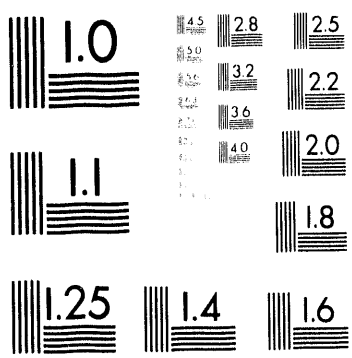

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

1 of 1

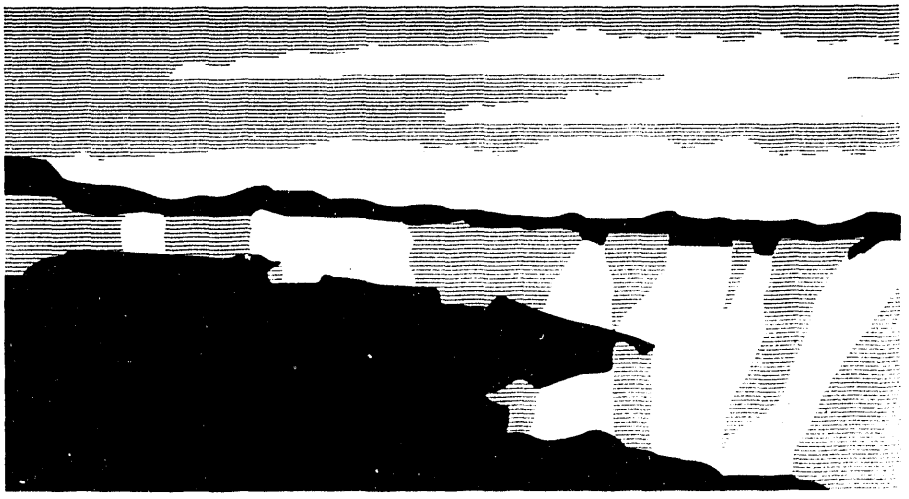
CONF-9310237--7

LA-UR 93-3697

Title: Virtual Data

Author(s): E. Bjorklund

Submitted to: International Conference on Accelerator and Large Experimental Physics Control Systems
Berlin, Germany
October 18-22, 1993



Los Alamos
NATIONAL LABORATORY

Form No. 836 R5
ST 2629 10/91

VIRTUAL DATA*

E. Bjorklund
Los Alamos National Laboratory
Los Alamos, New Mexico 87545, USA
bjorklund@lanl.gov

In the 1970's, when computers were memory limited, operating system designers created the concept of "virtual memory" which gave users the ability to address more memory than physically existed. In the 1990s, many large control systems have the potential for becoming data limited. We propose that many of the principles behind virtual memory systems (working sets, locality, caching, and clustering) can also be applied to data-limited systems – creating, in effect, "virtual data systems." At the Los Alamos National Laboratory's Clinton P. Anderson Meson Physics Facility (LAMPF), we have applied these principles to a moderately sized (10,000 data points) data acquisition and control system. To test the principles, we measured the system's performance during tune-up, production, and maintenance periods. In this paper, we present a general discussion of the principles of a virtual data system along with some discussion of our own implementation and the results of our performance measurements.

1. Introduction

Data acquisition in accelerator control systems can usually be characterized as either demand-driven or polled. At the Los Alamos National Laboratory's LAMPF-PSR accelerator complex, we have experience with both types of systems [1]. We have observed that a demand-driven system can suffer performance degradation during times of peak load. A polled system will degrade more gracefully with load. However, this can translate to consistently poor performance if the number of data points in the system becomes too large.

In 1989, we began a project to upgrade the interface computer for the RICE data acquisition system. RICE has been the main data acquisition system for the LAMPF linac since the machine was built in the late 1960s [2]. At the same time, we also decided to upgrade our data acquisition strategy.

Previously, the RICE system had been strictly demand driven. We had already been experiencing peak-demand-related performance problems. Because of the number of data channels in the RICE system, however (around 10,000), we knew that a purely polled system would not be able to give us adequate response time. Our proposed solution was a demand-based polling system -- one that would automatically determine which data channels were most frequently requested, poll those channels, and be demand driven for all the others.

Early in the design of the new system, we noticed, and then exploited, a number of similarities between our data acquisition system and the virtual memory computer operating systems that had become popular in the 1970s. Because of these similarities (along with an incorrigible proclivity for whimsy), we began calling our new system a "virtual data system."

* Work supported by the U.S. Department of Energy.

2. Virtual Memory, Virtual Data, and the Principle of Locality

A good overall description of virtual memory systems can be found in [3]. The major reason that virtual memory systems are successful is the "principle of locality" [4], which roughly states that "*processes tend to reference storage in non-uniform, highly localized patterns that are relatively stable over time.*" We suspected that a similar principle of locality might also hold true for data references in an accelerator control system. In particular, we noticed that as our control system became more geographically distributed, certain displays were reproduced at several locations throughout the site. We also observed that there were distinct activity patterns associated with the task of operating an accelerator complex. These observations led us to believe that it should be possible to construct a data acquisition system in which only a subset of the available data was polled. Furthermore, it should be possible to dynamically determine which data channels to poll based on observed reference patterns.

The principle of locality leads to the concept of the "working set" [5]. The formal definition of a working set in a virtual memory system is "*The subset of a program's address space that must be physically resident at a given time in order for that program to operate.*" Since this is difficult to determine *a priori*, most operating systems implement a slightly modified version of this definition, which is "*The subset of a program's address space which is currently resident in physical memory.*" This is a subtle difference, but it allows the operating system to make empirical judgments about how much physical memory to allocate to a program based on program behavior and system resources. It also allows the operating system to set a limit on the working set size and prevent the program from consuming all the system's memory resources. In a similar vein, we propose that the "theoretical" definition of a virtual data working set be "*The subset of all available data which is actively being referenced at the current time.*" It is important to keep in mind both the theoretical and "implementation" definitions since the success of the system will depend largely on how closely the implementation approximates the theoretical definition.

3. Caching, Clustering, and the Background Data Reader

Hatfield [6] identifies two kinds of locality -- temporal and spatial. Temporal locality implies that a storage location or a data channel that was recently referenced has a high probability of being referenced again within a short period of time. Spatial locality implies that the immediate neighbors of a recently referenced location or channel will also have a high probability of being referenced in the near future. Virtual memory systems deal with spatial locality by "clustering" -- reading multiple pages at a time from the backing store. Some data acquisition systems are also capable of reading data channels in clusters. When the RICE system issues a read, the operation is issued simultaneously to all RICE modules in the accelerator. Since there are currently 76 operational

RICE modules, each read operation returns 76 data values representing the same channel sampled along the length of the accelerator. This "longitudinal clustering" is useful for things such as obtaining a snapshot of beam spill along the linac. This may not, however, be the most efficient clustering mechanism for taking advantage of spatial locality. On the other hand, we feel that the question of just what constitutes a "neighboring data channel" is not very well understood at this point.

Virtual memory systems improve their performance by keeping the most recently referenced locations in a high-speed cache memory. Similarly, the idea behind polled systems is that it is faster to retrieve values from a memory-resident data cache than it is to read the values from the hardware. The utility of a cached data value, however, will decay with time. In fact, one of the problems with pure polling systems is that if the polling process dies, the operator can be misled by data values that are no longer current. A limit needs to be placed on how old a data value can be and still be considered valid. We call this limit the "valid data threshold time" (τ). Appropriate values for τ will depend on the needs of the system. If the primary consideration is operator response time, then a τ between $^1/_5$ and $^1/_4$ second should be adequate. Our system uses a τ of $^1/_4$ second. Given this, we now propose that the "implementation definition" of a virtual data working set be "*The subset of data available from a memory-resident cache whose values are less than τ seconds old.*"

We assume (by the principle of locality) that the contents of the working set will change relatively slowly with respect to τ. Consequently, we employ a "background reader" process to keep the contents of the working set refreshed in the memory-resident cache. The background reader is essentially a polling process, but it only polls those data channels that are in the current working set. To determine which data channels are in the current working set, the background reader first determines the average amount of time it takes to perform a single read ($\bar{t}_r$). From $\bar{t}_r$ and τ, the background reader can determine the maximum size of the working set (W_{max}) using the formula:

$$W_{max} = c\frac{\tau}{\bar{t}_r} \quad ,$$

where c is the cluster factor. The value of $\bar{t}_r$ is determined by counting how many reads the background reader actually performs in a given time interval. Therefore, $\bar{t}_r$ is automatically corrected for system loading. The number of hardware reads (N_r) that can be accomplished within the time interval, τ, is given by:

$$N_r = \frac{W_{max}}{c} = \frac{\tau}{\bar{t}_r} \quad .$$

The data channels in the current working set, and which will be polled by the background reader, are then the N_r most frequently requested channels along with any channels they are clustered with.

It is important to note that the size and contents of the working set are not determined by the size of the memory-resident cache, as they might be in a virtual memory system. The contents of a virtual data working set are determined by the age of the values in the memory-resident cache and not by the size of the cache (which presumably could be large enough to hold all the data in the system). The size of the working set could actually be smaller than W_{max} if N_r different channels were not referenced during the sampling period. Note also that a data channel does not have to be on the background reader's list of polled channels to be in the working set. If a data channel is requested whose current value is not in the working set, a "data fault" (analogous to a virtual memory "page fault") will occur, the data value will be fetched from the hardware, and that channel will become a part of the working set for the next τ seconds. This differs slightly from the traditional "replacement policies" of most virtual memory systems.

The background reader needs to periodically re-evaluate the working set to make sure it is still polling the correct channels. In our system, this re-evaluation arbitrarily occurs every 90 seconds. One could also envision a system that monitored the data fault rate to determine when to do the re-evaluation. As mentioned above, the process of re-evaluating the working set consists of first determining the values of $\bar{t}_r$ and N_r, and then selecting the N_r most frequently requested channels. The most obvious way to make this selection is to sort the channels based on access frequency. We can be more efficient, however, if we realize that the list of the N_r most frequently accessed channels does not have to be ordered. By making a simple modification to the "Quicksort" algorithm described in [7], we can obtain an unordered list of the N_r most frequently accessed channels in $\mathcal{O}(n)$ time -- where "n" is the total number of data channels in the system. This is opposed to the $\mathcal{O}(n \log n)$ time typical of the better sort algorithms.

4. Old Data, New Data, and Used Data

In a sense, all data is "old data," since there will always be a delay between the time a value is acquired and when it is used. We define "old data" to be data whose values were retrieved from the memory-resident cache and not directly from the hardware. The age of an "old" data value can therefore be anywhere between 0 and τ seconds (not counting system and network overhead). If the age of a cached data value is greater than τ seconds, the next request for that data value will be satisfied by reading the hardware. We call this a "new data" value.

Early on, we realized that there are circumstances in which the data value must always be read from the hardware and not from the cache. A data-averaging routine, for example, would be defeated if the same old-data value was reread for each sample. Another example might be a closed-loop control process with a time constant less than 2τ. In order to accommodate these cases, we had to modify our data access routines so that the caller could explicitly request "new data." When the system sees the "new data" flag, it always queues a read operation on the data

channel regardless of the time stamp on the cached data. In doing this, we made a conscious decision to make "old data" the default and "new data" the exception.

We also recognize a third category of data, which is in-between old and new data. When the system determines that a request must be satisfied with new data -- either because the cached value is too old or because the caller explicitly requested new data -- the request is placed on an appropriate hardware read queue. While the request is waiting on the queue, an earlier request may either read the same data channel or a channel in the same cluster. When this happens, the second request is removed from the queue and satisfied with the data from the earlier request. We call the data for the second request "used data" since it originally belonged to an earlier request.

5. Results

In order to test our assumptions about working sets and locality, we instrumented the system and collected statistics at various stages of the 1993 run cycle. These stages were: 1) Before startup -- when the accelerator is off but the control system is still involved with vacuum monitoring and program development; 2) Early startup -- doing equipment checks and the initial tune; 3) Late startup -- during the final production tune; and 4) Normal production -- delivering beam to the experimental targets. We also accumulated (for comparison purposes) statistics for a high-intensity accelerator development run. The results are summarized in Table 1.

	Fresh Data	Used Data	Old Data	Requests/Sec.
Idle	12 %	7 %	81 %	9
Early Startup	5 %	9 %	86 %	14
Late Startup	10 %	14 %	76 %	35
Development	25 %	11 %	64 %	31
Production	20 %	18 %	62 %	58

Table 1
RICE Data System Statistics for 1993 Run Cycle

The "Requests/Second" column gives a relative measure of the system load. The numbers in this column are somewhat misleading because our system allows for "aggregate requests." In some cases, a single "request" may return over 800 data items.

The "Old Data" column provides a good indication of the degree of locality present in the system. The highest degree of locality is observed during early startup. This phase typically involves running a limited number of tasks, for long periods of time, concentrated on a small subset of the machine. It is therefore not surprising to observe a high degree of locality during the early startup phase. We notice that locality decreases as the machine progresses towards

production status. As we approach production, the number and diversity of tasks increases in proportion to the percentage of the machine that is operational. Note that even during production, when locality is at its lowest, we still only have to go to the hardware for one request out of every five.

The "Used Data" column can give some insight into the effectiveness of our clustering mechanism. As mentioned above, the RICE system uses a "longitudinal" clustering scheme. Comparing the "Used Data" column with the "Fresh Data" column, we see that on the average, a single read will satisfy at most one other queued request. This could indicate that the longitudinal clustering is not a good match with the actual spatial locality of the system. It could also mean, of course, that the read queues are not very deep.

6. Conclusions

We feel that our experiment with virtual data has been successful. In particular, we have eliminated most of the "peak demand" bottlenecks we observed in the previous system. In some of our more data-intensive programs, we have measured a factor of five performance increase in their data acquisition times.

Based on our experience so far, we feel that there are two important parameters to consider in building a successful virtual data system. These are the maximum working set size (W_{max}), and the time required to read the hardware (t_h). If the value of W_{max} is larger than the total number of data channels in the system, then a pure polling system will probably yield acceptable response. On the other hand, if the value of W_{max} is significantly smaller than the quantity of data required (the theoretical definition of the working set), then the system may become overloaded and experience performance problems. This is analogous to the phenomenon of "thrashing" in a virtual memory system.

If the time required to read the hardware (t_h) is a large fraction of the total time required to access a data value (including system and network overhead), then a virtual data system will probably yield a significant performance improvement over a purely demand driven system. In our system, t_h is approximately $1/5$ the total data acquisition time, so the performance increase on an unloaded system was relatively modest. Where we realized our greatest performance increase (such as the factor of five mentioned above) was in aggregate requests -- requests that return more than a single data value. Note also that if t_h approaches or exceeds τ, the system may benefit from not having a background reader process.

A great deal more can be said about virtual data systems than space will allow. There are also a number of areas we feel require further study. In particular, we would like to better understand how spatial locality applies in an accelerator control system. Another interesting area of study

would be the effects of local versus global working sets. We hope that this paper can be a springboard for further discussion and work in this area.

7. Acknowledgments

I would like to thank Stuart Schaller for many hours of fruitful discussion regarding data access and the concept of virtual data. I would also like to thank David Schultz, who implemented the original RICE interface software and who had many useful suggestions about how it might be improved. When I use the word "we" in this paper, I am usually referring to these two gentlemen.

8. References

[1] S.Schaller, et.al., Proc. 1991 International Conference on Accelerator and Large Experimental Physics Control Systems, (KEK Proceedings 92-15, Tsukuba Japan, 1992) p. 7.

[2] D.R.Machen, R.A.Gore, and D.W.Weber, IEEE Trans. Nucl. Sci., NS-16 (1969) p. 883.

[3] P.J.Denning, Computing Surveys, vol. 2, no. 3 (ACM Press, New York NY, 1970) p. 153.

[4] L.A.Belady, IBM Systems Journal, vol. 5, no. 2 (1963) p. 78.

[5] P.J.Denning, Communications of the ACM, vol. 11, no. 5, (ACM Press, New York NY, 1968) p. 323.

[6] D.Hatfield, IBM Journal of Research and Development, vol. 15, no. 1 (1972) p. 58.

[7] C.A.R.Hoare, Computer Journal, vol. 5, no. 1 (1962) p. 10.

DATE
FILMED
1/3/94
END