Opportunistic Prioritised Clustering Framework (OPCF)

Zhen He, Alonso Marquez and Stephen Blackburn*

Department of Computer Science
The Australian National University
Canberra, ACT 0200

Email: {zhen, alonso}@cs.anu.edu.au

Abstract

Ever since the 'early days' of database management systems, clustering has proven to be one of the most effective performance enhancement techniques for object oriented database management systems. The bulk of the work in the area has been on static clustering algorithms which re-cluster the object base when the database is off-line. However, this type of re-clustering cannot be used when 24-hour database access is required. In such situations *on-line clustering* is required, which allows the object base to be reclustered 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-organisation; the *re-use of prior research* on static clustering algorithms; and the *prioritisation of re-clustering* so that the worst clustered pages are re-clustered first. In this paper, we present OPCF, a framework in which any existing off-line clustering algorithm can be made on-line and given the desired properties of opportunism and clustering prioritisation. In addition, this paper presents a performance evaluation of the ideas suggested above and in particular shows the importance of opportunism in improving the performance of on-line clustering algorithms in a variety of situations. The main contribution of this paper is the observation that existing off-line clustering algorithms, when transformed via a simple transformation framework such as OPCF, can produce on-line clustering algorithms that out-perform complex existing on-line algorithms, in a variety of situations. This makes the solution presented in this paper particularly attractive to real OODBMS system implementers who often prefer to opt for simpler solutions.

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. On the other hand, cheap disks mean object bases will become bigger as database designers realise that more data can be stored [Knafla 1999]. 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[Ailamaki et al. 1999]. 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 bottlenecks.

Ever since the 'early days' of database management systems, clustering has proven to be one of the most effective performance enhancement techniques [Gerlhof et al. 1996]. The reason for this is that the majority of object access in a object oriented database are navigational. Consequently, related objects are often accessed consecutively. Clustering objects in an object oriented database reduces disk I/O by grouping related objects into the same disk page. In addition to reduced I/O, clustering also uses cache space more efficiently by reducing the number of unused objects that occupy the cache.[1] Periodical re-clustering allows the physical organisation of objects on disk to closer reflect the prevailing pattern of object access.

The majority of existing clustering algorithms are static [Tsangaris and Naughton 1991; Tsangaris and Naughton 1992; Banerjee et al. 1988; Gerlhof et al. 1992; Gerlhof et al. 1993; Drew et al. 1990; Wietrzyk and Orgun 1998]. 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

*The authors wish to acknowledge that this work was carried out within the Cooperative Research Center for Advanced Computational Systems established under the Australian Government's Cooperative Research Centers Program.

[1] Throughout this paper we use the term 'cache' to refer to the in-memory portion of an object base.

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 organisation 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-organisation;
- The re-use of existing work on static clustering algorithms; and
- A prioritisation of re-clustering so the worst clustered pages are re-clustered first.

The goal of on-line clustering is to generate the minimum number of I/Os for a given set of database application access patterns. The clustering process itself may generate I/O, loading data pages for the sole purpose of object base re-organisation. However, most researchers have chosen to ignore these sources of I/O generation and instead concentrated on developing the on-line clustering algorithm that minimises the number of data pages[2] loaded during normal database operation[3].

Despite the great body of work that exists on static clustering[Tsangaris and Naughton 1991; Tsangaris and Naughton 1992; Banerjee et al. 1988; Gerlhof et al. 1992; Gerlhof et al. 1993; Drew et al. 1990; Wietrzyk and Orgun 1998], 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 outperform an existing and highly competitive on-line clustering algorithm (DSTC [Bullat and Schneider 1996]) in a variety of situations.

The disruptive 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-organisation, the clustering algorithm must select the portion to re-cluster first. We term this the selection problem. To our knowledge, this problem has not been previously identified in the literature. To solve this problem we propose that it should be the worst clustered memory-resident page that should be selected first for re-clustering. To this end, our on-line clustering framework, incorporates prioritisation.

2 Related Work

The re-organisation 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-organisation. However, we are aware of only one algorithm [Wietrzyk and Orgun 1999] that takes care to not to introduce extra I/O during the re-organisation phase. Wietrzyk and Orgun [1999] 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-organised and the remaining objects are only re-arranged as they are loaded into memory. However, the objects considered for re-organisation 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-organising those objects. This makes on-line clustering algorithms produced by OPCF more opportunistic[5] than existing algorithms.

The incremental nature of on-line clustering requires that only a small portion of the entire database to be re-clustered at each iteration. However, the choice as to which portion to re-cluster is where many existing algorithms differ. McIver and King [1994] suggest targeting the portion that was accessed after the previous re-organisation. However, this may involve a very large portion of the database if the re-clustering is not triggered frequently. Wietrzyk and Orgun [1999] re-cluster effected objects as soon as an object graph modification occurs. They use a threshold mechanism[6] to determine when re-clustering is worthwhile. However, this approach may still be too disruptive. An example of when its disruptiveness is likely to be felt is when the system is in peak usage and frequent object graph

[2] Pages where the objects reside.

[3] Normal database operation as opposed to re-clustering operation.

[4] Note that in an optimistic system this would translate to transaction aborts.

[5] By opportunistic we refer to our opportunistic use of in-memory data in order to minimise I/O.

[6] Based on the heuristic that says that a more frequently accessed object that is clustered badly is more worth while re-clustering.

modifications are occurring. In such a scenario the object graph would be continuously re-clustering during peak database usage. The algorithm thus lacks a means of controlling when the re-clustering takes place. In contrast, the on-line algorithms developed with OPCF can be easily made adaptive to changing system loads. This is due to the fact re-clustering can be triggered by an asynchronous dynamic load-balancing thread rather than a object graph modification.

A large body of work exists on static clustering algorithms [Tsangaris and Naughton 1991; Tsangaris and Naughton 1992; Banerjee et al. 1988; Gerlhof et al. 1992; Gerlhof et al. 1993; Drew et al. 1990; Wietrzyk and Orgun 1998]. However only relatively few static algorithms have been transfered into on-line algorithms. McIver and King [1994] combined the existing static clustering algorithms, Hudson and King [1989] and Banerjee, Kim, Kim, and Garza [1988] to create a new on-line clustering algorithm. However Hudson and King [1989] and Banerjee, Kim, Kim, and Garza [1988] are only sequence-based clustering algorithms which have been found to be inferrior when compared to graph partitioning algorithms[Tsangaris and Naughton 1992]. Wietrzyk and Orgun [1999] 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 [Bullat and Schneider 1996].

3 Organisation

In section 4 we describe a framework in which existing off-line clustering algorithms can be modified to operate on-line. In section 5 we describe two existing metrics for measuring clustering quality. In sections **??** and **??** we describe how OPCF can be applied to two existing off-line clustering algorithms, probability ranking principle algorithm (PRP) and greedy graph partitioning algorithm respectively. In section 8 we give a brief description of an existing on-line clustering algorithm, DSTC. In section 9 we present results obtained from running the algorithms on-line PRP, on-line greedy graph partitioning and finally DSTC [**?**].

4 The Opportunistic Prioritised Clustering Framework (OPCF)

The opportunistic prioritised clustering framework offers a generic way in which existing off-line clustering algorithms can be made on-line. There are two key properties of OPCF: I/O opportunism, and prioritisation of re-clustering, so the worst clustered page is re-clustered first. The framework ensures the resulting clustering algorithm can be made opportunistic but does not limit the algorithms to opportunism.

OPCF works at the page grain, instead of object grain. This means that when re-clustering occurs, all objects in an integer number of pages are re-clustered. This contrasts with on-line object grain algorithms like DSTC [**?**] where individual objects that are determined to need re-clustering are removed from existing pages and placed into new pages.

In order to apply OPCF, a series of steps must be applied. These steps are outlined below.

- *Define Incremental Reorganisation Algorithm*: In this step, a strategy is developed by which the existing off-line clustering algorithm is adapted to work in an incremental way. That is, at each iteration of reorganisation the algorithm must be able to operate with a limited scope.

- *Define Clustering Badness Metric*:

  OPCF prioritises re-clustering by re-clustering the worst clustered pages first. This means there must be a way of defining the quality of clustering at a page grain. We term this the clustering badness metric. The way in which clustering badness is to be defined for a particular off-line clustering algorithm depends on the goal of the clustering algorithm. For instance, the goal of graph partitioning algorithms is to satisfy the min-cut criteria and therefore for graph partitioning algorithms the min-cut criteria should be included in the clustering badness metric. In contrast, the PRP clustering algorithm has the goal of grouping hot objects together and therefore it may have a clustering badness metric that includes a measure of the concentration of cold objects in pages that contain hot objects.

  At each clustering analysis iteration[7] a user defined number of pages (NPA) have their clustering badness calculated. Once the page's clustering badness is calculated, the clustering badness is compared against a user-defined clustering badness threshold (CBT). If the page has a higher clustering badness value than the threshold then the page is placed in a priority queue sorted on clustering badness. At each reorganisation iteration a page is

[7] Cluster analysis simply refers to calculating clustering badness of pages of the store.

Parameter Abbreviation	Parameter Description
N	the number of objects accessed before a cluster analysis is triggered
CBT	cluster badness threshold
NPA	the number of pages analysed in each cluster analysis iteration
NRI	the number of re-reorganisation iterations performed after each clustering analysis iteration

Table 1: Description of user specified parameters of OPCF.

removed from the top of the priority queue and used to determine the scope of reorganisation for that reorganisation iteration. A user-defined number (NRI) of re-organisation iterations are performed at the end of each clustering analysis iteration.

- *Define Scope of Reorganisation*: To limit the work done in each reorganisation iteration of the on-line clustering algorithm, a limited number of pages must be chosen to form the scope of reorganisation. The scope of reorganisation should be chosen in such a way that re-organisation of those pages will produce the maximum amount of improvement in clustering quality while preserving the property of incrementality.

  The way the scope of reorganisation is chosen dictates whether the clustering algorithm is opportunistic or non-opportunistic. If the scope of reorganisation is chosen in such a way that only in-memory pages are included, then the resulting on-line clustering algorithm is opportunistic, otherwise it is not.

- *Define Cluster Placement Policy*:

  Because OPCF works at a page rather than object grain, the initial stages of each reorganisation iteration target an integer number of pages and so will, in general, identify multiple clusters, some of which may be small.[8] The existence of clusters which are smaller than a page size raises the important issue of how best to pack clusters into pages.

  A simple way in which cluster analysis can be triggered in OPCF is by triggering cluster analysis when a user specified number of objects has been accessed (N) this is similar to the technique used in **?**]. However any other triggering method may be used, including triggering via asynchronous thread for load balancing reasons.

  Table 4 provides a table of user specified parameters for OPCF.

5 Two Metrics Used to Measure Quality of Clustering

Tsangaris and Naughton [1991],[1992] proposed two metrics for measuring the quality of an object clustering—working set size and long term expansion factor.

Working set size (WSS(M)) [Tsangaris and Naughton 1991] is a metric for locality that is cache replacement policy independent. WSS(M) is evaluated by taking M frame requests, eliminating duplicates and computing the cardinality of the resulting set. Therefore the larger the cardinality, the fewer the duplicates, hence the lower the locality. A clustering algorithm that achieves a lower value for this metric will perform well on workloads that traverse a small portion of the database starting with a cold cache.

Long term expansion factor EF_∞ [Tsangaris and Naughton 1992] is an indicator of the steady state performance of an object clustering algorithm when the cache size is large. EF_∞ is the ratio of pages accessed in the steady state (N_∞) to the number of pages that would be required ideally to pack all active objects (n_∞).

These clustering metrics were designed for off-line clustering algorithms and were intended to be independent of buffer size and buffer replacement policy. These clustering metrics were designed for off-line clustering algorithms and were intended to be independent of buffer size and buffer replacement policy. However, for opportunistic on-line clustering algorithms where only objects in memory are used for clustering, these metrics are no longer independent of buffering effects. Despite this fact they are good tools for discussing the relative merits of existing static clustering algorithms

6 Probability Ranking Principle Clustering Algorithm (PRP)

The simplest off-line clustering algorithm is the probability ranking principle (PRP) algorithm. PRP just involves placing the objects in the object graph in decreasing heat (where 'heat' is simply a measure of access frequency). This

[8] When reorganisation occurs at an object grain, each reorganisation can be more strongly targeted towards a particular cluster or clusters, and so be less likely to identify small clusters.

surprisingly simple algorithm yields near optimal long term expansion factor [Tsangaris and Naughton 1992]. The reason that PRP achieves near optimal expansion factor is that it groups together those objects that constitute the active portion of the database. Therefore when the size of the active portion of the database is small relative to the available cache size and the steady state performance of the database is of interest, then this algorithm yields an near optimal solution. However, when a small traversal is conducted on a cold cache, PRP tends to perform poorly for working set size when compared to more complex clustering algorithms that take object relationships into consideration [Tsangaris and Naughton 1992].

The simplicity of the PRP algorithm combined with its minimal requirements with respect to statistics makes it particularly suitable for on-line clustering[9]. However, to our knowledge, an on-line version of PRP has been suggested before suggested in the literature.

6.1 On-line PRP Clustering Algorithm

In this section we describe the application of OPCF to the PRP clustering algorithm to give it on-line capability.

- *Reorganisation Algorithm*: In order to make PRP work in an incremental fashion, a logical ordering based on heat is placed on the pages of the store. The clustering algorithm incrementally re-arranges the objects so as to slowly migrate cold objects to cold pages and hot objects to hot pages.

  At each reorganisation iteration, the algorithm reorders the set of objects that lie within the pages targeted for that iteration according to heat order, the hottest objects moving to the hottest page, the coldest to the coldest page, etc.

- *Clustering Badness Metric*: The goal of PRP clustering algorithm is to map the active portion of the database into as few pages as possible. It accomplishes this by migrating hot objects towards one portion of the store while migrating cold objects in the other direction. In order to achieve this objective, we have defined a clustering metric which says a page is worse clustered if it contains both hot objects and a lot of waste. We define waste to mean space consumed by cold objects. The intuition behind this definition of clustering badness is that pages which contain hot objects but also a lot of waste is both very likely to be in cache and also wasting a lot of cache space, and thus unnecessarily displacing other hot objects.

  The definition of clustering badness is as follows:

$$CB(p) = \sum_{i \in p} heat_i \times \sum_{i \in p} (size_i / heat_i) \tag{1}$$

  The second term in the equation is a measure of the waste in the page. Therefore a larger and colder object in a page will contribute more waste.

- *Scope of Reorganisation*:

  The scope of each reoganisation is defined as three pages which are adjacent in heat-order, where the middle page is the target page for that iteration and the target page is chosen to be the page which is currently worst clustered. When I/O opportunism is used, the two *in-memory* pages closest to the target are selected (as the adjacent pages may be on disk). See figure 6.1 for an example.

  This definition of scope of reorganisation gives the clustering algorithm a high degree of incrementality. In addition, this gives the clustering algorithm an opportunity to improve the quality of clustering by placing the colder objects in the logically colder page and hotter objects in the logically hotter page.

- *Cluster Placement Policy*: Since PRP does not produce clusters of objects it does not have a cluster placement policy.

[9] Where CPU and I/O resources are precious

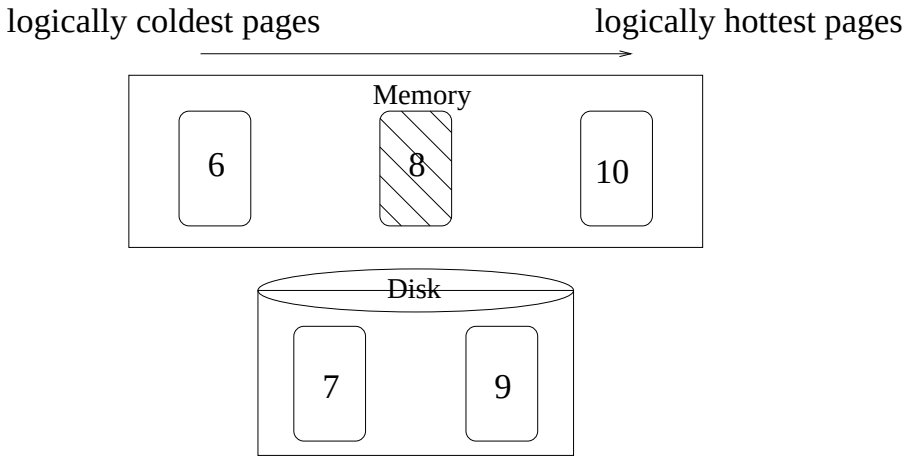


Figure 1: In this example the currently worst clustered page is 8, so the scope of reorganisation for opportunistic on-line PRP is pages 6, 8 and 10 (where page numbers reflect heat-order). If I/O opportunism is not used, the scope would be pages 7, 8 and 9.

7 Graph Partitioning

Partition-based clustering algorithms consider the object placement problem as a graph partitioning problem in which the min-cut criteria is to be satisfied for page boundaries. The vertices and edges of the graph are labeled with weights. Vertex weights represent object size and depending on the clustering algorithm, edge weights represent either the frequency with which a reference between the pair of objects is traversed, or the frequency with which the pair objects were accessed in close temporal proximity.

There are two types of partition based static clustering algorithms: *iterative improvement* and *constructive partitioning*. Iterative improvement algorithms such as the Kernighan-Lin Heuristic (KL) [Kernighan and Lin 1970], iteratively improve partitions by swapping objects between partitions in an attempt to satisfy the min-cut criteria. Constructive algorithms such as greedy graph partitioning (GGP) [Gerlhof et al. 1993] attempt to satisfy the min-cut criteria by first assigning only one object to a partition and then combining partitions in a greedy manner.

The study carried out by Tsangaris and Naughton [1992] indicates that graph partitioning algorithms perform best for both the working set size metric and long term expansion factor metric. However, they are generally more expensive in terms of CPU usage and statistic collection than sequence based algorithms [Banerjee et al. 1988; Drew et al. 1990; Benzaken and Delobel 1990].

7.1 On-line Graph Partitioning

This section outlines how we applied the opportunistic prioritised clustering framework onto off-line graph partitioning algorithms.

- *Reorganisation Algorithm*: At each reorganisation iteration the graph partitioning algorithm is applied to the pages in the scope of reorganisation as if these pages represent the entire database.
- *C*lustering Badness Metric: The off-line graph partitioning algorithms attempt to satisfy the min-cut criteria. This means that they minimise the sum of edge weights that cross page boundaries. In order to include this criteria into our clustering badness metric we have included external tension in the metric. External tension is the sum of weights of edges of the clustering graph which cross page boundaries. A page with higher external tension is worse clustered. In addition, heat is included in the metric to give priority for reorganising hotter pages. Below is a definition of clustering badness for graph partitioning algorithms:

$$CB(p) = \sum_{i \in p} heat_i \times \sum_{i \in p} external\ tension_i \tag{2}$$

The calculation of external tension differs between the opportunistic version of the on-line graph partitioning algorithm and the non-opportunistic version. In the opportunistic version, the external tension is calculated from only weights of edges that cross the page under consideration to other in-memory pages. By contrast, the non-opportunistic algorithm also counts edge weights that crosses page boundaries onto disk pages.

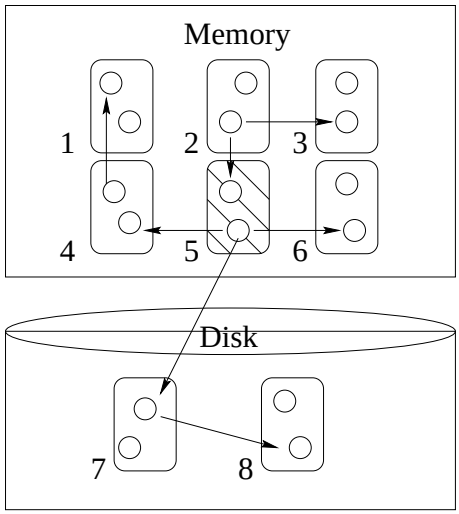


Figure 2: In this example the worst clustered page is 5 and the scope of reorganisation for opportunistic on-line graph partitioning are pages 2, 4, 5 and 6. The scope of reorganisation for non-opportunistic on-line PRP are pages 2, 4, 5, 6 and 7.

- *Scope of Reorganisation*: The scope of reorganisation is the worst clustered page and its related pages. Related pages are defined as pages which have at least one object link crossing the boundary between the two pages. If the on-line clustering algorithm is to be run opportunistically then only in-memory related pages are in the scope of reorganisation. See figure 7.1 for an example.
- *Cluster Placement Policy*: For this application of OPCF, we have chosen to place clusters into pages in order of heat. The reason for this choice is that cold clusters will be placed away from hot clusters and thus pages containing hot clusters which are more likely to be in memory will have less wasted space occupied by cold clusters. This is similar to the goal of PRP.

The particular graph partitioning algorithm implemented for the results section of this paper is the greedy graph partitioning (GGP) algorithm [Gerlhof et al. 1992]. However, the above methodology can be applied to any off-line graph partitioning clustering algorithm. GGP first places all objects in a separate partition and then iterates through a list of edges in descending edge weight. If the two objects on the ends of the currently selected edge are in different partitions and the sum size of the two partitions are smaller than a page then the partitions are joined.

8 Dynamic Statistical and Tunable Clustering Technique (DSTC)

DTSC is an existing dynamic clustering algorithm [**?**] which has the feature of achieving dynamicity without adding high overhead and excessive volume of statistics.

The algorithms is divided into five phases:

- *O*bservation Phase: In order to minimise disruptiveness of statistics collection, DSTC only collects statistics at predefined observation periods and the information is stored in a transient observation matrix.
- *Selection Phase*: In order to reduce the volume of statistics stored, at the selection phase the transient observation matrix is scanned and only significant statistics are saved.
- *Consolidation Phase*: The results of the selection phase are combined with statistics gathered in previous observation phases and saved in the persistent consolidated matrix.
- *Dynamic Cluster Reorganisation*: Using the information in the updated consolidated matrix, new clusters are discovered or existing ones are updated. In order to achieve dynamicity the re-organisation work is broken up into small fragments called clustering units.
- *Physical Clustering Organisation*: The clustering units are finally applied to the database in an incremental way (that is one clustering unit at a time). This phase is triggered when the system is idle.

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	20000
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

(a) OCB parameters

Parameter Description	Value
system class	centralised
disk page size	4096 bytes
buffer size	varies
buffer replacement policy	LRU-2
pre-fetch policy	none
multiprogramming level	1
number of users	1
object initial placement	optimised sequential

(b) VOODB parameters

Table 2: Parameters used for OCB and VOODB. VOODB parameters involving time have been omitted from this table, since the results reported are in terms of I/O performance.

DSTC is not an opportunistic clustering algorithm since its scope of re-organisation can be objects that are currently residing on disk. In order to make DSTC opportunistic we have chosen to restrict the scope of objects that can be chosen from when forming clustering units to those objects that are currently in memory. In the results section, results of both opportunistic and non opportunistic DSTC are presented.

9 Results

In this section we present results of experiments we conducted with the object clustering benchmark OCB [**?**] using the virtual object oriented database simulator, VOODB [**?**]. VOODB is based on a generic discrete-event simulation framework. Its purpose is to allow performance evaluations of OODBs in general, and optimisation methods like clustering in particular.

OCB is a generic object-oriented benchmark that was designed to benchmark OODBMS 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 experiments conducted in this paper the objects varied in size from 50 to 1200 bytes and the average object size was 268 bytes. The parameters of OCB and VOODB used to conduct the experiments in this paper are specified in table 9.

Throughout the remainder of this section, we will use the following abbreviations:

NC no clustering

DSTC dynamic statistical and tunable clustering technique [**?**]

GP greedy graph partitioning [Gerlhof et al. 1993]

PRP probability ranking principle clustering algorithm [Tsangaris and Naughton 1992]

The algorithm names are given the suffix '**N**' if the algorithm run was non-opportunistic and '**O**' if the algorithm was opportunistic.

The read workload used in the experiments consisted of simple traversals, hierarchical traversals and stochastic traversals. The update transactions used consisted of object attribute updates, object insertions, object deletion, object link insertions and object link deletions.

The parameters used for the clustering algorithms investigated the results of this paper are presented in table 9. For a description of DSTC parameters, see [**?**].

9.1 Varying Buffer Size Experiment

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. (Skewing the access distribution has the effect of highlighting the importance of clustering. An algorithm that does not cluster well will perform very poorly under such a workload.)

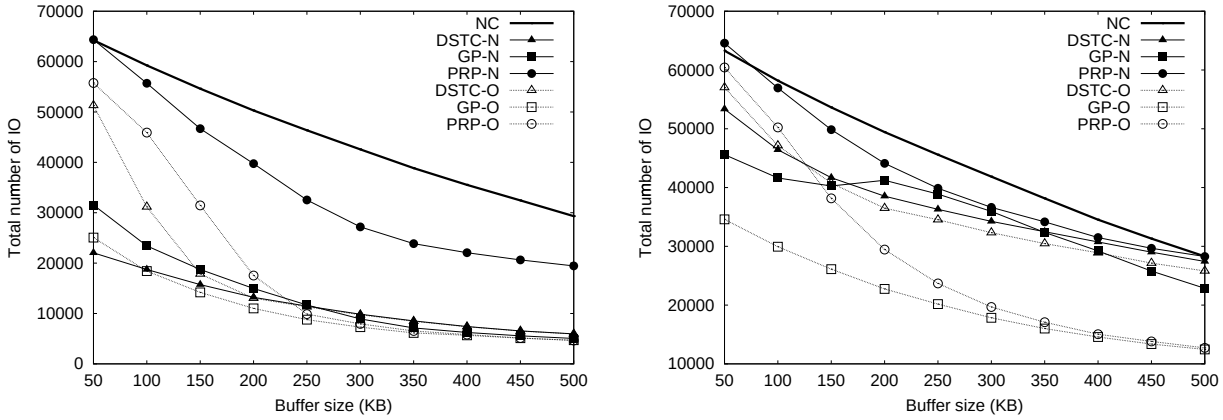
Parameter	Value
n	200
n_p	200
p	1000
T_{fa}	0.0
T_{fe}	0.0
T_{fc}	0.0
w	0.0
s	10

(a) DSTC parameters

Parameter	PRP Value	GP Value
N	200	200
CBT	0.1	0.1
NPA	50	50
NRI	25	25

(b) OPCF parameters (see table 4)

Table 3: Parameters used for the clustering algorithms DSTC, PRP and GP.



(a) Read only transactions

(b) 10% update transactions

Figure 3: Effects of varying buffer size

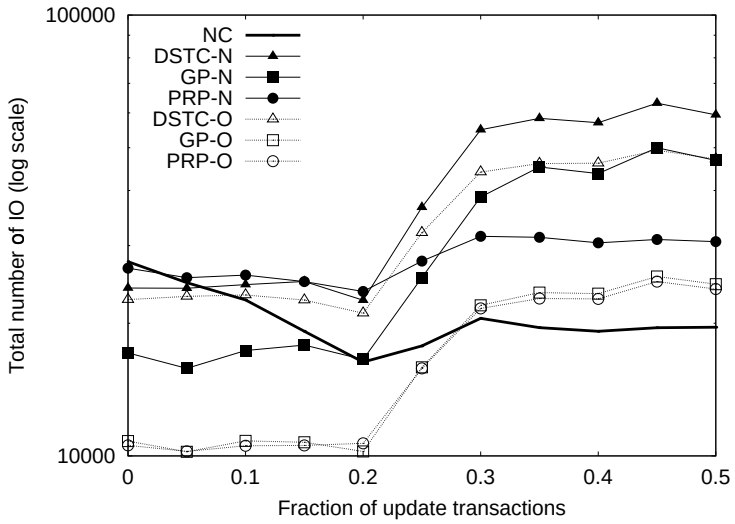


Figure 4: Effects of varying update fraction

The results are shown on figure 3. 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.

9.2 Varying Fraction of Updates

In this experiment we investigated what effect varying the faction of update transactions has on the performance of the on-line clustering algorithms. For this experiment we set the buffer size to 600 KB and the other system settings were the same as in section 9.1. The results of this experiment can be seen in figure 4.

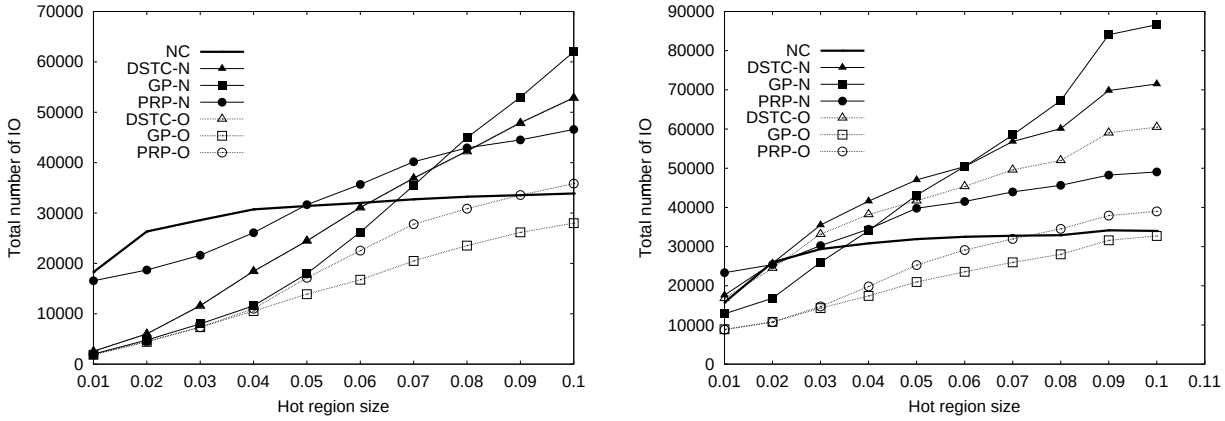
As observed in section 9.1, GP-O and PRP-O are the best performers when the buffer size is large. What is interesting here is the point at which clustering becomes worse than no clustering for the different algorithms. PRP-O and GP-O do not perform worse than no clustering until the fraction of update transactions is greater than 0.27 however DTSC-O becomes worse than no clustering at a fraction of 0.08.[10] This seems to indicate that the opportunistic algorithms proposed by this paper are more robust to update transactions when compared to DSTC. The reason for this can again be explained by the fact GP and PRP have less empty spaces among pages. Therefore random updates become less dispersed when compared to DSTC which places every cluster on a separate page.

9.3 Varying Hot Region Size

In this experiment we investigated the effect that varying the size of the hot region has on the performance of the on-line clustering algorithms. The buffer size was again set to 600 KB. The remaining settings with the exception of hot region size (which we will vary) were the same as for section **??**.

Figure 5 shows the results of running the experiment with both read only transactions and 10% update transactions. The most important observation that can be made from these graphs is that as the hot region's size increases, GP-O outperforms DSTC-O by an increasing margin. A possible reason for this observation is that when the size of the hot region is small, the hot objects are initially dispersed very thinly across the pages of the database and therefore a clustering algorithm such as GP which works at the page grain can only find a few clusters per re-clustering iteration. However DSTC-O, which picks out objects that belong to the same cluster from anywhere in memory, and DTSC-N, which picks from anywhere in the database, can quickly form large clusters even if the objects are dispersed among

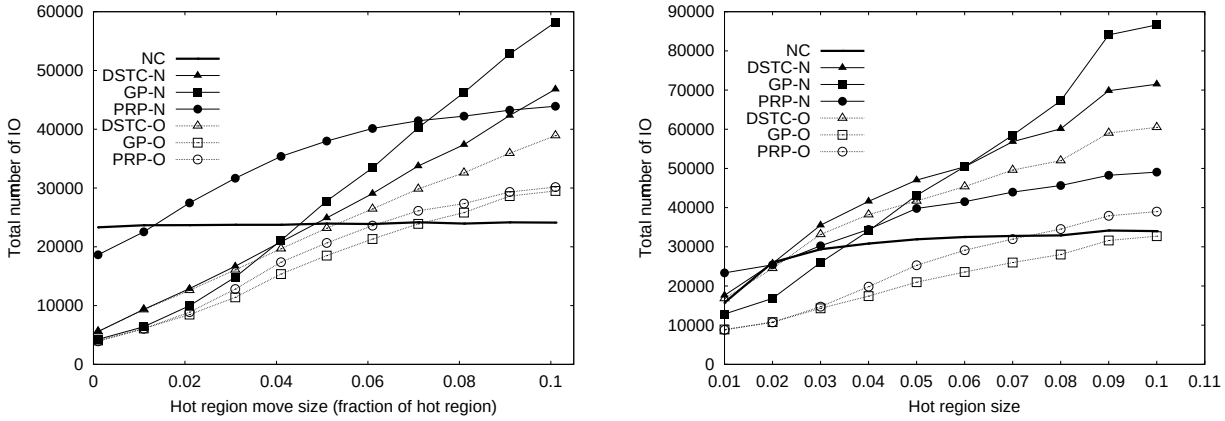
[10] This is not inconsistent with the results of section 9.1 since the buffer size used in this experiment is slightly larger than the maximum buffer size used in section 9.1



(a) read only transactions

(b) 10% update transactions

Figure 5: Effects of varying hot region size



(a) read only transactions

(b) 10% update transactions

Figure 6: Effects of varying the size by which the hot region moves in each move iteration

many pages. However this advantage begins to diminish as the hot region size increases and every page begins to have a larger portion of hot objects. That is to say, as more hot objects appear in every page, each re-organisation iteration of GP (page grain re-clustering) becomes more productive. The bad performance of GP-N may be attributed to its tendency to generate excessive read I/O as the hot region size increases. As the hot region size increases, GP-N's scope of reorganisation increases dramatically because its scope of reorganisation is all the worst clustered page's related pages (anywhere in the database). Therefore any productive work that the algorithm performs gets quickly overshadowed by the excessive I/O generated by the clustering processes.

9.4 Changing Transaction Access Patterns

This experiment investigates the effect of changing transaction access patterns on the various on-line clustering algorithms. Due to their on-line nature, it is desirable for the on-line clustering algorithms to perform well even when the transaction access patterns change.

In this experiment we gradually moved the hot region of the store to simulate changes in access patterns. After 1000 objects has been accessed the hot region is moved by a certain fraction of the size of the hot region. The size by which the hot region moves at each move iteration is varied in this experiment. The size of the hot region is set to 1.5% and the remaining parameters of the system are set to the same values as for section 9.3.

The results for this experiment are depicted on figure 6. The most important observation from figure 6 is that as the rate of change in the access pattern increases there is not much change in the relative performance difference between the opportunistic versions of the different algorithms. The performance difference between DSTC-O, PRP-O and GP-O stays almost the same as the amount by which access pattern increases. However, the performance difference between opportunistic and non-opportunistic versions of GP and DSTC increases as the amount of access pattern change increases. The greater the change in access pattern, non-opportunistic versions of GP and DSTC progressively lag further behind their opportunistic counterparts. This seems to indicate that opportunism dampens the negative effects that changes in access patterns have on the performance of clustering algorithms. This may be explained by the fact that opportunistic algorithms pay less for any unhelpful speculative re-organisation,[11] since it does not incur the overhead of the additional clustering read I/O.

10 Conclusions

In this paper we have presented OPCF, a generic framework which when applied to off-line clustering algorithms, can produce on-line clustering algorithms that possesses the two desirable properties of *opportunism* and *prioritisation of clustering*. In addition, application of the framework is straightforward and yet it produces clustering algorithms that outperform an existing highly competitive on-line clustering algorithm, DSTC [**?**], in a variety of situations.

When update transactions were introduced into the workload, the OPCF-derived algorithms, GP-O and PRP-O, were found to maintain an high level of performance. However DSTC-N and DSTC-O were found to respond poorly to the introduction of update transactions. This indicates that DSTC is less robust to updates when compared to GP-O and PRP-O.

When the hot region size increases DSTC-O and DSTC-N were found to progressively perform worse than GP-O. This indicates GP-O is more robust to increases of hot region size when compared to DSTC.

Overall the OPCF algorithms GP-O and PRP-O were found to be more robust to various changes in workload conditions than DSTC-O and DSTC-N. This suggests that GP-O and PRP-O are better candidates for inclusion in OODBMS systems which are likely to be run in varying workload conditions.

When choosing between PRP-O and GP-O, the better algorithm depends on the situation. In general, if the active portion of the database can fit into memory PRP-O is the better algorithm since its quality of clustering is about the same as GP-O and it has smaller overhead both in terms of statistics collection and clustering algorithm running time. However if the active portion of the database does not fit into memory GP-O may be the better algorithm since in this situation GP-O produces better quality clustering. These results are consistent with observations made for off-line graph partitioning and PRP algorithms.

When the pattern of access of changes, opportunistic algorithms were found to perform better than non-opportunistic algorithms. This is due to the fact opportunistic algorithms pay less for an unhelpful speculative reorganisation when compared to non-opportunistic algorithms.

For further work, we plan to add more existing on-line clustering algorithms to our simulation study. We would also like to experiment with transforming other off-line clustering algorithms using OPCF and see how they perform in relation to the algorithms presented in this paper.

Acknowledgement

Bibliography

AILAMAKI, A., DEWITT, D. J., HILL, M. D., AND WOOD, D. A. 1999. Dbmss on a modern processor: Where does time go? In *The 25th VLDB conference, Edinburgh, Scotland* (September 1999), pp. 266–277.

BANERJEE, J., KIM, W., KIM, S. J., AND GARZA, J. F. 1988. Clustering a dag for cad databases. In *IEEE Transactions on Software Engineering*, Volume 14 (November 1988), pp. 1684–1699.

BENZAKEN, V. AND DELOBEL, C. 1990. Enhancing performance in a persistent object store: Clustering strategies in o_2. In *Implementing Persistent Object Bases* (September 1990), pp. 403–412.

BULLAT, F. AND SCHNEIDER, M. 1996. Dynamic clustering in object databases exploiting effective use of relationships between objects. In *ECOOP'96, 10th European Conference on Object-Oriented Programming, July 8-12, 1996, Linz, Austria* (1996), pp. 344–365. Springer.

[11] A re-organisation which proves to be worthless since the access patterns for those just re-organised objects has changed.

DREW, P., KING, R., AND HUDSON, S. E. 1990. The performance and utility of the cactis implementation algorithms. In D. MCLEOD, R. SACKS-DAVIS, AND H.-J. SCHEK Eds., *16th International Conference on Very Large Data Bases* (Brisbane, Queensland, Australia, 13–16 Aug. 1990), pp. 135–147. Morgan Kaufmann.

GERLHOF, A., KEMPER, A., AND MOERKOTTE, G. 1996. On the cost of monitoring and reorganization of object bases for clustering. In *ACM SIGMOD Record*, Volume 25 (1996), pp. 28–33.

GERLHOF, C., KEMPER, A., KILGER, C., AND MOERKOTTE, G. 1992. Clustering in object bases. Technical report, Fakultat fur Informatik, Universitat Karlsruhe.

GERLHOF, C., KEMPER, A., KILGER, C., AND MOERKOTTE, G. 1993. Partition-based clustering in object bases: From theory to practice. *In Proceedings of the International Conference on Foundations of Data Organisation and Algorithms (FODO)*.

HUDSON, E. AND KING, R. 1989. Cactis: A self-adaptive, concurrent implementation of an object-oriented database management system. In *ACM Transactions on Database Systems* (September 1989), pp. 291–321.

KERNIGHAN, B. AND LIN, S. 1970. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 291–307.

KNAFLA, N. 1999. *Prefetching Techniques for Client/Server, Object-Oriented Database Systems*. PhD thesis, University of Edinburgh.

MCIVER, W. J. J. AND KING, R. 1994. Self-adaptive, on-line reclustering of complex object data. In R. T. SNODGRASS AND M. WINSLETT Eds., *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994* (1994), pp. 407–418. ACM Press.

TSANGARIS, M. M. AND NAUGHTON, J. F. 1991. A stochastic approach for clustering in object bases. In *Proceedings of the ACM SIGMOD conference on Management of Data* (1991), pp. 12–21.

TSANGARIS, M. M. AND NAUGHTON, J. F. 1992. On the performance of object clustering techniques. In *In Proceedings of the ACM SIGMOD conference on Management of Data* (1992), pp. 144–153.

WIETRZYK, V. S. AND ORGUN, M. A. 1999. Dynamic reorganisation of object databases. In *Proceedings of the International Database Engineering and Applications Symposium* (August 1999). IEEE Computer Society.

WIETRZYK, V. S. I. AND ORGUN, M. A. 1998. Clustering techniques for minimizing object access time. *Lecture Notes in Computer Science 1475*, 236–247.