Literature review

Cornell and Yu[1990] In this paper they shows the importance of integrating of buffer management and query optimization strategies so that reference information from the queries can be captured to manage the buffer, and buffer availability can be reflected in the access path selection. Furthermore, each query can he pre-analyzed separately to reduce the number of access plans to be considered in the global optimization. An integrated approach to buffer management and query optimization is proposed and analyzed. ‘The query strategy of all transaction types is simultaneously considered together with the buffer allocation strategy so as to optimize overall system performance. As the buffer allocation depends upon the buffer holding time or transaction response time which is determined by the buffer allocation and query optimization strategies, an optimization method combining a integer programming model with a queuing model applied iteratively is developed to capture this effect.

Hurson and Cheng [1991] In this paper they investigates the updating effects and suggests a dynamic re-clustering scheme to reorganize related objects on the disk. A cost model is introduced to estimate the benefit and overhead of re-clustering. Reorganizations are performed only when the overhead can be justified. For environments in which multiple relationships among objects exist, the paper proposes a leveled clustering scheme to order related objects into a clustering sequence. When the read/write ratio is high, a fully dynamic re-clustering strategy is affordable; when the ratio is low, an off-line re-clustering strategy is a better choice. Results show that the leveled clustering scheme has a better access time compared with a single-level clustering scheme.

Rahm and Ferguson [1993] presents a new algorithm for shared data objects that are accessed sequentially. This algorithm fall into three classes and one of them is prefetching this algorithm performs asynchronous prefetching. This algorithm uses the velocity estimates to predict future cache misses and attempts to preload data to avoid these misses.

Gerlhof and Kemper [1994] In this paper they shows the importance of clustering quality to the performance of pre-fetching algorithms in OODBMSs. They found for applications with high locality, when running on a well clustered database, pre-fetching often achieves only negligible
gains. However they did not vary the buffer size or report the buffer size they used. This leads us to suspect that at high locality and good clustering the working set size of the benchmark used was smaller than the buffer size. In these conditions the database does not need to load from the disk, hence the negligible impact of pre-fetching.

However, when objects are not well clustered, many useless objects may be transferred to the client. Many object-oriented databases, in such cases use page Server Architecture approach: ObjectStore [Lamb 1991], O2 [Deux1991] and Platypus [He 2000].

Additionally Gerholf and Kemper do not propose any algorithms that exploit the synergy between pre-fetching and clustering, they only report the affects that clustering has on pre-fetching.

Cao, Felten, Karlin, and Li [1995] study the implications of integrating pre-fetching and buffer replacement when perfect knowledge of the future access sequence is known. They argued that pre-fetching too early may be harmful since early pre-fetching results in early buffer replacement if the buffer is full. Early buffer replacement can be harmful since new and better replacement opportunities may open up as the program proceeds.

Using this observation they develop two new integrated pre-fetching and buffer replacement algorithms called aggressive and conservative. These strategies were found to reduce application running time by up to 50% compared to no pre-fetching. However, their algorithms assume perfect knowledge of future access sequence, an unrealistic assumption in the real world. Their study was done within the context of operating systems buffer management.

Patterson, Gibson, Ginting, Stodolsky and Zelenka[1995] present aggressive, proactive mechanisms that tailor file system resource management to the needs of I/O-intensive applications. They show how to use application-disclosed access patterns (hints) to expose and exploit I/O parallelism, and to dynamically allocate file buffers among three competing demands: prefetching hinted blocks, caching hinted blocks for reuse, and caching recently used data for unhinted accesses. Their approach estimates the impact of alternative buffer allocations on application execution time and applies cost-benefit analysis to allocate buffers where they will have the greatest impact. They have implemented informed prefetching and caching in Digital’s OSF/1 operating system and measured its performance on a 150 MHz Alpha equipped
with 15 disks running a range of applications. Informed prefetching reduces the execution time of text search, scientific visualization, relational database queries, speech recognition, and object linking by 20-83%. Moreover, applied to multiprogrammed, I/O-intensive workloads, informed prefetching and caching increase overall throughput.

**Bullat and Schneider [1996]** proposed an integrated dynamic clustering, buffer replacement and pre-fetching algorithm. Their integrated policy used the concept of a `cluster unit`. A cluster unit is a set of clustered objects that span one or more pages. They perform pre-fetching and buffer replacement at the cluster unit grain instead of the page grain. Alternatively pre-fetching and buffer replacement can occur at the `moving cluster-window` grain a sub-set of x pages of the cluster unit which has the requested object at its center.

An important drawback of this approach is that often dynamic clustering algorithms produce cluster units that are smaller than a page in size. Also, most dynamic clustering algorithms do not arrange objects in cluster units, hence precluding the use of this approach. The cluster unit concept does not incorporate direction of navigation.

Direction of navigation is very important for both pre-fetching and buffer replacement. In pre-fetching we are only interested in pre-fetching objects in the forward direction of navigation similarly, in buffer replacement we are only interested in retaining in memory objects in the forward direction of navigation. Lastly, their paper only reported the results of a performance study comparing no clustering against their new dynamic statistical clustering technique (DSTC). The performance gains from integrating pre-fetching and buffer replacement were not reported.

**Vitter and Krishnan[1996]** In this paper they apply a form of the competitive philosophy for the first time to the problem of prefetching to develop an optimal universal prefcaster in terms of fault rate, with particular applications to large-scale databases and hypertext systems. Our prediction algorithms for prefetching are novel in that they are based on data compression techniques that are both theoretically optimal and good in practice. Intuitively, in order to compress data effectively, you have to be able to predict future data well, and thus good data compressors should be able to predict well for purposes of prefetching. We show for powerful models such as Markov sources and $m$th order Markov sources that the page fault rates incurred by our prefetching algorithms are optimal in the limit for almost all sequences of page requests.
Knafila, N. [1997] presented a prefetching technique for complex object relationships in a page server. The object structure of the database is analysed and stored in a prefetch object table. During the run time of the application this table is consulted to make the right prefetches on time. We used the object pointers to make predictions for future access. If the application follows such an object reference chain, we know the object that points to an object in the next page therefore making this page a candidate for prefetching. They also use the branch information of the complex relationships to predict the next pages as accurately as possible. If there are more prefetches to do at the same time it use threads to get all prefetches before the application requires access.

Jiang, Kleinrock [1998] presents an adaptive prefetch scheme for network use, in which they download files that will very likely be requested in the near future, based on the user access history and the network conditions. This prefetch scheme consists of two parts: a prediction module and a threshold module. In the prediction module, they estimate the probability with which each file will be requested in the near future. In the threshold module, they compute the prefetch threshold for each related server, the idea being that the access probability is compared to the prefetch threshold. An important contribution of this paper is that they derive a formula for the prefetch threshold to determine its value dynamically based on system load, capacity, and the cost of time and system resources to the user. They also show that by prefetching those files whose access probability is greater than or equal to its server's prefetch threshold, a lower average cost can always be achieved.

Markl, Ramsak, Bayer [1999] presents a Multidimensional hierarchical clustering (MHC) of OLAP data overcomes these problems while offering more flexibility for aggregation paths. Clustering is introduced as a way to speed up aggregation queries without additional storage cost for materialization. Performance and storage cost of our access method are investigated and compared to current query processing scenarios. In addition performance measurements on real world data for a typical star schema are presented.

Weikum and Konig, Kraiss [1999] This paper gives an overview of self-tuning methods for a spectrum of memory management issues, ranging from traditional caching to exploiting
distributed memory in a server cluster and speculative prefetching in a Web-based system. The common, fundamental elements in these methods include on-line load tracking, near-future access prediction based on stochastic models and the available on-line statistics, and dynamic and automatic adjustment of control parameters in a feedback loop.

**Bernstein, Pal, Shutt [1999]** In this paper they propose using the context in which an object is loaded as a predictor of future accesses, where context can be a stored collection of relationships, a query result, or a complex object. When an object O’s state is loaded, similar state for other objects in O’s context is prefetched. They present a design for maintaining context and using it to guide prefetch. They give performance measurements of its implementation in Microsoft Repository, showing up to a 70% reduction in running time. We describe variations that selectively apply the technique, exploit asynchronous access, and use application-supplied performance hints.

**Xiaoyi Xu [2001]** presents the implementation of an access pattern data clustering tool. The data clustering algorithm takes the access pattern data from the trace analysis and partitions the objects into buffer pools. The data clustering tool provides an effective way of choosing an initial buffer pool configuration. Comparing the performance of different configurations statistically, we observed that of K-means clustering configuration performs significantly better than the random and the distributed configuration, and as well as the expert configuration. This supports our hypothesis that the clustering algorithm can do as well as a human expert. It can be an effective tool for automatic buffer pool configuration.

**Lee, Seung, Jeon [2001]** Proposed an integrated push/pull buffer mechanism able to manage both push and pull schemes in a single buffer at client-side. The proposed scheme can support various media playback devices using a single buffer space, which in consequences saves memory space compared to the case where a client keeps two types of buffers. Moreover, it facilitates the single buffer as a mechanism for absorbing network jitter effectively and efficiently. The proposed scheme has been implemented in an existing multimedia communication system called ISSA developed by the authors and has showed good performance compared to the conventional buffering methods in multimedia communication environments.
Han, Whang and Moon [2001] have proposed new notions of the typelevel access pattern and the type-level access locality and presented a new prefetching method based on these notions. We also have proposed a formal framework for understanding underlying mechanisms of capturing and prefetching. Results shows that the proposed method reduces roundtrips and improves performance drastically compared with the existing fetching methods: on-demand fetching and context-based prefetching.

Balan, Tickoo, Bajic, Kalyanaraman and Woods [2002] integrate an intelligent buffer management scheme with the congestion control scheme at the source for efficient multimedia streaming. The scheme exploits the fact that most of the transmission losses actually occur at the source and not in the network. The integrated model uses priority information from the encoder and network information from the congestion scheme and drops low priority packets and sends the most important packets in the available bandwidth. The packets are dropped when the source transmission buffer length exceeds a minimum threshold. This scheme ensures that the media transmitted has the highest possible quality under the given network conditions using a given coding scheme.

Safronov and Parashar [2002] presents a Page Rank based prefetching technique for accesses to web page clusters. The approach uses the link structure of a requested page to determine the most important linked pages and to identify the page(s) to be prefetched. The underlying premise of our approach is that in the case of cluster accesses, the next pages requested by users of the web server are typically based on the current and previous pages requested. Furthermore, if the requested pages have a lot of links to some important page, that page has a higher probability of being the next one requested. An experimental evaluation of the prefetching mechanism is presented using real server logs. The results show that the Page-Rank based scheme does better than random prefetching for clustered accesses, with hit rates of 90% in some cases.

Choi, Moon, Kim [2005] discuss the clustering issues for XML storage. They present the clustering method that clusters the data nodes with similar paths and thus can reduce page I/Os required for query processing. Proposed method set the data nodes that have the same absolute
paths as a base cluster unit, and then compares the path similarity between those units. Finally, clustering is done by applying the graph partitioning technique to the path similarity graphs. Through our experiments, we proved that the PSim clustering method is a flexible method which can process various types of queries efficiently.

Han, Moon [2005] present a technique in ORDBMSs to effectively reduce the number of fetches, thereby significantly enhancing the performance. They Show the new notions of the type level access pattern and the type-level access locality and presented a new prefetching method based on these notions. It also shows that the method reduces the number of fetches and improves elapsed time drastically compared with the existing fetching methods: on-demand fetching and context-based prefetching. There results indicate that our approach provides a new paradigm for prefetching that improves performance significantly in navigational applications and is a practical method that can be implemented in commercial ORDBMSs.

Han, Whang and Moon [2005] formally define the notion of prefetching. We also formally propose new notions of the type level access locality and type-level access pattern. The type-level access locality is a phenomenon that repetitive patterns exist in the attributes referenced. The type-level access pattern is a pattern of attributes that are referenced in accessing the objects. They then develop an efficient capturing and prefetching policy based on this formal framework.

Ibrahim, Cook [2006] presents AutoFetch, a technique for automatically generating prefetch specifications using traversal profiling in object persistence architectures. AutoFetch generates prefetch specifications based on previous executions of similar queries. In contrast to previous work, AutoFetch can fetch arbitrary traversal patterns and can execute the optimal number of queries. AutoFetch has been implemented as an extension of Hibernate. It demonstrates that AutoFetch improves performance of traversals in the OO7 benchmark and can automatically predict prefetches that are equivalent to hand-coded queries, while supporting more modular program designs.

Heung Seok Jeon [2006] present the SA-W²R scheme that integrates buffer management and prefetching, where prefetching is done constantly in aggressive fashion. The scheme is simple to
implement making it a feasible solution in real systems. In its basic form, for buffer replacement, it uses the LRU policy. However, its modular design allows for any replacement policy to be incorporated into the scheme. For prefetching, it uses the LRU-one block lookahead (LRU-OBL) approach, eliminating any extra burden that is generally necessary in other prefetching approaches. Implementation studies based on the GNU/Linux show that the SA-W^2R performs better than the GNU/Linux with a maximum increase of 23% for the workloads considered.

**Pallis, Vakali and Pokorny [2007]** present an integrated approach which combines effectively caching and prefetching. Specifically, Web caching and prefetching can complement each other since the first one exploits the temporal locality whereas the second one utilizes the spatial locality of the Web objects.

**Raju, and Satyanarayana [2007]** propose a novel approach called Cluster and PreFetch (CPF) for prefetching of web pages based on the Adaptive Resonance Theory (ART) neural network clustering algorithm. First, we cluster the users and then we prefetch the web pages for each cluster before the users request them. Experiments have been conducted and results show that prediction accuracy of our CPF approach is as high as 98.38 percent. Our CPF approach effectively reduces the user perceived latency without wasting the network resources.

**Dhawal Navinbhai Thakker [2009]** shows that clustering and prefetching techniques over the network so that streaming applications can run without jitter and demand accesses can be satisfied in reasonable time. Experimental evaluation demonstrated that the proposed design can provide the required QoS. It also showed that it can be easily integrated into existing infrastructure, at the file system level and at the block level showed that based on the networking environment it is possible to provide QoS for streaming applications to run without jitter and demand misses to be satisfied in reasonable time over modern computer networks.

**Lewis, Alghamdi, Assaf, Ruan, Ding, Qin[2010]** proposed an automatic prefetching and caching system (or APACCS for short), which mitigates all of these shortcomings through three unique techniques, namely: and one of the technique is prefetch buffer management.
In all experiments, APACS performed measurably better than the alternative strategies. Prefetch buffers are allocated dynamically according to an optimization analysis using the cost-benefit model. Interestingly, in many cases when the prefetch hit ratio is negligible, APACS still outperforms LRU. This happens because many of the files that LRU flushes from the cache remain in the prefetch buffer.