Visualizing Working Sets

Evangelos P. Markatos*
Institute of Computer Science (ICS)
Foundation for Research & Technology – Hellas (FORTH)
P.O.Box 1385, Heraklio, Crete, GR-711-10 GREECE
markatos@csi.forth.gr, http://www.ics.forth.gr/proj/avg/paging.html

1 Introduction

It is widely known that most applications exhibit locality of reference. That is, applications access only a subset of their pages during any phase of their execution. This subset of pages is usually called the **working set** of the application. In this note we present the working set of applications in pictorial form so that it can be easily viewed and understood. Based on these working set "pictures' we make observations about the size, the duration, and the regularity of the working sets of various applications. Our applications cover several domains, ranging from numerical applications, program development tools, CAD simulations, and database applications. Our results suggest that most numerical and some database applications have regular access patterns and good locality of reference. Although most database and program development applications seem to have little locality of reference, careful observations at the appropriate granularity reveal regular and sometimes periodic access patterns.

2 Experiments

We choose to visualize the working sets of several applications that are representative of long-running computations on large data sets:

- LU, SP, MG, BT are from the NAS suite of benchmarks which are long-running applications on large data sets.
- N-BODY implements the Barnes-Hut simulation for the N-BODY problem, an application that seems to have no regular access pattern, since it traverses hierarchical data structures.
- GZIP is a file compressing utility, and GCC is a compiler, which are both representative of a program development environment.
- XLATR and ESIM are part of the CAD Benchmarks distributed by the Collaborative Benchmarking Laboratory (`http://cbl.ncsu.edu/`).
- The last three applications are from the database domain: OO7-T1 (part of the OO7 benchmark [1]) is a traversal of a database, OO7-Q7 is a query on the same database, and SQL is an SQL server running the TPC-B benchmark.

The characteristics of the applications are summarized in the following table:

*The author is also with the University of Crete, Greece.

Application name	Application domain	Length (in million data refs)
LU	numeric	1,000
SP	numeric	1,000
MG	numeric	849
BT	numeric	754
N-BODY	numeric	1,000
GCC	compiler	111
GZIP	compression	1,000
XLATR	CAD	717
ESIM	CAD	1,000
OO7-T1	database traverse	50
OO7-Q7	database query	1
SQL	database TPC-B	5

We executed the above benchmark applications on top of the ATOM tracing environment [5] (apart from the SQL application for which we had the traces already available [4]), running on top of Digital UNIX V3.2D-1. To keep the tracing times reasonable, long running applications were run as long as necessary to collect one billion data references, which correspond to several billion instructions.

The working sets of the applications are shown in figures 1 to 12. Figures 1 to 9 have two views of the working set: The first one presents the memory access pattern of the application. This access pattern is calculated as follows: Time is divided in intervals (usually 2 million data references long). For each page accessed at least once, we plot a dot on the respective interval that the page was accessed. Thus, each dot in the figure corresponds to at least one access to a (4-KByte) page done sometime during a 2-million-references interval [1]. For example, in figure 1 we see that most of the pages the LU application accesses range between 262000 and 268000. In most figures we also plot the size of the working set for each time interval.

3 Observations

Looking at the access patters of the studied applications and their working sets, we observe:

- Several of the studied applications have **periodic access patterns**. For example, in fig. 1 we clearly see that the LU application has an initialization phase and six periods where LU accesses the same pages over and over again. Some other applications, like the BT (fig. 3), MG (fig. 4), and SP (fig. 5) also have periodic access patterns. It seems that these applications access the same pages over and over again.

- Most applications make **long sequential accesses**. For example, in fig. 1 we see that the LU applications accesses several pages in ascending order during some phases, and in descending order during other phases. Similar observations can be made for the BT (fig. 3), MG (fig. 4), SP (fig. 5), and XLATR (fig. 8) applications. Although other applications, like GCC and ESIM, do not seem to make long sequential accesses, careful observations of fig. 14(b), and fig. 16(c), also reveals that these applications make long sequential accesses to some of their data as well.

[1] To increase clarity of the presentation, sporadic accesses to pages outside the figure boundaries are not shown. Since most of the figures were several Mbytes long each, we reduced the level of detail plotted when this reduction did not change the observable access patterns of the applications.

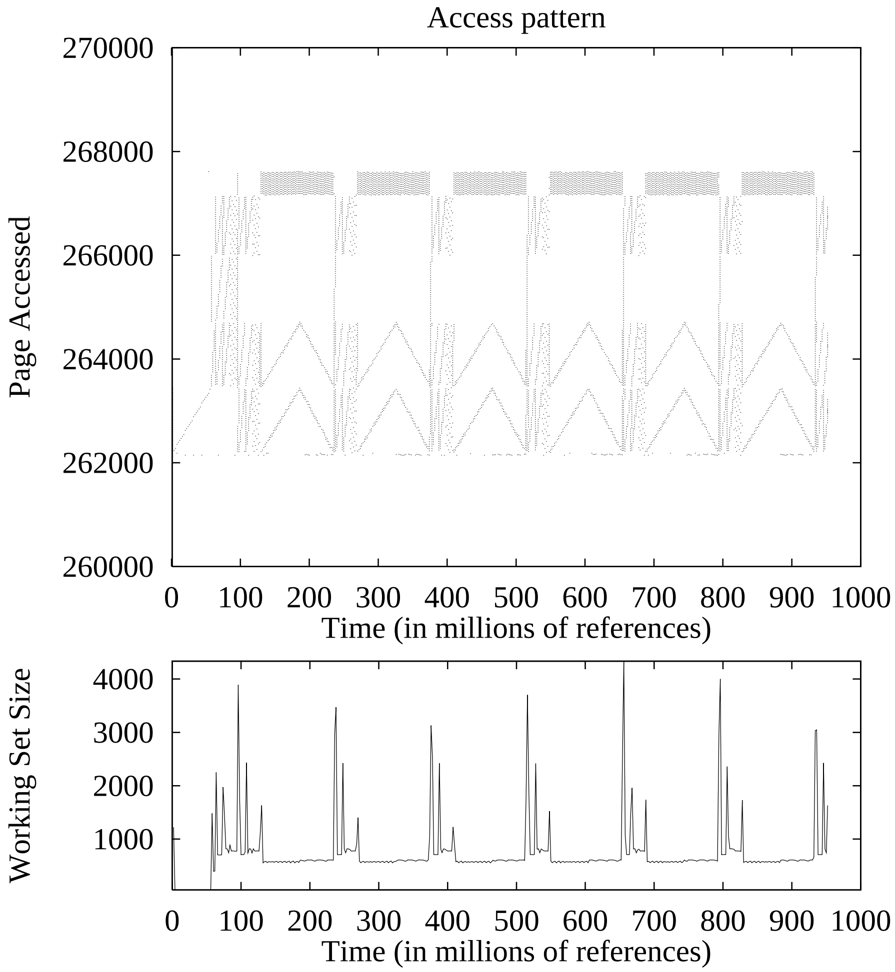


Figure 1: Access Pattern and working set size of LU

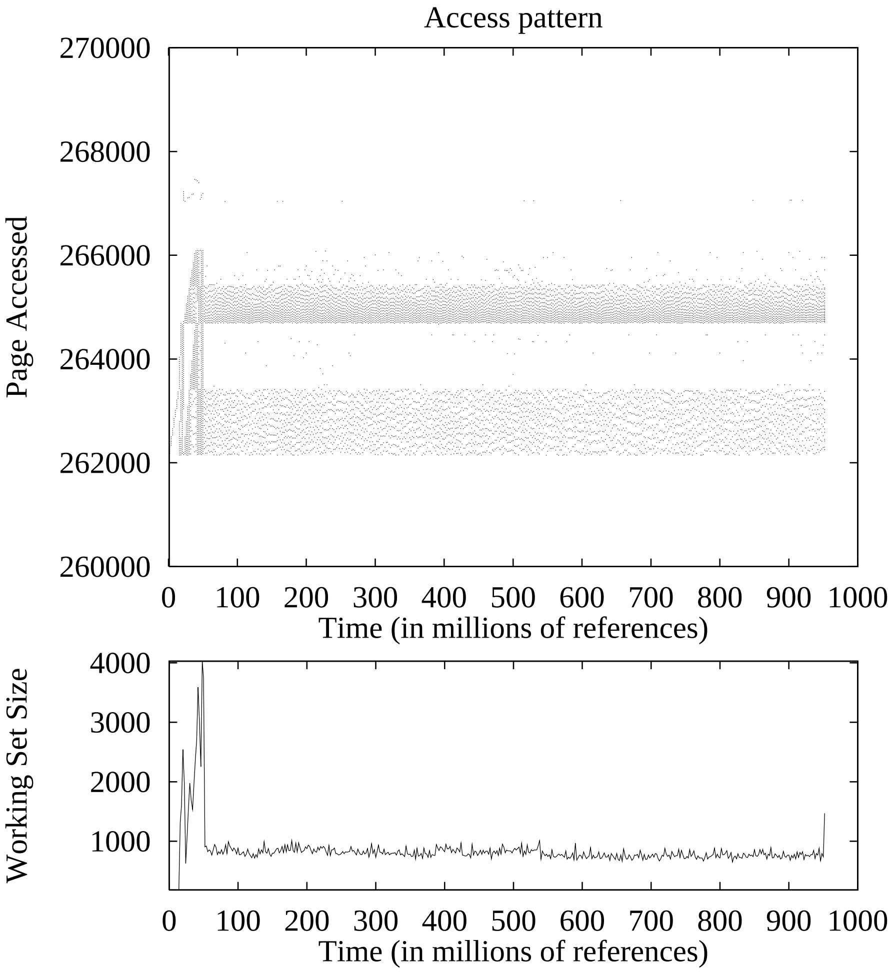


Figure 2: Access Pattern and working set size of N-BODY simulation

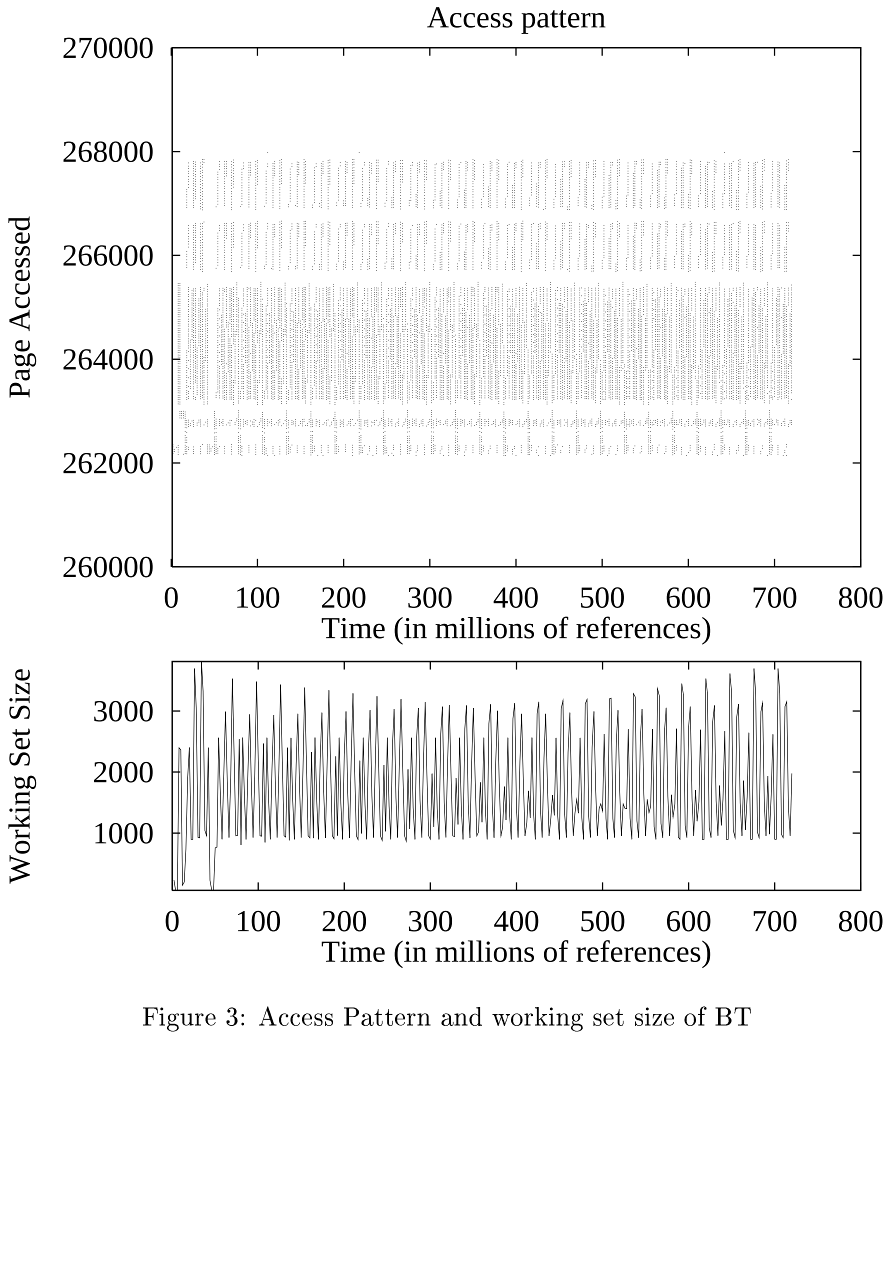


Figure 3: Access Pattern and working set size of BT

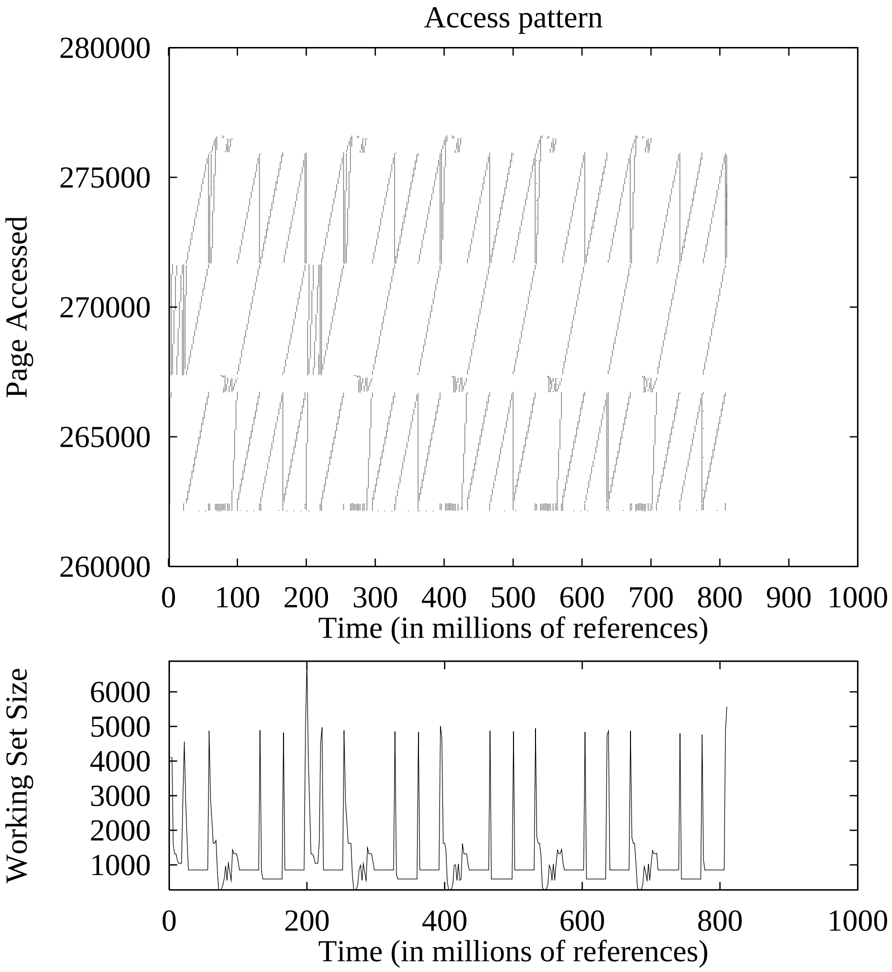


Figure 4: Access Pattern and working set size of MG

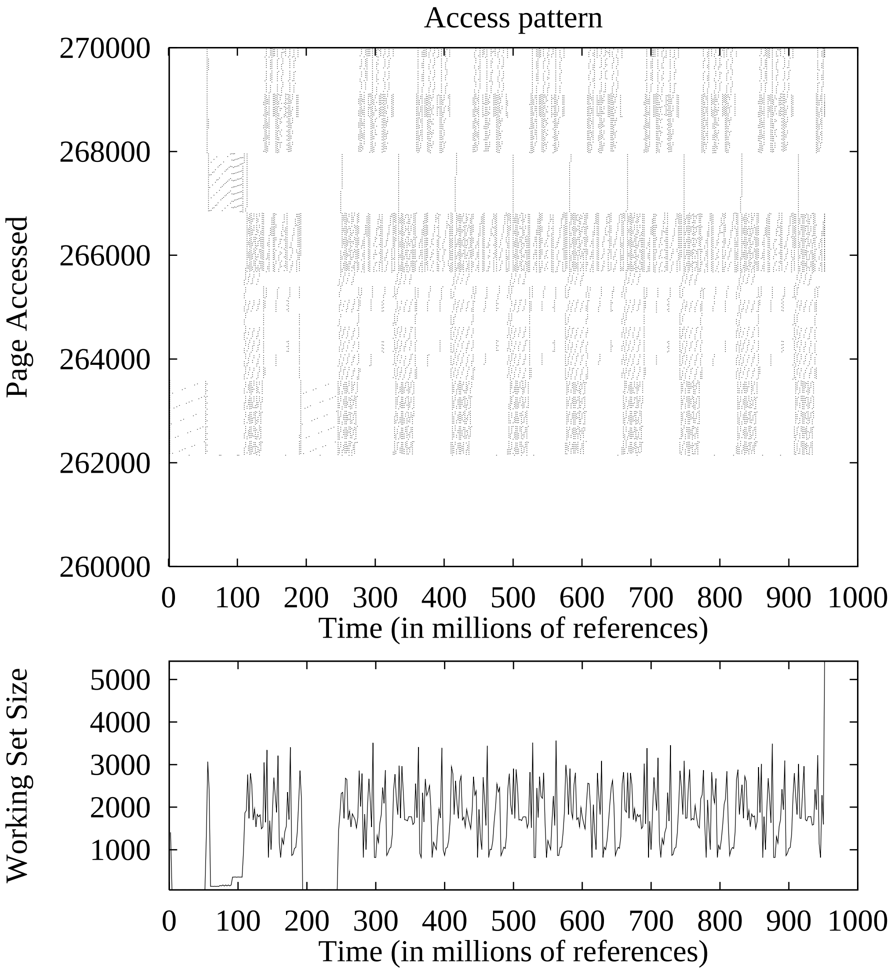


Figure 5: Access Pattern and working set size of SP

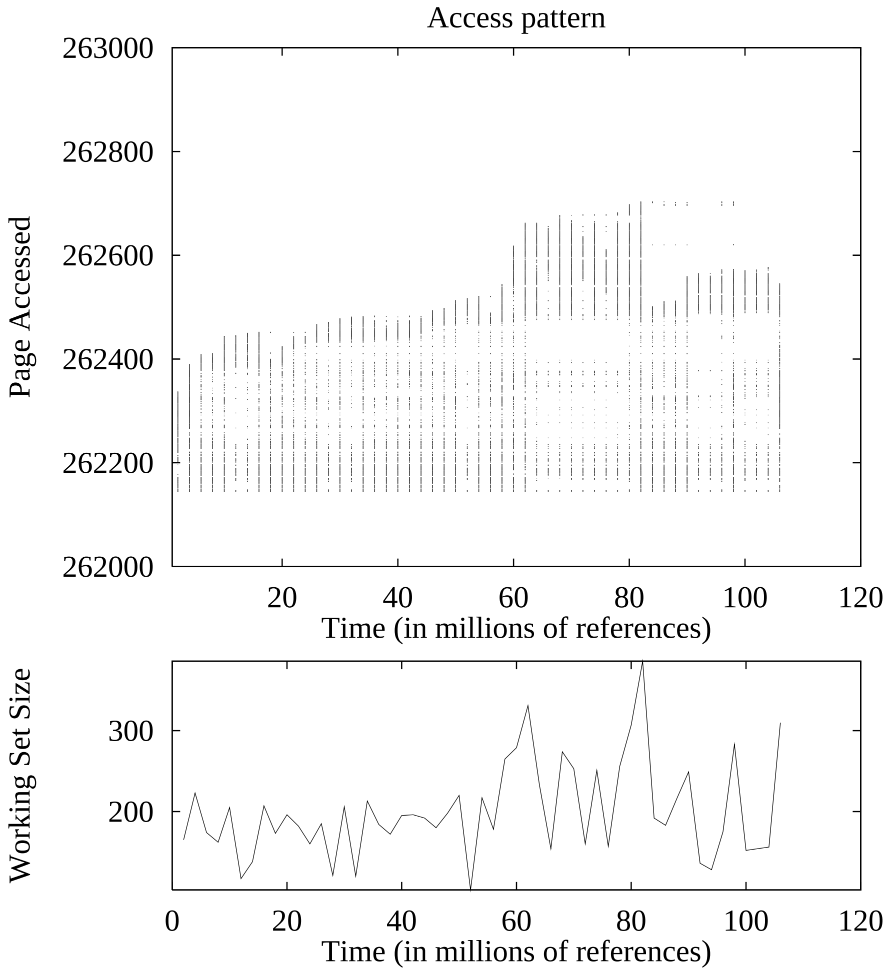


Figure 6: Access Pattern and working set size of GCC

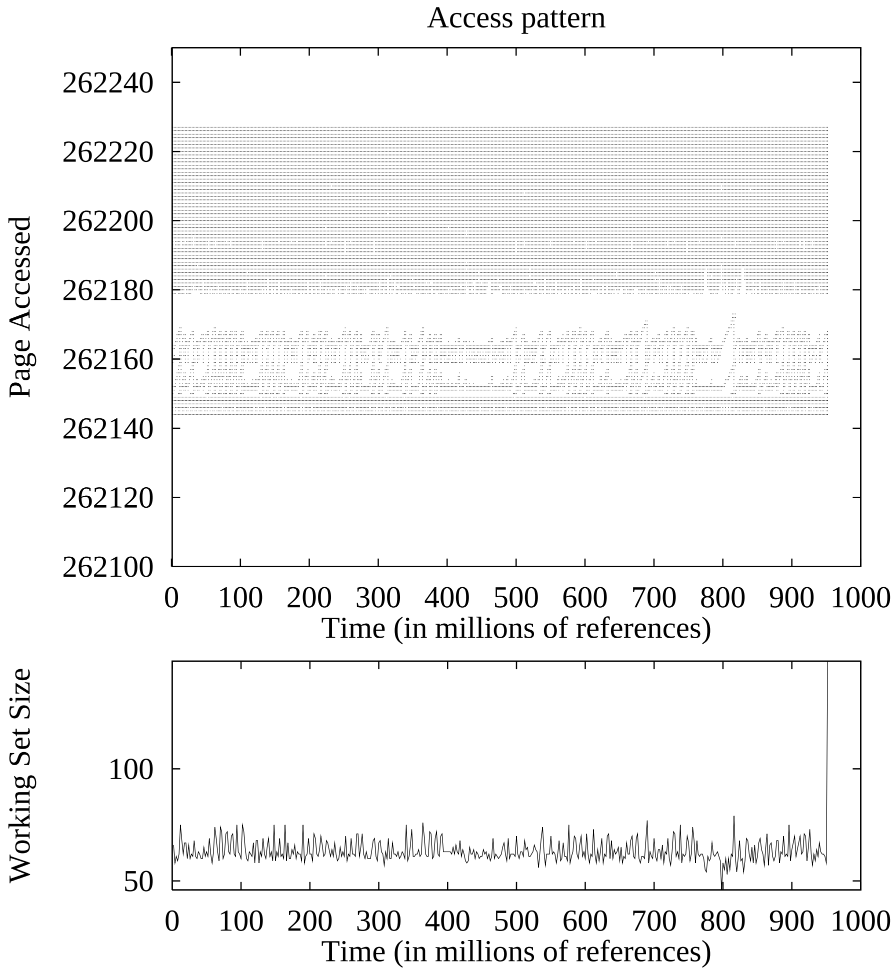


Figure 7: Access Pattern and working set size of GZIP

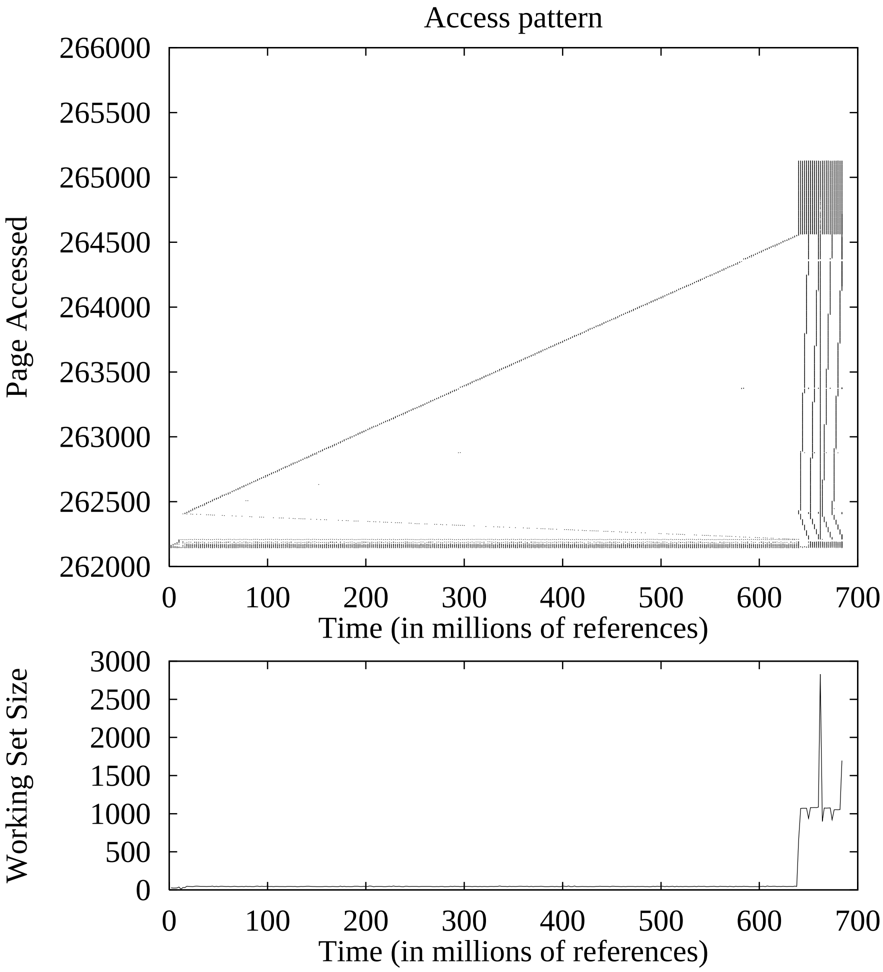


Figure 8: Access Pattern and working set size of XLATR

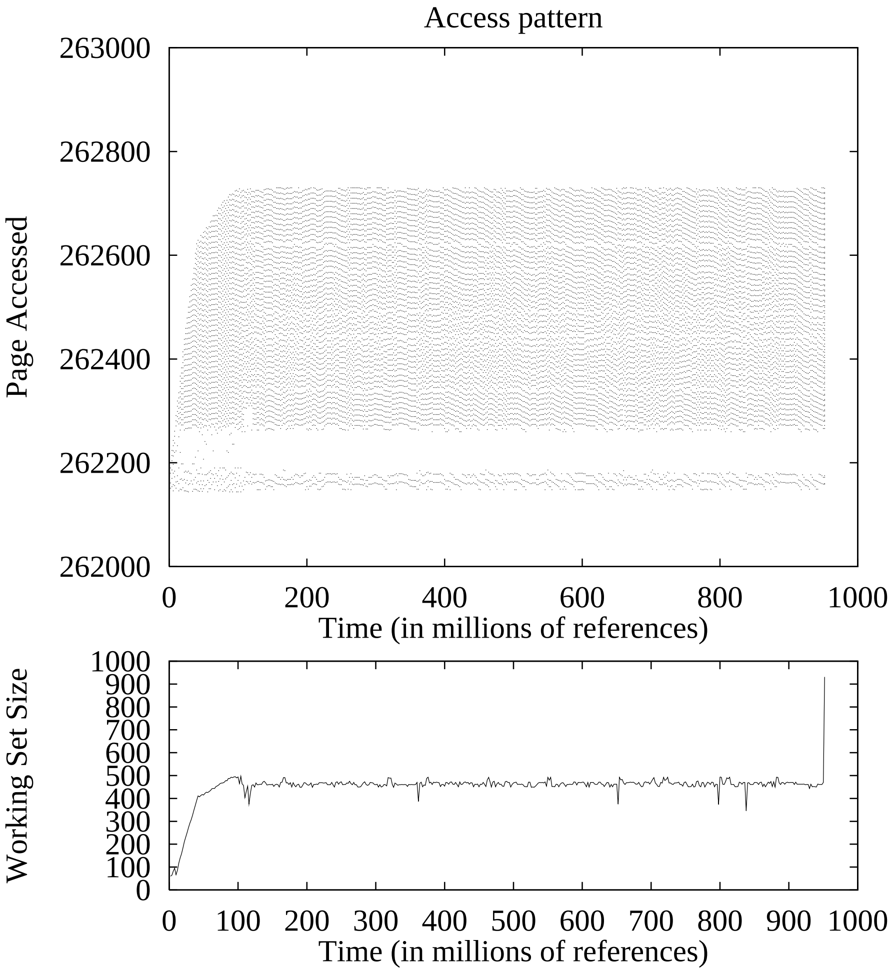


Figure 9: Access Pattern and working set size of ESIM

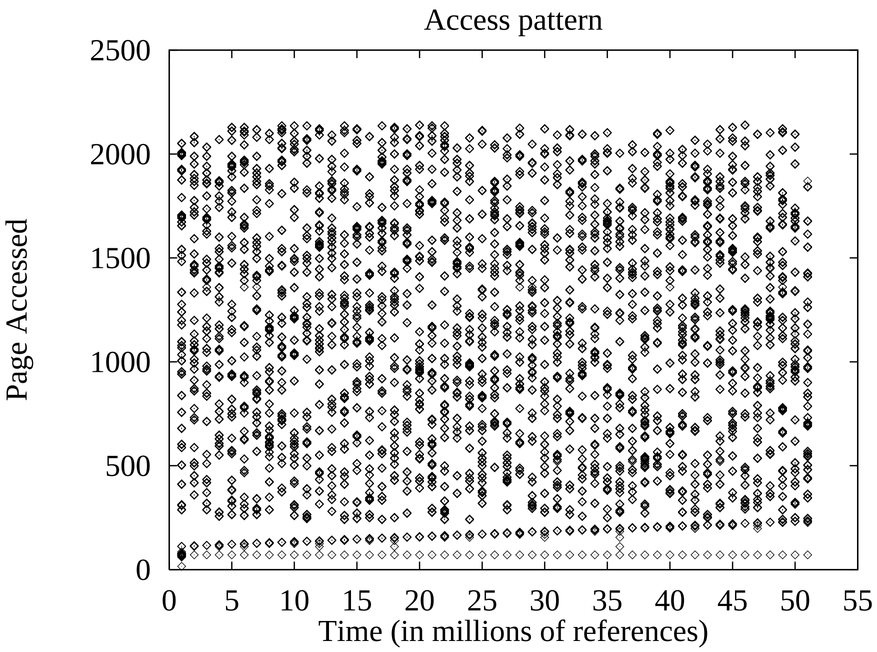


Figure 10: Access Pattern and working set size of OO7 T1

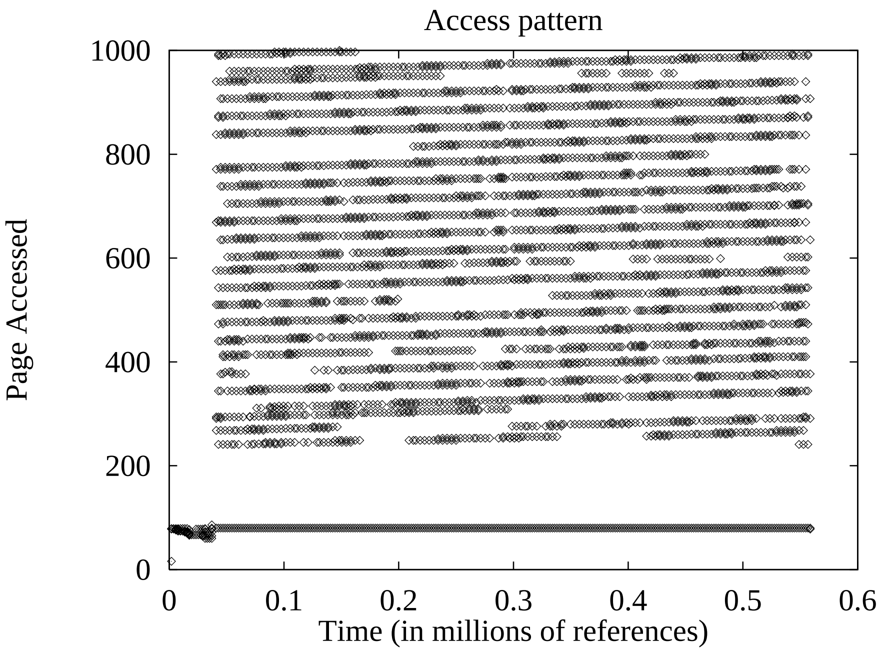


Figure 11: Access Pattern and working set size of OO7 Q7

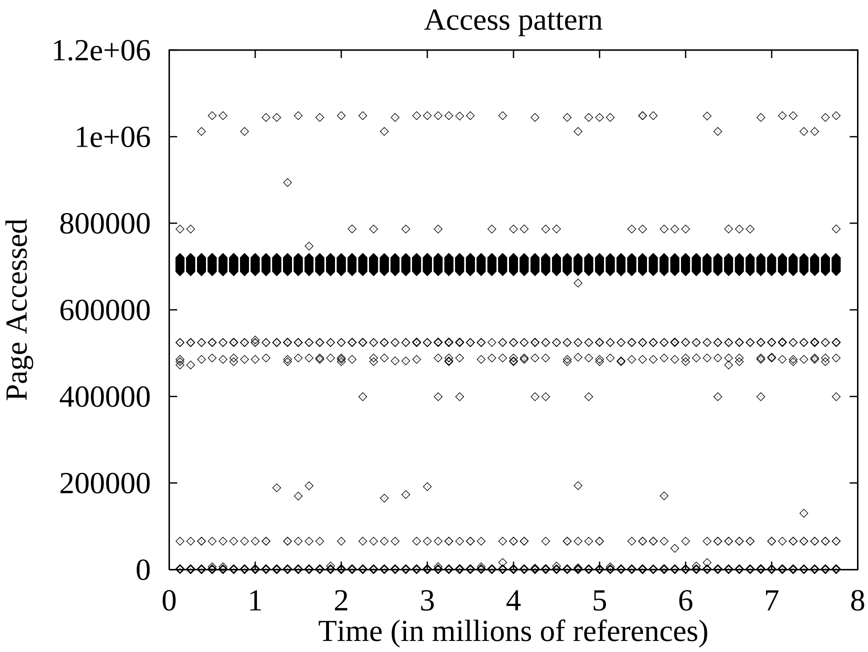


Figure 12: Access Pattern of SQL

- **Working set sizes may change rapidly**. Several of the applications depict abrupt changes in the size of their working set. For example, fig. 1 shows that most of the time the working set size of LU is around 600 pages; however, during short time intervals it may rise as high as 4000 pages - an order of magnitude increase. In most cases, between those abrupt working set changes, the application accesses a small number of (usually the same) pages. It is interesting to note that none of our examined applications depicts smooth changes between working sets. Applications move between different working sets so rapidly, that these moves appear as spikes in figures 1 to 5.

- **Applications that appear to have no regularity in the access patterns at all, may show very regular access patterns when viewed at the appropriate granularity**. For example, the N-BODY simulation seems to have little locality at first glance as figure 2 indicates. Although the N-BODY simulation has clear preference to two subsets of its pages (indicated by the two dark parallel stripes), there seems to be no regularity in accessing these pages. In search of any hidden regularity in the application, we "zoomed" in the access patterns as follows: we marked the accessed pages in intervals that are 128K references long, and plotted the results in figure 13(b). To allow the appropriate viewing granularity, figure 13(b) plots the access patterns only up to 100 million references. Dark rectangles and triangles that cover a part of the figure suggest that the application prefers a specific subset of its pages during certain time intervals. However, after the 50th million reference, the application appears to have no regular access pattern at all. To see if this is true, we "zoomed" in the access patterns once again as follows: we marked the accessed pages in intervals that are 1024 references long, and plotted the results in figure 13(c). We see that a periodic access pattern starts to emerge, with a period of about 25,000 references long. Similarly, GCC and GZIP seem to have no regularity in their access patterns. However, after appropriate zooming shown in figures 14(b), and 15(b), regular access patterns start to emerge. It can be easily seen now, that GCC and GZIP have clearly identifiable sequential (GCC) and periodic (GZIP) access patterns.

It seems that locality of reference and regularity in access patterns manifest themselves only when viewed at the appropriate distance.

- **Database applications may have more locality and regularity, from what can be seen at a first glance.** For example, the query OO7-Q7 shown in fig. 11 has very regular access pattern: it makes long sequential accesses to its data. On the other hand, the SQL application seems to have little regularity in its accesses, as shown in fig. 12 and 17(a), and as reported in [4]. To reveal any hidden regular access patterns that may reside in SQL, we renamed the page numbers as follows: the first page accessed by SQL was given the name 0, the second page to be accessed by SQL was given the name 1, etc. Figures 17(b) and 17(c) plot the page accesses of the SQL application to these re-numbered pages. We can easily see in figure 17(c) that SQL favors a specific subset of its pages, giving preference in accessing repeatedly the same clusters of pages. Simulation results show that the SQL trace has a significant amount of locality of reference - much more than what can be seen at a first glance [2]. Finally, the database benchmark OO7-T1 (fig. 10) appears to have no regular access pattern, but this should be expected, because this benchmark makes a random walk of the assembly hierarchy.

4 Discussion

The above observations can be useful to researchers, programmers, and practitioners that want to understand, visualize, and/or improve the performance of their applications.

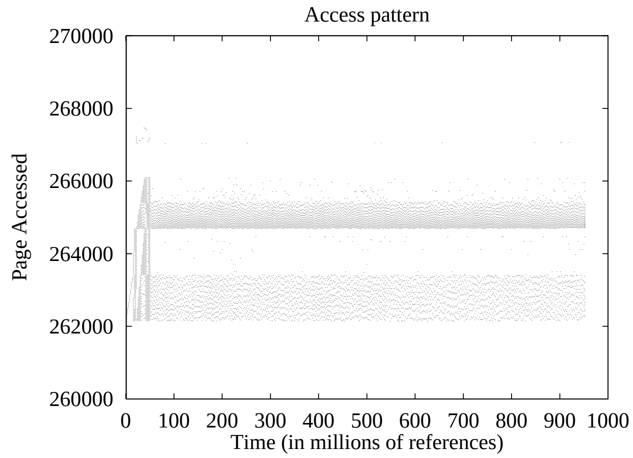
For example, the periodicity of page accesses and the long sequential accesses seen in several of the studied applications can be used by operating systems to implement policies that will **prefetch** pages before they are actually needed. Prefetching pages has shown to reduce the high latency observed in virtual memory systems [3]. The existence of regular access patterns makes prefetching more effective. The regularity of access patterns can also be used in disk **block placement** policies so that nearby-accessed pages be placed in nearby physical disk blocks. Similar observations can also be used by stripping policies in RAIDS and in the organization of segmented caches. The abrupt changes in working sets that were observed in most applications, can also be taken into account by page replacement algorithms.

5 Summary

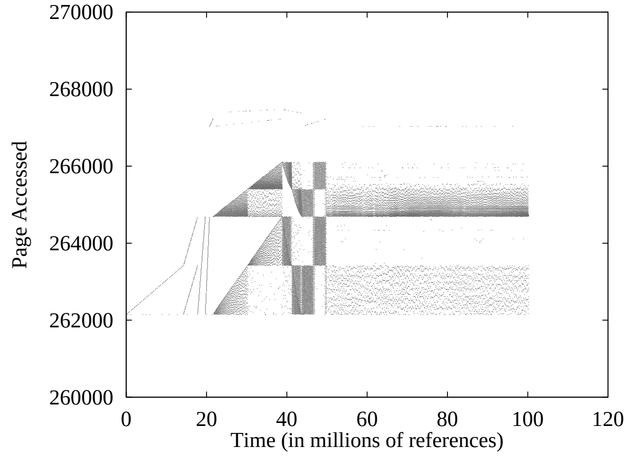
In this note we visualize the working set of several different applications, that are representative of various domains including numerical computations, databases, program development, and CAD simulation. We observe that most of the studied applications have regular and sometimes periodic access patterns. Even though some applications appear to have no regular access patterns at all, careful observation at the appropriate granularity reveal an impressively regular access pattern.

Acknowledgments

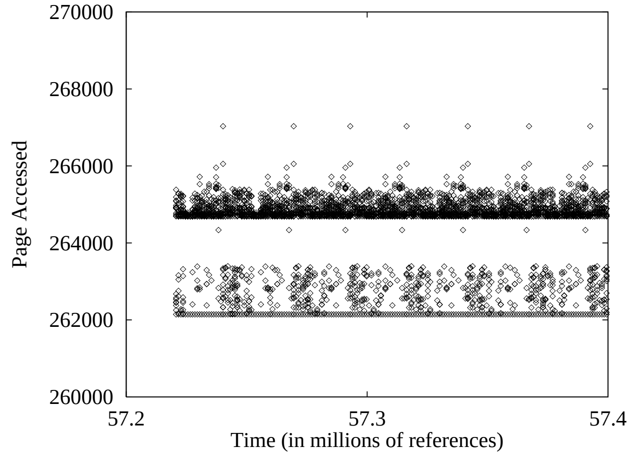
This work was supported in part by PENED project "Exploitation of idle memory in a workstation cluster" (2041 2270/1-2-95), and in part by the USENIX



(a)

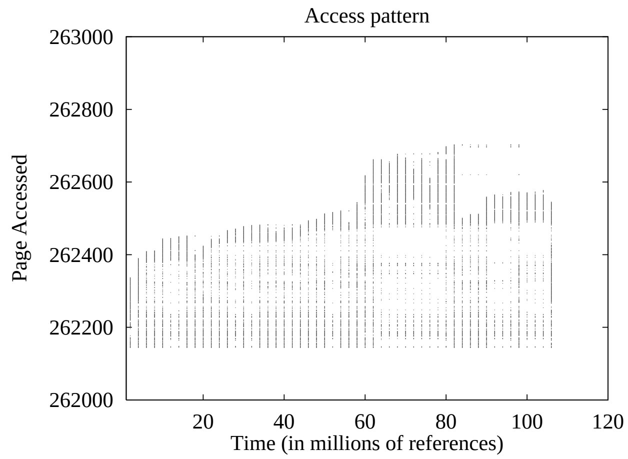


(b)

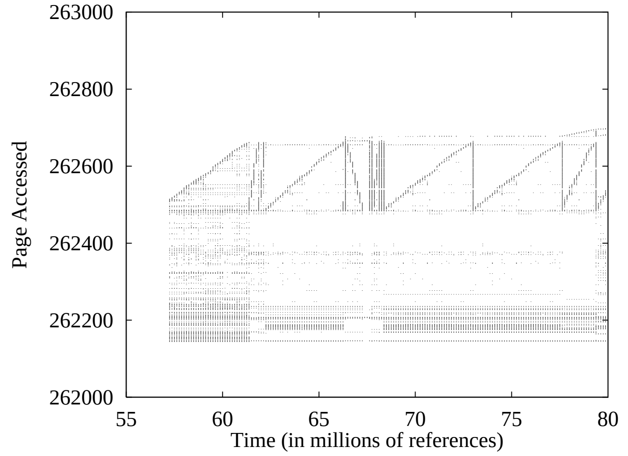


(c)

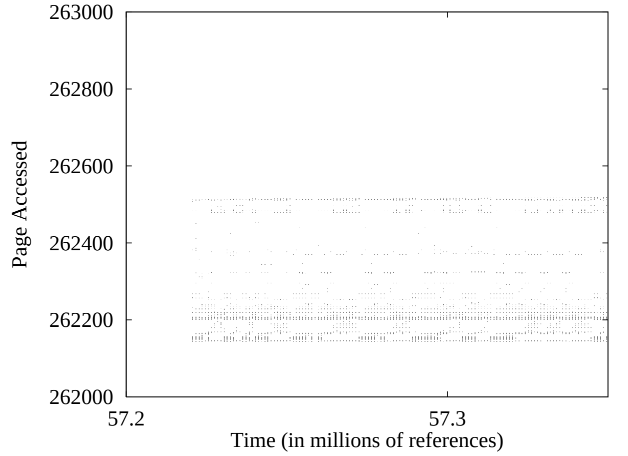
Figure 13: Access Patterns of N-BODY Simulation at three different zooming levels. We see that although the figure (a) suggests that there is little locality in the application, the rest two figures reveal surprisingly regular access patterns.



(a)

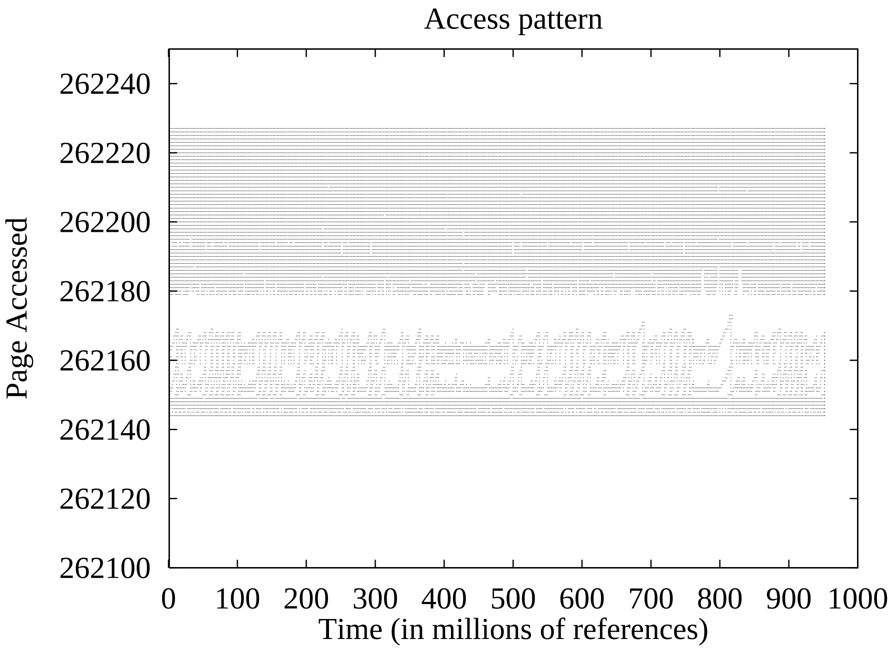


(b)

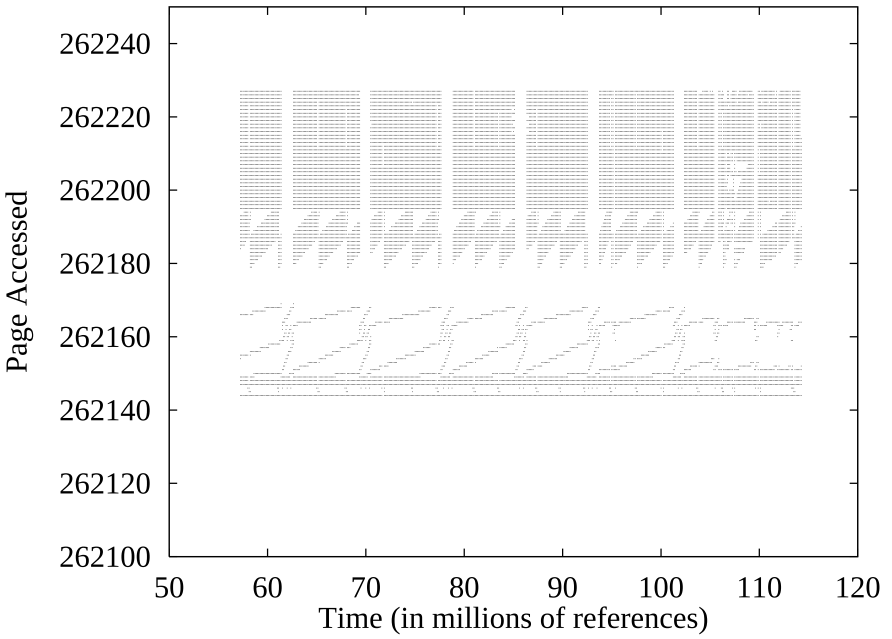


(c)

Figure 14: Access Patterns of GCC at three different zooming levels. We see that although the fig. (a) suggests little regularity in the access patterns of GCC, fig. (b) shows long sequential accesses which can be seen only at the appropriate zooming level.



(a)



(b)

Figure 15: Access Patterns of GZIP at two different zooming levels. We see that although the fig. (a) suggest that there is little regularity in the application, fig. (b) reveals surprisingly regular access patterns.

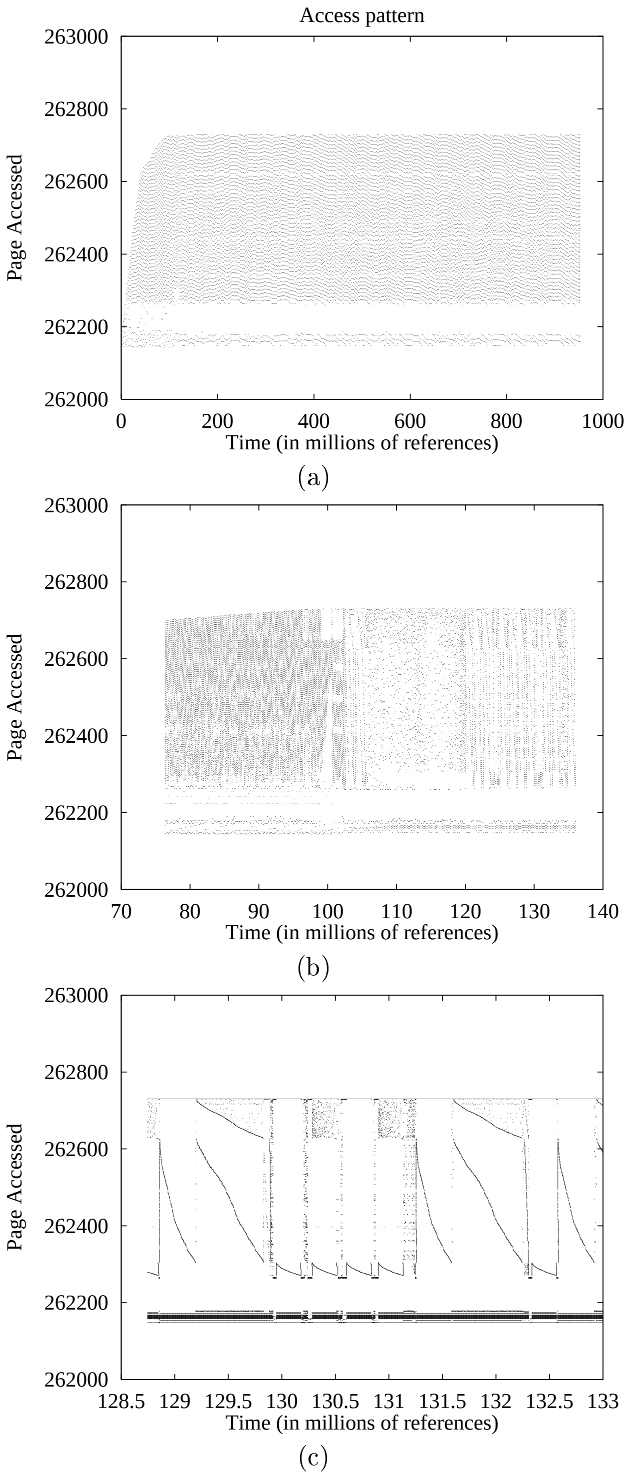
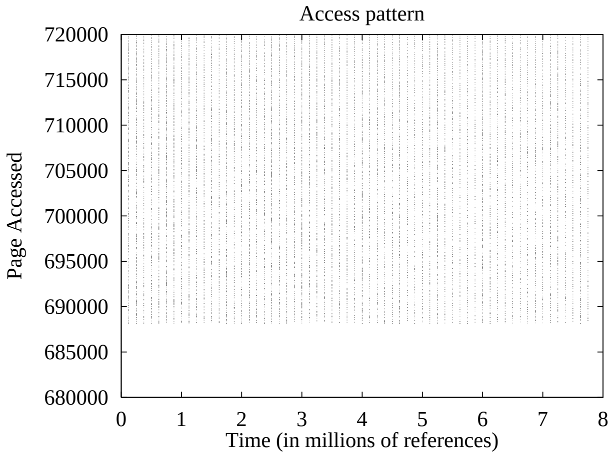
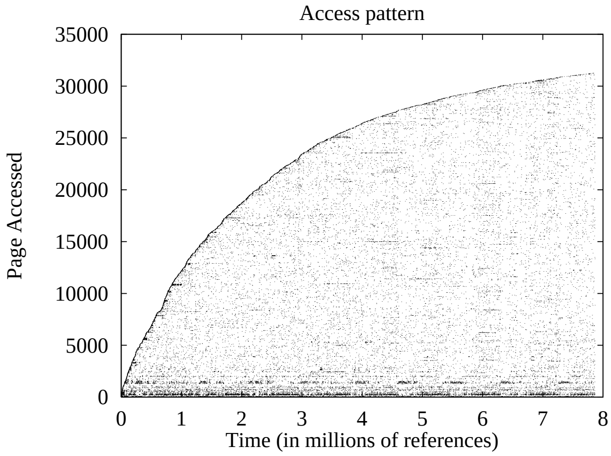


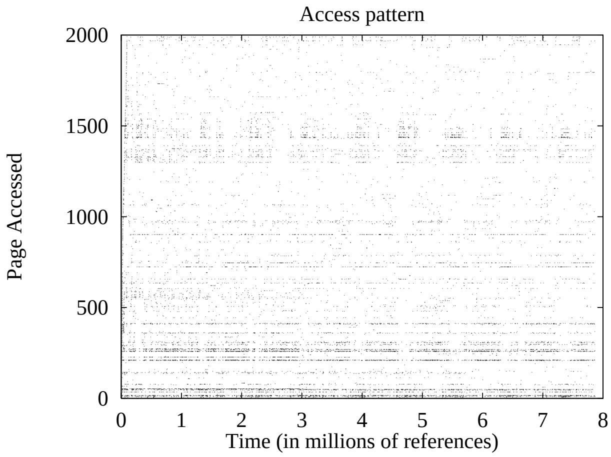
Figure 16: Access Patterns of ESIM at three different zooming levels. Figures (b) and (c) reveal regular and sequential access patterns.



(a)



(b)



(c)

Figure 17: Access Patterns of SQL at three different zooming levels. Figure (a) plots the virtual page accesses of the application. To reveal any hidden regular access patterns that may reside in SQL, we renamed the page numbers as follows: the first page to be accessed was assigned number 0, then second page to be accessed was assigned number 1, etc. Figures (b) and (c) show the (re-numbered) page accesses of the application. Figure (c) shows a significant amount of locality, and some repeatability in accessing some page clusters.

Association through project "Network RAMDISK". We deeply appreciate this financial support. Mike Feeley provided us the code for OO7, and DEC SRC provided us the SQL traces. Manolis Katevenis, Dionisis Pnevmatikatos and Catherine Chronaki provided useful feedback during earlier stages of this work. We thank them all.

References

[1] M. Carey, D. DeWitt, and J. Naughton. Ther OO7 Bechmark. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 12–21, 1993.

[2] S. Gribble and M. Kornacker. Memory Hierarchies and Real-World Applications, 1996. UC Berkeley.

[3] T.C. Mowry, A.K. Demke, and O. Krieger. Automatic Compiler-Inserted I/O Prefetching for Out-of-Core Applications. In *Second Symposium on Operating System Design and Implementation*, pages 3–17, Seattle, WA, October 1996.

[4] S.E. Perl and R.L. Sites. Studies of Windows NT Performance Using Dynamic Execution Traces. In *Second Symposium on Operating System Design and Implementation*, pages 169–184, October 1996.

[5] A. Srivastava and A. Eustace. ATOM: A System for Building Customized Program Analysis Tools. In *Proceedings of the SIGPLAN '94 Conference on Programming Language Design and Implementation*, pages 196–205, Orlando, FL, June 1994.