal mode with an ASCII character set. Drawing and refreshing the diagonals is clearly a motivating consideration for an X-window tool.

Performance indices must conform to a sum rule: This is a conceptual limitation and from the early discussion, there is probably a practical limit of supporting about 4 indices or 60 parameters. Such a limitation also means that it is not clear hour to use this approach for monitoring other aspects of computer system performance, e.g., I/O and communication subsystems. This is both a strength and a weakness.

Important processor statistics are not displayed: Currently, there is no indication of other important processor statistics e.g., context switch rates. One possibility would be to show a single number representing the context-switch rate averaged over all processors. This number could be displayed as a user-selecable option near the triangle.

We have introduced the notion of barycentric (center of mass) coordinates and shown how it can be as a performance monitor for the efficient, dynamic, diplay of MP ultilization statistics. The greatest benefit appears to be the presence of simple visual "signatures" which correlate with the workload to a degree where the performance analyst's peripheral attention is mostly drawn to significant changes in performance. Although reading these signatures is not intuitive, learning is minimal.

7 ACKNOWLEDGEMENTS

The author to thank the many engineers at Pyramid who "road tested" *barry* and provided the feedback which improved the content of this paper.

References

[1] 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.

[2] K. M. Nichols. "Performance Tools". *IEEE Software*, pages 21–30, May 1990.

[3] A. D. Malony. "Performance Observability. Technical Report UIUCDCS-R-90-1630, Univ. Illinois, Urbana-Champagne, IL, 1990.

[4] N. J. Gunther. "The CASE for future network systems' performance". Invited paper. Systems Design and Network Conference, Santa Clara, California, May 1990.

[5] M. Loukides. *System Performance Tuning.* O'Reilly & Assoc. Inc., Sebastopol, California, 1992.

[6] C. P. Thacker. "Alto: A personal computer". Technical Report CSL-79-11, Xerox PARC, Palo Alto, CA, August 1979.

[7] N. J. Gunther. "A distributed workstation approach to IC process characterization". In *Proceedings of WORKSTATIONS'85 Conference*, page 188, San Jose, California, Nov 1985.

[8] W. Teiteleman. "The cedar programming environment: A midterm report and examination". Technical Report CSL-83-11, Xerox PARC, Palo Alto, CA, Juune 1984.

[9] A. D. Malony. "An integrated performance data collection, analysis and visualization system. In *Proc. Fourth Conference on Hypercube Concurrent Computers and Applications*, March 1988.

[10] J. Gray, editor. *The Benchmark Handbook for Database and Transaction Processing Systems.* Morgan Kaufmann, San Mateo, CA, 1991.

[11] R. D. Trammell. "The big picture: Visualizing system behavior in real time". In *Proc. USENIX Summer Conference*, pages 257–266, June 1990.

[12] R. Jain. *The Art of Computer Systems Performance Analysis.* Wiley, New York, 1990.

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

[14] D. Ferrari, editor. *Computer System Performance Evaluation.* Prentice-Hall, 1978.

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

[16] N. Hirsh and B. L. Brown. "A holistic method for visualizing computer performance measurements". In G. Nelson and G. Shriver, editors, *Visualization in Scientific Computing.* IEEE Press, 1990.

[17] N. J. Gunther. "PARCbench: A benchmark methodology for multiprocessors". In *Proc. ACM SIGMETRICS Conference*, Santa Fe, New Mexico, May 24–27 1988.

[18] TPC Benchmark B. "Full disclosure report for the MIServer MIS 12S/12 using UNIFY 2000 release 2". Technical report, Pyramid Technology, San Jose, CA, January 1992.

[19] V. Quercia and T. O'Reilly. *X Window System Users' Guide, Vol.3.* O'Reilly & Assoc., 1991.

[20] O. Serlin. "FT systems". A monthly newsletter published by ITOM International, 1992.

[21] A. D. Malony and D. A. Reed. "Visualizing parallel computer system performance". Technical Report UIUCDCS-R-88-1465, Univ. Illinois, Urbana-Champagne, IL, 1988.

[22] H. Moravec. *Mind Children.* Harvard Univ. Press, 1988.