Issues in Informing Science and Information Technology **Volume 6, 2009**

How Anomalous Is Belady's Anomaly?

Kirby McMaster
CS Dept, Weber State University, Ogden, UT, USA

kmcmaster@weber.edu

Samuel Sambasivam
CS Dept, Azusa Pacific University, Azusa, CA, USA

ssambasivam@apu.edu

Nicole Anderson
CS Dept, Winona State University, Winona, MN, USA

nanderson@winona.edu

Abstract

In a virtual memory system using demand paging, the page fault rate of a process varies with the number of memory frames allocated to the process. When an increase in the number of frames allocated leads to an increase in the number of page faults, Belady's anomaly is said to occur. In this study we used computer simulation to examine four conditions that affect the incidence of Belady's anomaly: (1) page replacement algorithm (FIFO vs. Random Page), (2) process size, (3) reference string length, and (4) memory frames allocated to the process.

We found that over a wide range of process sizes and reference string lengths, Belady's anomaly occurred for up to 58.6% of the (random) reference strings under FIFO, and up to 100% of the reference strings for Random Page. Under conditions where anomalies occur most often, the average frame allocation level was around 75% of the process size for FIFO, but just over 50% of the process size for Random Page. Throughout the study, Belady's anomaly occurred so frequently that it no longer seems anomalous. This is especially true for the Random Page algorithm.

Keywords: virtual memory, demand paging, Belady's anomaly, page fault, reference string, FIFO, Random Page.

Introduction

Most Computer Science textbooks introduce concepts that are not fully explained or understood. Operating System textbooks often give a brief example of a phenomenon called *Belady's anomaly* and mention situations where Belady's anomaly does not occur. Rarely do they discuss conditions where anomalies are likely to occur or explain why they occur.

Belady's anomaly arises in some algorithms that implement *virtual memory*. Virtual memory is a service provided by an operating system that allows programs larger than physical memory to execute. In a virtual memory system, processes are divided into fixed-size pages. Whenever a page is needed by a process, a page fault occurs, and the required page is loaded into memory. If all memory frames have been allocated, a page in memory must be replaced each time a new page is requested. This is referred to as *demand-paging*. The choice of which page to replace is specified by a page replacement algorithm.

The page fault rate of a process varies with the number of memory frames allocated to the process. Having more available memory usually leads to fewer page faults. However, when an increase in the number of frames allocated leads to an increase in the number of page faults, Belady's anomaly is said to occur (Belady, 1966; Belady, Nelson, & Shedler, 1969). It is known that the occurrence rate for Belady's anomaly depends on which page replacement algorithm is implemented. Examples of Belady's anomaly for the FIFO page replacement algorithm are presented many Operating Systems textbooks (Deitel & Choffnes, 2004; Dhamdhere, 2008; Schlesinger & Garrido, 2007; Silberschatz, Galvin, & Gagne, 2004; Stuart, 2008). On the other hand, Belady's anomaly cannot occur when the page replacement algorithm is a stack algorithm (Mattson, Gecsei, Slutz, & Traiger, 1970).

The use of the word *anomaly* suggests that a phenomenon is "unusual" or "unexpected (Merriam-Webster, 2009). In this study, we demonstrate that Belady's anomaly occurs frequently under various system conditions. Future research will explain why the phenomenon occurs so often. Our research is not intended to change how virtual memory is implemented in current operating systems. Instead, the main goal is to improve our understanding of how demand-paging and Belady's anomaly behave. This research provides an additional educational benefit in the form of interesting programming projects for Operating Systems courses. In Computer Science, a substantial part of learning occurs during the process of designing, writing, and testing software (Knuth, 2008; McMaster, Anderson, & Rague 2007).

Methodology

This research study used computer simulation to examine conditions that affect how often Belady's anomaly will occur. We focused on four operating system conditions that affect the incidence of Belady's anomaly:

1. Page replacement algorithm.
2. Process size in pages.
3. Reference string length.
4. Number of memory frames allocated to the process.

The primary objective of the study was to provide answers to the following questions:

1. How does the *frequency* of Belady's anomaly depend on the process size and the reference string length?
2. How many memory *frames* are typically allocated when an anomaly occurs?

Each of these questions was examined for both the FIFO and Random Page replacement algorithms.

The computer simulations were performed using a program written specifically for this research study. In the simulation runs, the process size ranged from 20 to 100 pages in increments of 20. The length of the reference string varied from 25 to 12800 page references, with each value being twice the preceding value. Two page replacement algorithms were examined, FIFO and Random Page—algorithms where Belady's anomaly is known to occur.

In each simulation run, one page replacement algorithm, one process size, and one reference string length were selected. Then 1000 random reference strings were generated using a uniform random number generator. For each reference string, the number of page faults was computed for each memory allocation from 1 frame up to the process size.

For each algorithm and memory frame level, frames were "pre-filled" before any reference string page faults were counted. Specifically, K memory frames were filled with page numbers K (first out) through 1 (last in). Similarly, K+1 frames were filled with page numbers K+1 (first out) through 1 (last in). These initial allocations insured that the pages in K frames were a subset of the pages in K+1 frames, and that the pages were in the same order in the two queues (with page 1 being the most recent arrival). This initial "pre-fill" method was necessary for FIFO comparisons (although irrelevant for Random Page).

Frequency of Belady's Anomaly

For the **FIFO** page replacement algorithm, the occurrence rate of Belady's anomaly over the design region of process sizes and reference string lengths is summarized in Table 1. The body of the table gives the number of reference strings (out of 1000 randomly generated strings) that exhibited Belady's anomaly. Here, a reference string was considered to be *anomalous* if the page fault count increased (a *bump*) at least once as the number of allocated frames increased.

Table 1: Anomalous Reference Strings – FIFO
Number of reference strings exhibiting Belady's anomaly.

	Reference String Length									
Process Size	**25**	**50**	**100**	**200**	**400**	**800**	**1600**	**3200**	**6400**	**12800**
20	0	14	53	58	16	3	0	0	0	0
40	0	8	49	155	188	136	30	2	1	0
60	0	3	23	140	281	339	210	58	6	0
80	0	0	16	107	326	467	447	208	49	0
100	0	0	2	76	301	502	586	450	149	23

The number of anomalous strings over the design region in Table 1 ranges from 0 to 586 (58.6%). The maximum anomalous string count of 586 is for a process size of 100 and a reference string length of 1600. Belady's anomaly may be counter-intuitive, but under these conditions it is not rare or unusual.

Most of the cases having 0 anomalies are when the reference string length is very short or very long. In these FIFO simulations, Belady's anomaly does not occur when the reference string is short because of the way memory frames are pre-filled before the reference string is applied. As long as the page numbers for K frames are a subset of the page numbers for K+1 frames, any page fault for K+1 frames is also a page fault for K frames. Incoming page numbers from the reference string must "flush out" some initial memory pages unevenly (e.g. from K+1 frames but not from K frames) before any pages in the K frames can differ from the pages in K+1 frames.

Individual page faults are less likely for K+1 memory frames than for K frames. Thus, long (random) reference strings decrease the chance that aggregate page fault counts will exhibit Belady's anomaly. This is an instance of a Central Limit Theorem effect (Feller, 1968).

Reading Table 1 vertically, as the process size becomes larger, the maximum number of anomalous strings increases (often at different reference string lengths). One reason for the increase in maximum anomalous string counts is that larger process sizes have more pairs (K vs. K+1) of memory frames where bumps can appear.

However, for a fixed reference string length, as the process size grows larger, the anomalous string counts eventually decrease. Table 1 shows this decrease only for reference string lengths below 800, since process sizes above 100 are not included in the table. Additional reasons for the increase and decrease in anomalous string frequencies are less obvious and will be explained in a later paper.

In his original paper, Belady (1969) speculated: "Some random selection superimposed on FIFO might avoid the anomaly." Table 2 presents the number of reference strings (out of 1000) that exhibited Belady's anomaly when the **Random Page** algorithm was used. The design region of process size and reference string lengths is the same as in Table 1. Again, a reference string is counted as anomalous if it exhibits at least one bump.

Table 2: Anomalous Reference Strings – Random Page
Number of reference strings exhibiting Belady's anomaly.

	Reference String Length									
Process Size	**25**	**50**	**100**	**200**	**400**	**800**	**1600**	**3200**	**6400**	**12800**
20	995	984	917	684	292	45	2	0	0	0
40	1000	1000	1000	1000	998	945	553	99	2	0
60	1000	1000	1000	1000	1000	1000	996	795	248	11
80	1000	1000	1000	1000	1000	1000	1000	997	841	213
100	1000	1000	1000	1000	1000	1000	1000	1000	996	713

The number of anomalous strings in Table 2 ranges from 0 to 1000 (100%). The maximum anomalous string count of 1000 appears in 25 of the 50 cells, but never for a process size of 20. The only 0's in Table 2 occur when the process size is small and the reference string is long.

For each process size in Table 2, as the reference string length increases, the anomalous string count starts near 1000 and gradually decreases. As with FIFO, this is due to a Central Limit effect, which suggests that the number of anomalous strings will approach 0 regardless of the process size. The high number of anomalous strings for relatively short reference strings demonstrates that the Random Page algorithm takes fewer page references to "flush out" the initial "pre-filled" memory frames. Even so, if the reference string is short enough (near 1), the anomalous string count approaches 0 (not shown in the table).

Viewing Table 2 vertically for fixed reference string lengths, as the process size grows larger, the anomalous string counts always increase. These counts often reach the maximum value of 1000 within the design region. Overall, it is apparent that the use of the Random Page replacement algorithm leads to a dramatic increase in the number of reference strings exhibiting Belady's anomaly. Belady's anomaly occurs so often that it is unusual not to observe it.

Multiple Bumps per Reference String

Because the number of anomalous strings was so high for the Random Page algorithm, we decided to count the total number of page fault bumps generated by the 1000 reference strings. Using this approach, when an anomaly bump appears more than once for a reference string (at different frame allocation levels), each bump is included in the total. Table 3 summarizes the number of anomaly bumps for the FIFO algorithm. The maximum bump frequency for each process size (row) is shown in **bold**.

Table 3: Total Anomaly Bumps – FIFO
Number of page fault bumps for 1000 reference strings.
Some reference strings exhibit more than one bump.

	Reference String Length									
Process Size	**25**	**50**	**100**	**200**	**400**	**800**	**1600**	**3200**	**6400**	**12800**
20	0	14	57	**60**	16	3	0	0	0	0
40	0	8	49	166	**207**	144	31	2	1	0
60	0	3	23	147	334	**399**	233	59	6	0
80	0	0	16	112	374	**585**	565	232	51	0
100	0	0	2	80	357	693	**869**	533	161	23

The anomaly bump pattern in Table 3 is similar to the anomalous string pattern in Table 1, except that some frequencies are higher (due to multiple bumps per string). Anomaly bumps under FIFO appear as often as 869 times in 1000 reference strings. Comparing Table 3 with Table 1, we see that the 869 bumps occurred in 586 reference strings (for process size 100 and reference string length 1600). The detailed FIFO simulation results for these 1000 reference strings had multiple bumps distributed as follows:

```
Bumps    Strings   Total
  1        367      367
  2        161      322
  3         52      156
  4      ___ 6     __ 24
           586      869
```

For each process size in Table 3 (as in Table 1), as the reference string length grows longer, the number of anomaly bumps increases from 0 to a maximum value and then decreases back to 0. Vertically, larger process sizes have bigger maximum bump counts, but for increasingly longer reference strings. In this simulation, process size 20 reached a maximum bump count of 60 when the reference string length was 200, whereas process size 100 required 1600 page references to obtain a maximum bump count of 869.

The pattern for anomaly bump data is easier to visualize when presented graphically. Figure 1 displays the FIFO anomaly bump frequencies from Table 3 as a line graph, where each of the five lines represents a different process size. This graph summarizes the FIFO relationships between process size, reference string length, and anomaly bump counts.

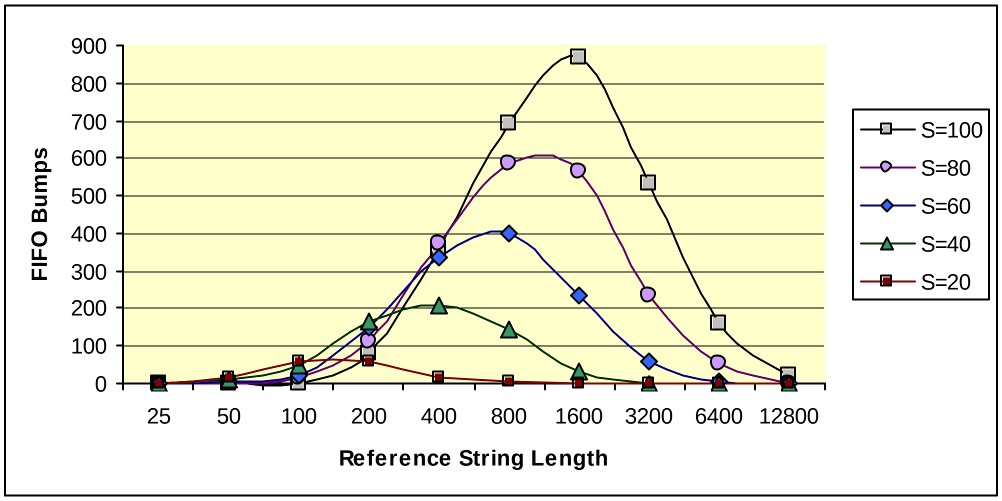

Figure 1: Total Anomaly Bumps – FIFO
Number of page fault bumps for 1000 reference strings.
Some reference strings exhibit more than one bump.

Table 4 presents the anomaly bump counts for the Random Page algorithm. The maximum bump frequency for each process size is again shown in **bold**. The Random Page bump pattern in Table 4 is similar to the FIFO results in Table 3, except the Random Page bump counts are much larger, and the maximum frequencies on each row occur for shorter reference strings.

Table 4: Total Anomaly Bumps – Random Page
Number of page fault bumps for 1000 reference strings.
Many reference strings exhibit multiple bumps.

	Reference String Length									
Process Size	**25**	**50**	**100**	**200**	**400**	**800**	**1600**	**3200**	**6400**	**12800**
20	**3236**	2857	2016	1027	331	45	2	0	0	0
40	9184	**9547**	8707	7020	4856	2439	780	103	2	0
60	14770	**16638**	16391	14861	11809	8070	4268	1549	271	11
80	20054	23615	**24400**	22929	19965	15269	10213	5131	1660	242
100	24936	30351	**32152**	31229	28204	23427	17294	10416	4829	1226

In Table 4, anomaly bumps under the Random Page algorithm occur as often as 32152 times in 1000 reference strings. Comparing Table 4 with Table 2, we see that in the case of 32152 bumps, all 1000 reference strings had at least one bump. More detailed Random Page simulation results determined that the number of bumps for these 1000 reference strings ranged from a low of 23 to a high of 40, with an average of 32.2 bumps per reference string. When a phenomenon occurs this often, can it be considered to be an anomaly?

For each process size in Table 4, as the reference string grows longer, the number of anomaly bumps increases to a maximum value and then decreases toward 0. Larger process sizes have bigger maximum bump counts, but require fairly short reference strings. In this simulation, process size 20 had a maximum bump count of 3236 when the reference string length was only 25, whereas process size 100 reached a maximum bump count of 32152 for a reference string length of 100. This again demonstrates that the Random Page algorithm takes fewer page references than FIFO to "flush out" the initial "pre-filled" memory frames.

Figure 2 displays the Random Page anomaly bump counts from Table 4 as a line graph. This graph summarizes the Random Page relationships between process size, reference string length, and anomaly bump frequencies. Figure 2 suggests that the number of anomaly bumps decreases (toward 0) when the reference string is shorter than 25, as well as for long reference strings.

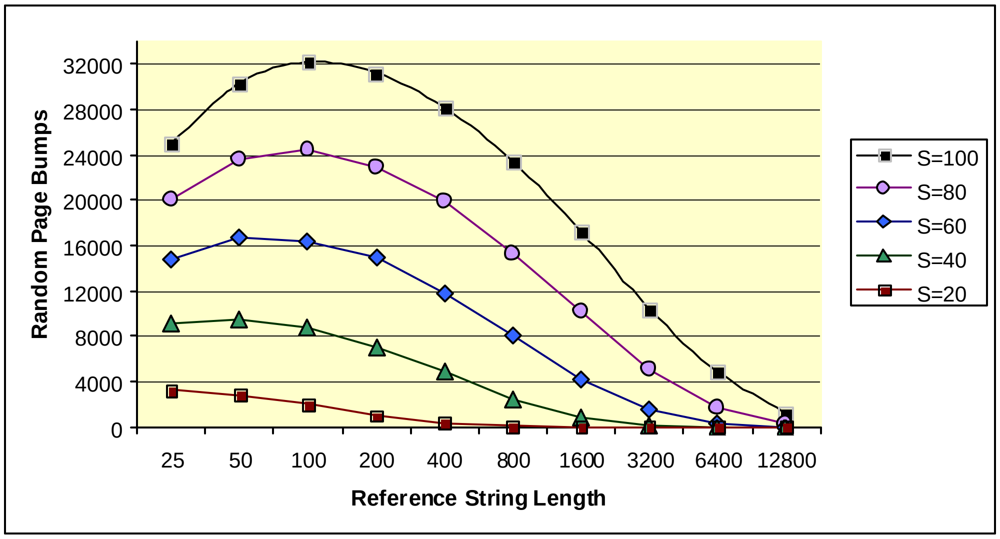

Figure 2: Total Anomaly Bumps – Random Page
Number of page fault bumps for 1000 reference strings.
Some reference strings exhibit more than one bump.

Frame Allocation and Belady's Anomaly

The second question to be answered in this study concerns the number of memory frames allocated when an anomaly bump occurs. The smallest number of frames that can be allocated is one. The largest practical number of frames to allocate is the process size, which allows the entire process to be stored in memory.

In each simulation run, page fault counts for the entire range of frame allocation levels were obtained. For each anomaly bump, the number of frames allocated when the bump occurred was recorded. The smallest frame allocation in which a bump can appear is 2, since the anomaly occurs only if the current page fault count is higher that the previous one.

Table 5 gives the average number of frames allocated for all of the bumps listed in Table 3. These results are for the FIFO page replacement algorithm. A separate average is shown for each process size and reference string length. The "--" symbol indicates that no bumps occurred, so no average frame level was calculated. The highlighted (**bold**) values in Table 5 are the average frame allocations in each row where the maximum bump frequencies appear in Table 3. With FIFO page replacement, the anomalies occur at larger average frame allocation levels as the process size and reference string length increase.

Table 5: Average Frame Level – FIFO
Average number of memory frames allocated
when an anomaly bump occurs.

	Reference String Length									
Process Size	**25**	**50**	**100**	**200**	**400**	**800**	**1600**	**3200**	**6400**	**12800**
20	--	11.6	13.1	**15.1**	16.1	16.7	--	--	--	--
40	--	15.9	21.0	26.2	**28.9**	30.8	31.8	34.0	36.0	--
60	--	16.0	25.9	34.3	41.0	**45.5**	47.2	48.7	49.2	--
80	--	--	32.7	41.2	51.8	**57.8**	62.2	64.3	65.5	--
100	--	--	35.0	46.7	60.2	69.4	**76.7**	80.4	82.2	82.3

Table 6 gives the average number of frames allocated for all of the bumps listed in Table 4. These results are for the Random Page algorithm. The highlighted (**bold**) values in Table 6 are the average frame allocations in each row at the point where the maximum bump frequencies occur in Table 4.

Table 6: Average Frame Level – Random Page
Average number of memory frames allocated when an anomaly bump occurs.

	Reference String Length									
Process Size	**25**	**50**	**100**	**200**	**400**	**800**	**1600**	**3200**	**6400**	**12800**
20	**10.7**	11.1	11.3	11.5	11.7	12.4	12.5	--	--	--
40	20.0	**20.7**	21.4	21.6	22.1	22.5	23.1	23.3	23.0	--
60	29.0	**30.3**	30.9	31.5	31.9	32.5	33.2	34.2	35.2	29.7
80	37.4	39.3	**40.5**	41.4	42.2	42.7	43.2	44.3	45.0	44.7
100	46.2	48.8	**50.3**	51.2	51.9	52.5	53.5	54.4	55.4	56.2

The Random Page pattern in Table 6 is similar to the FIFO pattern in Table 5, in that the average frame allocation level for anomalies is higher for large process sizes and long reference strings. We note again that, compared to FIFO, Random Page maximum bump points occur for smaller frame levels and for shorter reference string lengths.

The upper section of Table 7 summarizes the relationships between process size, reference string length, average 1st frame, and average frame level at conditions where FIFO anomaly bumps are most frequent. As the process sizes moves from 20 to 100 pages, the reference string length increases from 200 to 1600, and the maximum number of bumps grows from 60 to 869. In these cases, the average number of frames allocated using FIFO is around 75% of the process size.

Table 7: Anomaly Summary – FIFO and Random Page
Reference string length, maximum bumps, 1st frame, and average frame, by process size

Process Size	Ref Str Length	Max Bumps	Avg 1st Frame	Average Frame	Frame/ Size(%)
FIFO					
20	200	60	15.0	15.1	75.5%
40	400	207	28.7	28.9	72.3%
60	800	399	45.0	45.5	75.8%
80	800	585	56.7	57.8	72.3%
100	1600	869	74.5	76.7	76.7%
Random					
20	25	3236	6.5	10.7	53.5%
40	50	9547	6.3	20.7	51.8%
60	50	16638	6.1	30.3	50.5%
80	100	24400	6.2	40.5	50.6%
100	100	32152	6.0	50.3	50.3%

In addition to the average frame level for all bumps, this table includes the average 1st bump frame. Here we are concerned with the smallest number of frames for which an anomaly bump appears (especially for strings with multiple bumps). For FIFO, the average 1st bump frame is slightly below the average frame level for all bumps, due primarily to the fact that few reference strings have more than one bump.

The lower section of Table 7 describes the same relationships at conditions where Random Page anomaly bumps are most frequent. For process sizes between 20 and 100 pages, the reference string lengths of 25, 50, and 100 are much shorter than for FIFO. On the other hand, the maximum number of bumps is substantially larger, ranging from 3236 to 32152. In these cases, the average number of frames allocated using Random Page is just over 50% of the process size.

An interesting pattern is manifest for average 1st bump frame in the Random Page section of Table 7. The average 1st frame for anomaly bumps does not increase with process size, but instead remains almost constant at about 6 frames. Thus, for larger process sizes, the average 1st bump frame becomes more distant from the average frame level for all bumps. Under Random Page, many reference strings have multiple bumps, so anomaly bumps occur across a wide range of frame levels.

Figure 3 provides a visual summary of most of the data in Table 7. This figure implies that the relationships between process size and the average frame level variables are approximately linear, but with different slopes. For FIFO, the slope is approximately 0.75 for both the average frame level for all bumps and the average 1st bump frame. For Random Page, on the other hand, the slope is about 0.50 for the average frame level for all bumps, whereas the slope is slightly negative for the average 1st bump frame.

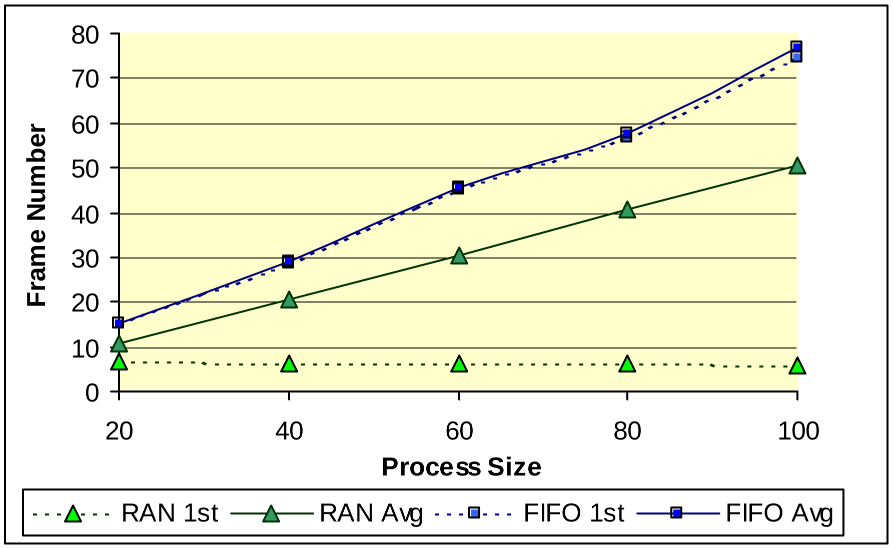

Figure 3: Frames vs. Process Size – FIFO and Random Page
Relationship of 1st frame and average frame number with process size for reference string lengths having maximum anomaly bump counts.

Summary and Conclusions

In this study we examined four conditions that affect the occurrence of Belady's anomaly: (1) page replacement algorithm (FIFO vs. Random Page), (2) process size, (3) reference string length, and (4) memory frames allocated to the process. Computer simulation was used to determine how often Belady's anomaly occurs and how the anomaly depends on the above conditions.

We found that over the range of process sizes and reference string lengths examined, Belady's anomaly occurred for up to 58.6% of the reference strings under the FIFO algorithm, and up to 100% of the reference strings under the Random Page algorithm. The highest total number of anomaly bumps per 1000 reference strings was 869 for FIFO and 32152 for Random Page.

At maximum anomaly bump conditions, the average memory frame allocation level was proportional to the process size. For FIFO, the average frame allocation was around 75% of the process size. For Random Page, the average frame allocation was just over 50% of the process size.

The overall results indicate that across a broad range of process sizes and reference string lengths, Belady's anomaly occurred so frequently that it no longer seems anomalous. This is especially true for the Random Page algorithm.

Future Research

The primary purpose of this study was to describe *how often* Belady's anomaly occurs and at what frame allocation levels, given design conditions for process size, reference string length, and page replacement algorithm. The previous sections of this paper provide very little explanation of *why* the observed patterns occur. We did suggest that larger process sizes have more pairs (K vs. K+1) of frames, that Central Limit Theorem effects may apply for longer reference strings, and that pre-filled frames must be unevenly flushed by new page references before anomalies can appear.

An analysis of why anomalies are more frequent for the Random Page algorithm and at specific frame levels K vs. K+1 requires an appropriate memory model. At any point in time, each of the process pages will be in precisely one of four distinct locations:

1. Unshared pages in K frames.
2. Unshared pages in K+1 frames.
3. Shared pages in K frames and K+1 frames.
4. Not in memory.

In future research, we will develop a probability framework for the above memory model. We plan to extend our computer simulation to examine the average number of shared and unshared pages under the design conditions employed in this study. It is expected that the number of unshared pages in K frames will usually be larger (and shared pages smaller) for the Random Page algorithm than for FIFO. If true, this would explain the higher occurrence rate of Belady's anomaly for the Random Page algorithm.

References

Belady, L. A. (1966). A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, *5*(2).

Belady, L. A., Nelson, R. A., & Shedler, G. S. (1969). An anomaly in space-time characteristics of certain programs running in a paging machine. *Communications of the ACM*, *12*(6).

Deitel, P., & Choffnes, D. (2004). *Operating systems*. Prentice Hall.

Dhamdhere, D. (2008). *Operating systems*. McGraw-Hill.

Feller, W. (1968). *An introduction to probability theory and its applications* (3rd ed.). Wiley.

Knuth, D. (2008). Donald Knuth: A life's work interrupted. *Communications of the ACM*, *51*(8).

Mattson, R. L., Gecsei, L., Slutz, D. R., & Traiger, I. L (1970). Evaluation techniques for storage hierarchies. *IBM Systems Journal*, *9*(2).

Merriam-Webster. (2009). *Merriam-Webster online dictionary*. Retrieved from www.merriam-webster.com

McMaster, K., Anderson, N., & Rague, B. (2007). Discrete mathematics with programming: Better together. *Proceedings of SIGCSE 2007*.

Schlesinger, R., & Garrido, J. (2007). *Principles of modern operating systems*. Jones & Bartlett.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2004). *Operating system concepts* (7th ed.). Wiley.

Stuart, B. (2008). *Principles of operating systems: Design and applications*. Course Technology.

Biographies

Dr. Kirby McMaster recently retired from the CS Dept. at Weber State University. His primary research interests are in Database Systems, Discrete Mathematics, and Mathematical Frameworks for Computer Science.

Dr. Samuel Sambasivam is chairman of the CS Dept at Azusa Pacific University. His research interests include Optimization Methods, Expert Systems, Client/Server Applications, Database Systems, and Genetic Algorithms.

Dr. Nicole Anderson is an Assistant Professor in the CS Dept. at Winona State University. Her teaching and research interests involve Database Systems, Software Engineering, and the development of collaborative online learning communities.