Towards Quality of Service Based Resource Management for Cluster-Based Image Database

A. Brüning[1], F. Drews[2], M. Hoefer[1], O. Kao[3], and U. Rerrer[3]
[1]Department of Computer Science, Technical, University of Clausthal, Germany
E-mail: bruening@informatik.tu-clausthal.de, martin.hoefer@tu-clausthal.de
[2] Ohio University, Athens, USA
E-mail: drews@ohiou.edu
[3] Department of Computer Science, Paderborn University, Germany
E-mail: {okao,urerrer}@upb.de

Abstract
The main research in the area of image databases addresses the improvement of retrieval quality and the speedup of the query processing. A number of image retrieval systems reached meanwhile a level, where these are used in real-world applications and thus create new demands. In particular dealing with a large, simultaneous number of users and queries requires concepts for resource management, which strongly consider the underlying architecture and the various approaches for image retrieval. This paper is our first step in this direction, as it provides a formal problem specification and a proof that this problem is $\mathcal{NP}$-hard.
Keywords: resource management, image retrieval

1 Introduction

A standard approach for the retrieval in image databases is based on the extraction and comparison of a priori defined features. These can be combined and weighted in different ways resulting in advanced features representing the image contents on a higher abstraction level. The similarity degree of a query image and the target images is determined by calculation of a multidimensional distance function of the corresponding features. An acceptable system response time is achieved, as no further processing of the raw data is required during the retrieval process. However, the extraction of such simple features results in a disadvantageous reduction of the image contents. Due to the comparison of whole images only global features like dominant colors, shapes, and textures define the similarity, but detailed object and topological information is not sufficiently considered.

A retrieval following a human-like approach of analyzing and searching images can be realized using dynamically extracted features. The user marks his/her region of interest (ROI), which is then separated from the background and transformed by algorithms for dynamic feature extraction. The result is a set of features describing the object. All other elements of the source image are ignored. In order to find the object in different environments a template matching with the gained ROI is applied. The ROI slides over the image and for each section a similarity value is computed. The similarity computation is repeated for all image sections with respect to the given step size (e.g. 5 pixels). Finally the most similar section within the image is found. A result ranking based on the values of the most similar sections in all images is presented to the user.

Summarizing, a typical image retrieval chain consists of an operation for a static retrieval s and of an operation for dynamic retrieval d combined as $d \circ s(I)$, where I is the total set of stored images. As already noted the operation s is executed in a couple of seconds using the existing index structures. The re-

sult of s is a selection of the most promising images for the current query, which are subsequently analyzed using methods for dynamic image retrieval. The processing time for this step is user/query-specific and depends on the applied operation type, the selected parameters (e.g. step size) and the settings of the preselection algorithm, which produces the image subset for the dynamic retrieval. This second step is accompanied by a large computational load [**?**] resulting in enormous processing time per query.

In order to speedup this process, powerful parallel architectures are necessary. According to simulation results [**?**], a cluster-based architecture was selected and a cluster-based prototype called CAIRO has been developed. The organisation of the underlying cluster architecture is depicted in Figure **??**.

The node functionalities are subdivided into three classes:

1. Query stations host the user interfaces and provide a web-based access.

2. Master node provides the web-based interface, controls the cluster, and performs the pre-selection with the a-priori extracted features. Subsequently, it distributes the dynamic retrieval tasks to the nodes and unifies the intermediate results into the final ranking.

3. Compute nodes perform the comparisons with the sample image. Each of these nodes contains a disjoint subset of the images and executes all operations with the local data.

The images are distributed over the cluster nodes using a content-independent, size-based partitioning strategy. The total memory size of the images stored on the local devices is approximately the same for all nodes. This even data distribution is distorted by the preselection, as only a part of the images has to be processed dynamically. As result, varying response times of the individual nodes occur. In order to equalize the processing times and to balance the workload a temporal or permanent migration of the images has to be performed. For this purpose, a number of load balancing strategies was developed and examined [**?**]. The results have shown that the response times of a typical database can be reduced noticeably, if the queries are executed sequentially in batch mode. However, this is not acceptable for real-world applications, as couple of queries would block the system for days. In case of multiple queries a well-defined QoS concept is necessary.

The main contribution of this paper is a formal statement for the described problem. Furthermore, we prove that this problem is $\mathcal{NP}$-hard and thus heuristic strategies have to be applied.

2 Quality of Service for multiple queries

The processing of multiple queries in the described database must be analyzed from two points of view. The concurrent execution of the queries can be performed easily with well-known synchronization mechanisms from traditional database. It is rather a simple problem, as all operations are executed image-wise and the majority of the operations is read-only. Hence, the decisive aspect is the processing time. Each query blocks the system for a long time, which will be substantially extended, if multiple queries are allowed. Load balancing – e.g. combination of queries considering images of disjoint subsets placed on different nodes – promises a speedup, however this approach depends on the characteristics of the current query and it is thus not well-suited for real-world applications. Therefore, a utility measure and a concept for quality-of-service is designed, which allows for the reallocation of resources depending on the actual set of queries issued to the system. By this means we are able to react flexibly to changes in global system load.

In an overload situation, the main goal is to keep the response time within desired time in-

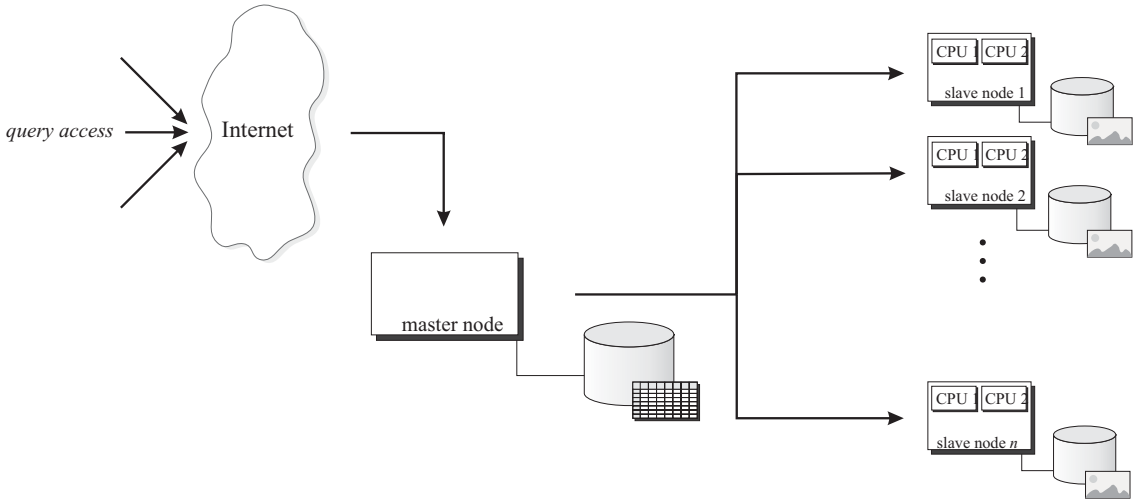

Figure 1: Schematic of the underlying cluster architecture

tervals by reducing the number of images to be processed with time-intensive dynamic operators. Thereby, the retrieval quality will only slightly be reduced, as the best images will still be found and only images with (very) low chances for success will be ignored. However, it is possible, that particular objects or persons will not be found, as the images are not contained in the pre-selected image set. The reason for this is, that static operators evaluate the entire image, whereas the dynamic operators consider only small parts of the image which might fit the query section. On the other hand, in an underload situation, the unused resources are utilized to increase the retrieval quality of the remaining queries by enlarging the set of pictures processed by dynamic operators.

The adaptation of the image sets to be searched is realized by adapting the threshold values for the static image retrieval s. A low value for the threshold s leads to a high-number of images as a result of the preselection and a high threshold value delivers solely the most promising images. The threshold interval is discretized and the resulting n discrete threshold values are denoted as "service level settings". A higher setting is equivalent to a lower threshold value and results in a higher search quality. Note, that the service level setting of a query can be adapted *online*, consequently providing a tool for adapting the system load. The user should be able to define a minimal service level setting in order to define a minimal processing quality to be met for his query. If no minimal service level setting is provided a (low) standard setting is assumed. In order to determine whether it is beneficial to enhance the retrieval quality of a query or to complete it earlier, a user-provided utility function is needed. The definition of the utility function underlies certain restrictions in minimal and maximal utility values. Depending on the user status these restrictions may vary.

Hence, a query can be characterized by the sample objects, the user-provided utility function, its minimal service level setting and eventually a hard deadline.

3 Resource Management Architecture

In this section we describe the resource management architecture for the retrieval system. Users issue queries to the system via internet. The optimization and acceptance module (OAM) controls the admission of queries into the system and maintains an input queue

of queries. For each query the OAM checks whether the query can be processed by the system or not. The OAM's decision may be based on the time constraints of the queries or the overall utility of the system obtained by admitting the query. Once a query is accepted, the system has to assure completion of this query without violating neither resource constraints nor the query's deadline (if provided). If a query is refused, the system can return information to the user about how to lower the quality and time requirements of the query for acceptance. The system architecture allows to consider his case as a new query issued by the user.

Note that the sets of images on the compute nodes are usually not equally reduced by the static retrieval operator. Therefore, equilibration of processing demand across the cluster nodes is needed. The set of images of a query is subject to change when the service level of the query is adapted by the system. In order to determine which image has to be processed we provide the OAM with a service level table containing the maximal service level for each image in which it is still processed.

This table can be generated by appliying the static retrieval operator on each image of the system. Once the table is generated, no further run of the static retrieval operator is necessary for this query. The result of the static retrieval operator on an image is compared with the threshold values of the different service levels in increasing order.

As we will show later, the problem of determining an optimal schedule with optimal service level settings is $\mathcal{NP}$-hard. Hence, the decision making process performed by the OAM has to be based on a good approximation of an optimal solution. Therefore, the optimization is accomplished by several modules tackling different sub-problems:

(i) the load balancer determines which host processes which object,

(ii) the service level optimizer determines the service level settings for the queries, and

(iii) the query optimizer determines the schedule of the queries on the hosts.

Every time a query is issued, a new solution for the system is calculated. If the new solution permits the acceptance of the new query without violating any constraint, the new process schedule and service level settings are passed to the application software for execution. Then, for each query a retrieval application is started on each host, processing the subset of objects as determined by the OAM. Consequently, q different queries result in q different retrieval applications on each host, each of which processing a subset of the stored images.

It may be reasonable to repeat the optimization after a certain period of time independent of arriving queries. Thus, the predicted schedule can be adjusted according to system dependent delays in order to ensure that the completion time of queries obeys the hard deadlines.

4 Formal System Model and Problem Statement

For a systematic discussion of the problems associated with the architecture introduced in the last section, we define a formalized optimization problem of the OAM to be solved dynamically on a repeated basis. The database D of objects is distributed over the hosts $\mathcal{H}_1, \ldots, \mathcal{H}_h$ of the system. Let D_l be the set of objects on host $\mathcal{H}_l$ with $D = D_1 \cup \ldots \cup D_h$ and $size(D_l) \approx size(D_m)$. If a query $\mathcal{Q}_i$ enters the system, all images of the database are examined with a static operator. The maximal service level setting (threshold value) for each image is determined. For a query i on each host l and each service level j we get a different set of images $Q_{ijl} \subseteq D_l$. Generally, this leads to an unbalanced image distribution for a query as we have $size(Q_{ijl}) \not\approx size(Q_{ijm})$ for two hosts l and m. With q queries the formal description of a query $\mathcal{Q}_i$ consists of

(i) a set $S = \{s_1, \ldots, s_n\}$ of service level settings,

(ii) a set of images $Q_{ij} = Q_{ij1} \cup \ldots \cup Q_{ijh}$ to be processed with setting s_j,

(iii) user defined monotonic utility functions $U_i(r_i)$ with r_i being the response time of the system for this query, and

(iv) (possibly) a deadline π_i for the response time.

Note that due to our assumption about the threshold values service level settings can be totally ordered, i.e. an ordering $s_1, \ldots, s_n$ exists, such that for each query i $Q_{i1} \subseteq Q_{i2} \subseteq \ldots \subseteq Q_{in}$ holds. Modelling the system we will make a few simplifying assumptions:

(i) Apart from the images located on them all hosts have identical features,

(ii) all queries have hard deadlines,

(iii) each image appears on exactly one host ($D_i \cap D_j = \emptyset$ for $i \neq j$), and

(iv) if an image is scheduled to be processed by a query on a different host, it is copied and available on the other host only for the corresponding query.

For the utility functions we assume that the user is able to specify one utility function $U_i(r_i)$ per query, which is monotonically decreasing in the response time. It will be adjusted by a multiplicative factor monotonically depending on the choice of the service level setting, i.e. the number and type of images processed. With this construction we can ensure that the ordering of service level settings holds not only for image sets but also for utility values. Especially the notion of minimum quality associated with a minimum service level can be justified, because for each query and any given response time a minimum service level setting results in minimum utility. The overall system utility is constructed by an aggregation of query utilities.

The main resources of the system are CPU-time, memory and network bandwidth. For each combination of settings for the service attributes, a resource profile as a function of the object size has to be given. All utility functions are monotonically decreasing in the response time. Therefore delaying queries by simultaneous execution of multiple queries on a host (multi-tasking) or preemption can never lead to optimal system utility. Thus, we will assume that in each point in time a host processes only one non-preemptive image processing task of one query. Hence, memory constraints can be neglected when our system is designed such that each host provides the maximum memory requirement of a dynamic feature extraction operator applied on an image in $\mathcal{D}$. Considering current practical computers this is a reasonable assumption.

Note, that every time a new query is arriving the OAM is invoked to solve a new instance of this problem. It has adjusted image sets for the existing queries in the system. The currently processed query may be preempted when the solution of the OAM is put into action. In this situation preemption of a query is allowed.

Communication has been a major issue in previous attempts to optimize response time in image retrieval systems [**?**]. Here we introduce the notion of a communication module on each host i, which is informed about the times when queries need to process images of D_i on another host j. The communication modules initiate transfer such that the image arrives only shortly before the processing starts. Images are only transferred when they are really needed, which is appealing in a case when the allocation of image processing tasks to hosts might change dynamically with each call of the OAM. Thus, the transfers are distributed over the time of processing and it is very unlikely that reasonably sized bandwidth will be used up resulting in processing delays. Nevertheless we will relax the assumption of identical hosts and introduce host-dependent processing times for each retrieval operation. Processing of an image $k \in D_i$, $k \notin D_j$ on host j might be delayed by communication. The processing time on host j should reflect this fact and be equal to the processing time on host i plus some extra amount depending on bandwidth and image size, which we call "expected com-

munication delay".

With processor time being the only remaining resource we can find the following lemma which facilitates the development of efficient optimization algorithms for the scheduling problem of finding an optimal schedule with optimal service level settings:

Lemma 1
Under the given assumptions there is an optimal solution with the following properties:

1. Image processing tasks of a query form *blocks* on the hosts to which they are allocated, i.e. all images of a query on a host are scheduled back-to-back.

2. There is no idle time on any host between the processing of any two images/blocks.

3. Blocks of all queries are ordered the same way on every host.

Proof of Lemma 1:

1. In an optimal schedule for host l, in which images of queries are not scheduled in blocks, consider the non-block query i which finishes last. We move all other images of i on l back-to-back directly before the start of the last finishing image of i. Thus, we construct a block at the end of the schedule of l. The completion time of i will remain unchanged, because the starting time of the last image is not altered. All other queries are able to finish no later than before - the images of the last query are moved to the end of the schedule, therefore images of other queries can be moved to earlier start times on host l. This change results in equally good or better utility values. Repeating this argument for all queries and all hosts yields the superiority of block-schedules. □

2. Move the idle time to the back of the schedule by scheduling earlier all images after the idle slot. The utility of the new schedule is at least as good as before. □

3. Without loss of generality, we assume that each query has a block on each host. Let all blocks with processing times 0 be scheduled on each host at time 0. We assume to have an optimal schedule, in which blocks do not have the same ordering on every host. Now consider the latest position α, in which the orderings on two hosts differ. Let query i be the query of the block with the latest response time of all the blocks scheduled in these positions (break ties arbitrarily). Now consider switching all other blocks of query i to position α on their hosts. The overall length of the schedules at position α on the hosts does not change. So the new completion time of the switched blocks of query i does not exceed the completion time of the non-switched block. Therefore switching all the blocks of query i to position α on all hosts does not increase the response time of query i. It leaves the completion times of all blocks scheduled at positions greater than α unchanged. The blocks of queries scheduled at positions smaller than α might be moved to earlier start and completion times on the hosts. This change results in equally good or better utility values. The lemma follows from repeating this argument until a common ordering on the hosts is found. □

Now we are able to give a formal proof for the complexity of the optimization problem.

Lemma 2
The problem of finding an optimal schedule with optimal service level settings is $\mathcal{NP}$-hard.

Proof of Lemma 2:

We consider a special case with one host $H = \{1\}$ and one service level setting $S = \{1\}$, in which each query i must process only one image $k \in D$. The deadlines are denoted d_i. The utility function is given as $U_i(C_i) = -w_i C_i$ with C_i representing the completion time of the query on the host and w_i an arbitrary

weight. Then the problem reduces to the well known scheduling problem $1 \mid \tilde{d}_j \mid \sum w_j C_j$ which was proven to be strongly $\mathcal{NP}$-hard [**?**]. As our problem is a generalization, it is strongly $\mathcal{NP}$-hard, too. □

5 Conclusions

This paper presents an formal model for introduction of multiple queries in a cluster-based database with object retrieval. In opposite to general models for resource management such as QRAM [**?**], the specific characteristics of image retrieval with dynamically extracted features and cluster processing are considered. The main contribution of the paper is a formal problem statement as well as prove that the problem is $\mathcal{NP}$-hard.

Future work includes in first line the development and evaluation of heuristic approaches for the solution of the specified problem.

References

[1] C.C. Venters and M. Cooper. A review of content-based image retrieval systems, Tech. Rep. jtap-054, University of Manchester, 2000.

[2] O. Kao, G.R. Joubert. Efficient Dynamic image retrieval using the À trous wavelet transformation, Advances in Multimedia Information Processing, LNCS 2195, 2001, pp. 343-350, Springer

[3] J.K. Lenstra Sequencing by Enumerative Methods, Mathematical Centre Tract 69, Mathematisch Centrum, Amsterdam, 1977.

[4] T. Bretschneider, S. Geisler, O. Kao. Simulation-based Assessment of Parallel Architectures for Image Databases, Proceedings of the International Conference on Parallel Computing (ParCo 2001), pp. 401-408, 2002, Imperial College Press.

[5] F. Drews, K. Ecker, O. Kao and S. Schomann. Strategies for Workload Balancing in Cluster-based Image Databases, Parallel Processing Letters, to appear.

[6] O. Kao, S. Stapel. Case Study: Cairo A Distributed Image Retrieval System for Cluster Architectures, T.K. Shih (Edt.): Distributed Multimedia Databases: Techniques and Applications, pp. 291-303, 2001, Idea Group Publishing

[7] R. Rajkumar, C. Lee, J. Lehoczky, D. Siewiorek. A Resource Allocation Model for QoS Management, Proceedings of the IEEE Real-Time Systems Symposium, pp 298-307, 1997