Edith Cowan University
CSG1102
Operating Systems
Assignment 1

Martin Ponce
Student 10371381

Tutor: Peter Hannay

August 24, 2015

Contents

1 Annotated Bibliography

1.1 Virtual memory managers

Blanchet, G., & Bertrand, D. (2012). Computer Architecture. Hoboken, NJ: Wiley.

Blanchet and Bertrand's (2012) book is aimed at readers wishing to learn about the essential architecture of a computer as a whole. However, chapter 9 (p. 175 - 204) provides insight into the functionality of virtual memory management. The authors introduce virtual management with a brief history, then explain the purpose of virtual memory. Advantages of virtual memory are listed, but on the other hand state that the cost of use is the decreased performance during the access of slower mass stoarge.

They explain how physical memory and a mass storage device interact to provide virtual memory through the use of paging, and briefly cover fetch and page fault algorithms. Page size considerations are also listed, and referred to as "conflicting criteria". The chapter also introduces the concept of multi-level paging.

Blanchet and Bertrand (2012) then conclude the chapter by provide a detailed and technical step-by-step view of virtual memory management at work, using a program execution as an example.

Blunden, B. (2003). Memory Management Algorithms and Implementation in C/C++. Plano, TX: Wordware.

Although Blunden's (2003) book is primarily for C/C++ programmers looking to implement their own memory management system, it provides an overview of operating system memory management in chapters 1 (p. 1 - 43) and 2 (p. 45 - 126), titled "Memory Management Mechanisms" and "Memory Management Policies", respectively.

Blunden introduces memory management in chapter 1 at the processor level, which provides functionality for segmentation and paging. While explaining memory hierarchy, he outlines the purpose of the L1 and L2 cache, and as with Blanchet and Bertrand (2012), states a similar disadvantage of virtual memory: Sacrifice performance for memory space. The author then identifies the differences between page frames and pages.

Chapter 2 compares virtual memory management and paging (or lack thereof) between four operating systems: MS-DOS, MMURTL, Linux, and Windows XP. Interestingly, MMURTL's use of paging was not related to virtual memory, but to provide memory protection and allocation.

While the text can be very technical at times (C/C++ code snippets), there is sufficient information in these two chapters relating to virtual memory management and paging for use in further research, particularly the comparisons between different operating systems.

Jacob, B., Ng, S. W., & Wang, D. T. (2008). Memory Systems - Cache, DRAM, Disk. Burlington, MA: Morgan Kaufmann Publishers.

Jacob, Ng, and Wang's (2008) book provides a complete description of memory systems, and dedicates a section to virtual memory in chapter 31 (p. 883 - 920).

Jacob et al. (2008) cite many academic sources throughout their book, and choose to express pseudo-code rather than code specific to a programming language (where applicable) to provide ease of use.

Jacob et al. (2008) begin the chapter by describing a brief history of virtual memory management, and then the technique of combining the cache, main memory and disk to provide virtual memory to a computer system. They point out the advantages of virtual memory, however, unlike Blanchet and Bertrand (2012) or Blunden (2003), they fail to mention any disadvantages.

This chapter defines the relationship between address spaces and main memory cache. They explain the various designs of main memory cache which dictate how a virtual page is mapped to the main memory cache. Jacob et al. (2008) cite various sources attempting to improve Translation Lookaside Buffer lookup times.

The authors then describe the functionality of page tables and the information they must store to enable the operating system to perform paging. Jacob et al. (2008) then compare hierarchical and inverted page tables structures and their corresponding algorithms to manage page tables.

This book functions well as a dependent source for research, with many academic citations, clear explanations and diagrams for visual representation of concepts.

1.2 Paging / swap

Babaoglu, Ö., & Joy, W. (1981). Converting a swap-based system to do paging in an architecture lacking page-referenced bits. In SOSP '81 Proceedings of the eighth ACM symposium on Operating systems principles (Vol. 15, pp. 78–86). New York. doi:10.1145/800216.806595

Babaoglu and Joy (1981) describe the challenges and their methods converting UNIX from a segmented, swap-based system to a paging system, in a computer with architecture that does not support page-referenced bits. The authors explain the use of page-referenced bits as essential to page replacement algorithms such as clock page replacement and sampled working set (SWS). They compare various page replacement algorithms, and set out to implement their variation of clock page replacement in UNIX by simulating page-referenced bits through software.

Babaoglu and Joy justify their design and optimization decisions by comparing the performance of the clock page replacement, and identify opportunities for improvement, based on the given hardware and operating system. For instance, Babaoglu and Joy (1981, p. 80) state the original algorithm only seeks to replace a single page when triggered by a page fault, and through testing, found examples where page requests spiked due to UNIX's non-uniform operations. They modified the algorithm by implementing a free page pool containing page frames not currently in the clock loop, and set a minimum free page pool size as a threshold. When this threshold is reached, clock page replacement is triggered and pages are replaced until the free page pool size reaches the threshold again. As the free page pool size decreases, the scan rate of the clock page replacement implementation increases until it reaches a maximum scan rate, which is "determined by the time it takes to simulate the setting of a referenced bit" (Babaoglu & Joy, 1981, p. 80).

The authors compared their clock paging system to the original swap-based system and present their findings with graphs comparing performance between the two. They found that under lower load levels, their clock page implementation out performed the swap-based method. However, under heavy load, while clock page replacement exhibited much lower page traffic, the overhead required to support it was higher than the swap-based method. The authors counter this finding by stating that the CPU utilization was greater for the clock paging system compared to the swap-based system.

Babaoglu and Joy conclude the article with a recommended requirement when designing a page replacement algorithm for architecture with no support for page-referenced bits, the results of their comparison of the two memory management algorithms, and limitations of their global clock replacement algorithm.

Denning, P. J. (1967). The Working Set Model for Program Behavior. In SOSP '67 Proceedings of the first ACM symposium on Operating System Principles (pp. 15.1–15.12). New York: ACM.

Although the working set is neither a virtual memory manager or paging algorithm itself, it is essential to the efficiency of virtual memory (Silberschatz, Galvin, & Gagne, 2013). Denning's (1967) proposal is a landmark article in the field of computer science. According to google scholar, Denning's paper has been cited 1076 times.

This source introduces the concept of the working set model, which provides the ability for a system to determine what information is being used, or not being used by a program during execution. This ability allows a computer to make better decisions in allocating or deallocating resources to that program.

At the time, the user and the compiler were commonly proposed to be used as input for dynamic memory allocation. Denning (1967, p. 15.1) claims that neither sources are adequate. He argues that the user cannot possibly provide reliable estimates of resource requirements of his or her program. Additionally, due to the nature of modularised programming, it is possible that the compiler may not be able to decide which modules are required until run time, and therefore not be able to approximate the required resources appropriately.

Denning (1967) explores the current body of work in memory management strategies and states that no paging algorithms have been proposed which involve anticipating resource requirements. The author attributes the cause to the lack of reliable information that the system can use to judge allocation requirements, and returns to the point of the user and compiler as inputs.

Denning (1967, p. 15.2) proposes “new mechanisms” to monitor behaviours of programs, and allocate resources based on those observations. The proposal begins by defining the current environment, briefly explaining the process of paging and what occurs during a page fault. He defines the “core memory management” problem as deciding which pages remain resident in main memory. Denning (1967, p. 15.3) offers a strategy - to “minimize page traffic” to reduce overhead required during page swaps, and reduce processing time, which affects processor efficiency if processing pages takes too long. Thus, Denning (1967, p. 15.3) defines the working set as “the minimum collection of pages that must be loaded in main memory for a process to operate efficiently, without ‘unnecessary’ page faults”. A more explicit definition is provided as “the set of its pages a process has referenced within the last τ seconds of its execution” (Denning, 1967, p. 15.4).

He provides hardware and software implementations of the working set model, and justifies the proposal through its practicality in resource allocation for both processor and memory demand strategies.

Denning (1967) concludes the paper with brief comparisons of current allocation strategies, the ideologies of the working set model and iterates the various parameters and properties defined in the implementation the proposal.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2013). Operating System Concepts Essentials (2nd ed.). Hoboken, NJ: Wiley.

Silberschatz et al.’s (2013) book provides overview of operating systems, and contains a chapter describing virtual memory (p. 371 - 438). The authors present a brief history of virtual memory management and explain the organisation of the virtual address space.

The chapter explains the concept of paging, while exploring the functionality of demand paging and page faults. The book offers several mathematical equations to calculate demand paging efficiency and provides considerably more detail of page replacement and frame allocation algorithms compared to other books in this annotated bibliography. Silberschatz et al. (2013) defines thrashing and the working set model. The chapter concludes by comparing the implementation of virtual memory between two operating systems, Windows and Solaris.

Similar to Jacob et al. (2008), this book provides clear and concise explanations and diagrams to visually represent concepts.

References

Babaoglu, O., & Joy, W. (1981). Converting a swap-based system to do paging in an architecture lacking page-referenced bits. In *Sosp '81 proceedings of the eighth acm symposium on operating systems principles* (Vol. 15, pp. 78–86). New York. doi: 10.1145/800216.806595

Blanchet, G., & Bertrand, D. (2012). *Computer Architecture*. Hoboken, NJ: Wiley.

Blunden, B. (2003). *Memory Management Algorithms and Implementation in C/C++*. Plano, TX: Wordware.

Denning, P. J. (1967). The Working Set Model for Program Behavior. In *Sosp '67 proceedings of the first acm symposium on operating system principles* (pp. 15.1–15.12). New York: ACM.

Jacob, B., Ng, S. W., & Wang, D. T. (2008). *Memory Systems - Cache, DRAM, Disk*. Burlington, MA: Morgan Kaufmann Publishers.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2013). *Operating System Concepts Essentials* (2nd ed.). Hoboken, NJ: Wiley.