Intensify the I/O Performance of OODBS by Collaboration between Opportunism and Prioritization

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

As we all know that clustering has demonstrated to be one of the most effective performance enhancement techniques for object oriented database systems. The bulk of work is done on static clustering, that is re-clustering the object base when the database is off-line. When 24-hour database access is required this type of re-clustering cannot be used. In these cases clustering is required which can recluster the object base while the database is in operation. We believe that most existing on-line clustering algorithms lack three important properties. These include: the use of opportunism to imposes the smallest I/O footprint for re-organization; the re-use of prior research on static clustering algorithms; and the prioritization of re-clustering so that the worst clustered pages are re-clustered first. In this paper we present a opportunistic priority clustering framework in which any existing off-line clustering algorithm can be made on-line and given the desired properties of opportunism and clustering prioritization. Most importantly it allows the created algorithm to have the properties of I/O opportunism and clustering prioritization which are missing in most existing dynamic clustering algorithms. We have used OPCF to make the static clustering algorithms Graph Partitioning and Probability Ranking Principle into dynamic algorithms. We have used the latest version of VOODB 2007 and OCB 1998 in comparison to other researchers.

Keywords: Object-oriented database system, Static clustering, Dynamic clustering algorithms, Prioritization, Opportunism.

1. INTRODUCTION

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

Clustering has been demonstrated to be one of the most effective performance enhancement techniques [3]. In early days research of database management systems. Steering object access in a object oriented database is a reason behind this. Consequently, related objects are often accessed consecutively. The grouping of related objects into the same disk page reduces disk I/O by clustering objects in an object oriented database. By reducing the number of unused objects that occupy the cache clustering also uses cache space more efficiently. Periodical re-clustering allows the physical organization of objects on disk to closer reflect the prevailing pattern of object access.

The majority of existing clustering algorithms are static [4-11]. Static clustering algorithms require that re-clustering take place when the database is not in operation, thus prohibiting 24 hour database access. In contrast, on-line clustering algorithms re-cluster the database while database applications are in operation. Applications that require 24 hour database access and involve frequent changes to data access patterns may benefit from the use of on-line clustering. The problem is compounded when the size of the database is large. In such situations the initial physical database organization will become obsolete over time, which may result in the object cache containing many objects that are never used.

In our view, there are a number of properties that are missing from most existing on-line clustering algorithms. These properties include: The use of opportunism to impose the smallest I/O footprint for re-organization; The re-use of existing work on static clustering algorithms; and a prioritization of reclustering so the worst clustered pages are re-clustered first.

Prioritization refers to choosing the worst clustered page to recluster first. This is very important since at each reorganization iteration we can only recluster a small portion of the object base (to minimize disruption to the database application), this means we need to be very selective on what we recluster first. Reclustering the worst clustered page first ensures that we get the largest benefit from reclustering as soon as possible.

Opportunism refers to only reclustering pages that are currently in memory. This simple rule is of critical importance, since the goal of dynamic clustering is to reduce the number of I/Os generated by the database application. So if we generate addition I/O overhead while performing the dynamic clustering then it is very possible that the cost of clustering may outweigh the benefits. By doing clustering opportunistically (only reclustering in memory pages) we can make sure no I/O is generated during the clustering process itself.

Although the great body of work that exists on static clustering [4-11], there has been little transfer of ideas into the on-line clustering literature. In this paper we address this omission by incorporating two existing and yet vastly contrasting types of static clustering algorithms into our on-line clustering framework. These are the ‘probability ranking principle’ algorithm (PRP) and the ‘graph partitioning’ algorithms. We show that by using our framework these two existing static clustering algorithms surpass an existing and highly competitive on-line clustering algorithm DSTC [12] and DRO [13] in a variety of situations.

The troublesome nature of re-clustering dictates that on-line clustering algorithms must be incremental. This means that only a small portion of the object base can be re-clustered in each iteration. When faced with a variety of different portions to target for re-organization, the clustering algorithm must select the portion to re-cluster first.

The re-organization phase of on-line clustering can incur significant overhead. Two of the key overheads are increased write lock contention [4] and I/O. To reduce write lock contention, most on-line clustering algorithms are designed to be incremental and thus only consider a portion of the object base for clustering during each re-organization. However, we are aware of only one algorithm Wietrzyk and Orgun [14] that takes care to not to introduce extra I/O during the re-organization phase. Wietrzyk and Orgun [14] accomplish this by calculating a new placement when the object graph is modified, either by a link (reference) modification or object insertion. The algorithm then reclusters the objects that are effected by the modification or insertion. Once the new placement is determined, only the objects in memory are re-organized and the remaining objects are only re-arranged as they are loaded into memory. However, the objects considered for re-organization can include any object in the store. The drawback of this approach is that information about any object in the store may be needed. OPCF produces algorithms that differ from these algorithms by only needing information on objects that are currently in memory and only re-organizing those objects. This makes on-line clustering algorithms produced by OPCF more opportunistic than existing algorithms.

A large body of work exists on static clustering algorithms [4-11]. However only relatively few static algorithms have been transfered into on-line algorithms. McIver and King [15] combined the existing static clustering algorithms. Cactis Hudson and King [16] and DAG Banerjee et al. [5] created a new dynamic clustering algorithm. However Hudson and King [16] and Banerjee et al. [5] created only sequence-based clustering algorithms which have been found to be inferior when compared to graph partitioning algorithms [4]. Wietrzyk and Orgun [14] developed a new dynamic graph partitioning clustering algorithm. However their graph partitioning algorithm was not compared with any existing static graph partitioning clustering algorithm. In this paper two existing static clustering algorithms were transformed into on-line clustering algorithms using OPCF and compared to an existing on-line clustering algorithm, DSTC [12].

2. METHODOLOGY

2.1. Simulation Setup

We describe simulations designed to compare the performance of algorithms produced using OPCF with two existing state of the art dynamic clustering algorithms, DSTC and DRO. We have chosen DSTC and DRO because they are two of the latest dynamic clustering algorithms. They are built based on the lessons learnt from many older dynamic clustering algorithms.

The simulations are conducted using the virtual object oriented database simulator, VOODB [17]. VOODB is based on a generic discrete-event simulation framework. Its purpose is to allow performance evaluations of OODBs in general, and optimization methods such as clustering in particular. VOODB simulates all of the traditional OODBMS components such as: the transaction manager, object manager, buffer manager, and I/O subsystem. The correctness of VOODB has been validated for two real-world OODBs, O2 [18] and Texas [19].

VOODB is very user tunable, offering a number of system parameters such as: system class (distribution configuration), page size, buffer size, buffer replacement strategy, etc. The parameters relevant to our study along with the values used are depicted in Table 1(a). The centralized system class refers to the stand -alone system configuration, in which the clients and server both reside on the same machine. Optimized sequential placement refers to compact placement (every page has a low percentage of empty space), which is mostly placed in creation order.

Table 1(a): VOODB parameters used.

Parameter Description	**Value**
System class	Centralized
Disk page size.	4096 bytes
Buffer size.	Varies
Buffer replacement strategy	LRU-2
Pre-fetching strategy	None
Multiprogramming	1
Number of users	1
Object initial placement	Optimized sequential

In this paper the OCB benchmark [20] is chosen as the tool used for evaluating buffer management techniques under stationary access patterns. Here we provide a brief description of OCB. In addition, we describe the OCB parameter settings that are used for the simulations in this paper.

The OCB benchmark was initially designed for benchmarking clustering algorithms but its rich schema and realistic workloads make it particularly suitable for benchmarking prefetching algorithms too. 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. The parameter shown in Table1(b) which are used in the simulations reported in this paper, a total of 200,000 objects

are generated. The objects varied in size from 50 to 1600 bytes and the average object size is 232 bytes. The total database size is 23 MB. Since this is a small database size we also use small buffers (1MB and 4MB) to keep the database to buffer size ratio large. Since we are interested in the caching behavior of the system, the database to buffer size ratio is a more important parameter than database size alone.

Table1(b): 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	200,000
Number of references types	4
References types random distribution	Uniform
Class reference random distribution	Uniform
Objects references random distribution	Uniform

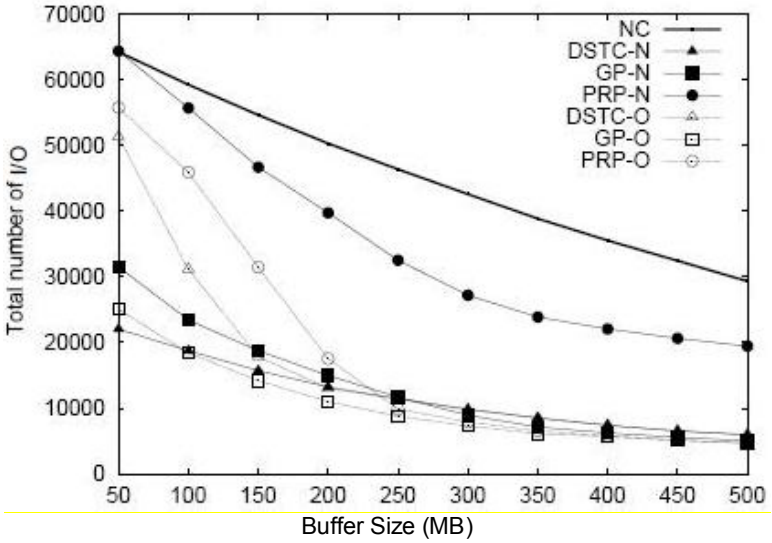
In this paper we compare the performance of four dynamic clustering algorithms. These include two existing algorithms, dynamic statistical tunable clustering (DSTC), detection & re-clustering of objects (DRO) and two new OPCF algorithms, dynamic greedy graph partitioning (OP-GP) and dynamic probability ranking principle (OP-PRP).

We introduce skew into the way traversal roots are selected. Roots are partitioned into hot and cold regions. In all simulations the hot region is set to 3% of the size of database and has an 80% probability of access. These settings are chosen to represent typical database application behavior. G. Graefe [21] cited statistics from a real videotext application in which 3% of the records got 80% of the references. Carey et al. [22] used a hot region size of 4% with a 80% probability of being referenced in the HOTCOLD workload used to measure data caching trade-offs in client/server OODBMSs. Franklin et al. [23] use a hot region size of 2% with a 80% probability of being referenced in the HOTCOLD workload used to measure the effects of local disk caching for client/server OODBMSs.

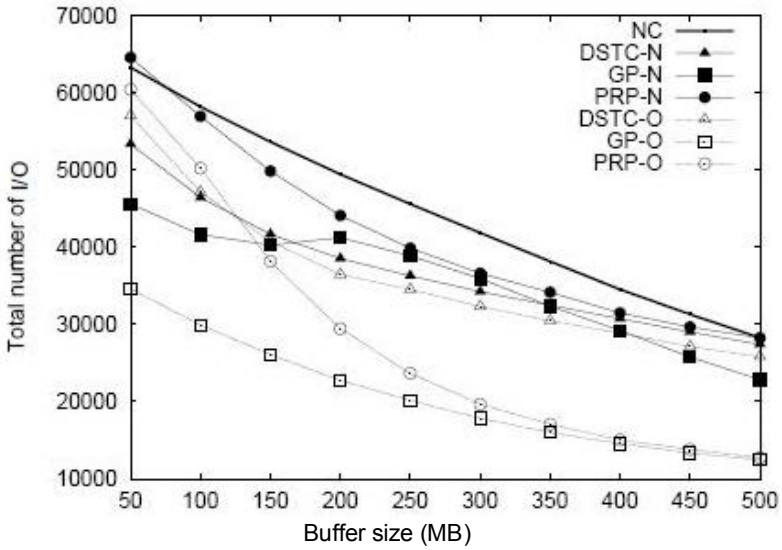
3. SIMULATION RESULTS

Here we report the results of simulations comparing the performance of two existing highly competitive dynamic clustering algorithms (DSTC, DRO) and two new algorithms produced using the OPCF framework (OP-PRP, OP-GP).

3.1. Varying Buffer Size



(a) Read only transaction.



(b) 10% update transactions.

Fig.1: Effects of varying buffer size.

This experiment was designed to investigate the effects of changing buffer size in two different conditions, read only transactions and 10% update transactions. We divided the 20MB (20000 object) database into hot and cold regions. The hot region was made to be 1.5% of the total size of the database and 99% of transactions were directed at the hot region.

The results are shown on Figure 1. When updates are introduced into the workload, GP-O appears to outperform DTSC-O and DSTC-N algorithms by a large margin. A possible explanation for this behavior is that DSTC works at the object grain and thus places each newly constructed cluster of objects into a new page. This generates a lot of empty space in pages where the cluster size is small. The end result is that objects are more spread out and when random updates occur, a larger number of pages are updated, resulting in a larger number of write I/Os. This contrasts to the graph partitioning algorithm where multiple clusters may reside in the same page and thus random updates are confined to a smaller number of pages.

Secondly, when the buffer size is small the PRP algorithms do not perform as well as GP. This result is consistent with the off-line behavior of the algorithms. The reason for this is that PRP does not take inter-object relationships into consideration when clustering. In our simulations, increasing the parameters T_{fa}, T_{fe} and T_{fc} by the smallest possible increment causes the algorithms to go from over excessive clustering to almost no clustering (both settings result in about the same performance). Here

T_{fa} = Threshold value under which the number of accesses to individual objects is too small to be considered in elementary linking factors computation.

T_{fe}= Threshold value under which elementary linking factors are not consider for updating consolidation factors.

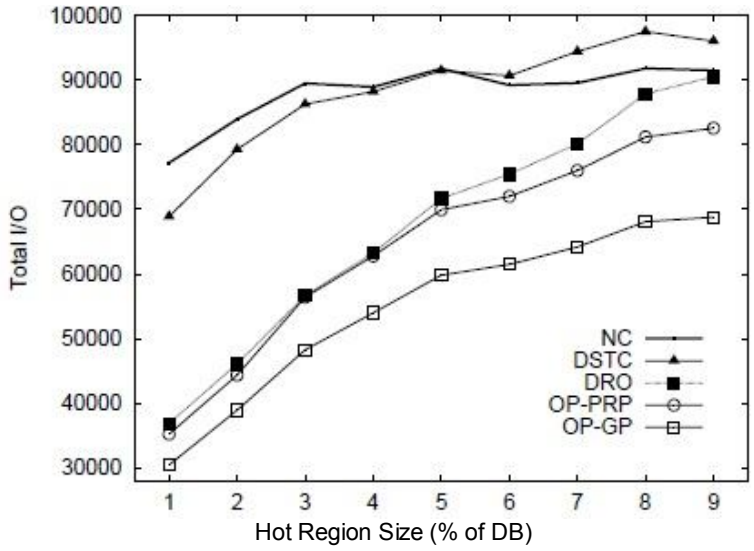
T_{fc}= Threshold value under which a consolidated linking factors are not considered significant.

The results shown are when the more aggressive clustering setting is used. We have tested all algorithms extensively with different parameters settings and only used the settings that gave the best overall performance for each algorithm.

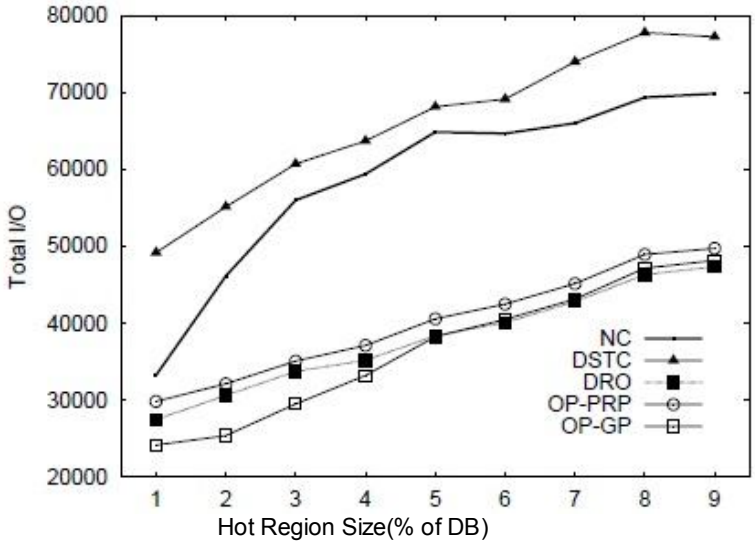
3.2. Varying Hot Region Size

Here we examine the effect of changing hot region size on the dynamic clustering algorithm performance. The hot region probability of access is set to 80%. The results for two buffer size settings of 1MB and 4MB are reported in Figures 2(a) and 2(b), respectively. The results in this simulation showed that OP-GP providing best overall performance. We attribute this mainly to its use of opportunism and its use of sequence tension to model access dependencies between objects. At the small buffer size of 1MB, OP-GP's performance degrades at a slower rate than DRO, the best existing dynamic clustering algorithm. The reason for this is that increasing the hot region size has the effect of increasing the working

set size. A larger working set means more objects will have higher usage rates. This in turn means DRO, which only clusters pages with usage rates above the minimum UR threshold, clusters more aggressively as working set size increases. As the hot region size increases, a larger portion of the working set is disk resident. DRO, which is not opportunistic, performs more clustering read I/O since a larger portion of the working set is disk resident. DRO also performs more clustering write I/Os as the hot region size increases. This is because as more pages are loaded into memory for clustering purposes, more dirty pages are evicted (working set does not fit in memory). Dirty page evictions causes write I/O. In contrast, OP-GP's opportunistic behavior combined with its bounded scope of re-organization results in a smaller clustering I/O footprint, for both small and large hot region sizes. At the larger buffer size of 4MB (Figure 2(b)), almost all of the working set fits in memory (even when the hot region size is 9% of database size). In this environment, DRO's performance approaches that of OP-GP as the hot region size increases. This is because as more of the working set fits in memory, DRO's lack of opportunism is less damaging to its performance. Since most of the pages it wants to cluster are memory resident, less clustering read I/O is required. Clustering write I/Os are also decreased due to fewer dirty page evictions.



2(a): 1MB buffer size.



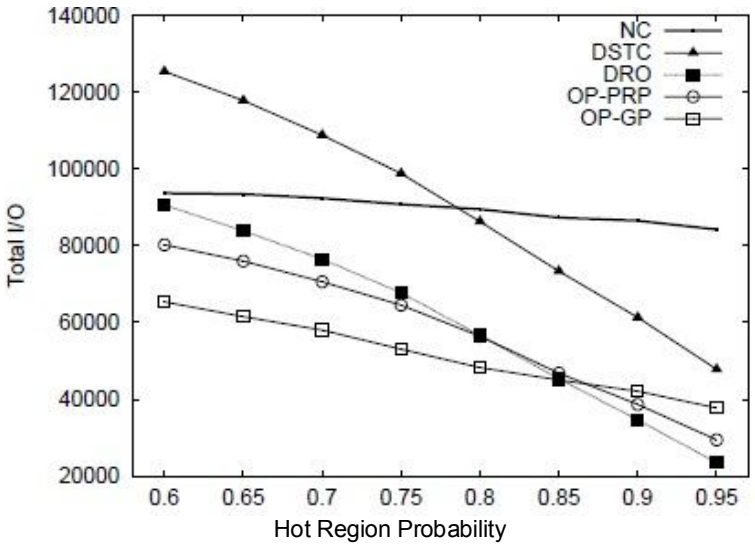
2(b): 4MB buffer size.

Fig. 2: Hot region size versus Total I/O.

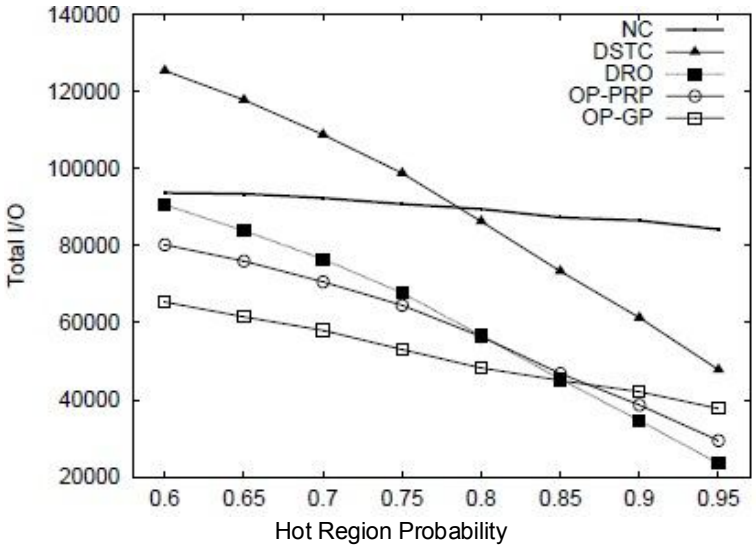
3.3. Varying Hot Region Access Probability Size

Here we report the effects of changing hot region access probability. The hot region is set to 3% of the database size. The results of two buffer size settings of 1MB and 4MB are reported on Figures 3(a) and 3(b), respectively. The results for 1MB buffer size show OP-GP offering the best performance at small hot region access probabilities. However, when hot region access probability is high, OP-GP's performance degrades to be worse than DRO. The reason for OP-GP's success at low access probabilities is OP-GP's ETT threshold and opportunism protects it from aggressive re-clustering of the cold region. In contrast, DRO does not have these features. Thus at these settings DRO finds it difficult to distinguish between the hot and cold region (the difference between access probability of hot and cold region is small), and consequently re-clusters much of the cold region aggressively. The result is that DRO generates a lot of clustering read and write I/O overheads for marginal transaction read I/O gains. However, when hot region access probability is high, OP-GP's ETT threshold starts to work against it. This is because ETT restricts re-clustering (even in the hot region) to those pages that give the largest performance improvement. In contrast, DRO, which aggressively re-clusters the hot region, benefits from improved transaction read I/O gains. At high hot region access probabilities, DRO no longer has difficulty separating the hot and cold region for re-clustering. Thus it no longer spends clustering I/O resources for re-clustering the cold region. The results for the 4MB buffer size as seen in Figure 3(b) again showed that OP-GP offering best performance when hot region access probability is low. When hot region access probability is high, OP-GP and DRO perform approximately the

same. The reason for OP-GP's superior performance at low hot region access probabilities is the same as for the 1MB buffer size results. However, at high hot region access probabilities, the larger buffer size is more forgiving for OP-GP's lack of aggression in re-clustering the hot region. Thus OP-GP's performance no longer degrades to worse than DRO at high hot region access probabilities. DSTC, which has been set to re-cluster aggressively (at low aggression settings it performs almost no re-clustering and there is no middle ground), shows rapid performance improvement when hot region access probability increases. This is because at high hot region access probabilities, aggressive re-clustering is confined to primarily the hot region (the cold region rarely gets touched and thus is rarely re-clustered). This fact means DSTC's clustering I/O overheads rapidly diminish as the hot region access probability increases.



3(a): 1MB buffer size.

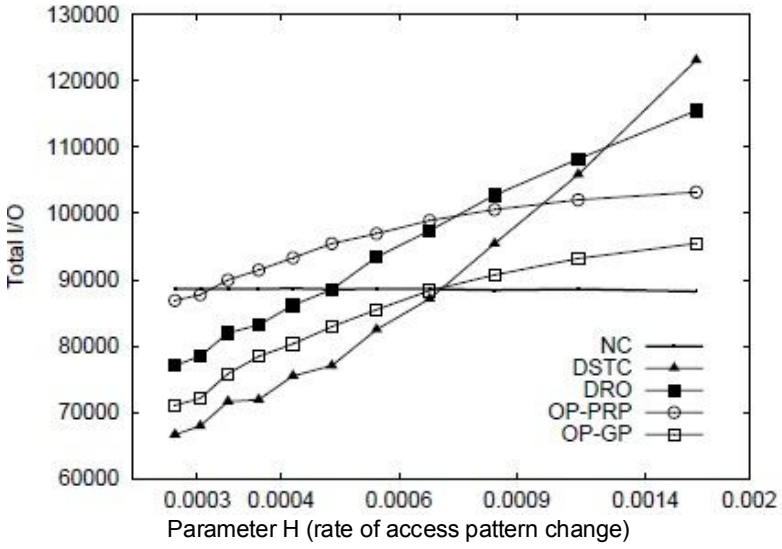


3(b): 4MB buffer size.

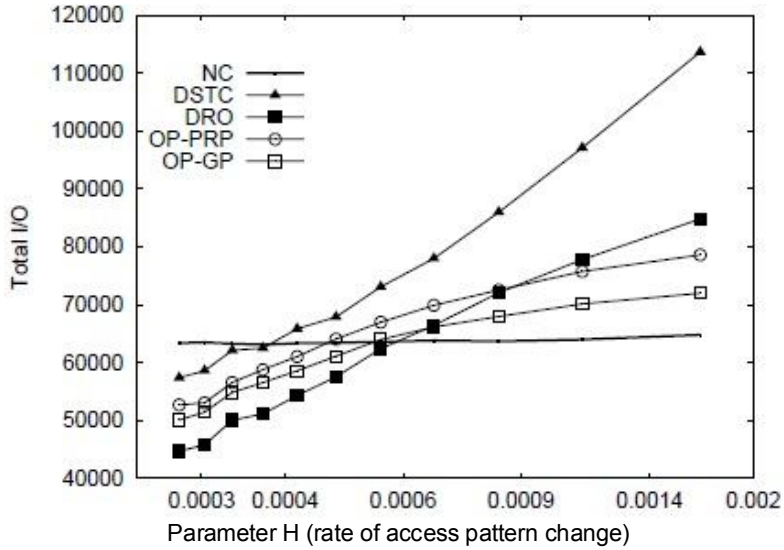
Fig. 3: Hot region probability Vs Total I/O.

3.4. Moving Window of Change

In this simulation, we used the moving window of change protocol to test each of the dynamic clustering algorithms' ability to adapt to changes in access pattern. In this simulation we varied the parameter H, rate of access pattern change. We report the results using 1MB and 4MB buffer sizes. The results are shown on Figure 4(a) and 4(b), respectively. The X-axis in the graphs is in $\log_2$ scale. The general observation is that OP-GP offers poor performance when the rate of access pattern change is small but offers best performance (when compared to the other dynamic clustering algorithms) when the rate of access pattern change is high. The poor performance of OP-GP under small rate of access pattern changes is that OP-GP has not been designed for this vigorous style of change (where the hot region changes suddenly from 0.8 probability of access to 0.0006 probability of access). The other algorithms, DSTC and DRO, periodically delete gathered statistics which makes them cope much better in this style of change. However, when the rate of access pattern change is very high, DSTC and DRO suffer from over aggressive re-clustering. Both DSTC and DRO do not place hard bounds on the scope of re-organization. This causes them to re-cluster vigorously at high rates of access pattern change (the clustering on many pages appear to be outdated). Much of the hard re-clustering work goes to waste, since re-clustered pages quickly get outdated. In contrast, OP-GP and OP-PRP, which place hard limits on the scope of re-organization, perform better at higher rates of access pattern change.



4(a): 1MB buffer size.

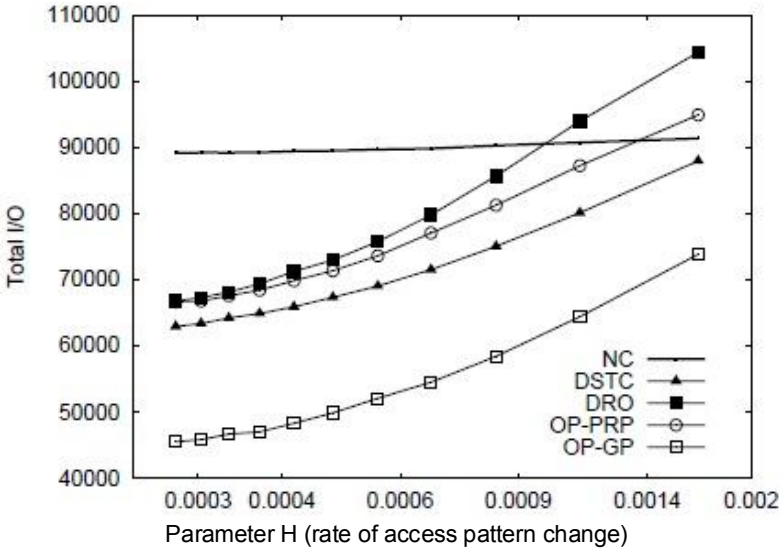


4(b): 4MB buffer size.

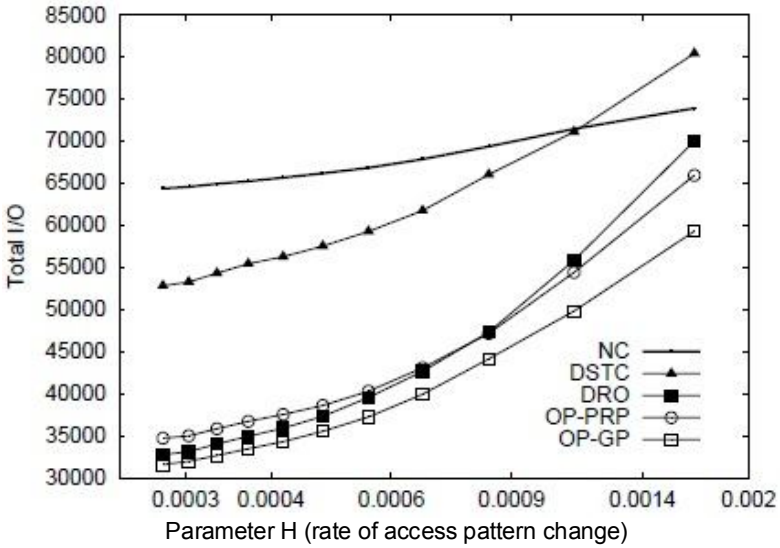
Fig. 4: Moving window of change.

3.5. Gradual Moving Window of Change Simulation

In this simulation, we used a less vigorous style of access pattern change, the gradual moving window of change protocol. Unlike the previous simulation, the hot region cools gradually instead of suddenly. In this simulation we varied the parameter H, rate of access pattern change. We report the results using 1MB and 4MB buffer sizes. The results are shown on Figure 5(a) and 5(b), respectively. The X-axis in the graphs is in $\log_2$ scale. The results show OP-GP consistently outperforming the other dynamic clustering algorithm when this gradual style of change is used. This is due to OP-GP's policy of not deleting old statistics, which is beneficial instead of detrimental to clustering quality in this simulation. The gradual cooling of the hot region means that as the hot region cools a lot of residual heat remains. OP-GP, which never deletes old statistics, continues to re-cluster the slowly cooling hot region.



5(a): 1MB buffer size.



5(b): 4MB buffer size.

Fig. 5: Gradual moving window of change.

4. CONCLUSION

In this paper we have presented Opportunistic Priority Clustering Framework, which when applied to static clustering algorithms, can produce dynamic clustering algorithms that possesses the two desirable properties of opportunism and prioritization of clustering. In accumulation, application of the framework is uncomplicated and yet it produces clustering algorithms that surpass an existing highly cutthroat dynamic clustering algorithm, DSTC [3], in a variety of situations. The outcome in this paper test the dynamic clustering algorithms in ample variety of situations, hot region sizes, hot region access probabilities, including various buffer sizes and various rates of access pattern change. The results show that the OPCF algorithm, OPCF with greedy graph partitioning offers best recital in nearly all situations. OPCF with greedy graph partitioning derives its recital advantage mainly from its opportunistic behavior and its use of sequence tension to model access enslavement between objects. Opportunism reduces OP-GP's clustering I/O overhead, while sequence tension allows OPCF with greedy graph partitioning to achieve high clustering quality. The combination of low clustering I/O overhead and high clustering quality gives OPCF with greedy graph partitioning the best overall performance. The result in this paper is same as in [24] but our parameter is different from other.

REFERENCES

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

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

[3] A. Gerlhof, A. Kemper and G. Moerkotte; "On the cost of monitoring and reorganization of object bases for clustering", ACM SIGMOD Record, Vol. 25, pp. 28-33,1996. www.acm.org

[4] E. Tsangaris; "M.M. Principles of Static Clustering For Object Oriented Databases", PhD thesis, Computer Science, University of Wisconsin–Madison, 1992. http://en.scientificcommons.org/4912466

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

[6] C. Gerlhof, A. Kemper, C. Kilger and G. Moerkotte; "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. www.citeseerx.ist.psu.edu

[7] P. Drew, R. King and S.E. Hudson; "The performance and utility of the cactis implementation algorithms", Proceedings of the International Conference on Very Large Databases, Brisbane, Queensland, Australia, pp. 135-147, 13-16 Aug1990. www.vldb.org

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

[9] W.M. Shui, D.K. Fisher, F. Lam and R.K. Wong; "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

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

[11] A.S. Darabant and A. Campan; "Advanced Object Database Design Techniques", Carpathain journal of mathematics, Vol. 1, pp. 21- 30, 2004, www.carpathian.ubm.ro

[12] F. Bullat and M. Schneider; "Dynamic clustering in object databases exploiting effective use of relationships between objects", Proceedings of the European Conference on Object-Oriented Programming (ECOOP 1996), Linz, Austria, Springer, pp. 344–365, 1996. www. arxiv.org

[13] J. Darmont, C. Fromantin, S. Regnier, I. Gruenwald and M.Schneider; "Dynamic clustering in object oriented databases: An advocacy for simplicity", Proceedings of the International Symposium on Object and Databases, Vol. 1944 of LNCS, pp.71-85,June 2000. www. arxiv.org

[14] V.S. Wietrzyk and M.A. Orgun; "Dynamic reorganisation of object databases", Proceedings of the International Database Engineering and Applications Symposium, IEEE Computer Society, August 1999. www. dl.acm.org

[15] W.J.J. Mciver and R. King; “Self-adaptive, on-line reclustering of complex object data”, Proceedings of the International Conference on the Management of Data, Minneapolis, Minnesota, R. T. Snodgrass and M. Winslett (Eds.), ACM Press, pp. 407-418, 1994. www.dl.acm.org

[16] E. Hudson and R. King; “Cactis: A self-adaptive, concurrent implementation of an object oriented database management system”, ACM Transactions on Database Systems, pp. 291-321, September 1989. www. dl.acm.org

[17] J. Darmont and M. Schneider; “VOODB:A generic discrete-event random simulation model to evaluate the performances of OODBs”, hal.archives-ouvertes.fr (hal 00144233, version1 -3), May 2007.

[18] O. Deux; “The O2 system”, Communications of ACM, Vol. 34(10), pp. 34-48, 1991. www. dl.acm.org

[19] V. Singhal, S.V. Kakkad and P.R. Wilson; “Texas: An efficient, portable persistent store”, Proceedings of the International Workshop on Persistent Object Systems, pp.11-33,1992. www. dl.acm.org

[20] J. Darmont, B. Petit and M. Schneider; “OCB: A generic benchmark to evaluate the performances of object-oriented database systems”, Proceedings of the International Conference on Extending Database Technology, Vol. 1377, pp. 326-340, March 1998. www.springerlink.com

[21] G. Graefe; “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), 15 June 2007. www.citeseerx.ist.psu.edu

[22] M.J. Carey, M.J. Franklin, M. Livny and E.J. Shekita; “Data caching tradeoffs in client-server dbms architectures”, Proceedings of the International conference on the Management of Data, J. Clifford and R. King (Eds.), pp. 357-366, 1991. www.portal.acm.org

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

[24] He Zhen, Lai Richard, Alonso Marquez and Stephen Blackburn; “Opportunistic prioritised clustering framework for improving OODBMS performance”, Journal of System and Software, Vol. 80(3), pp. 371-387, March 2007. www.portal.acm.org