Automated Thin Object Management System

Master's Thesis Proposal

Huffman, Michael J.
Advisor: Dekhtyar, Alexander

23 September 2008
California Polytechnic State University, San Luis Obispo
1 Grand Ave. San Luis Obispo, CA 93407
mhuffman@calpoly.edu

Keywords: Orthogonal persistence, memory management, virtual memory, garbage collector, buffer management, database

Abstract

Memory intensive applications, constrained by fixed-size or low memory environments, can combat this limitation by leveraging persistent disk-based storage systems (most notably relational databases). Others, however, may incur significant penalties at runtime as main memory overflows and the use of extended virtual memory leads to thrashing. Worse yet, some systems may choose a database solution while not necessarily requiring the benefits of a Database Management System, such as querying or transaction management.

Object-relational mapping tools, such as the Hibernate, ease the persistence and retrieval of data into and from a relational database. However, such tools are ill suited to systems using a database for relieving main memory. Manually managing the offloading of main memory objects to disk is tedious, error prone, and unmaintainable. A new classification of objects is proposed–Thin Objects; that is objects whose data is partially stored in main memory and on disk. Thin Objects are divided into critical (thick) and non-critical (thin) portions, allowing critical pieces of data to remain resident in main memory while non-critical data is stored on disk.

The Automated Thin Object Management System (ATOMS) is proposed as a means for transparently storing and retrieving data from the non-critical region of Thin Objects in order to reduce the main memory footprint of a running application. Designed for the Java programming language, ATOMS utilizes the garbage collection facility of the Java Virtual Machine as an optimal page replacement algorithm in a variable sized buffer–the heap. Paralleling the buffer management component in Database Management Systems [11, 15], ATOMS monitors changes in non-critical data values so that those lacking reachability beyond the scope of their defining object are persisted to disk.

Using aspect-oriented programming principles, ATOMS maintains the obliviousness [18] quality while maintaining the utmost flexibility, reusability, and configurability. Adhering to the principles of orthogonal persistence [3] within persistent programming languages, Thin Objects reduce the complexity and increase maintainability by separating this persistence functionality as a cross-cutting concern. ATOMS supports the use of any persistent data storage component; however, a reference implementation includes one relational database and one non-relational database.

Related Work

The proposed solution closely relates to three separate areas of research: (1) buffer management (2) memory management and (3) orthogonal persistence. Buffer management has been studied intensely over the recent decades beginning with [15] and [11]. The buffer manager is the component of most Database Management Systems that is responsible for interacting with the disk to perform physical I/O operations. The buffer manager manages a set (usually static in size) of buffers in main memory that serve as a cache for physical disk access. Much research has been completed in this area and is beyond the scope of this proposal. See [7, 8, 11, 15–17, 24, 27] for further reference.

Memory management, since the separation of main memory from disk memory, has been studied vigorously throughout much of the history of modern computing [12]. Of particular significance is two key areas of research: (1) virtual memory management and (2) garbage collection. Virtual memory, the decoupling of physical addresses from logical addresses, is paramount in the buffer management systems previously noted. [12] provides a thorough, yet outdated survey of the research in this field.

Much attention in the recent two decades has been given to garbage collection algorithms in virtual-machine based languages, used by high-level languages such as Java. Garbage collection was devised as a means of freeing the developer from the constraints, complexity, and difficulties of manual memory management. [9] has proposed a garbage collector that performs heap compression to reduce the memory footprint of Java applications. [20] addresses the fundamental of many garbage collection algorithms–objects that lose reachability are no longer usable. The authors contend that reachability is not synonymous with usability. They consequently propose an algorithmic extension to the Java garbage collector that automatically detects loitering objects in main memory, i.e. those considered idle or obsolete. Further research related to garbage collection can be found in [6, 9, 10, 13, 21, 22, 28, 30–32]. Related to garbage collection, the use of weak reference objects within the Java language is detailed in [1, 14, 19].

Lastly, the concept of orthogonal persistence was introduced in [5]. Otherwise known as the Orthogonal Persistence Hypothesis [4], the goal of orthogonal persistence is "allowing programs to be expressed independently of the longevity of the data they manipulate, and conversely, so that the longevity of an object is independent of the way it is manipulated" [23]. Three aspect-oriented implementations have been developed in effort to support this concept [25, 26, 29]. [2] identifies several flaws with these approaches with respect to type orthogonality, persistence independence and transitivity, suggesting that persisting containers and path expression pointcuts overcome these flaws.

References

[1] Agesen, O., and Garthwaite, A. Efficient object sampling via weak references. *SIGPLAN Not. 36*, 1 (2001), 121–126.

[2] Al-Mansari, M., Hanenberg, S., and Unland, R. Orthogonal persistence and aop: a balancing act. In *ACP4IS '07: Proceedings of the 6th workshop on Aspects, components, and patterns for infrastructure software* (New York, NY, USA, 2007), ACM, p. 2.

[3] Atkinson, M., and Morrison, R. Orthogonally persistent object systems. *The VLDB Journal 4*, 3 (1995), 319–402.

[4] Atkinson, M. P. Persistence and java - a balancing act. In *Proceedings of the International Symposium on Objects and Databases* (London, UK, 2001), Springer-Verlag, pp. 1–31.

[5] Atkinson, M. P., Bailey, P., Chisholm, K. J., Cockshott, P. W., and R.Morrison. An approach to persistent programming. *The Computer Journal 26*, 4 (Nov 1983), 360–365.

[6] Brecht, T., Arjomandi, E., Li, C., and Pham, H. Controlling garbage collection and heap growth to reduce the execution time of java applications. *ACM Trans. Program. Lang. Syst. 28*, 5 (2006), 908–941.

[7] Cai, F. F., Hull, M. E. C., and Bell, D. A. Buffer management for high performance database systems. In *HPC-ASIA '97: Proceedings of the High-Performance Computing on the Information Superhighway* (Washington, DC, USA, 1997), IEEE Computer Society, p. 633.

[8] Casas, I. R., and Sevcik, K. C. A buffer management model for use in predicting overall database system performance. In *Proceedings of the Fifth International Conference on Data Engineering* (Washington, DC, USA, Feb 1989), IEEE Computer Society, pp. 463–469.

[9] Chen, G., Kandemir, M., Vijaykrishnan, N., Irwin, M. J., Mathiske, B., and Wolczko, M. Heap compression for memory-constrained java environments. In *OOPSLA '03: Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications* (New York, NY, USA, 2003), ACM, pp. 282–301.

[10] Chin, W.-N., Craciun, F., Qin, S., and Rinard, M. Region inference for an object-oriented language. *SIGPLAN Not. 39*, 6 (2004), 243–254.

[11] Chou, H.-T., and DeWitt, D. J. An evaluation of buffer management strategies for relational database systems. In *Proceedings of the 11th international conference on Very Large Data Bases* (1985), VLDB Endowment, pp. 127–141.

[12] Denning, P. J. Virtual memory. *ACM Comput. Surv. 2*, 3 (1970), 153–189.

[13] Domani, T., Kolodner, E. K., Lewis, E., Salant, E. E., Barabash, K., Lahan, I., Levanoni, Y., Petrank, E., and Yanorer, I. Implementing an on-the-fly garbage collector for java. *SIGPLAN Not. 36*, 1 (2001), 155–166.

[14] Donnelly, K., Hallett, J. J., and Kfoury, A. Formal semantics of weak references. In *ISMM '06: Proceedings of the 5th international symposium on Memory management* (New York, NY, USA, 2006), ACM, pp. 126–137.

[15] Effelsberg, W., and Haerder, T. Principles of database buffer management. *ACM Trans. Database Syst. 9*, 4 (1984), 560–595.

[16] Faloutsos, C., Ng, R., and Sellis, T. Flexible and adaptable buffer management techniques for database management systems. *IEEE Trans. Comput. 44*, 4 (Apr 1995), 546–560.

[17] Feng, L., Lu, H., and Wong, A. A study of database buffer management approaches: towards the development of a data mining based strategy. *IEEE International Conference on Systems, Man, and Cybernetics 3* (Oct 1998), 2715–2719.

[18] Filman, R. E., and Friedman, D. P. Aspect-oriented programming is quantification and obliviousness. In *Aspect-Oriented Software Development*, R. E. Filman, T. CElrad, S. Clarke, and M. Aksit, Eds. Addison-Wesley, Boston, MA, 2005, pp. 21–35.

[19] Gabay, Y., and Kfoury, A. J. A calculus for java's reference objects. *SIGPLAN Not. 42*, 8 (2007), 9–17.

[20] Goldstein, M., Shehory, O., and Weinsberg, Y. Can self-healing software cope with loitering? In *SOQUA '07: Fourth international workshop on Software quality assurance* (New York, NY, USA, 2007), ACM, pp. 1–8.

[21] Hertz, M., and Berger, E. D. Quantifying the performance of garbage collection vs. explicit memory management. In *OOPSLA '05: Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications* (New York, NY, USA, 2005), ACM, pp. 313–326.

[22] Hirzel, M. Data layouts for object-oriented programs. *SIGMETRICS Perform. Eval. Rev. 35*, 1 (2007), 265–276.

[23] Hosking, A. L. Main memory management for persistence. In *The OOPSLA '91 Workshop on Garbage Collection* (1991).

[24] Levy, H., Messinger, T., and Morris, R. The cache assignment problem and its application to database buffer management. *IEEE Trans. Software Eng. 22*, 11 (Nov 1996), 827–838.

[25] Pawlak, R., Seinturier, L., Duchien, L., Florin, G., Legond-Aubry, F., and Martelli, L. Jac: an aspect-based distributed dynamic framework. *Softw. Pract. Exper. 34*, 12 (2004), 1119–1148.

[26] Rashid, A., and Chitchyan, R. Persistence as an aspect. In *AOSD '03: Proceedings of the 2nd international conference on Aspect-oriented software development* (New York, NY, USA, 2003), ACM, pp. 120–129.

[27] Sacco, G. M., and Schkolnick, M. Buffer management in relational database systems. *ACM Trans. Database Syst. 11*, 4 (1986), 473–498.

[28] Shuf, Y., Gupta, M., Bordawekar, R., and Singh, J. P. Exploiting prolific types for memory management and optimizations. In *POPL '02: Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (New York, NY, USA, 2002), ACM, pp. 295–306.

[29] Soares, S., Laureano, E., and Borba, P. Implementing distribution and persistence aspects with aspectj. In *OOPSLA '02: Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (New York, NY, USA, 2002), ACM, pp. 174–190.

[30] Vardhan, A., and Agha, G. Using passive object garbage collection algorithms for garbage collection of active objects. In *ISMM '02: Proceedings of the 3rd international symposium on Memory management* (New York, NY, USA, 2002), ACM, pp. 106–113.

[31] Yang, T., Berger, E. D., Kaplan, S. F., and Moss, J. E. B. Cramm: virtual memory support for garbage-collected applications. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation* (Berkeley, CA, USA, 2006), USENIX Association, pp. 103–116.

[32] Yu, Z. C. H., Lau, F. C. M., and Wang, C.-L. Object co-location and memory reuse for java programs. *ACM Trans. Archit. Code Optim. 4*, 4 (2008), 1–36.