QoS Resource Management for Cluster-Based Image Retrieval Systems

A. Brüning[1], S. Geisler[1], M. Hoefer[2], and O. Kao[3]
[1]Department of Computer Science, Technical, University of Clausthal, Germany
E-mail: {bruening,geisler}@tu-clausthal.de
[2]Department of Computer and Information Science, University of Konstanz, Germany
E-mail: hoefer@inf.uni-konstanz.de
[3]Department of Computer Science, Paderborn University, Germany
E-mail: okao@upb.de

Abstract *This paper presents a novel concept for resource management in cluster-based image retrieval systems. First, the paper describes image retrieval using static and dynamic feature extraction. The complexity of dynamic feature extraction requires the utilization of powerful parallel architectures and in order to provide the user with reasonable response times. Most existing methods for resource management in parallel image retrieval systems are based on sinlge query execution and do not take quality of service (QoS) aspects into account. This appears not to be practical in large-scale and commercial applications of image databases having a large number of users at any time. In order to allow an efficient utilization of the parallel system and to meet user-defined QoS demands associated with queries, we need to develop a new concept and a novel resource management architecture. Interesting aspects of the model include utility theory, flexible computations, QoS levels, and a hierarchical resource management architecture. Finally, an approach for algorithmic solution is described.*

Keywords: resource management, image retrieval

1 Introduction

Image management systems handle large, general sets of images and allow to search for a number of images that are similar to a given sample image. The importance of image databases rose enormously in recent years. Progresses in digital technologies caused a vast spreading of multimedia applications. This can be seen from the production of Petabytes of pictorial information per year in numerous application areas such as museums, research institutions, photo agencies for publisher houses, press agencies, and civil services. A survey of existing image databases is e.g. provided in [1].

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, because no further processing of the raw data is required during the retrieval process. Moreover, pre-compiled index structures accelerate the search, as only a small part of the image set has to be considered [2]. The extraction of such simple features results, however, 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 detailed retrieval following a human-like approach of analyzing and searching images can be realized using dynamically extracted features. Any manually or automatically selected elements of the query image are transformed into feature vector and compared with all sections of the selected target images during runtime. The user marks his/her region of interest (ROI), which is then separated from the background and transformed by various algorithms for dynamic feature extraction. The result is a set of features – e.g. Gabor-Jets [3] – 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 couple of seconds using the existing index structures. The result 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 pre-selection algorithm, which produces the image subset for the dynamic retrieval. This second step is accompanied by a large computational load [4] resulting into enormous processing time per query.

In order to speedup this process, powerful parallel architectures are necessary. According to simulation results [5], a cluster-based architecture was selected and a cluster-based prototype called Cairo has been developed. The nodes of the underlying architecture are subdivided into three classes:

- Query stations host the user interfaces and provide a web-based access.
- A master node controls the cluster and receives the query. It performs the pre-selection, distributes the dynamic retrieval tasks to the nodes and unifies the intermediate results into the final ranking.
- 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.

For the initial distribution of the images over the cluster nodes a partitioning strategy is selected which is content-independent and where 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 pre-selection, as only a part of the images has to be processed dynamically. Due to the applied size-based partitioning a random distribution of the relevant images occurs and leads to varying response times of the individual nodes. 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 [6]. The results have shown that the response times of a typical database can be reduced noticeably, if the queries are executed non-preemptively and 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.

2 A Quality of Service concept

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 databases. 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, hence full processing of every query can be computationally and timely infeasible. Load balancing – e.g. combination of queries considering images of disjoint subsets placed on different nodes – can yield a speedup, which however is not able to yield sufficient improvement. Therefore, a QoS concept is designed, which helps to determine for each query, how much processing time is allocated to it – i.e. how many and which of the images are processed with time-intensive dynamic retrieval operations. The aim is to reduce the retrieval quality as little as possible by ranking the images such that the best images will still be found and only images with (very) low chances for success will be ignored. However, it still might be possible, that particular objects or persons will not be found, as the images are not contained in the pre-selected image set. 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.

Our QoS-concept is manifested in the adaptation of the image sets to be searched. This 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 pre-selection 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, ensuring a reasonable amount of images to be processed. 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. The optimization and acceptance module (OAM) controls the admission of queries into the system and maintains an input queue of queries. In case of several queries arriving simultaneously, they are processed either in the order of arrival or according to user status. 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. From the system architecture point of view this case can be considered as a new query with different settings issued by the user. The sets of objects on the hosts are usually not equally reduced by the static retrieval operator. Therefore, equilibration of processing demand across the compute nodes is needed. Note that the set of objects of a query is subject to change when the service level of the query is adapted by the system.

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:

- Load balancer determines which host processes which object
- Service level optimizer determines the service level settings for the queries
- 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. 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 ensuring 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 h hosts. Let D_l, D_m be the set of objects on hosts l and m with $D = D_1 \cup \ldots \cup D_h$ and $size(D_l) \approx size(D_m)$. If a query enters the system, all images of the database are examined with a static operator. The maximal service level setting (threshold value) for the 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 different hosts l and m. With q queries the formal description of a query consists of

- Set $S = \{s_1, \ldots, s_n\}$ of service level settings,
- Set of images $Q_{ij} = Q_{ij1} \cup \ldots \cup Q_{ijh}$ to be processed with setting s_j,
- Monotonic (user defined) utility function $U_i(r_i)$ with r_i the response time of the system for this query, and (possibly)
- Deadline π_i for the response time.

Note that due to our assumption about the threshold values service level settings can be totally ordered, meaning that 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. Modeling the system we will make a few simplifying assumptions:

- Apart from the images located on them all hosts have identical features
- All queries have hard deadlines
- Each image appears on exactly one host: $D_i \cap D_j = \emptyset$ for $i \neq j$
- 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 capturing his benefit if the query is processed with the best service level and finished at time r_i. U_i is monotonic decreasing and 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 object size has to be given.[1] All utility functions are monotonic decreasing in the reciprocal of the response time, therefore delaying a query by simultaneous execution (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 any image. Considering current practical computers this is a reasonable assumption.

Communication has been a major issue in previous attempts to optimize response time in image retrieval systems [6]. 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 communication delay".

[1]We assume a functional dependency between the resource usage and the object size, as it exists for many image and audio processing operators.

With processor time being the only remaining resource we can find the following: Lemma Under the given assumptions there is an optimal solution in which

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

2. 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.

To get a better understanding of the system we present an approach for an algorithmic solution. Due to the complexity of the problem the optimization is divided in two parts: the service-level-optimization and the image scheduling between the hosts. We suggest a tabu search approach [7] for the optimization of the service-level settings. The search space is the space S of all service-level settings meeting the minimal quality demands of all queries. A specific setting can be represented by a vector $s \in S$ consisting of the settings s_i for each query $\mathcal{Q}_i$. The tabu search neighborhood can be realized by increasing or decreasing one setting s_i without violating neither the minimal quality constraints nor the deadlines of all queries.

The changing of a service-level setting during the service-level optimization results in changed sets of images which have to be scheduled. Thus, a fast scheduling algorithm has to be applied. As mentioned above the scheduling of the queries has to be done in *blocks* in order to achieve optimal utility. For each query $\mathcal{Q}_i$, these blocks can be predicted by identifying the sets $Q'_{ij} = Q_{ij} \setminus \bigcup_{s=1}^{j-1} Q_{is}$, $j = 1, \ldots, n$, of images additionally processed when switching from service-level $j-1$ to j. The sets Q'_{ij} are balanced over the hosts by the LPT algorithm [8], considering the host-dependent processing times. We assume the sets Q'_{ij} to be large compared to the average image size so that LPT gives nearly optimal equilibrated results. This reduces the problem of image-scheduling to block-scheduling.

As stated above, all blocks Q'_{ij} of a query $\mathcal{Q}_i$ have to be scheduled back to back in order of increasing service-level settings. Optimal utility can be achieved when the same order is maintained on all hosts. For a given service-level setting $s \in S$ we obtain a fast deadline feasibility test:

feasible(s)

1. Queries are scheduled in EDF-ordering on all hosts.
2. Schedule the Q'_{ij}, $j = 1, \ldots, s_i$ for every query i in order of increasing service-level settings for obtaining the default schedule $\tilde{\mathcal{S}}(s)$.
3. If all deadlines are met return *true* else $false$.

Usually, best utility values are not obtained for the EDF ordering of the queries. For optimizing the order of queries we suggest a local search heuristic based on swapping adjoining queries:

switch-optimize(s)

1. Start with $\mathcal{S} = \tilde{\mathcal{S}}(s)$. If no utility-increasing feasible swap exists, return $\mathcal{S}$.
2. Calculate new schedule $\mathcal{S}$ by executing best feasible swap and return to 1.

This leads to the following algorithm:

1. A new query $\mathcal{Q}_i$ arrives.
2. Identify the Q'_{ij} for all service level settings of $\mathcal{Q}_i$ and equilibrate the images in the Q'_{ij} by LPT with respect to their host-dependent processing times.
3. Start service-level optimization with minimal feasible settings $s_{min} \in S$ for all queries.
4. If **feasible**$(s_{min}) = false$ then exit by declining the query
5. Perform tabu-search with *iter* iterations:
   (a) Alter one service-level setting in s
   (b) If **feasible**(s) and *utility*$(\mathcal{S}) >$ *opt* with $\mathcal{S} =$ **switch-optimize**(s) then *opt* $=$ *utility*$(\mathcal{S})$ and *best* $= \mathcal{S}$
6. *best* is best solution, *opt* is the best utility value

An implementation of the proposed algorithm and performance measurements in real-world environments using the cluster-based image database Cairo [9] are currently in progress.

5 Conclusions and Future Work

Current research on the design of image databases addresses almost exclusively the increase of retrieval quality and the efficient execution of isolated queries in batch mode, e.g. image partitioning and load balancing. However, the growing number of fully-operational image databases in real applications and web services requires studies of resource management with respect to the current number of waiting queries. The main aim is to adjust the *global* load and to provide reasonable response time. Such mechanisms already exist for internet search engines for documents, but also for general systems such as QRAM [10]. However, these approaches do not consider specific image database characteristics resulting from large computational and network load. Therefore we developed a formal system model and an algorithmic approach, which takes the specific problems of image retrieval – in particular with dynamic image operators – in account.

Future work includes in first line the evaluation of the proposed algorithm in different scenarios and comparison with other strategies, which have to be developed. In particular set-up of practical limits for the maximum allowed pre-selection without significant loss of retrieval quality are mandatory.

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] V. Krüger, G. Sommer. Gabor wavelet networks for object representation, Proceedings of the DAGM Symposium, pp. 13-15, 2000.

[4] A. Reuter. Methods for parallel execution of complex database queries, Parallel Computing 25 (1999) pp. 2177-2188.

[5] 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.

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

[7] F. Glover and M. Laguna. Tabu Search, Kluwer Academic Publishers, 1997.

[8] R.L. Graham. Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17, pp. 263-269, 1969

[9] 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

[10] 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