An evolphin® Software Technology White Paper

evolphin software

Revolution in asset versioning & workflow

2410 Camino Ramon #228

San Ramon, CA 94583

+1 888-386-4114

info@evolphin.com

www.evolphin.com

Why Database Matters in Digital Asset Management

Rahul Bhargava

Chief Technical Officer

evolphin Software, Inc.

Contents

Executive Summary

According to an IDC 2011 paper, "Extracting Value from Chaos," the digital universe is exploding. In 2011 the world created a staggering 1.8 zettabytes. By 2020 the world will generate 50 times that amount of information. Creative organizations and their IT staff are scrambling to manage this avalanche of digital assets.

The need for a scalable Digital Asset Management system (DAM) is now more important than ever. The challenge however is that traditional database technologies will not support this level of scalability creating a problem for current DAM solutions. Existing database solutions were designed to manage structured data and not the types of digital files that are now proliferating the creative organization.

DAM software, the primary tool for managing large amounts of data and an equally large amount of users, is an unfortunate victim of yesterday's database technology. Most traditional databases were designed when hardware was much more expensive and did not have nearly the processing power that is available today.

A new breed of database has emerged that can scale to meet the needs of creative organizations and still provide the necessary assurances that the data will be consistent.

evolphin's RevDB database has been rigorously evaluated, tested and proven to scale to massive proportions as demonstrated in recent benchmarking tests performed by Azul, an evolphin partner and leader in no-pause ultra-low latency Java Virtual Machine (JVM) technology.

Brief History of Database Technology

Database management systems (DBMSs) have been around for many years. In the 1970s Relational Database Management Systems (RDBMS) came into vogue for storing structured data such as employee records or sales records, each of them well suited for a RDBMS. SQL (Structured Query Language), a mechanism for searching and finding such records in a RDBMS such as Oracle, DB2, or MySQL has also become an industry standard.

The ubiquity of SQL and RDBMS make them the database solution of choice for software architects building enterprise applications. It is no surprise therefore that most Digital Asset Management (DAM) systems in the marketplace have been built on top of a RDBMS. Unfortunately, when these systems were first designed, the software architects did not consider the implications of storing unstructured data in a database designed specifically for structured data.

In the past decade NoSQL database systems have been gaining traction. These databases have the following characteristics:

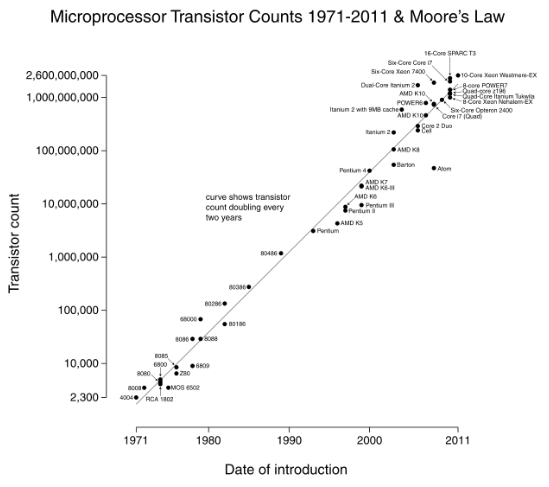
- They do not require a fixed table schema
- They avoid costly join operations common in RDBMSs by storing de-normalized structured data
- They are often targeted at applications without the need for strong consistency[1] requirements.

This presents a dilemma to software architects who require the benefits of an SQL free approach but also want strong consistency guarantees on the stored data.

Supported by the latest research in very large database systems (VLDB), a new breed of database solutions are emerging to address the shortcomings of NoSQL databases while providing the atomicity, consistency, integrity and durability (ACID) guarantees needed by applications such as a DAM. evolphin Software's RevDB is a new breed of database.

[1] Consistency as defined in the theory of database systems.

CPU & Hardware Trends



In the graph above it is evident that the number of transistors that can be fitted onto a single chip has grown from around 10 million to over 2 billion in the last decade. This means that a larger number of cores can exist within a single processor. IT departments now have the ability to consolidate multiple servers into a single massively scalable system.

Despite the advances in both processing power and memory, one bottleneck has remained; hard disk. While SSD systems providing fast performance are gaining traction in the consumer space, storing terabytes of enterprise content is still the realm of magnetic disks. These disks remain the most cost effective way of storing large digital assets. Therefore, disk I/O will continue to be a bottleneck for the foreseeable future.

Clearly there are major implications for applications such as Digital Asset Management that require high compute power in addition to fast disk access.

Why not to use a Traditional SQL RDBMS with DAM?

Not optimized for today's hardware

In 2000 a Sun Microsystems SPARCstation 2000 with 5GB of memory was considered prohibitively expensive. Today, at a similar price point, a system with 512GB of RAM is feasible. Beyond memory, the number of processors (with multiple cores) that can be embedded onto a single motherboard has increased significantly. A system with 100+ cores is relatively affordable today whereas just 5 years back such a system would have been out of reach for most businesses.

SQL RDBMS were designed in an era when memory and computing power were both costly and limited. Most traditional RDBMS still use expensive disk based paging schemes to bring data from disk to memory. The number of open connections to the database driver limits the concurrency. It is common to have a database connection pool that is a fraction of the number of concurrent users accessing the system, making the database the major bottleneck from a performance & scalability perspective.

Applications end up creating their own caching layer for read-only data in order to circumvent the paging schemes and concurrency limitations presented by traditional RDBMS architectures.

Not optimized for blobs

Large unstructured data like digital assets cannot be managed efficiently using a RDBMS because the file data for digital assets is not a structured record. For instance a video asset is an example of unstructured data because it has many uncorrelated frames. An RDBMS would store this unstructured data as a blob (binary large object). However, searching and retrieving blobs can be slow, inefficient and cumbersome when using an interface like SQL that is optimized for serving tabular data.

Cost of SQL

SQL plays a major role in an RDBMS. All database operations are typically accessed through the SQL interface. The database must interpret each SQL query and then perform expensive join operations in order to return the search results. The application then consumes the records and creates an O-R (relation to object) mapping in the code to create usable objects for manipulating and streaming the results back to the client. RDBMS try to avoid costly SQL calls by caching the queries but with varying queries this model is difficult to scale.

Locking

A RDBMS has a complex locking mechanism for concurrency control to ensure data consistency and integrity. For instance, if a metadata row needs to be modified, the RDBMS may obtain a row level write lock that can block readers from accessing the database row causing "lock contention". In a DAM context this could mean locking thousands of rows when for instance metadata records need to be updated for an entire creative project. This in turn, reduces the overall throughput and performance of the DAM system due to significant lock contention.

Inconsistencies

Traditional DAMs often store metadata in the RDBMS and the actual file data on a network share. Some DAMs even store the actual metadata in a side file, with the database used for just indexing the actual metadata. In effect traditional DAMs behave like a glorified network file system.

This presents a challenge in keeping the file data, metadata and metadata index in sync. For example, if a user changes the metadata and file data of multiple files and then tries to ingest them using the above scheme, there is no way for the DAM to guarantee an atomic update to both the file system and the RDBMS tables.

The reason is that the RDBMS provides transactional semantics for its operations unlike the file system. There is no overarching "transaction manager" that coordinates the write operations to both systems. This can lead to scenarios where files may be modified or deleted but metadata tables are not, causing the two systems to become out of sync. The system administrator then has to manually rebuild the database tables to make them consistent with the file system.

The traditional DAM software architect is faced with two very clear problems:

1. If they use the RDBMS blob mechanism to store the file data then performance will be adversely affected.
2. Conversely, if they only store metadata in the RDBMS (and not the file data), they will not be able to guarantee data integrity and consistency.

Deployment headaches

Traditional DAM systems are often complex to deploy and manage; various components including a RDBMS that need to be setup and managed by a database administrator (DBA), makes operational maintenance and support more difficult than it should be. It is one thing to tune database queries to get the best performance out of the RDBMS used by the DAM, it is quite another to conduct fault diagnoses when multiple components and vendors are involved.

Why evolphin Built a Database from Scratch!

The evolphin engineering team has an extensive background in building scalable databases and distributed systems; experience gained with Silicon Valley companies that recognized the challenges and pitfalls of using RDBMS in a DAM system.

Several large companies like Google and Facebook have recognized the same issues and have invested heavily in building their own custom database and file systems to scale to large volumes of unstructured data. Those systems are proprietary and are not available to the public at large. In addition, the use case for these companies does not require strong consistency guarantees and transactional behavior from the database and the file system. DAM systems, on the other hand, store both structured and unstructured data that needs to be modifiable with strong consistency guarantees.

The evolphin engineers have created a next generation database technology that has already addressed the challenges and concerns outlined in this paper. With 8 patents pending, the RevDB database brings the capabilities outlined below to the DAM industry.

Designed for today's hardware

The first assumption is that 64-bit server systems will scale to large amounts of RAM (64GB+) as well as processor cores (32 cores or more) at a reasonable cost. At the time of writing this article, a hardware configuration with 64 GB RAM, 8x8 hyper-threaded Intel Xeon processors (64 virtual cores) would be a fraction of the cost compared to 10 years ago. This trend is likely to continue with the accelerated growth in the number of cores and amount of RAM on server machines. The RevDB database system takes advantage of hardware with massive parallel processing, using concurrent threads, as well as large in-memory cache of indexes that takes advantage of a copious amount of RAM in contrast to RDBMS indexes that reside on disk and therefore run into costly disk paging calls. This ensures a DAM application written on top of RevDB gets the maximum benefit from today's hardware.

Optimized for blobs

The Zoom database works in tandem with its built-in overlay transaction file store. Large digital assets can be stored as a blob (binary large object) in the Zoom file store without performance degradation and without the loss of any strong consistency guarantees. Zoom's deduplication technology is coupled with the RevDB blob storage mechanism to make it even more efficient to store revisions of work-in-progress digital assets.

Zoom ships with its own middleware/application server that further optimizes the network protocol for transferring large file data. This proprietary network transfer protocol tunnels over HTTP, ensuring that HTTP aware network devices such as firewalls do not have any issues.

No SQL
Zoom's RevDB exposes an object oriented interface to the database. There is no SQL to object mapping needed, as the database schema is directly compiled into objects. Therefore the Zoom RevDB completely avoids all the issues surrounding SQL based systems like traditional RDBMS.

Multiversion Concurrency Control
Zoom RevDB implements a state-of-the-art multiversion concurrency control scheme for data consistency in the presence of multiple readers and writers. This lock-free approach eliminates lock contention that is common in RDBMS. In addition the in-memory index model ensures RevDB does not need to hold page or range locks. Performance improvements are dramatic with this approach. Searching through millions of metadata records can be accomplished in milliseconds.

Atomicity
Zoom RevDB implements a transactional interface across the file system and the database thereby eliminating the biggest weakness of the traditional DAM approach of keeping the file system separate from the metadata index that is stored in an RDBMS. Now, when the end-user updates metadata and file data across multiple files, partial failures due to issues such as a network outage on the file system, do not cause the DAM state to become inconsistent because of partial or incomplete updates.

The Evidence

evolphin worked with Azul Systems, a provider of high performance Java Virtual Machine (JVM) technology, to benchmark the Zoom system in a real-world scenario with vast numbers of files and users.

The Setup at Azul
The Azul team setup an independent lab with a massively scalable server machine that included:

- 64 cores of compute power
- 512 GB of RAM
- 10 Gbps Ethernet LAN
- 10K RPM SAS disks
- Linux (CentOS 6.4 64-bit) operating system
- 10 million digital assets ingested into Zoom server

The evolphin Zoom server was configured to use Azul enhanced JVM, Zing, instead of the default Oracle JVM. Azul JVM implements a highly-scalable, no-pause GC (Garbage Collector) that can scale to TB+ RAM.

Benchmark
Zoom clients were setup on a Linux machine with similar specs as the server machine. An automated script was deployed to run a massive number of concurrent clients to simulate real-world load. Each Zoom client was setup to programmatically run several Zoom commands, such as browse and search, in parallel using the Zoom command line API. The parameters for these commands ensured the worst case scenario of searching through every single record in Zoom looking for matching assets. The Zoom server was configured to return 10,000 search results to simulate a large response that would place significant load on the server and to stream over the 10 Gbps network.

Results
Zoom server was injected with an increasing client load starting at 100 concurrent requests up to 3,000 concurrent requests within a second or less. At the peak that would translate to 180,000 requests every minute. In the real world, for every concurrent user there would typically be 10 users. This would translate to a single Zoom server supporting **over** 100,000 active users.

In the end, the Zoom server could not be loaded any further because resources outside the control of the Zoom server became saturated. These included: 1) Ethernet network, 2) Client machine acting as the load driver, and 3) Linux OS on the client side. Zoom server did not crash nor slow down, in fact, it was able to scale to 384GB of RAM using the Azul Zing JVM before the resources became saturated with the load.

Conclusions

Traditional RDBMS are not well suited to managing large volumes of digital assets. A fresh approach is needed and is now available in the form of VLDB systems. The evolphin RevDB database is one of the first such databases and when used for Digital Asset Management can provide on a single server the scalability, performance and reliability required by today's creative organizations.