Architecture of a Highly Scalable Data Warehouse Appliance Integrated to Mainframe Database Systems

Knut Stolze[1] Felix Beier[1,2] Kai-Uwe Sattler[2]
Sebastian Sprenger[1] Carlos Caballero Grolimund[1] Marco Czech[1]

[1]IBM Research & Development, Böblingen, Germany
[2]Ilmenau University of Technology, Germany

Main memory processing and data compression are valuable techniques to address the new challenges of data warehousing regarding scalability, large data volumes, near realtime response times, and the tight connection to OLTP. The IBM Smart Analytics Optimizer (ISAOPT) is a data warehouse appliance that implements a main memory database system for OLAP workloads using a cluster-based architecture. It is tightly integrated with IBM DB2 for z/OS (DB2) to speed up complex queries issued against DB2. In this paper, we focus on autonomic cluster management, high availability, and incremental update mechanisms for data maintenance in ISAOPT.

1 Introduction

For supporting management decisions, more and more companies gather business data in data warehouses (DWH) and evaluate it with the help of modern business intelligence (BI) tools. Reports are used for identifying market trends, conducting risk assessments, performing customer segmentations, and many other analytic tasks. These analyses are considered as business critical by an increasing number of companies and the trend evolves from static analyses towards ad-hoc reporting by a large user group with varying skill levels [Gar07]. The data warehouse systems are therefore faced with new challenges for handling enormous amounts of data with acceptable response times for analytical queries.

Thus, several techniques have been developed for achieving near realtime response times for reporting queries. They reduce the number of required operations for computing results and try keeping processing units fully utilized with reducing the I/O bottleneck when transferring data between different layers in the memory hierarchy. In combination with massive parallel computations, data throughputs can be increased by orders of magnitude.

Special data structures and algorithms have been developed that exploit access characteristics of critical system workloads for reducing the amount of data that has to be processed and efficiently applying typical operators on it. Prominent examples are compressed, read optimized data structures, cache conscious indexes and query operators, as well as operations which are executed directly on compressed data.

Since prices for RAM chips drastically decreased within the last years [Joh02], it nowadays becomes feasible to store the entire warehouse data – or at least the critical part of it – in fast main memory. More and more vendors of modern data warehouse solutions therefore pit on the application of main memory databases (MMDB) or hybrid solutions using a large main memory buffer for the data.

For scalability reasons, typically computer clusters consisting of several nodes are used. Further nodes can be added when the amount of data which has to be handled increases. Moreover, increasing the cluster size may reduce query response times when parallel computations can be efficiently utilized [DG92].

These approaches lead to new challenges in the system architecture which have to be addressed. Because companies rely on the availability and fast response times of their decision support systems, downtimes of latter can become very expensive and have to be avoided. But sometimes errors occur which must be handled internally and transparently, especially in the context of an appliance. Nodes in the cluster might fail and others may take over their parts for keeping the overall system alive. On the other hand, in case the cluster size is increased to scale up the system, the workload should be redistributed for an equal utilization for each of the nodes to minimize overall query response times. Along the same line, mechanisms must be available, which allow a simple upgrade and – equally important – downgrade of the installed software, including automatic forward and backward migration of all internal data structures, ranging from the actual data managed by the system, over the catalog to the actual management of the operating system.

Further problems occur when updates have to be processed on the read optimized DWH systems. They can lead to degenerations of the typically compressed and densely packed data structures. These degenerations have to be removed from time to time for maintaining the system operational and keeping the high performance commitments to user applications. Because the system must be available all the time, offline maintenance operations are not an option. Both – updates to the data and reorganizations of data structures – therefore have to be executed online while concurrent queries are running, without major negative impacts on latter.

All these challenges should be addressed without increasing complexity of system administration. Therefore, an appliance-like approach where most of the maintenance tasks are performed transparently and autonomously without needing interventions by the administrator is the most promising. In this paper we focus on the autonomic cluster management implemented by IBM Smart Analytics Optimizer (ISAOPT) data warehouse appliance [IBM10], which is a cluster-based main memory database management system (MMDBMS).

The remainder of the paper is structured as follows. In section 2 we summarize related work for high availability and administration questions of main memory DWH systems. In section 3 we give an overview on ISAOPT and its integration with IBM DB2 for z/OS. Section 4 discusses the autonomic cluster management features of ISAOPT and how software management (including forward/backward migration) is realized. Data management using incremental updates to synchronize ISAOPT with DB2 is described in section 5. Finally, section 6 concludes the paper and gives an outlook to further research directions.

2 Related Work

ParAccel Analytic Database (PAADB) [Par09] is an analytical high performance DBMS comparable to ISAOPT, implementing a hybrid architecture for both, disk-based query execution with a large main memory buffer and for databases that reside completely in RAM. The data is stored in a cluster with either a shared nothing or shared disk architecture. PAADB uses three types of cluster nodes, all running on commodity hardware. *Leader nodes* provide the interface for communicating with external applications. They are responsible for parsing and scheduling queries, as well as cluster workload management. *Compute nodes* actually store the data highly compressed in a columnar layout and execute tasks scheduled by leader nodes. The main query performance is achieved without tuning mechanisms like MQTs or indexes, but with massive parallel computations. Last, *hot standby nodes* are used for cases other nodes fail to take over their work. Lost data in a failover scenario is recovered automatically depending on the cluster architecture. It is either reloaded from shared disk or from backup replications stored on other nodes.

Vertica [Ver08] also implements a parallel DBMS on a cluster with either a shared nothing or shared disk architecture. High availability is achieved like in PAADB with replicating data. If nodes fail, other ones can take over their workload. The Vertica cluster implements k-safety, i. e. k nodes may fail without causing a system downtime. The overhead for storing replicated data is compensated using aggressive compression schemes on columnar data structures. Since updating these read optimized structures causes some problems, Vertica introduces a write optimized staging area where updates are written to. These changes are applied to the read optimized part by a *tuple mover* during runtime. Several algorithms have been investigated for performing these modifications online in the background to concurrent query execution. [SI09]. Vertica uses a snapshot isolation approach for hiding modifications from concurrent queries without locking overhead.

3 Overview of the IBM Smart Analytics Optimizer

ISAOPT is an appliance consisting of a computer cluster and a parallel MMDBMS. It was developed based on the Blink query processor [RSQ+08] for reacting to the trends in modern analytical data warehousing environments. It is aimed to improve warehousing on System z. Because customers using DB2 for z/OS typically pay for computations on mainframes, executing computationally intensive ad hoc reporting or data mining workloads have a noticeable price tag. Thus, ISAOPT is a cost-efficient solution for them, which also delivers a significant performance improvement for such SQL queries.

The main idea is offering an hybrid approach for supporting both OLTP and OLAP workloads with quality of service guarantees close to those for System z. As illustrated in fig 1, OLTP transactions are executed directly by DB2 for z/OS and expensive OLAP queries are offloaded by DB2 transparently to ISAOPT. The appliance is attached to the mainframe via TCP/IP network. DB2 recognizes it as an available resource and offloads query blocks only if it is deemed to be beneficial, based on the decision by DB2's cost-based optimizer.

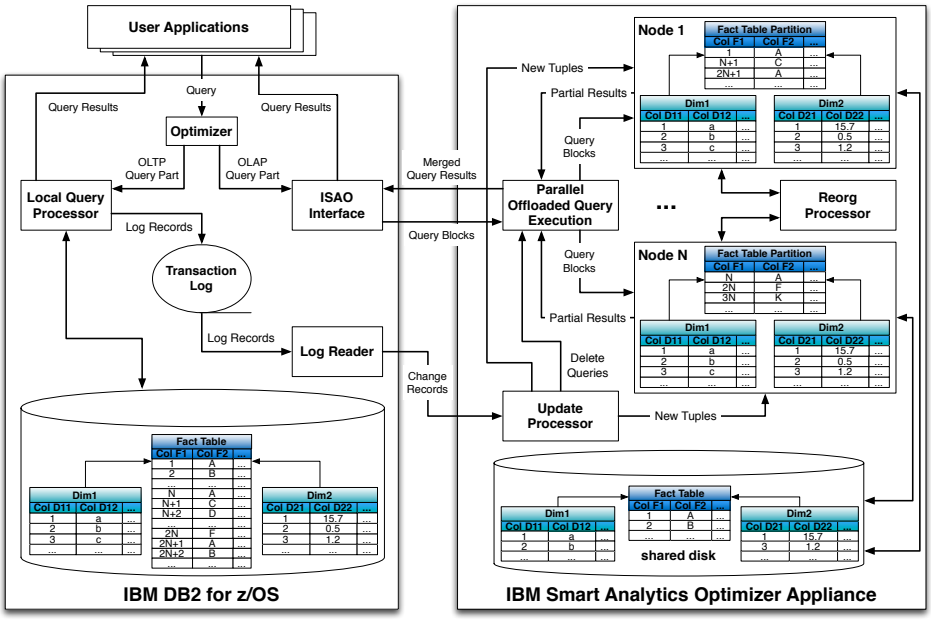

Figure 1: Overview: IBM Smart Analytics Optimizer

Complex analytical queries are split into query blocks which are executed in parallel by ISAOPT on several physical nodes in a cluster. Their partial results are merged and transferred back to DB2 completely transparent to user applications which still use DB2's SQL interface without further configuration or source code modifications. ISAOPT's available main memory and processing power can be scaled by extending the cluster size. Each node uses specialized and dedicated commodity hardware (blades) and therefore executing data warehouse tasks on them is more cost efficient than on general purpose systems like mainframes – with the penalty of lower hardware quality and, thus, a higher chance for outages. But latter is compensated by ISAOPT's cluster monitoring and recovery mechanisms.

Those parts of the warehouse data (called data marts in the following) which are critical for the analytical workload are replicated to ISAOPT. While the data is primarily stored on disks in DB2 (and supported by DB2's buffer pools), it is completely held highly compressed in the main memory of the appliance (and only backed with disk storage for recovery purposes). The used compression technique allows a fast evaluation of equality and range predicates using bitmasks directly on the compressed values and hence avoids an expensive decompression in many cases. [RSQ+08] Furthermore, a cache efficient data structure is used which enables the predicate evaluation of queries with vector operations. [JRSS08] Through these techniques, queries can be accelerated by orders of magnitude despite the fact that no workload specific tuning techniques like indexes or materialized views are used. [Dra09] Due to that, no special workload knowledge is necessary by database administrators for guaranteeing high performance of their systems.

4 Cluster Management

ISAOPT uses two different types of nodes. Each query being processed by ISAOPT runs on a single coordinator, which orchestrates the overall query execution, and on all workers. Due to the integration with DB2 for z/OS, the DB2 for z/OS acts as the client sending each query to another coordinator. The decision is based on the current workload of each coordinator, in order to achieve good workload balancing behavior.

Coordinators. A coordinator node does not hold any data in its main memory. It uses its memory to merge partial query results received from worker nodes and to apply post-processing, e. g. sort rows of a result set or compute aggregates like `AVG`. Since it is not known in advance, how big the intermediate result sets may be, all the coordinators' main memory has to be available – and still it may be necessary to resort to external sort algorithms using external storage, for example.

Another task for coordinator nodes comes with system failures. Since multiple coordinators are available, one of them can take over the data and workload of a failed worker, reducing the impact on other workers – which may already run with high memory pressure. In contrast to PAADB, no hot standby nodes are employed because the hardware resources of such nodes can be put to the task of workload processing without loosing any time in case of a system failure.

Workers. A worker node reserves most of its physical main memory to hold the mart data. Only about 30% of the total memory is available for temporary intermediate query results as well as all other operations like monitoring and autonomic system maintenance.

4.1 System Failures

The IBM Smart Analytics Optimizer appliance realizes a shared-disk architecture so that it can easily recover system failures without losing data. All mart data kept in main memory is also stored on disk for backup purposes. The used commodity hardware cannot match the stability of System z, resulting in an increased risk of hardware failures. Reloading the data from DB2 and re-compressing it would be too expensive in terms of time needed for the recovery and CPU overhead for IBM DB2 for z/OS.

Storing the compressed data on a dedicated storage system as-is expedites the recovery process with the least amount of overhead. Mirroring the data in the cluster appears to offer another solution with even faster recovery, but it comes with the penalty that it impacts the most scarce resource in a main-memory database system in terms of scalability: main memory.

4.1.1 Process Failures

A dedicated process is running on each node in the MMDBS cluster. That process may fail, e. g. due to programming errors. Very fast recovery is often achievable with a simple restart of the process and the node is operational again almost instantly. To accomplish that in ISAOPT, all data is loaded into shared memory and will always survive a failing process because it is not owned by its address space. The recovery time is mostly dominated by the time to detect the process failure and to restart it (a few seconds) and not by the time needed for reloading the backup from shared disk (potentially several minutes). Thus, the outage of the overall cluster will be much shorter, delivering on the promise of high availability that is expected by customers today even for unplanned outages.

4.1.2 Node Failures

If a complete node fails, e. g. due to hardware problems encountered at run-time or a problem in the operating system kernel and its drivers, it may happen that the node cannot be restarted (automatically or even manually) and needs to be replaced. In order to keep the cluster operational, redundancy mechanisms are required to detect node failures and initiate recovery steps if necessary. Waiting for the hardware problem to be repaired is not an option in customer environments. The recovery steps include attempts to restart the process on the failing node first, then rebooting the blade, and if that also fails, autonomically removing the node from the cluster and distributing its data and workload to other nodes. All these steps are implemented in the IBM Smart Analytics Optimizer.

The redistribution of the data is based on the shared file system holding the backup information, just created for these purposes. ISAOPT employs GPFS [SH02], a high-performance and fail-safe cluster file system, which has been further enhanced to be fully auto-configurable and auto-recoverable in the context of the ISAOPT appliance. Since GPFS does not yet support IPv6, the requirement came up to automatically configure an IPv4 network for it. The IPv6 automatic configuration features are used to detect which nodes exist in the cluster and to set up an IPv4 class C subnet. That is done during the boot process of each node. A small stand-alone tool is employed for the node detection. DHCP is not being used because its full power was not needed and a DHCP server must not become a single point of failure either. Furthermore, configuring a fail-safe DHCP server automatically would have been a more involved task as well.

4.1.3 System Inconsistencies

In rare cases, it may happen that one of the nodes in the cluster is not running with the correct software version, e. g. because that node was not active in the cluster when a newer (or older) version was installed and activated. Corruptions on the operating system are another problem that the ISAOPT appliance has to be able to deal with automatically. Manual intervention from a support specialist can only be the very last resort. The expectation of users is that such problems are usually detected and recovered by the system itself, i. e. a self-healing approach is required. For that purpose, ISAOPT maintains a so-

called *reference system image*, in which all administrative tasks are applied first. During each node's boot process, the local installation is verified with this reference image and re-synchronized when necessary.

Avoiding such system inconsistencies by using a network boot mechanism may sound attractive, but the network boot server would introduce a single-point of failure. The storage system holding the *reference system image* does not suffer from this problem due to its own redundancy and failure recovery mechanisms.

4.2 Cluster Administration

Cluster administration sums up tasks that are typically performed by a support specialist as a follow-up for hardware failures or newly detected and fixed vulnerabilities. It ranges from replacing broken hardware (e. g. defective DIMMs) over installing new firmware versions and drivers for components like network cards to adding new nodes to an existing cluster for increasing its memory capacity and processing power. Therefore, it may be necessary to remove an existing node from the cluster – while not interrupting the cluster's availability for query processing – or to add a new node to it.

The preparations for such tasks are built on the above described recovery mechanisms, except that the redistribution of the data and workload is explicitly initiated. Due to the shared-disk architecture, other nodes load the data of a node to be removed from the cluster from shared disk and take over the node's workload. The loading of the node's data and workload draining is achieved within a few minutes and can be done in advance so that there is no externally visible outage of the appliance.

Newly added nodes exploit the consistency checks of the operating system level (described above) to make sure the correct version is running. That means, only a very small base system (just 50MB) is needed on new nodes. This base system has to be able to boot the operating system kernel and to perform synchronizations with the reference system image.

4.3 Software Updates & Automatic Migration

A cluster-based MMDBMS must have the capability to switch between different versions of the appliance. That does not only mean to move from the current version of the software to a newer one – it should also be possible to easily rollback to a previous software version. The latter requirement is not often found in todays software products, but it is simply expected by customers of System z, who have the capabilities to upgrade a system like IBM DB2 for z/OS without taking it offline at all if it runs in a cluster. Since ISAOPT is integrated with DB2 like any other service (cf. fig 2), those requirements are transferred from System z directly to it as well.

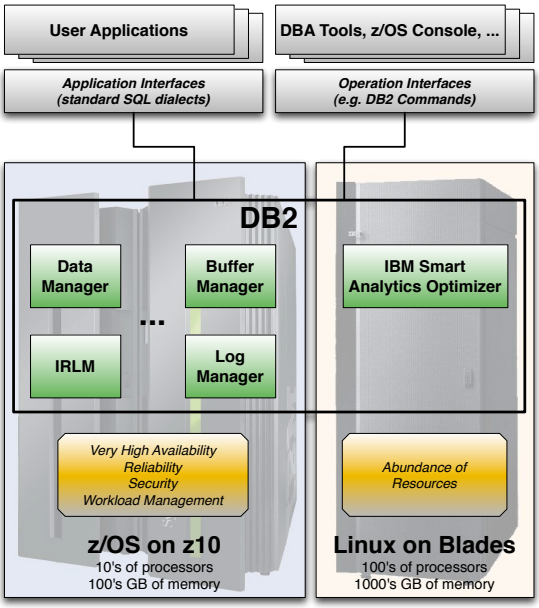

Figure 2: DB2 Integration with zHybrid Architecture

A stored procedure executing in DB2 for z/OS is used to deploy new software versions and to switch between already installed versions in ISAOPT. Forward migration, i. e. migrating structures from an older to a newer version is straight-forward. It can be applied transparently in most cases when the new version is activated for the first time. However, backward migration may be necessary if the newer version may not work as expected (e. g. performance degradations, stability issues, . . .).

Transparent migration can not be done when an older version is activated if there were differences in the data structures or the meta data compared to the currently running software version. Therefore, migration functionality is decoupled from the specific version of the ISAOPT server code into its own, light-weight executable. This tool is *always* used in its most-current state. That way, it is ensured that the tool can perform a migration step, even if the current ISAOPT version may not be capable.

5 Incremental Updates and Data Maintenance

Naturally, the data in a data warehouse has to be maintained incrementally. It does not matter if it is a standalone data warehouse system or an accelerator like ISAOPT, which is integrated with DB2 for z/OS and stores its own replica of the original warehouse data. In both cases, reloading a complete mart would just take too long because multiple TBs of data would have to be extracted, transformed, transferred, and loaded again.

Two techniques are being implemented by ISAOPT to update the data and autonomic reorganization is realized as well. Those tasks are running in the background to normal query processing without any downtime to the latter. ISAOPT implements snapshot semantic for all tuples like Vertica (cf. Sect. 2). Unique timestamp IDs (called *epochs*) and an additional range predicate on them are used to guarantee consistent views on the data for running queries while updates are active. [SRSD09] This approach allows a lock-free parallel execution of these concurrent tasks. Furthermore, failing updates can easily be rolled back by undoing the modifications marked with the respective epoch ID.

5.1 Log-based Update

The first update strategy is illustrated in fig. 1. When data stored in source tables in IBM DB2 for z/OS is modified, change records are written into the database log. This delta for committed transactions is read by a log reader, potentially compressed to merge multiple update operations on the same tuples, sent to ISAOPT, and applied there. [SBLC00].

To mark tuples as deleted in ISAOPT, the data of the updated table is scanned in its entirety and the delete epoch for affected tuples is set. Those tuples are not physically deleted and remain in place so that concurrent queries are not interrupted, i. e. setting the delete epoch is transparent for them. New tuples are distributed in the cluster, dictionary encoded, and written directly into the read optimized blocks. A special handling is only required for new values which have no corresponding dictionary code. [SRSD09]

Unfortunately, the prerequisite that updates are logged is not common for customer warehousing environments where many operations usually may bypass the transaction log in order to improve performance, for example when loading a new set of data for sales of the previous day. Nevertheless, if applicable, log-based update is the most efficient method with minimal overhead and can be used for realtime updates.

5.2 Partition-based Update

The second update approach implemented in ISAOPT can be used in all environments where no log records are available or where real time updates are not important and can be processed batch wise in larger time intervals. This solution exploits the fact that big tables are usually horizontally partitioned by ranges where each partition contains a disjoint subset of the table's data. This range partitioning is exploited by ISAOPT to track partitions, which were newly added or have been removed. For removed partitions, the corresponding data is deleted in each replica and all tuples from new partitions are automatically loaded. Additionally, the administrator can specify which table partitions have data modifications e. g. resulting from a LOAD or similar operation. These updates are locally limited to the affected partitions and can be applied with simply re-loading the outdated data.

This solution has a higher overhead than log-based updates because it may happen that some data will be transferred to ISAOPT's replica even if that data actually didn't change. The amount of the overhead depends on the table partitioning granularity, the update frequency on the data warehouse data, and the time intervals when synchronizations between source tables and replicas are initiated.

5.3 Mart Reorganization

When updates are executed, several degenerations can occur which have a negative impact on system performance and scalability. For example, the following degenerations can be observed in ISAOPT:

- New attribute values that could not be considered for dictionary computation in the frequency partitioning algorithm will not have order preserving dictionary codes assigned. They are covered with special cell types as described in [SRSD09] which can not be used for fast range predicate evaluations directly on compressed data. If a tuple has to be stored in the *catch-all cell* no compression can be applied for it.
- Because the frequency partitioning algorithm determines code lengths depending on attribute value histograms when the mart is loaded [RSQ+08], shifts in data distributions through updates can have negative impacts on the overall efficiency of the dictionary encoding.
- During updates, tuples must not be physically removed as long as previously started queries are running. With the snapshot semantics, the affected tuples are marked as deleted and leave gaps in the data structures as soon as the last query needing them in it's data view finishes. These gaps waste memory and negatively influence query runtimes because the deleted tuples have to be tested for validity in each query snapshot. Since tracking these gaps for overwriting them during following updates is too expensive, they remain in the data structures and require a clean-up.

To address such problems, a reorganization process is needed in cluster-based main memory database systems. Following autonomic principles, a self-scheduled process is used, which is triggered as soon as a reorganization is needed. The decision whether – and which – reorganization should be performed is based on statistics that need to collected in the system. For example, ISAOPT gathers the number of tuples marked as deleted or stored in the *catch-all cell* and *extension cells* and it also collects automatically information about the executed query workload. Thus, the reorganization task can be a simple removal of cell blocks that only contain deleted tuples, compacting the data by physically removing deleted tuples no longer needed by any query in cell blocks, a rearrangement of the storage layout based on the statistics derived from the query workload, or even the computing and application of a a new compression scheme.

Like updates, the internal reorganizations in a main-memory database system like ISAOPT have to be executed online without major impact on query execution. But required system resources like main memory and CPU cycles are critical resources. The reorg process of ISAOPT therefore implements an autonomic adaptation and throttling of resources available for the task at any time, based on the current workload that is active in the system.

6 Conclusion

A highly scalable cluster-based MMDBMS like the IBM Smart Analytics Optimizer has many similarities to traditional disk-based database systems in terms of its architecture. In this paper we presented two important aspects of ISAOPT: cluster management and data maintenance with incremental updates and online data reorganization. Cluster management comprises the automatic detection of system failures and initiating of recovery procedures.

Since ISAOPT is a cluster-based MMDBMS, system consistency across nodes in the cluster has to be ensured, for example in the context of software updates. Mismatches in the software versions are detected and recovery is triggered. The appliance form factor relieves the administrator of all such tasks because they are run autonomically. Thus, administrators only have to take care of deploying software versions and activating the desired version. Migration of data structures and meta data itself is handled transparently by ISAOPT.

Furthermore, no administration efforts are required for maintaining the internal data structures after updates. The appliance autonomically detects degenerations and repairs latter in the background to query processing without major noticeable performance impacts. This guarantees the efficient data structure layout for query processing as well as optimal compression rates to save system resources.

Future directions in the product development will focus on further improving system recovery times and also develop strategies for disaster recovery. Given that the ISAOPT appliance is tightly integrated with DB2, thus it also has to fit into the recovery mechanisms of DB2 itself. Regarding the autonomic management of user data, using ISAOPT's maintenance process for reorganizations as self tuning mechanism is a topic for current research activities. These operations can be exploited to restructure and recompress the internal data structures depending on the monitored system workload with intent to reduce query response times and a more efficient utilization of available system resources.

References

[DG92] David DeWitt and Jim Gray. Parallel database systems: the future of high performance database systems. *Commun. ACM*, 35(6):85–98, 1992.

[Dra09] Oliver Draese. IBM Smart Analytics Optimizer Architecture and Overview. Presentation Slides IOD 2009 Session Number TDZ-2711, IBM Corp., October 2009.

[Gar07] Intelligent Users Use Business Intelligence: From sports teams to traffic police to banks to hospitals, organizations are using analytical tools to turn data into knowledge. www.computerworld.com, 2007.

[IBM10] IBM Corp. *IBM Smart Analytics Optimizer for DB2 for z/OS V1.1 User's Guide*, November 2010.

[Joh02] McCallum John. *The Computer Engineering Handbook*, chapter Price-Performance of Computer Technology, pages 4–1 – 4–18. CRC Press, 2002.

[JRSS08] Ryan Johnson, Vijayshankar Raman, Richard Sidle, and Garret Swart. Row-wise parallel predicate evaluation. *Proc. VLDB Endow.*, 1(1):622–634, 2008.

[Par09] The ParAccel Analytic Database: A Technical Overview. Whitepaper, ParAccel Inc, 2009.

[RSQ+08] Vijayshankar Raman, Garret Swart, Lin Qiao, Frederick Reiss, Vijay Dialani, Donald Kossmann, Inderpal Narang, and Richard Sidle. Constant-Time Query Processing. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 60–69, Washington, DC, USA, 2008. IEEE Computer Society.

[SBLC00] Kenneth Salem, Kevin Beyer, Bruce Lindsay, and Roberta Cochrane. How to roll a join: asynchronous incremental view maintenance. *SIGMOD Rec.*, 29(2):129–140, 2000.

[SH02] Frank Schmuck and Roger Haskin. GPFS: A Shared-Disk File System for Large Computing Clusters. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, page 19, Berkeley, CA, USA, 2002. USENIX Association.

[SI09] Gary H. Sockut and Balakrishna R. Iyer. Online reorganization of databases. *ACM Comput. Surv.*, 41(3):1–136, 2009.

[SRSD09] Knut Stolze, Vijayshankar Raman, Richard Sidle, and O. Draese. Bringing BLINK Closer to the Full Power of SQL. In Johann Christoph Freytag, Thomas Ruf, Wolfgang Lehner, and Gottfried Vossen, editors, *BTW*, volume 144 of *LNI*, pages 157–166. GI, 2009.

[Ver08] The Vertica Analytic Database Technical Overview White Paper: A DBMS Architecture Optimized for Next-Generation Data Warehousing. Whitepaper, Vertica Systems, 2008.