The Simulation Evaluation of Heat Balancing Strategies for Btree Index over Parallel Shared Nothing Machines

Hisham Feelifl and Masaru Kitsuregawa
Institute of Industrial Science, The University of Tokyo
{hisham, kitsure} @tkl.iis.u-tokyo.ac.jp

In shared nothing machines the data are typically declustered across the system processing elements (PEs) to exploit the I/O bandwidth of the PEs. Notwithstanding, the access pattern is inherently dynamic; therefore, the system performance may degrade as some PEs become hot spot. Consequently reorganization for heat balancing is essential and should be online as well. Meanwhile, in order to achieve efficient query and transaction evaluation, data at each PE are indexed based on the Btree structures. Thus, data reorganization should satisfactorily deal with the index modification as moving data from "hot spot" PEs to cold PEs. In this paper, we propose online heat balancing strategies for Btree index over parallel shared nothing machines based on the principle of distributing the given heat as evenly as possible across the system PEs. Furthermore, the proposed strategies have the capability to reduce the instantaneous migration cost during the heat balancing process, which in turn avoid the harmful migrations that may degrade the system performance. We evaluate the performance of the proposed strategies in comparison with the well-known strategies. The conducted simulation shows their efficiency in correcting the system performance degradation.

1. Introduction

Shared-nothing parallel architectures are more able than other architectures to achieve two important objectives: high performance and the ability to smoothly increment the system growth by adding new nodes [OV91]. In such architectures, there are many processing elements (PEs); each PE has its processor, exclusive memory modules and one or more disk units. The PEs communicate with each other by sending messages via the communication network, the only shared resource. The data are typically declustered across the system PEs and the execution of a transaction or a query is distributed over the network. However, the access pattern is inherently dynamic, which in turn can lead to performance degradation as some PEs become "hot spot" (frequently accessed). Therefore, reorganization for heat balancing is essential. The basic motivation to investigate and realize heat-balancing facilities comes from simple experience that several applications in shared nothing systems usually do not exploit the system very good. Heat balancing is particularly challenging for evolving workloads, where the hot and cold data change over time. Data reorganization can only counteract such situations, and such reorganizations should be performed online without requiring the system to be quiescent [WZS91 & SWZ93]. Additionally, to achieve efficient query and transaction evaluation, data at each PE are indexed. Therefore, data reorganization should satisfactorily deal with the index modification as moving data from "hot spot" PEs to cold PEs (infrequently accessed) [AON96]. In [FK99], they propose an online heat balancing strategy for parallel indexed database, in which the data migration process is based on distributing the given heat as evenly as possible across the system PEs. In the paper, we extend their work by increasing the alternatives during the heat balancing process and tuning their strategy performance. The organization of the paper is as follows. In the next section, we briefly discuss the related work for online data reorganization and index modification. Section 3 establishes the system search structure and the considered migration strategy. Section 4 clarifies our considerations to the system workload and the proposed heat balancing strategies as well. Sec. 5 deals with the experimental work and finally, we conclude the paper.

2. Related work

Data reorganization should take place only when the benefit outweighs the cost [CABK88]. Though there has been much work in the area of online reorganization in the recent years In [WZS91 & SWZ93], the authors present an online method for the dynamic redistribution of data, which is based on reallocation of file, fragments. A limitation of their study is that they do not consider index modification. Perhaps [SD92] is the first paper that discusses a solution for online index reorganization. They outline the issues involved in changing of all references to a record when its primary identifier is changed due to a record move. The techniques in the [SD92, ZS96] are limited to centralized DBMS and require the use of locks, where using locks during reorganization can degrade performance significantly [AON96]. In [AON96] they examined the problem of online index reorganization. They present two alternatives for performing the necessary index modifications, called one-at-a-time OAT page movement and BULK page movement. While these alternatives are extremes on the spectrum of the granularity of data movement, they both use the conventional B+-tree algorithms which slowdown the migration process. To minimize the index modification cost, in [YKM99] they suggest the Fat-Btree as a powerful search structure that supports the data reorganization and speeds up the migration process. In [FK99] they propose an online strategy to reorganize the data with minimal cost of modifying the indexes, as distributing the given heat as evenly as possible across the system PEs. However, the migration decisions are carried out through one step, which may be harmful for the system performance. In contrast, this paper extends their algorithm by balancing the system in an incremental approach and preserving the principle of distributing the given heat as evenly as possible, as well.

3. The System Distributed Search structure

We assume that data are initially range partitioned across all the system PEs so that the access method can associatively access data for strict match queries, range queries and cluster data with similar values together. Using a B-tree based index enables more efficient processing of range queries than a hashed index, where only the nodes containing data in the specified range are accessed. One solution to associative access is to have a global index mechanism replicated on each PE [OV91]. Conceptually, the global index is a two-level index with a major clustering on the PE range and a minor clustering on some attribute of the relation. The first level directs the search to the PE wherein the data is stored. The second level of the index is a collection of Fat-Btrees, one at each PE; each Fat-Btree independently indexes the data at its PE [YKM99].

If the height of the Fat-Btree at the migration source and destination are the same, then the amount of data to be migrated correspond to the entirety of one or more branches of the Fat-Btree at the source PE. So that, it would be easy to prune the entirety of the branches from the Fat-Btree at the source PE as well as attaching these branches into the Fat-Btree at the destination PE using bulk-loading technique without excess overhead. We adopt our reorganization strategy on the advantages of the Fat-Btree in speeding up the migration process and consequently minimizing the reorganization cost.

3.1. Data Migration Strategy: Branches Migration

A source PE (PE*r*) can be defined as the PE from which the data pages (through the corresponding index branches) have to be moved to other PEs. Similarly, a destination PE (PE*d*) can be defined as the PE at which the data pages (and index branches) have to be stored. Since data is range partitioned, thereby we can only move data from one PE to its neighboring PEs (left or right or both), which hold the preceding or succeeding ranges, with two exceptions, the first deals with the "rightmost" PE, which has the capability of data migration only in the left direction, while the second deals with the "leftmost" PE, which has the capability of data migration only in the right direction. These migration directions represent the heat balance horizon. Assume that we initiate the migration process between a PE*r* and a PE*d*, where each of these PEs has its own search structure. Traditionally, the migrated data are inserted one at time (one record at time, one page at time), which can be very costly especially if the number of records (or pages) is very large. The idea behind the branch migration strategy is as follows:

1. Since every branch in the search structure represents a unique sub-range form the whole range of the key attribute, so branches migration means migration of sub-ranges from the PE*r* to the PE*d*. Every sub-range has its own access history (heat), therefore, if we balance the hot sub-ranges as even as possible across the system PEs, then, the system performance is higher.
2. The hot sub-ranges migration may be very costly as it includes index modification at the PE*r* and PE*d*. However, this critical cost could be kept minimal if both the PE*r* and PE*d* have flexible search structures that can support this kind of migration with minimal cost. Our objective with the search structures is just fulfilled with the distributed Fat-Btree.

1. Online Heat Balancing

Online heat balancing is done in four basic steps: monitoring PE workload, exchanging this information between PEs if it is necessary, calculating new distribution and making the work migrating decision, and the actual data migration.

4.1. The workload

The system workload is reflected by a metric, called *heat* [CAB88]. We define the heat of a range R = *{Rmin .. Rmax}* as the access frequency of R over some period of time. A range R as a logical or abstract quantity could be achieved at any physical quantity in the system such as a data page, an index branch, an index tree, and a PE, so that: Heat *(***R***)* = *Heat (***O***)*, where **O** = *{ data page, index branch, index tree, PE* } that holds the range **R** = {*Rmin.. Rmax*}.

The complexity of maintaining heat statistics on a range R varies from maximal to minimal cost cases, depending upon the physical quantity **O** that holds the range R. The maximal cost cases, may be appeared, if we maintain the heat statistics for every data page in the system, **O** = {data page}, which roughly requires maintaining statistics for every possible point in the whole range. Although this approach for the workload is very costly, but it has its own advantage in estimating an accurate figure of the workload. On the other extreme, the minimal cost cases could be achieved if we maintain the heat statistics for every tree (or PE) in the system, **O** = {PE}, which requires only information proportional to the number of the system PEs. Although this approach is not costly, but it has its own disadvantage in inaccurate estimation of the workload. On the middle of the spectrum, there are mid-cost cases, e.g., maintaining the heat statistics for every index branch in the system or even for every sub-tree at every root node in the system, O = {Sub-tree}. The main advantage of these mid-cost alternatives is; they provide some compromise solution in term of cost and accuracy for measuring the system workload and migration cost. In our simulation, we use the mid-cost approach of [FK99], in which they maintain the heat of every sub-tree at the root node of every PE in the system. Then, in order to minimize the heat statistics information, we assume a uniform heat distribution in the deeper levels (than the root). However, we consider the selection of the workload is a "design parameter" which depends upon the applications and their requirements.

4.2. The Heat Balancing Strategies

In this section, we consider the heat balancing strategies for Btree index over parallel shared nothing machines.

4.2.1. The Full Window Strategy.

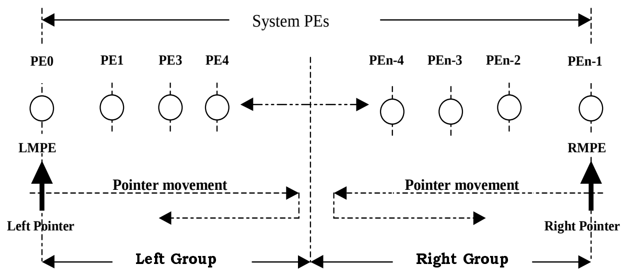


Figure 1. The "full window" strategy notation,

In [FK99], they observe the special migration directions at the "right most" PE (RMPE) and the "left most " PE (LMPE) that exist with the range partition strategy. Their algorithm starts by recording the possible migrations at both the RMPE and LMPE of the system. Then, by dropping virtually these RMPE and LMPE from the system, there will be new RMPE and LMPE at which the process can be repeated again, see Fig. 1. The process is repeated until the system is virtually vanished after storing the migration sequence -which is needed to balance the system- into a structure called "migration directory" [FK99]. After executing the migration directory decisions, the resulting system is heat balancing as evenly as possible. We refer to this strategy as the "*full window*" strategy.

4.2.2. Window of PEs Heat Balancing Strategies.

The number of PEs in the system is one of the parameters influencing heat-balancing decisions. More PEs make it more probable for any "hot spot" PE to find a cold PE to migrate its data. This positive effect competes with the negative effect produced by the larger message passing overhead and highly cost migration created by the large system size. In order to optimize the migration cost with the number of PEs, we have to clarify first, what are the major parameters that greatly affect the migration cost in the "full window" strategy. With the "full window" strategy, the migration cost to balance a PE in the system of "N" PEs is proportionally related to the following quantities; the "excess/missing heat" and the "balancing distance" at this PE. The "excess/missing heat" (*EMH*) is the amount of excess or missing heat that to be migrated to balance this PE, which equals *Heat (PEi) – Average heat.* The balancing distance (BD), expressed in term of the number of PEs, is the distance between the encountered PE and the PE at which it will donate (or absorb) the "excess/missing heat" of the encountered PE. So that, the migration cost to balance this PEi can be approximated as:

$$\textit{Migration cost (PEi)} = C * \textit{EMH(PEi)} * \textit{BD (PEi)}$$

Where *C* is a constant depends on the system communication and search structure parameters. The system migration cost is the summation of the migration cost at every PE in the system.

Clearly, as the system PEs number increases as the probability of highly "balance distance" increases thus increasing the migration cost. In addition, as the data skew increases the "excess/missing heat" increases thus increasing the migration cost. Generally, if we assume distributing the given heat as evenly as possible, then in order to decrease the system migration cost we have to decrease the "excess/missing heat" or the "balancing distance" or both. This can be achieved by shrinking the balancing horizon, the considered number of PEs, during the heat balancing process. I.e., instead of considering the whole PEs as in the "full window" strategy, we can only consider some of PEs or simply a window of PEs. Within a window of PEs, we can fit the principle of distributing the given heat as even as possible as applying the "full window" strategy within the window. Therefore, the instantaneous migration cost is reduced as a result of reducing the "excess/missing heat" and the "balancing distance" at each PE. Although this will imply some sacrifice in the heat balancing, but it may provides some compromise between the heat balancing requirements and the requirement for minimizing the instantaneous migration cost. In the following subsections, we consider the alternatives in selection of both the window location and its size.

4.2.2.1. The Three Windows Strategy.

In this strategy, we first reduce the balancing horizon by partitioning the system PEs into two windows; right window and left window as shown in Fig. 2. Each window has as size of N/2, where N is the system PEs number. We apply independently the "full window" strategy on each window. Then, in order to preserve the principle of distributing the heat as evenly as possible across the system, we introduce a third window called the "link window". The link window is a virtual window that has some members that belong to the right window and the other members belong to the left window (see Fig. 3). The "link window" provides a balancing linkage between two virtually independent balancing windows (horizons) and it gives the compensation for partitioning the balancing horizon. Thus if we apply the "full window" strategy within the link window we have instantaneously (to some extend) a heat-balanced system. By repeating the above procedure, we finally have a completely heat-balanced system. The balancing iterations are reached to its final as the "link window" finalizes completely the transfer of the "excess/missing heat" at one window (left or right) to the opposite window. The "link window" size is one parameter that influences the convergent time of the "three windows" strategy, so that small sizes lead to higher convergent time than that of large sizes. Generally, it can be chosen to satisfy the convergent time requirement, however, in our simulation, we assume the "link window" size is N/2, where N is the PEs number. The advantage is balancing the system without any additional overheat for searching for the optimum window (location and size). Partitioning the system PEs into "M" disjoint windows can extend the above strategy; each has a size of K, where K= N/M. The value of K ranges form 2 to N ("full window"), see Fig. 2.B. These "M" windows will consequently require "M-1" link windows, so that the total windows number = 2M-1, each of K size. Fig. 5 outlines the mentioned strategy.

One of the main characteristics of the "three windows" strategy over the "full window" strategy its incremental balancing to balance the system. Therefore, instead of doing the balancing decisions in one step, which may be harmful for the instantaneous system performance, incremental balancing is done, with some convergent time, so that the partial migrations may not harmful for the instantaneous system performance. This is so beneficial in highly-dynamically changing access pattern environments. Meanwhile, because the nature of the "three windows" strategy is based on local heat balancing, therefore the useless migrations (but without instability) is one of its disadvantages.

4.2.2.2. The Bottleneck Window Strategy.

With this strategy, we introduce some heuristic into the "full window" strategy for tuning its performance as reducing its instantaneous migration cost. First, we reduce the balancing horizon, by initially searching for the window that covers all the bottleneck PEs in the performance (their heat is higher than a threshold value), we call this window as the "bottleneck window". Balancing of such window may be not so beneficial to the system performance, because in general most of its candidates are "hot" PEs that need some other cold PEs to balance their heats with them. This suggests the extension of the "bottleneck window" in the right or left direction or both to include other cold PEs in the "bottleneck window". The extension of the "bottleneck window" is mainly dependent on the objective of the balancing strategy, so that if the objective were reduction in the migration cost, then it would be better to include the cold neighbors, which have minimum requirements (in term of the migration cost) on their balancing.

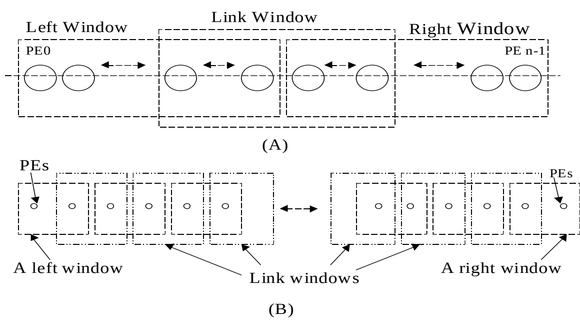


Figure 2. The three windows strategy: (A) Notation (B) Extension.

```
void    FullWindowAlgorithm (int Left, int Right)
{ // apply the algorithm of [FK99] on the window defined
  // by the "Left" and "Right" variables. }

void 3WindowsAlgorithm (Left, Right,K)
{  // K= Number of PE on every considered window
    WindowSize=Right-Left+1;
    if(WindowSize==K) FullWindowStrategy(Left,Right);
    else if(WindowSize>K)
    { // Left window
      3WindowsAlgorithm(Left,(Left+Right-1)/2,K);
      // Right window
      3WindowsAlgorithm((Left+Right+1)/2,Right,K);
      if(Right>Left)       // Link window
     { Right-=WindowSize/2;    Left+=WindowSize/2;
       FullWindowStrategy(Left,Right);}}
```

Figure 3. A Description of the "three windows" strategy.

Since the migration cost is proportionally related to the "excess/missing heat", then it would be better to give the membership to the neighbors which have minimum missing heat. The "bottleneck window" extension procedure is continued as long as its total heat is less than some threshold value of the total given heat. Logically, the initial state of the "bottleneck window" covers where is the problem that degrades the system performance. Then the " bottleneck window" extension gives the horizon to solve the system problem with reasonable effort. As we obtain the "bottleneck window" in its final state as we can apply the "full window" strategy within it, so that the system is heat-balanced with some extent of distributing the given heat as evenly as possible. Fig. 4 gives its high level description.
The main advantages of the "bottleneck window" strategy are reduction in the instantaneous migration cost without the possibility of the useless migrations that could be arise with local schemes based strategies (e.g. the "three windows" strategy). In addition, its "balancing horizon" (location and size) is dynamically changing with the heat distribution and the predefined threshold so that the membership is changed dynamically. The limitation of this added heuristic comes whenever the bottleneck window covers all the system PEs so that its behavior in balancing and migration cost is identical to that of the "full window" strategy.

5. Simulation Result

In this section, we describe our experiments to study the performance of the online data reorganization with the mentioned heat balancing strategies: "full Window", "three windows" and "bottleneck window". The metric used is the impact on the response time of queries and the system migration cost. Table 1.0 shows the major system, database and query configuration parameters with their default values and variation settings. We use a functional shared nothing-parallel database system where each PE has its own disk(s) and memory. The system PEs communicate with each other by exchanging messages across the interconnection network, set at 120 Mbit per second, the communication bandwidth is hardly a bottleneck during reorganization, therefore, the communication delay is set at 2 msec. We first create an initial Fat-Btree with the tuple key values generated using a uniform distribution. Then we generate 2,000,000 range queries using Zipf- distribution, which concentrates the queries in a narrow key range. These queries are generated with skew that defined by the skew factor (τ) of Zipf-distribution. Therefore, there are more range queries issued at one PE than the other PEs, depending on the skew factor τ. The heat skew will initiate the migration of branches between the PEs, depending on the heat balancing strategy's threshold. We model each of the PEs as a resource and the queries as entities. We assume the heat balancing is done in centralized scheme and it is initiated after every 1000 queries.

```
// find the initial state of the bottleneck window
int BottleNeckWindow(int*Lef,int *Right)
{  int Max,Position,ck;  HeatType MaxHeat;
if (! ValidWindow(*Left,*Right) return –1;
Max=WindowMax(*Left,*Right);
if ( MaxHeat >= ThreSholdHeat )
  {  Position=Left+1;
     if((ck=BottleNeckWindow(&Position,Right)== -1 )
    *Right=Max;   else *Right=ck;
    Position=Left-1;
    if((ck=BottleNeckWindow(&Position,Right)== -1 )
   *Left=Max; else *Left=ck;
    return Max;
else return –1;}
// extent the bottleneck window, if it is possible.
HeatType ExtendWindow(int*Left,int*Right)
{  LeftHeat=RightHeat=0;
    if(*Left>0)  LeftHeat=Heat(*Left-1);
    if(*Right<ProcessorsNumber-1) RightHeat=Heat(*Right+1);
    if(RightHeat>LeftHeat)  { (*Right) ++;
         WindowHeat+=RightHeat;}
   else {(*Left)--;WindowHeat+=LeftHeat;}
   return WindowHeat
}
// bottleneck window algorithms – as a three steps algorithm.
void BottleNeckWindowAlgorithm (int*Left,int*Right )
{ BottleNeckWindow(Left,Right);
   while((WinHeat=ExtendWindow(*Left,*Right)<Threshold))
   FullWindowStrategy(Left,Right);}
```

Figure 4. A Description of the "bottleneck window" strategy.

Since all the considered strategies are based on the "full window" strategy, we first consider the "full window" strategy under two migration schemes: *coarse migration* where the migration granularity is the corresponding branches (or even sub-trees) to the amount of the migrated heat, and *incremental migration* where migration granularity is within two limits. The minimum granularity (other than zero) is only one index branch, while the maximum is only one sub-tree at the root node. These limits are globally unified across the system PEs; thus, any amount of excess heat is normalized such that the corresponding migration granularity is interpreted within these two limits. Fig. 5.0 shows the result of this experiment, we observe the system performance is much better with the *incremental migration* than with the *coarse migration*. In the same time, these figures affirm the effectiveness of reducing the instantaneous migration cost during the heat balancing process regardless of the total migration cost, if the assumed strategy evenly balance the system. This result leads us to consider *the incremental migration* in all of the next experiments.

Table 1: the parameters and their used values.

Parameter	Default values / variation
System Parameters:	
Number of PEs in the cluster	16 /32
Index node size	4K page
Network bandwidth	120 Mbits/s
Time to read or write a data page	8 ms
Database Parameters:	
Number of records	1 million
Key size	4 bytes
Query Parameters:	
Number of queries	2,000,000
Zipf distribution decay factor	0.2/ 0.5; 0.1 → 0.9
Mean arrival rate	5.0/second / 5.0 → 60.0
Mean service time	500 ms
Strategies threshold	
"Full window" threshold = heat standard deviation/average heat.	0.07
"Three window" threshold	0.07
"Bottleneck window" threshold	0.8/0.85;0.9 of the total heat

Second, we study the effect of heat balancing on the response time. Figure 6 compares the average response time without and with heat balancing of the considered strategies. It shows the effectiveness of heat balancing in reducing the average response time and increasing the system bandwidth. We observe that the response time of a query in the "hot spot" PE differs greatly from the system response time. Figure 6 shows the effect of migration on the response time of the "hot spot" PE. Figure 7 shows the scalability on the average response time with a cluster of 32 PEs and skew factor 0.5. These figures affirm the effectiveness of distributing the given heat as evenly as possible, on the response time of the system and the "hot spot" PE as well.

To evaluate the performance of the considered strategies; we record the system migration cost and we define a factor, namely, the average migration factor which equals the overall migration cost divide by the experiment time. Figure 7 compares the average migration factor for each strategy under the data skew variation. From this figure, we can observe that on average, the "bottleneck window" strategy has the minimum migration cost evaluation as result of decreasing the balancing horizon in the system. The reason for the steep variation in its cost is that; as the skew increases as the bottleneck PEs number and its extended candidates number decrease which decreases the "bottleneck window" size. This is also affirms the dynamic behaviour of the "bottleneck window" strategy. Figure 9 shows the "bottleneck window" sensitivity to its threshold. Figure 8 compares the strategies' convergent time, it shows that the "three windows" strategy has the highest convergent time as result of its incremental balancing.

6. Conclusion

In this paper, we have proposed a heat balance model including data migration with minimal cost of index modification on shared nothing machines that can be employed in many practical applications. Heat balancing is necessary because the optimal data placement changes with time and usage of the parallel database system, and significant performance improvements can be obtained by heat balancing. Heat balancing has been studied before, but no work addresses the critical issue of balancing the given heat across the system PEs with minimal index modification and reducing the instantaneous migration. This paper fills an essential void.

References

[**AON96**] K.J. Achyutuni, E. Omiecinski, and S. B. Navathe. *Two techniques for On-line Index Modification in Shared Nothing Parallel Databases*. Procs ACM SIGMOD 1996.

[**CABK88**] G. Copeland, W. Alexander, E. Boughter, and T. Keller. Data Placement in Bubba. *Proc. of ACM SIGMOD Conference, pages 99-108,1988.*

[**FK99**] H. Feelifl and M. Kitsuregawa. Online Heat Balancing for Parallel Indexed Database On Shared Nothing System, *Proceedings of 1999 database workshop, pp. 85-92, 1999.*

[**OV91**] M.T. Ozsu and P. Valduriez. Principles of Distributed Database Systems, *Prentice Hall, 1991.*

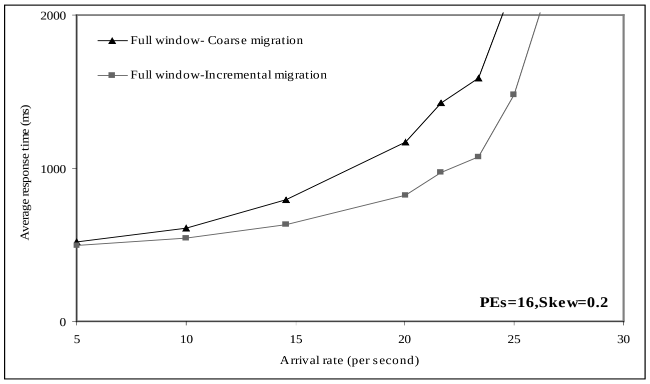
[**SD92**] B. Salzberg and A. Dimock. Principles of transaction-based on-line reorganization. *Procs. of the 18th Inter. Conf. on Very Large Databases, pages 511-520, 1992.*

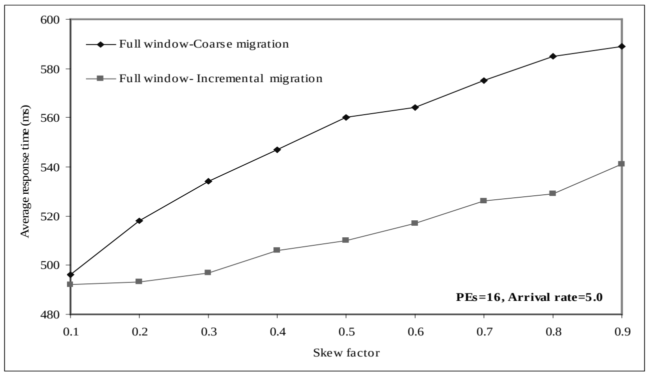
[**SWZ93**] P. Scheuermann, G. Weikum, P. Zabback. Adaptive Load Balancing in Disk Arrays. *Proceedings of the 4th Inter. Conf. on Foundations of Data Organization and Algorithms (FODO), 1993.*

[**WZS91**] G. Weikum, P. Zabbak, and P. Scheuermann. Dynamic file allocation in disk arrays. *Procs. of the ACM SIGMOD Conference, 1991.*

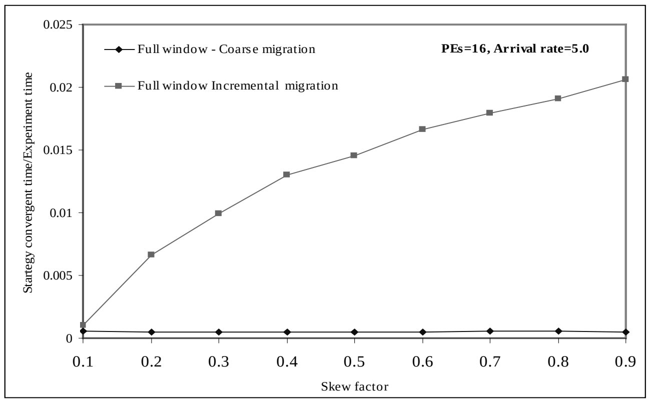
[YKM99] H. Yokota, Y. Kanemasa, J. Miyaazaki. Fat-Btree: An Update-Conscious Directory Structure. *Procs. of IEEEthe 15th IEEE Conf. on Data Engineering, pp. 448-457,1999.*

[**ZS96**] C. Zou and B. Salzberg. On-Line Reorganization of Sparsely-Populated B+ Trees, *Procs. ACM, pages 115-124,1996.*

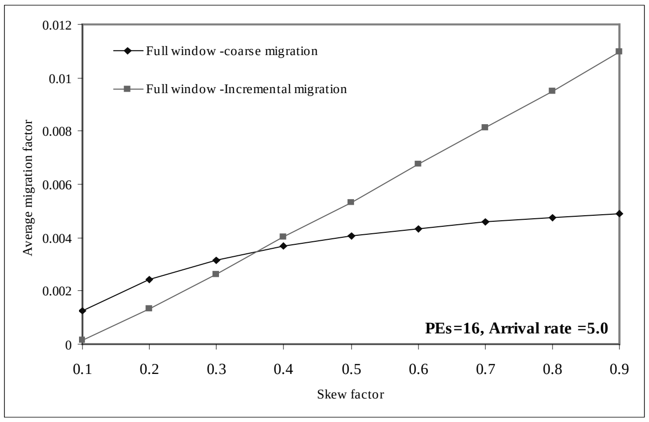




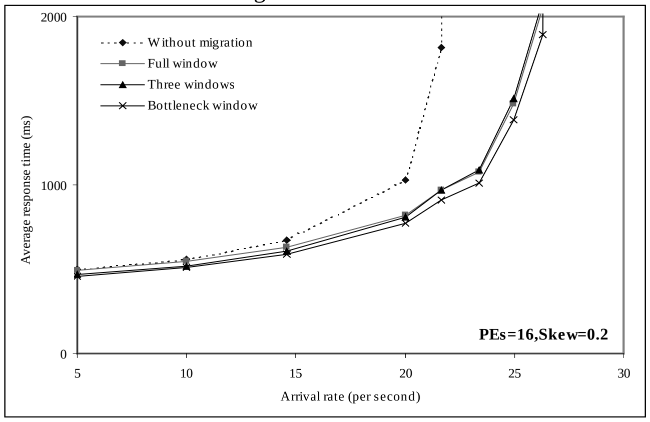
(5-A) Effect of the migration granularity on the average response time.



(5-B) Effect of the migration granularity on the strategy convergent time.



(5-C) Effect of the migration granularity on the total migration cost.
Figure 5: The “full window strategy performance” with two different migration schemes.



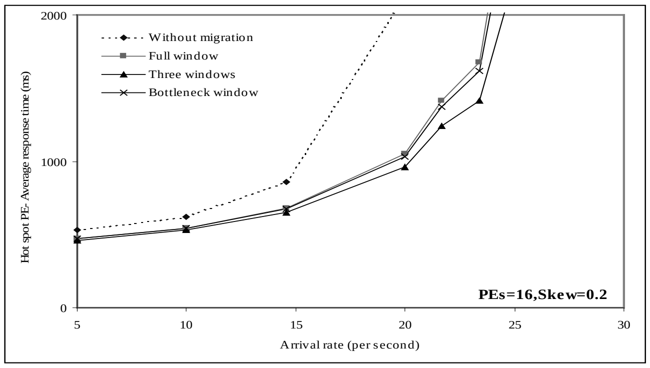


Figure 6: Effect of heat balancing on the average response time of a 16-PEs cluster (top) and its “hot spot” PE (bottom).

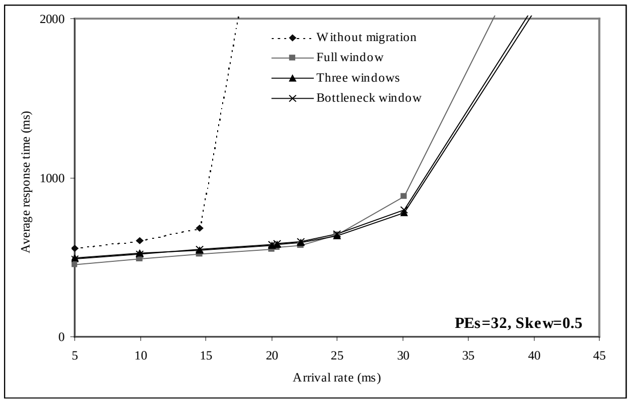


Figure 6: Effect of heat balancing on the average response time of a 32-PEs cluster.

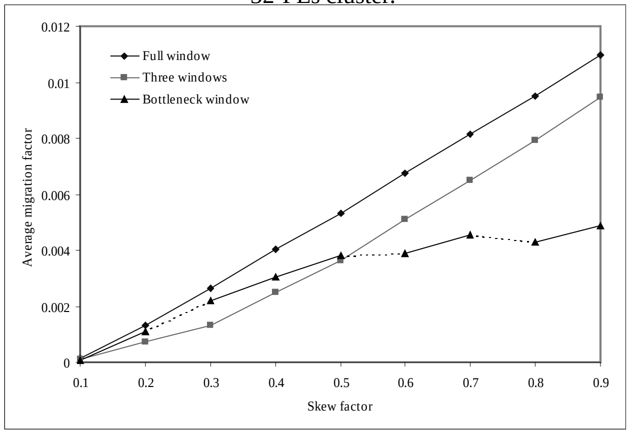


Figure 7: Effect of the heat balancing on the total migration cost

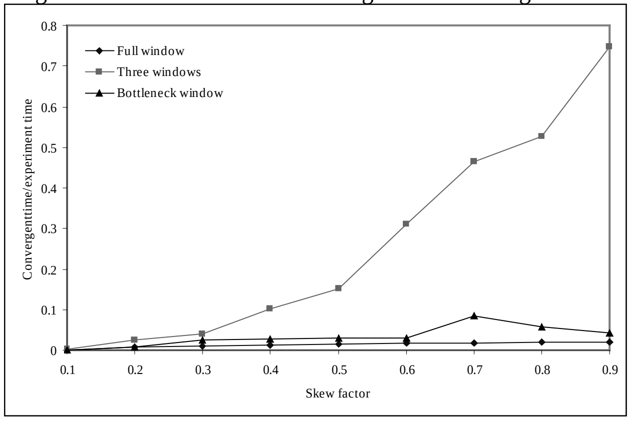


Figure 8: Comparson of strategy convegent time when the skew vary.

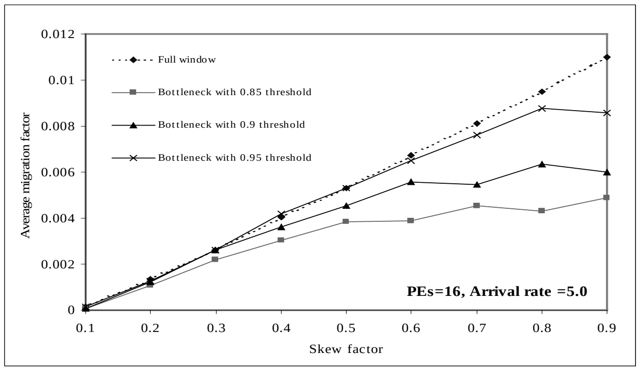


Figure 9: Effect of the threshold on the total migration cost of the “bottleneck window” strategy.