Intensify the I/O Performance of OODBS by Collaboration between Clustering and Buffer Replacement

Dheeraj Chooramani[1,*] and Dr. D.K. Pandey[2]
[1,*]Research Scholar, Department of Computer Science, JJTU Rajasthan
[2] Director, Dr. Pandey Professional College Ghaziabad

There are different techniques for improving I/O performance of Object oriented database Management Systems (OODBMS). Over 15 years of research into OODBMS design, performance remains as one of the major problems. I/O reduction has proven to be one of the most effective ways enhancing performance. The two main techniques of improving I/O performance of Object Oriented Database Management Systems (OODBMS) are clustering and buffer replacement. Clustering is the placement of objects accessed near to each other in time into the same page. Buffer replacement involves the selection of a page to be evicted, when the buffer is full. The page evicted ideally should be the page needed least in the future. These two techniques both influence the likelihood of a requested object being memory resident. We believe an effective way of reducing disk I/O is to take advantage of the synergy that exists between clustering, and buffer replacement. Hence, we design a framework, whereby clustering algorithms incorporating buffer replacement cache behaviour can be conveniently employed for enhancing the I/O performance of OODBMS. We call this new type of clustering algorithm, Cache Conversant Clustering (C3). In this paper, we present the C3 framework, and a C3 algorithm that we have developed, namely C3-GGP Greedy Graph Partioning. We have tested C3-GGP against three well known clustering algorithms. The results show that C3-GGP out performs them by up to 42% when using popular buffer replacement algorithms such as LRU,FCFS and CLOCK. C3-GGP offers the same performance as the best existing clustering algorithm when the buffer size compared to the database size is very small.

Keywords: Object-oriented databases, clustering, buffer replacement, caching, database optimization.

1. INTRODUCTION

The current rate of performance improvement for CPUs is much higher than that for memory or disk I/O. CPU performance doubles every 18 months while disk I/O improves at only 5-8 % per year. In addition, cheap disks mean object databases will become bigger as database designers realize that more data can be stored [1]. A consequence of these facts is that disk I/O is likely to be a bottleneck in an increasing number of database applications. It should also be noted, memory is also becoming a more prevalent source of bottleneck on modern DBMS [2]. However their study was conducted on relational DBMS. We believe for object-oriented DBMS where navigation is common, I/O may be a more common source of bottleneck.

We believe the best way of reducing disk I/O is taking advantage of the synergy that exists between the optimization techniques, 'clustering', and 'buffer replacement'. Clustering is the arrangement of objects into pages so that objects accessed close to each other temporally are placed into the same page. This in turn reduces the total I/O generated. 'Buffer replacement' involves the selection of a page to be evicted, when the buffer is full. The page evicted ideally should be the page needed furthest in the future. Selection of the correct page for eviction results in a reduction in the total I/O generated by the system. This paper will show that synergy does indeed exist between the two techniques. In addition, exploitation of that synergy results in improvements in performance.

Traditionally static clustering algorithms have generally been designed to place objects likely to be co-referenced into the same page [3, 4, 5, 6, 7, 8, 9, 10]. On the surface this seems like a reasonable approach, since it confines object graph traversals as much as possible to one page. By minimizing the likelihood of traversing out of the current page, the chances of requiring a disk load are minimized. However, upon closer inspection we can criticize this approach as being too conservative. This is because the assumption that navigations out of the current page have a high probability of causing a page load is only valid when either the cache size is one page or the cache size is larger but the buffer replacement algorithm only keeps pages cache resident for very short durations. This papers aims to demonstrate simple synergistic modifications to existing algorithms can result in improved performance. To this end we exploit our knowledge of buffer replacement algorithm behaviour to design static clustering algorithms that tolerate navigations out of the current page so long as the navigation proceeds into another cache resident page. We term this approach cache conversant clustering (C3) [11]. In order to make our approach more generally applicable we created the C3 framework. The C3 framework produces a 'family' of static clustering algorithms that all possess the property of cache conversant.

Experimental results show a C3 algorithm called C3-GGP outperforms the three existing static clustering algorithms, probability ranking principle algorithm (PRP) [3], greedy graph partitioning (GGP) [5], and Wisconsin greedy graph partitioning (WGGP) [3] in a variety of situations. We believe an effective way of reducing disk I/O is to take advantage of the synergy that exists between 'clustering', and 'buffer replacement'. Hence, we design a framework, whereby clustering algorithms incorporating buffer replacement behaviour can be conveniently employed for enhancing the I/O performance of OODBMS. We call this new type of clustering algorithm, "Cache Conversant Clustering" (C3).

In this paper, we present the C3 framework [11], and a C3 algorithm, namely C3-GGP. We have tested C3-GGP against three well known clustering algorithms. The main contribution of this paper is the development of the C3 framework that incorporates buffer replacement behaviour into clustering and the simulation to include comparisons with two additional clustering algorithms (WGGP, PRP); and three more simulation results, which allow us to gain a better understanding of the performance trade-offs of the proposed algorithm. Cache Conversant clustering algorithms can be developing using this framework. The work in this paper makes the following assumptions, the entire database can be shut off for rearrangement to take place, Patterns of object access between rearrangements bear some degree of similarity. All objects are smaller than one

page in size. Since large objects (larger than one page in size) do not benefit from clustering, we choose to focus our study on objects smaller than a page in size. However, the techniques in this paper can still be applied when large objects are present. For example, large objects can be placed in a separate area of the object store and dynamic clustering algorithms can ignore them, Objects are moved and mapped from one consistent state to another.

There has been a number of existing studies on static clustering algorithms [3, 5]. The simplest static algorithm is the probability ranking principle algorithm [3].Probability Ranking principle just involves placing the objects in the object graph in decreasing heat. This simple approach groups objects of similar heat into the same page. When the buffer is large and the working set of the database completely fits into memory this algorithm provides the optimal solution.

Graph partitioning clustering algorithms consider the object placement probable as a graph partitioning problem in which the min-cut criteria is to be satisfied for page boundaries. The edges of the graph are weighted using tension. A large range of buffer replacement algorithms were used in this study and they include least recently used, First In First out, Least frequently used, G-CLOCK [12], LRU-K[13], Belady optimal buffer replacement algorithm[14].

2. METHODOLOGY

2.1. Simulation Setup

The simulations are conducted using the Object Clustering Benchmark (OCB) [15] and the Virtual Object Oriented Database simulator, (VOODB) [16]. VOODB is based on a generic discrete-event simulation framework. Its purpose is to allow performance evaluations of OODBMs in general, and optimizing methods like clustering in particular. OCB is designed to benchmark OODBM systems and clustering polices in particular. The OCB database has a variety of parameters which make it very user-tunable. A database is generated by setting parameters such as total number of objects, maximum number of references per class, base instance size, number of classes, etc... Once these parameters are set, a database conforming to these parameters is randomly generated. The database consists of objects of varying sizes. In the simulations conducted in this paper the objects varied in size from 50 to 1200 bytes and the average object size was 268 bytes. A total of 20, 000 objects are used, resulting in a database.

The parameters of OCB and VOODB used to conduct the simulations in this paper are specified in Table 1(a) and Table 1(b). VOODB parameters involving time have been omitted from this table, since the results reported are in terms of I/O performance.

Table 1 (a): OCB Database Parameters.

Parameter Description	Value
number of classes in the database	50
maximum number of references, per class	10
instances base size, per class	50
total number of objects	10000
number of reference types	4
reference types random distribution	Uniform
class reference random distribution	Uniform
objects in classes random distribution	Uniform
Objects references random distribution	Uniform

Table 1 (b): VOODB Parameters.

Parameter	Value
System class	Centralized
Disk page size	4096 bytes
Buffer size T	Varies
Buffer replacement policy	LRU
Pre-fetch strategy	None
Multiprogramming level	1
Object initial placement	sequential

The results are generated via three steps. The first *training* step runs the database and collects statistical data of object access. The second *clustering* step uses the training data with the clustering algorithm to rearrange objects. The third *evaluation* step measures I/O generated from running the workload on the newly clustered database.

In this paper we compare the performance of the C3 algorithm C3-GGP with three existing static clustering algorithms. The three existing static clustering algorithms are the probability ranking principle algorithm (PRP) [3], greedy graph partitioning (GGP) [5], and Wisconsin greedy graph partitioning WGGP [3]. The reason for choosing these algorithms is they all use the SMC clustering graph. The Simple Markov Chain (SMC) model clustering graph algorithms has been shown by [3] to give best general performance.

In the simulations we set C3-GGP's hot region size parameter to 90% of main memory (with the exception of one simulation in which we investigate the effect of varying hot region size of C3-GGP). This is because we found C3-GGP performs best when we set its hot region size parameter to 90%.

The OCB workload used in this study included simple traversals, hierarchical traversals and stochastic traversals [15]. The depths of the traversals are 2, 4, and 6 respectively. 10000 transactions are run for every result reported in this paper. Each transaction involved execution of one of the three traversals.

We introduced skew into the way in which roots of traversals are selected. Roots are broken up into hot and cold regions. In all simulations the hot region was set to 2 % size of database and had a 98 % probability of access (i.e. there was a 98 % probability that the root of a traversal is from the hot region). However the number of hot objects generated by this pattern of access is greater than 2 % of database size since each traversal access more objects than just the root. These settings are chosen to represent typical database application behavior Gray [17] cites statistics from a real videotext application in which 3% of the records got 80% of the references. Carey [18] use a hot region size of 4% in the HOTCOLD workload have been used to measure data caching trade-offs in client/server OODBMSs. Franklin [19] use a hot region size of 2% in the HOTCOLD workload used to measure the effects of local disk caching for client/server OODBMSs.

The evaluation metric used is total I/O. In the simulations total I/O equals the total transaction read I/O. The reason for using total I/O instead of WSS as our evaluation metric is that ultimately we are interested in how well the algorithms can reduce total I/O. In this paper the WSS metric is only used as a guide to explain the intuitions that led to the design of our algorithms.

3. SIMULATION RESULTS

Here we report the results of simulations comparing the performance of three existing highly competitive static clustering algorithms (PRP, WGGP, GGP) with the C3 produced C3-GGP algorithm.

3.1. Varying Buffer Size

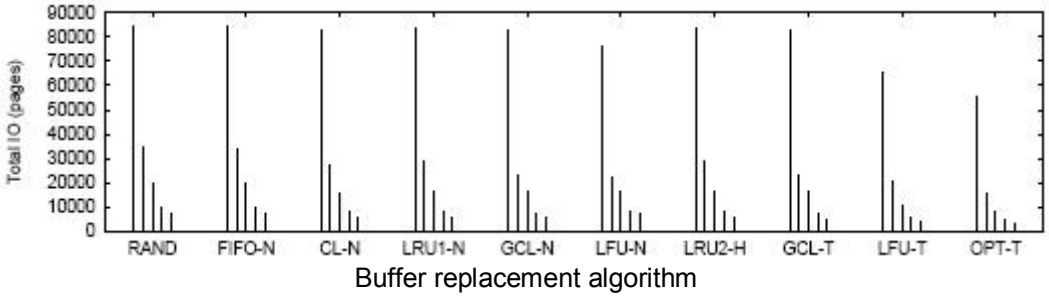
Here we report the effects of varying buffer size on the performance of the static clustering algorithms. The buffer replacement algorithm used is the LRU algorithm. The general observation is that C3-GGP always performs better than or as well as the existing algorithms. C3-GGP outperforms all existing algorithms between buffer sizes of 0.5MB and 5.8MB and shows equal performance for other buffer size settings. The reason for C3-GGP performing the same as GP when the buffer size is less than 0.5MB is that at this smaller buffer size the buffer replacement algorithms find it difficult to retain pages belonging to the hot region of C3-GGP in memory. This failure means clustering hot objects together is less profitable since even when many hot objects are clustered into the same page, the page still has a high probability of being evicted due to the small buffer size. When the buffer size is larger than 5.8MB almost all of the active portion of the database fits in memory and thus all static clustering algorithms perform about the same.At its best C3-GGP produces 42% less I/O than GGP (when buffer size is 2.4MB). The performance advantage can be attributed to C3-GGP's ability to retain hot objects in memory by creating hot pages with high concentrations of hot objects. This avoids thrashing of pages containing hot objects.

PRP's poor performance at buffer sizes below 6.6MB can be attributed to the fact that it does not attempt to cluster based on object transition information. However, when the buffer size is large enough to fit in the entire active portion of the database (beyond 6.6MB), it performs just as well as the other algorithms. This is because it is just as effective as the other algorithms at mapping the entire active portion of the database into a minimum number of pages.

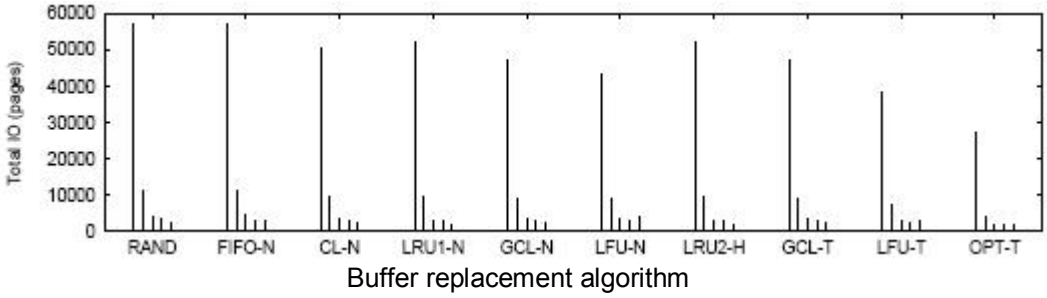
3.2 Varying Buffer Replacement Algorithm

We explore the performance of the clustering policies: no clustering, PRP, WGGP, GGP and C3-GGP on ten different buffer replacement algorithms. The ten different buffer replacement algorithms investigated include: random (RAND); First In First Out (FIFO-N); CLOCK (CL-N); traditional Least Recently Used (LRU1-N); GCLOCK (GCL-N) [20]; Least Frequently Used (LFU-N); Least Recently Used K algorithm with K set to 2 (LRU2-H) [18]; GCLOCK algorithm using training data (GCL-T); Least Frequently Used algorithm using training data (LFU-T); Belady's optimum algorithm (OPT-T) [14]. Algorithms with a 'T' suffix use information gathered in the training step of the simulation to help make more accurate replacement decisions during the evaluation step. The 'N' suffix is used for algorithms that do not use training data and also reset statistics for a page when it is first loaded into memory. Algorithms with a 'H' suffix retains history information for a page when it is evicted from memory. However, 'H' suffix algorithms do not use training data.

The results of using 1MB and 4MB buffer sizes are reported on Figure 1 (a) and (b) respectively. The results for five different static clustering policies are reported for each buffer replacement algorithm result. The static clustering results are reported in the following order, no clustering, PRP, WGGP, GGP and C3-GGP. The results show that for the 1MB buffer size case, C3-GGP offers best performance for all buffer replacement algorithms used. When the buffer size is 4MB, C3-GGP is the best performer for 8 of the 10 buffer replacement algorithms used. The only cases in which C3- GGP is not the best performer is when the buffer size is 4MB and the LFU-N and LFU-T buffer replacement algorithms are used. This is because at 4MB buffer size, almost all of the pages containing hot objects fit in memory, even when the hot objects are spread across many pages (the case with the NC, PRP, WGGP and GGP clustering algorithms). LFU algorithms which keep frequently accessed pages in memory prevent the pages containing hot objects from thrashing. Thus at these settings, C3-GGP's ability to prevent thrashing of pages containing hot objects from thrashing. Thus at these settings, C3-GGP's ability to prevent thrashing of pages containing hot objects no longer gives it an advantage over the other algorithms. The result is that GGP and WGGP, which cluster solely based on relatedness, are able to meet the second sub-objective of better than C3-GGP but do not suffer the negative consequences of not meeting the first sub-objective.



1(a): 1MB buffer size.



1(b): 4MB buffer size.

Figure 1(a) & 1(b): Comparing the effect of using different buffer replacement algorithm on total IO (pages) for five different static clustering policies .The results are reported in the following order: no clustering, PRP, WGGP, GGP and C3-GGP.

3.3. Varying Database Hot Region Size

In this simulation we varied the hot region size of the database and kept the probability of hot region access at a constant 0.8. The buffer replacement algorithm used is the LRU algorithm. It is encouraging to observe C3-GGP offers best performance for both 1MB and 4MB buffer sizes.

When using a buffer size of 1MB, C3-GGP's performance lead over GGP diminishes as the database hot region size increases. This is because as the hot region size increases, it becomes increasingly difficult for C3-GGP to fit hot objects into *its* hot region. Thus many hot objects end up in cold pages. The result is that C3-GGP is no longer able to prevent the thrashing of many of the pages that contain hot objects.

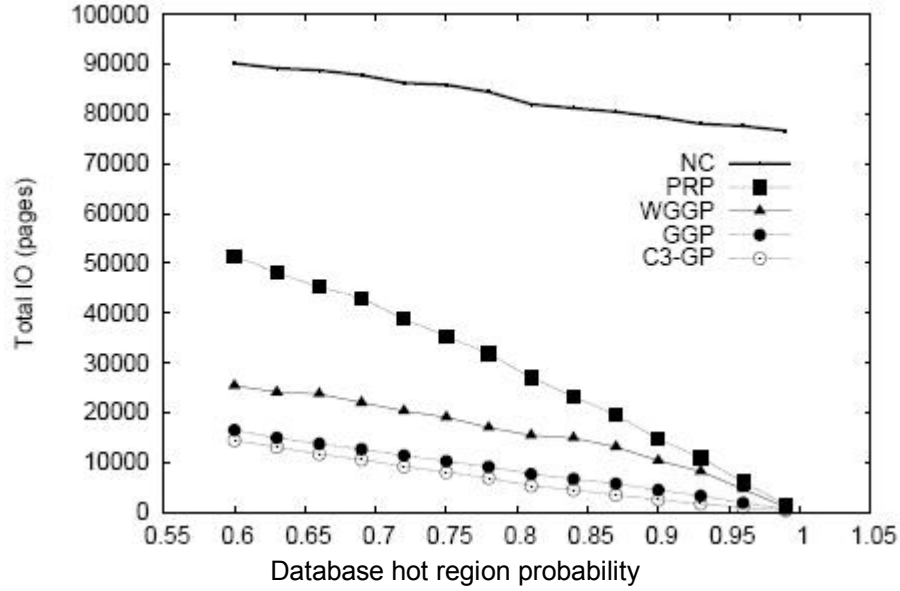
At the large buffer size of 4MB, C3-GGP's lead over the other static clustering algorithms increases as the database hot region size increases. The reason behind C3-GGP performing about the same as WGGP and GGP at small hot region sizes is that most of the active portion of the database fits in memory at this setting thus most pages containing hot objects are kept in memory even if hot objects are dispersed among many pages (as is the case for WGGP and GGP). However, as the hot region size increases,

C3-GGP's ability to compact hot objects into fewer pages becomes an increasingly larger advantage when compared to WGGP and GGP.

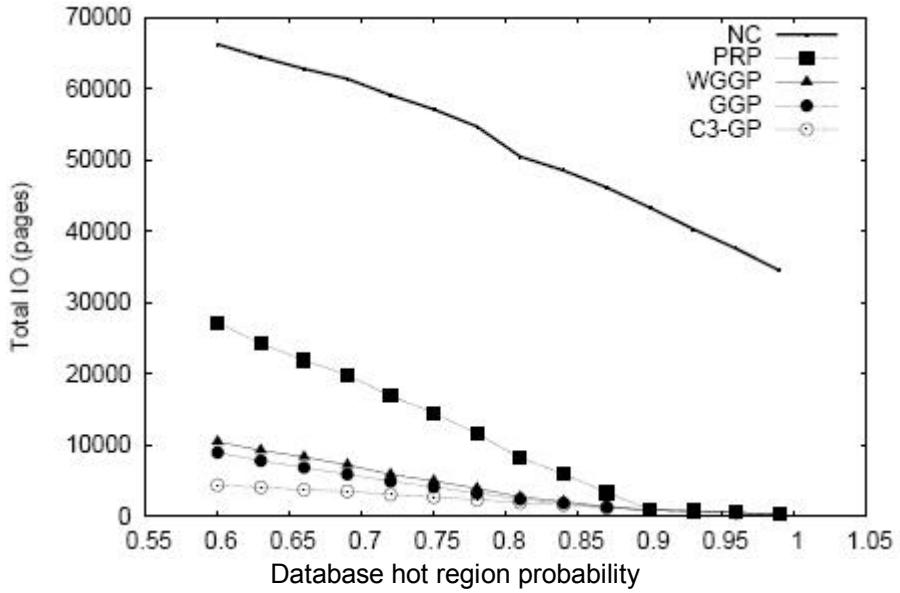
3.4. Varying Database Hot Region Access Probability

In this simulation we vary the probability of accessing objects in the hot region of the database. The size of the hot region is kept constant at 3% the size of the database. The buffer replacement algorithm used is again the LRU algorithm. The results when using the 1MB and 4MB buffer sizes are shown in Figure 2(a) and (b). The results show that C3-GGP offers the best performance in general.

At the 1MB buffer size, C3-GGP exhibits the best performance for all the results reported. However, at the 4MB buffer size, C3-GGP starts off well in front of the other algorithms but its lead diminishes as the database hot region probability increases. Eventually, at 0.9 all clustering algorithms perform the same. This is because at above 0.9 hot region access probability, almost all queries are confined to the hot region. Since the hot region is relatively small compared to the 4MB buffer size, the entire active portion of the database fits in memory. All of the static clustering algorithms are able to group the active portion of the database together and away from the non-active portion. This explains why all of the algorithms perform the same when the database hot region access probability is above 0.9.



2(a): 1MB buffer size.



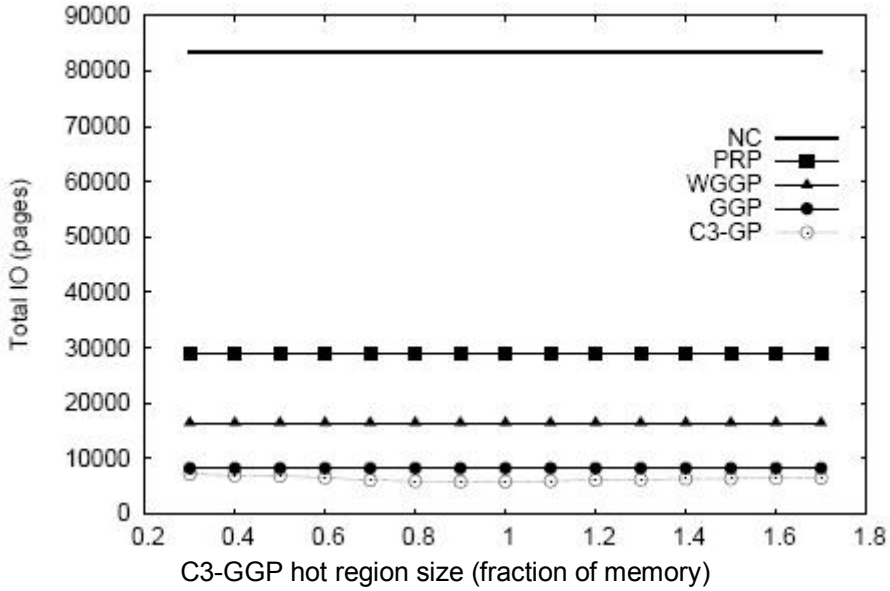
2(b): 4MB buffer size.

Figure 2(a) & 2(b): Comparing the effect of varying database hot region access probability on the total IO (pages) for five different static clustering polices. Uses the LRU replacement policy.

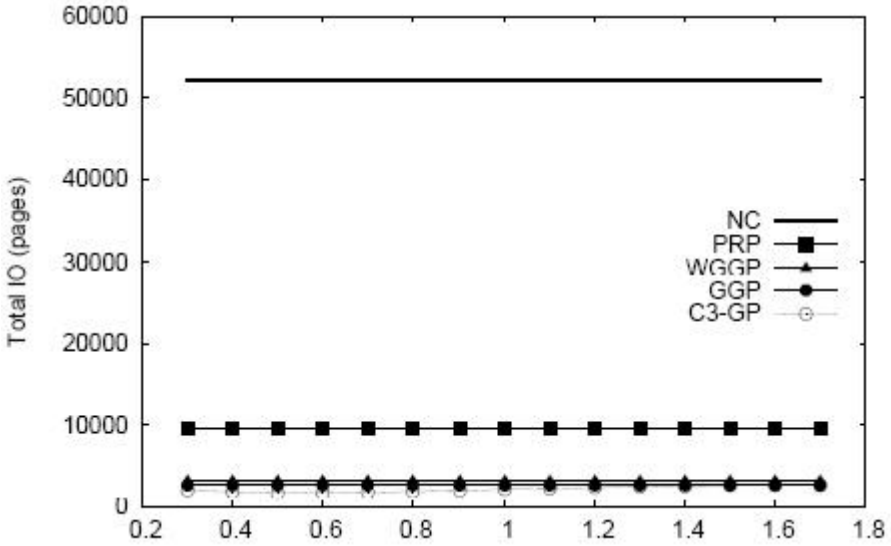
3.5. Varying C3-GGP Hot Region Size

In this simulation we varied the size of C3-GGP's hot region; C3-GGP's hot region is created by sorting the objects in decreasing heat and then taking the top *x* objects as belonging to the hot region. In the definition of C3-GGP, *x* is chosen so that all of the objects just fit into memory. In this simulation we vary the place where the sorted list of objects is cut. The buffer replacement algorithm used is the LRU algorithm. The results when using the 1MB and 4MB buffer sizes are shown in Figure 3 (a) and (b). The results for NC, PRP, WGGP and GGP do not change when C3-GGP hot region size is varied, since these algorithms do not use this parameter.

The results show that the optimal C3-GGP setting is dependent on the buffer size used. At 1MB buffer size, the optimal setting is approximately 0.9 and at 4MB buffer size the optimal setting is approximately 0.6. This is because the hot region size of the database is the same for both graphs; however the place at which C3-GGP divides its hot and cold region is a function of the buffer size, which is different for the two graphs. A possible direction of future work is to develop a method of dividing C3-GGP's hot and cold regions based on both the detected database hot region size and the buffer size.



3(a): 1MB buffer size.



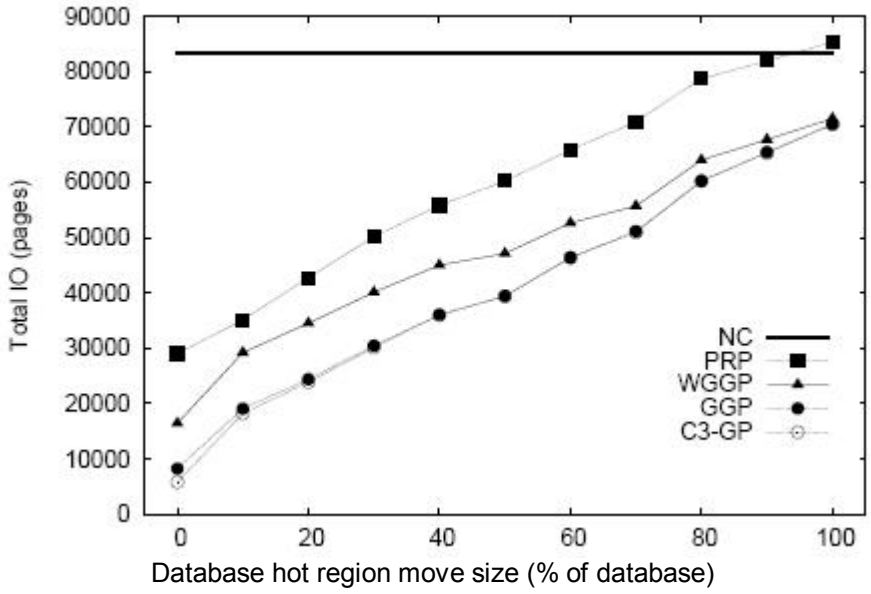
3(b): 4MB buffer size.

Figure 3(a) & 3(b): Comparing the effect on the total IO (pages) when C3-GGP's hot region size is varied. Uses the LRU replacement policy.

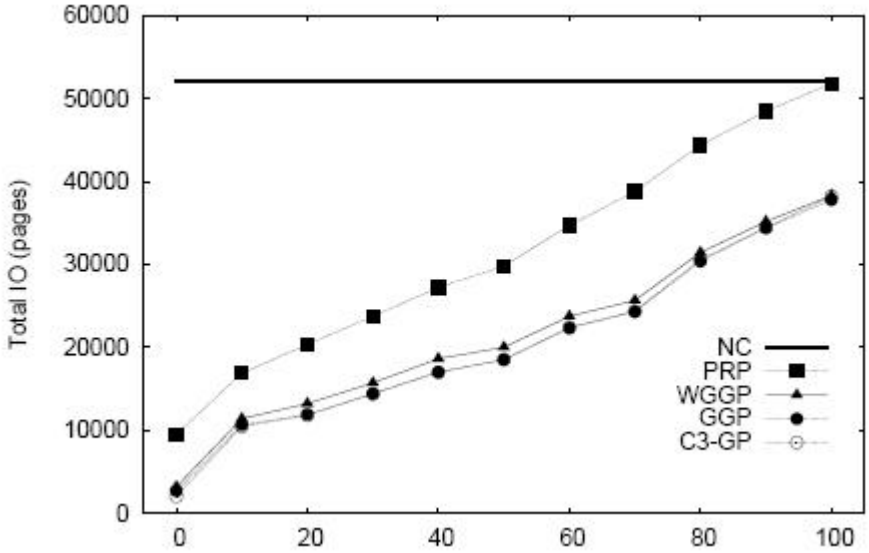
3.6. Training Skew

Until now all of the simulations involved running the same set of transactions for both the training and evaluation steps. In contrast, this simulation explores the effect of running a different set of transactions for the training and evaluation steps. This is achieved by moving the hot region of the database. The numbers on the x-axis of Figures 4 (a) and (b) show by how much the database hot region is moved. For example, a value of 20% means 20% of the hot region used for the training step became part of the cold region used for the evaluation step. This gives an indication of the degree of difference between transactions used in the training and evaluation steps. The hot region size of database was set to 3% of database size. The buffer replacement algorithm used is the LRU algorithm.

The results show that C3-GGP's performance advantage over GGP and WGGP rapidly diminishes as the level of training skew increases. This implies that C3-GGP is more sensitive to the quality of training data used. When poor heat information is supplied, C3-GGP places hot objects into the cold region and vise-versa. The consequence of this behaviour is that C3-GGP begins to loss its ability to keep a higher concentration of hot objects in memory. This explains the diminishing of C3-GGP's lead over GGP and WGGP when training skew is increased. However, it is encouraging to note that C3-GGP's performance never degrades to be worse than GGP or WGGP.



4(a): 1MB buffer size.



4(**b):** 4MB buffer size.

Figure 4(a) & 4(b): Comparing how the different static clustering algorithms perform (in terms of total IO (pages) under different amounts of training skew. Uses the LRU replacement policy.

4. CONCLUSIONS

In this paper, we have described the C3 framework, whereby clustering algorithms incorporating buffer replacement cache behaviour can be conveniently employed for enhancing the I/O performance of OODBMS. With this framework, a family of clustering algorithms (C3) can be developed. To demonstrate the soundness of the C3 framework, we have developed the C3-GGP algorithm.

Our simulation results show that C3-GGP out-performs existing static clustering algorithms in a variety of situations. Among the situations tested are 10 different buffer replacement algorithms, various buffer sizes, database hot region sizes, access probabilities, C3-GGP's hot region sizes and various amounts of training skew. Among all of the simulation results, C3-GGP performed at least as good as the existing algorithms for all but one particular situation (when the LFU-N and LFU-T buffer replacement algorithms are used and the buffer size is large). In particular, C3-GGP outperforms GGP (the best existing static clustering algorithm) when the buffer size is large as compared to the database size but offers similar performance when the buffer size is very small. This ability to perform consistently either better or the same as existing algorithms makes C3-GGP ideal for deployment in general purpose OODBMS in which workload conditions and system settings are not known a-priori.

REFERENCES

[1] Knafla, N.; "Prefetching Techniques for Client/Server, Object-Oriented Database Systems", PhD thesis, Computer Science, University of Edinburgh, 1999.

[2] Ailamaki, A., Dewitt. D.J., Hill. M.D. and Wood, D.A.; "DBMSs on a modern processor: Where does time go?", Proceedings of the International Conference on Very Large Databases, (VLDB 1999), Edinburgh, pp. 266–277, Scotland, September 1999.

[3] Tsangaris, E.; "M.M. Principles of Static Clustering For Object Oriented Databases", PhD thesis, Computer Science, University of Wisconsin- Madison, 1992.

[4] Banerjee, J., Kim, W., Kim, S.J., and Garza, J.F.; "Clustering a DAG for CAD databases", IEEE Transactions on Software Engineering, Vol. 14, pp. 1684-1699, November 1988.

[5] Gerlhof, C., Kemper, A., Kilger, C. and Moerkotte. G.; "Partition-based clustering in object bases: From theory to practice". Proceedings of the International Conference on Foundations of Data Organisation and Algorithms, FODO, pp. 301–316, 1993.

[6] Drew, P., King, R. and Hudson, S.E.; "The performance and utility of the cactis implementation algorithms", Proceedings of the International Conference on Very Large Databases, (Brisbane, Queensland, Australia, 13-16 Aug.1990), D. McLeod, R. Sacks-Davis, and H.-J. Schek, Eds., Morgan Kaufmann, pp. 135–147, 1990.

[7] Abuelyaman, E.S.; "An Optimized Scheme for Vertical Partitioning of a Distributed Database", In International Journal Of Computer Science And Network Security, Vol. 8(1), pp. 310, January 2008.

[8] Shui, W. M., Fisher, D.K., Lam, F. and Wong, R.K.; "Effective Clustering Schemes for XML Databases", Database and Expert Systems Applications, Vol. 3180 of LNCS, pp. 569-579, 2004. (DOI: 10.1007/978-3-540-30075-5_55), www.springerlink.com

[9] Darabant, A.S., Campan, A. and Cret, O.; "Hierarchical Clustering in Object Oriented Data Models with Complex Class Relationships", Proc of 8th IEEE Int. Conf. on Intel-ligent Engineering Systems, Romania, pp.307-312, 2004.

[10] Darabant, A.S. and Campan, A.; "Advanced Object Database Design Techniques.In Carpathain journal of mathematics", Vol. 1, pp. 21-30 , Print Edition: ISSN 1584 – 2851, Online Edition: ISSN 1843 – 4401, 2004, www.carpathian.ubm.ro

[11] Z., He., Marquez, A. and Lai, R.; "Cache conscious clustering", Database and Expert Systems Applications, Vol. 2113 of LNCS, pp. 815-825, 2001, (DOI: 10.1007/3-540-44759-8_79), http://www.springerlink.com

[12] Nicola, V.F., Dan, A. and Dias. D.M.; "Analysis of the generalized clock buffer replacement scheme for database transaction processing", Proceedings of the International Conference on the Measurement and Modeling of Computer Systems, ACM SIGMETRIC, pp. 35-46, 1992.

[13] O'neil, E.J., O'neil. P.E., and Weikum, G.; "The LRU-K page replacement algorithm for database disk buffering", Proceedings of the International Conference on the Management of Data (ACM SIGMOD), Washington, D.C, P. Buneman and S. Jajodia, Eds., ACM Press, pp. 297-306,1993.

[14] Belady, L.A.; "A study of replacement algorithms for a virtual-storage computer", IBM Systems, Vol. 5(2), 1966 .

[15] Darmont, J. and Schneider, M.; "Benchmarking OODBs with a generic tool", Journal of Database Management, Vol. 11(3), pp.16-27, 2000.

[16] Darmont, J. and Schneider, M.; "VOODB- A generic discrete-event random simulation model to evaluate the performances of OODBs", hal.archives-ouvertes.fr (hal 00144233, version1-3, May 2007. http://arxiv.org/abs/0705.0450

[17] Graefe, G.; "The five-minute rule twenty years later, and how flash memory changes the rules", Proceedings of the Third International Workshop on Data Management on New Hardware, Beijing, China (DaMoN 2007), June 15, 2007.

[18] Carey, M.J., Franklin, M.J., Livny. M. and Shekita, E.J.; " Data caching tradeoffs in client-server dbms architectures", Proceedings of the International conference on the

Management of Data (ACM SIGMOD 1991), J. Clifford and R. King, Eds., pp. 357-366, 1991.

[19] Franklin, M.J., Carey, M.J. and Livny, M.; "Local disk caching for client-server database systems", Proceedings of the International Conference on Very Large Databases (VLDB 1993), R. Agrawal, S. Baker and D.A. Bell, Eds., pp. 641-655, 1993.

[20] Z, He., Marquez, A. and Blackburn. S.; "Opportunistic prioritised clustering framework (OPCF)", Objects and Databases, Vol. 1944 of LNCS, pp. 86-100, June 2001. (DOI: 10.1007/3-540-44677-X_6), http://www.springerlink.com