Online Journal of Applied Knowledge Management
A Publication of the International Institute for Applied Knowledge Management

Volume 1, Issue 2, 2013

Managing Documents with NoSQL in Service Oriented Architecture

Milorad P. Stević, The Higher Education Technical School of Professional Studies, Novi Sad, Serbia, milorad.stevic@live.com

Abstract

The need for including ever more information and business processes in current information systems brought the necessity of introducing large-scale document handling in relationally organized systems. This leads into thinking what kind of architecture would satisfy the need for stability and consistency from one hand and high data throughput and effective document management from the other, without the change in already developed client interfaces for various platforms. Another concern was the need for efficient backup procedures for considerable amount of documents. This paper shows that it is possible to implement architecture that satisfy these needs - it is a merge of relational database system, service oriented architecture and appropriate NoSQL database system, with several client interface applications communicating with databases through services oblivious of the fact that they stopped using documents stored in relational database, but rather in NoSQL database. Adding NoSQL to service-oriented architecture gave the system stability and consistency using relational engine for structured data types, effective document management and full text search using RavenDB for unstructured data types and services for achieving uniform data delivery to heterogeneous client applications.

Keywords: Unstructured data types, NoSQL, Service-oriented architecture, Document management, Information systems, Partition tolerance, Scalability

Introduction

The information system (IS) analyzed in this paper is designed using service-oriented architecture approach with three layers: the database layer realized with Microsoft SQL Server database, the service layer realized with C# programming language and the front layer with several client interfaces for different platforms. In the beginning, successful modeling was very important for the core business processes and rules. Later on, requests for efficient document handling involving storing, archiving and searching for documents arose. Searching for selected text within previously stored documents and full text search capabilities became very important in the later stages of the development. Particular business requests are storing and archiving student thesis, seminars, projects and other school-related documents with mechanisms for detection the same or similar documents that would signalize fraud attempt. This involves some very large multimedia documents, especially within the graphic design and graphic engineering departments. Further, all documents regarding the distance learning system, also involving many large multimedia documents, must be stored in the system. The challenge has been even greater due to very limited funds, preventing the purchase of expensive equipment essential for storing and retrieving huge amounts of documents. Backup issues and availability of the system posed great demands on the system as well.

Whereas the middle and front layers could support newly introduced features without any difficulties, the database layer was harder to adapt. Relational database management system

(RDBMS) was built for handling structured data, but has limitations when it comes to managing large and numerous unstructured data like, in this case, documents (Stonebraker et al., 2007). Storing vast amount of unstructured data in binary type columns in RDBMS tremendously increases demands in both hardware and human resources. The future growth of data can be addressed through vertical scaling, but further burdens the resources (Kossmann, Kraska & Loesing, 2010). This requires complex replication, which is costly. Distributed systems based on RDBMS tend to be rigid and hard to administer. Using static scheme in RDBMS is excellent for managing structured data. However, it is rather demanding when it comes to unstructured data. For unstructured data, schema-free systems are better and easier for handling (Hellerstein, Stonebraker & Hamilton, 2007). NoSQL databases offer a different approach to this problem. They are designed for handling large amount of unstructured data by horizontal scaling, thus removing the obstacles regarding future growth of data (Hellerstein, Stonebraker & Hamilton, 2007). In addition to that, some of them offer a full text search capability, which is essential for this project (Kovacevic et al., 2011). On the other hand, most NoSQL databases are not ACID (Atomicity, Consistency, Isolation, Durability) but CAP (Consistency, Availability, Partition Tolerance), which is relevant when deciding whether to use file system, RDBMS or NoSQL (Wada et al., 2011; Stonebraker, 2010). In addition, isolation levels are different on various NoSQL platforms, if implemented at all, while considered mandatory on RDBMS (Fekete, Goldrei & Asenjo, 2009).

Since the most important goal in this project is to efficiently store and handle large-scale documents, the choice of NoSQL for handling unstructured documents must be justified. RDBMS is surely capable of doing the job, but it is not efficient enough in handling unstructured data. Although file system can be used for storing documents, it lacks mechanisms for handling documents that can be used automatically in IS. It is necessary to analyze all relevant technical information in order to confirm that NoSQL best suits the needs of this project. In addition, the great question is how to organize the system in order to RDBMS and NoSQL work together for the best. Because the next great task is full text search capability, the choice of RavenDB has been made because it incorporates Apache Lucene features (http://lucene.apache.org/) and is suitable for full text search based on fuzzy search (Kovacevic et al., 2011; Milosavljević, Boberić & Surla, 2010). This method of searching allows implementation of intelligent search methods in order to disclose fraud attempts. Considering RavenDB is equipped with horizontal scaling capabilities that require minimum administrative intervention, it became the choice for document handling issues.

Methods

Problem outline

This study was part of a greater project that dealt with building and implementing IS in High Technical School of Professional Studies in Novi Sad. There are 80 teachers and 1600 active students in 4 years of studies with average of 12 courses per year. Additionally, there are 2 distance learning programs actively held. In most courses it is expected from students to write at least 1 seminar and 1 project material per course and every student must write their thesis as final effort towards graduation. It is requested that every single document must be preserved in database so later access to material could be achieved. This is important from both educational and fraud detection reasons. In addition to these requests, it is necessary to provide 2 GB of personal space for every student and 20 GB of personal space for every teacher. Access to all files must be provided exclusively from various client interfaces that are part of the IS, both from within institution and from home and no other type of access is allowed for security reasons. Document version control must be enabled in order for a teacher to track down the progress of a student regarding every written assignment. Full text search capability is essential since it is a great amount of electronic material in stake. All these requests were combined with the demand of inexpensive implementation, so there were no rack-based servers and no storage area networks with high volume and high-speed disks.

Proposed hybrid structure

RDBMS was built to handle all the structured data about entities and processes in this project, but handling documents was something that was a matter of great importance and careful research prior making final decisions. It was important to ensure enough space for holding predicted amount of documents, safe access to documents and document existence in case of any kind of disaster. All those requests eliminated document handling based on a file system because of complex use and access control. Another circumstance, which denied use of file system approach, was the need for exclusive management of documents from within the IS. RDBMS was rejected as a potential solution for document management since there is no easy and inexpensive way of distributing RDBMS across many nodes. Replicating such distributed nodes would be even harder in some disaster-recovery scenario. There are ways to distribute RDBMS, but they are expensive and time consuming.

RavenDB was acceptable solution for managing documents: it can handle great amount of potentially large files and has no schema so it is suitable for unstructured data. It is distributed by design and running RavenDB on multiple nodes is the default way of running it. This was important because it was much easier to plan hardware acquisition for the project. Nodes need not be the same and it was possible to use almost any hardware infrastructure (Baker et al., 2011). All these properties helped achieving availability without complex, expensive hardware infrastructure. This enabled use of many small servers and adding or removing servers was easy task. Disaster recovery strategy was determined by the way distributivity was achieved - termination of a server was acceptable since all data was on multiple servers instantly and all that was needed to recover full availability is adding another server in the system (Abadi, 2012). At the same time, dealing with disaster recovery scenario solved backup issues - if the system was able to survive the disaster scenario, it was protected and well backed up.

Dimensioning the data

All findings were based on 1 academic year of observation, with 1600 students, 80 teachers, 10 study programs where multimedia and graphic design were not involved as mayor courses, 4 study programs based on multimedia and graphic design as mayor courses. There were 350 students that graduated in that academic year. Total of 477 different courses were held. In addition, for clarity reasons, all courses that required both text-based and multimedia-based assignment were shown as multimedia-based courses. Since these were only a few and multimedia files are much bigger that text-based files, this did not affect accuracy of the results. No course required more than one multimedia-based assignment.

Table 1: Structure of courses based on type of seminar work required

Type of course	**Number of courses**	**Number of participants**	**Number of documents**	**Avg size**	**Space**
Requires text-based seminar work	251	9650	1	1,7 MB	16,4 GB
Requires multimedia-based seminar work	51	3460	1	78 MB	269,9 GB
Does not require documents	175	6090	-	-	-
				Total	286,3 GB
Although sum of number of courses is equal to the total number of courses (477), it doesn't have to be this way, this is due to approximation taken for clarity reasons					

Table 2:Structure of courses based on type of project work required

Type of course	**Number of courses**	**Number of participants**	**Number of documents**	**Avg size**	**Space**
Requires text-based project	202	13928	1	2,3 MB	32 GB
Requires multimedia-based project	41	2205	1	83 MB	183 GB
Does not require documents	234	3067	-	-	-
				Total	215 GB
Although sum of number of courses is equal to the total number of courses (477), it doesn't have to be this way, this is due to approximation taken for clarity reasons					

Table 3: Structure of courses based on type of documents included in electronic material

Type of course	**Number of courses**	**Avg number of documents**	**Avg size**	**Space**
Contains text-based documents	455	7	3 MB	9,5 GB
Contains multimedia-based documents	359	15	20 MB	107,7 GB
Does not require documents	22	-	-	-
			Total	117,2 GB

Table 4: Structure of space needed for private use

User	**Number of users**	**Avg size**	**Space**
Teacher	80	20 GB	1,6 TB
Student	1600	2 GB	3,2 TB
		Total	4,8 TB

To the complete amount of space needed to handle documents it was necessary to add space for student thesis: 350 students times 5,2 MB (average space for one thesis) equals 1,8 GB of needed space on year basis.

Discussion

Implementation challenges

Since this project was not about achieving great speed and efficiency, the problem is reduced to finding the best solution for handling large-scale amount of documents in terms of storing and performing full text search on them. Storing documents means usage of disk space for documents in efficient manner regarding economy, security and scaling possibilities. Full text search was a request that was essential for the project since fraud attempt control was very important in educational institution. Storing is directly correlated with amount of space needed for keeping documents, while full text search is connected with amount of documents that are the target group for this operation. Further, it is important to segregate amount of documents that will represent increment on yearly basis so this trend could be tracked for scale-planning purposes. Separation of documents that are target to full text search from those that are not is important for search-planning purposes. This is where storage rules intersect with document segregation: it is important vise efficiency reasons to place documents that are either in group that is incrementing or in group that is targeted for full text search on storages that are most efficient, because it is expected that these documents will have heaviest access frequency.

This means that text-based documents are subject to full text search and multimedia-based documents are not. Seminars, projects and thesis are type of documents that will require new space every year since new students will take place of the old ones and generate new amounts of seminars, projects and thesis. Course content and student/professor private space will not increase over time, because it is expected that number of courses, students and professors will remain the same over time.

Table 5: Structure of documents that will and will not cause new space consumption every year

Document types	**Causing new space consumption every year**	**Static**
Seminars	286,3 GB	0
Projects	215 GB	0
Thesis	1,8 GB	0
Course content	0	117,2 GB
Private space	0	4,8 TB
Total	503,1 GB	4,9 TB

Table 6: Structure of documents regarding full text search

Document types	**Subject of full text search**	**Not subject of full text search**
Text-based (seminars, projects)	48,4 GB (16,4 GB + 32 GB)	0
Multimedia-based (seminars, projects)	0	452,9 GB (269,9 GB + 183 GB)
Thesis	1,8 GB	0
Course content	0	117,2 GB
Private space	0	4,8 TB
Total	50,2 GB	5,4 TB

Total amount of space needed for the project to start is 5,4 TB and amount of documents that is expected to be accessed very frequently is 503,1 GB which means that storage structure should be in accordance with this - 503,1 GB on as fast as affordable disks and the rest of 4,9 TB on slower disks. It is expected in 10-year period of time that space demands will be increased by 10 times 503,1 GB = 5 TB which gives a total of 10 TB of storage space.

Benefits of the selected solution

This trend shows that use of file system for storing and manipulating documents will be more and more tiring over time and must be based on expensive rack-based hardware configuration with very fast disks. Combining with redundancy needed to achieve disaster recovery capabilities, the request for storage space is even greater. In the case of disaster recovery scenario involving natural disasters (earthquake, flood, fire...) where distant storage is needed for recovery, it is very expensive choice. If security issues are added to this, where access control to documents is required it shows that in order to achieve this request, big administrative effort is expected to conduct access control for documents stored in the system. Another problem is accessing documents from within IS where location of a document is recorded inside RDBMS, but actual documents are stored on several disk locations. This poses the problem due to the need to achieve transactional manipulation of both meta data about the document and the document itself, for they are stored in different locations. Full text search on documents stored on a file system could be achieved using some proven technology like Lucene (Prasad & Patel, 2005; Milosavljević, Boberić & Surla, 2010).

Another potential solution is to store documents inside RDBMS, which is problematic from several points of view. Documents are unstructured data and RDBMS is not meant to deal well with these kinds of data, although it can be done. Horizontal scaling is hard to achieve in modern RDBMS and when it is to be achieved, it is by consuming a lot of human (administrative), hardware and financial resources. Disaster recovery scenarios are as problematic as with file system solution, since it has to be done using replication to a distance place with heavy load. This is problematic from capacity and bandwidth point of view. Good aspect of this solution is when backup is performed, both meta data and documents are backed up at once. Transactional manipulation of meta data about the documents and documents itself is well supported. This solution provides good access control mechanisms, which could be made internally inside IS. Full text search is a problem, since RDBMS is not made for this task, mainly because documents are unstructured data that can be very different in structure. However, modern RDBMS evolved and included some tools for this, depending of the vendor.

Using RavenDB for handling documents was the most appropriate solution because it deals with large number of documents and it is built for managing unstructured data. RavenDB runs as distributed system so several servers are serving documents at the same time, which is suitable for load balancing purposes (Wada et al., 2011). Server termination is acceptable by design, thus it can run on inexpensive hardware (Wada et al., 2011). Distributivity relies on horizontal scaling possibilities, which are almost automatic in RavenDB. A good approach for developing disaster recovery scenarios is enabled through support for efficient replication among distant servers. Unlike some other NoSQL solutions, RavenDB supports ACID characteristics. Backup solutions are as safe and efficient as with RDBMS since documents are stored within database. RavenDB allows full text search through proven Apache Lucene technology.

SOA Integration

Having RDBMS handling structured data and RavenDB handling unstructured data, it was necessary to integrate those two in the existing service-oriented architecture (SOA) based IS. For effective appliance of the proposed solution, it is important to integrate those platforms into a single database platform, from the client point of view. Since the development of various client platforms was planned as service-dependent, clients were not getting data from databases directly, but rather through the service layer. This data manipulation is done by the service layer of IS, as shown (Fig. 1). Because there were several different client interfaces to handle, it was important to come out with a solution to the problem that would require minimum effort for adoption. Service layer written in C# was appropriate solution, since it was possible to connect to either RDBMS or NoSQL from the service layer. This enabled the service layer to create the unique abstract data layer for various client interfaces. The service layer was doing data abstraction in accordance with client requests and sometimes it was gathering data from RDBMS, sometimes from NoSQL and sometimes from both. As a result, service layer was building objects and delivering those to the client layer. Client layer was not aware of the physical location of the data. When organized like this, the development of the client layer was much less error-prone, because data abstraction was done on the service layer of the SOA. Furthermore, it was possible to extend data management optimization because data could be handled and transferred as SOAP or as REST, as appropriate. Combining SOAP and REST based services into unique service layer widened the business potential of the platform, enabling simplified development of mobile-targeted client interfaces.

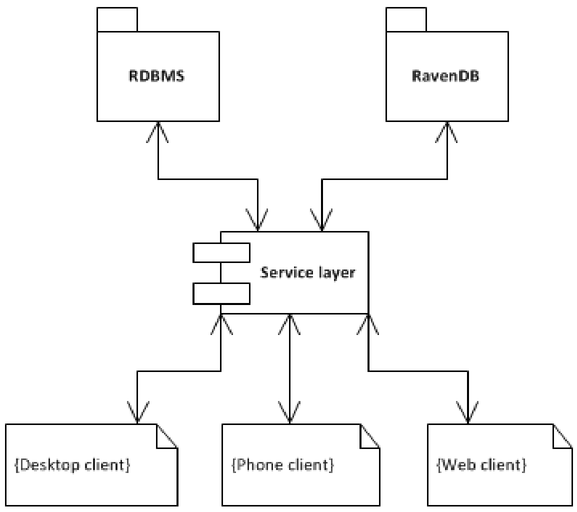

Fig. 1. Integration of RavenDB in existing SOA

Security constraints and data integrity

The proposed architecture is complex and requires appropriate security solution. Handling structured and sensitive data was the task entrusted to RDBMS, since it is robust and reliable database platform. RDBMS was in charge of conducting all authentication processes in the system. Every request that implied sensitive data manipulation generated from the client layer, regardless of the client type, was first checked for security authorization handled by the RDBMS. Only requests that were marked as valid on the RDBMS could perform data manipulation. Enforcing these rules enabled the system to be trustworthy since it was relying on RDBMS for security constraints. Service layer was central part of the process once again, since all client requests for data manipulation were going through this layer and it was possible to segregate security check processes and data manipulation processes.

Data integrity and access control was guaranteed because documents were kept in the NoSQL database, rather than on the file system. This approach denied unauthorized access to documents from the file-system level since documents were not stored on the file system. This means that although system administrator has high system privileges, data integrity of documents involved is preserved.

Implications for research

Further research could take place in measuring performances of a system that rely on different kinds of NoSQL, respectively. Since this project did not depend on performance characteristics of the solution, but rather on usability of the system based on scaling and full text search capabilities, measuring performances did not take place. Nevertheless, it would be very important to explore effectiveness of systems based on other NoSQL solutions in order to create recommendations and best practices for different usage scenarios.

Another field of research could be development of a system for automatic fraud attempt detection which would detect situations in which students could try to use prohibited amount of others work in their thesis, seminar or a project or try to use others work without proper approval. Rules would have to be set for these situations and if students break any of those rules, the system would take predesigned automated action.

Ecology is nowadays very important field of research. Possibility for deployment of used hardware as server instances for NoSQL, knowing that system is partition tolerant and can survive server termination is very interesting application of ecology in IS.

Implications for practice

Expanding existing IS to an area of document management is demanding and sensitive because it is expected from IS to handle new requests with ease, but securely at the same time. It is important to make expansion as inexpensive as possible. Implementing solution that includes NoSQL database for managing documents in SOA has significant implications for practitioners.

- SOA based systems can be extended with NoSQL database for handling unstructured data without changes on client layer because communication is realized between service and database layer and various client interface development continues without change.
- NoSQL is suitable for handling large amount of documents because it has horizontal scaling possibilities.
- Horizontal scaling and distributive characteristics enable the system to survive server termination.
- Disaster recovery capabilities are present even with large amount of data.
- Even old and used hardware could take place in a system like this because of partition tolerance part of CAP theorem.

Limitations

The study shows acceptable adoption of NoSQL technology inside particular SOA and any insight is limited to this context. When applied in different surroundings, the model could require adaptation.

Notes

1 Apache Lucene, available at: http://lucene.apache.org (accessed 01 February 2013).
2 RavenDB, available at: http://ravendb.net (accessed 01 February 2013).
3 The projects using Apache Lucene, available at: http://wiki.apache.org/jakarta-lucene/PoweredBy/ (accessed 01 February 2013).
4 Project example using RavenDB, available at: http://gorodinski.com/blog/2012/08/28/porting-from-sql-with-nhibernate-to-ravendb/ (accessed 01 February 2013).

References

Milosavljević, B., Boberić, D. & Surla, D. (2010). Retrieval of bibliographic records using Apache Lucene. *The Electronic Library, Vol. 28 No: 4*, pp. 525-539.

Prasad, A.R.D. & Patel, D. (2005). Lucene search engine: an overview. *Proceedings of the DRTC-HP International Workshop on Building Digital Libraries Using DSpace*, 7-11 March 2005, DRTC, Bangalore. Retrieved from http://drtc.isibang.ac.in/xmlui/bitstream/handle/1849/244/I_lucene%20search.pdf?sequence=1

Abadi, J. (2012). Consistency Tradeoffs in Modern Distributed Database System Design. *The IEEE Computer Society, Vol. 45 No: 2*, pp. 37-42.

Stonebraker, M. et al. (2007). The End of an Architectural Era (It's Time for a Complete Rewrite). *Proc. VLDB Endowment (VLDB 07), ACM.* pp. 1150-1160.

Baker, J. et al. (2011). Megastore: Providing Scalable, Highly Available Storage for Interactive Services. *Proc. 5th Biennial Conf. Innovative Data Systems Research (CIDR 11), ACM.* pp. 223-234.

Stonebraker, M. (2010). Errors in Database Systems, Eventual Consistency, and the CAP Theorem. *blog, Comm. ACM,5.* Retreived from http://cacm.acm.org/blogs/blog-cacm83396-errors-in-database-systems-eventual-consistency-and-the-cap-theorem.

Wada, H. et al. (2011). Data Consistency Properties and the Trade-offs in Commercial Cloud Storage: The Consumers' Perspective. *Proc. 5th Biennial Conf. Innovative Data Systems Research (CIDR 11), ACM,* pp. 134-143.

Fekete, A., Goldrei, S. & Asenjo, J. P. (2009). Quantifying isolation anomalies. *Proc Very Large Databases (VLDB'09)*, pages 467–478.

Kossmann, D., Kraska, T., & Loesing, S. (2010). An Evaluation of Alternative Architectures for Transaction Processing in the Cloud. *ACM International Conference on Management of Data*, pages 579–590. ACM.

Kovacevic, A. et al. (2011). Automatic extraction of metadata from scientific publications for CRIS systems. *Program: electronic library and information systems, Vol. 45 Iss: 4,* pp. 376 - 396.

Hellerstein, J. M., Stonebraker, M. & Hamilton, J. R. (2007). Architecture of a database system. *Foundations and Trends in Databases, 1(2),* pp. 141–259.

Biography

Milorad P. Stević is a Database Architect with 14 years of experience (Progress versions 8,9; Oracle version 10g, MSSQL Server versions 2005, 2008, 2012), He is also a .NET Software Architect with a 8 years of experience in development for the Windows platform, Web, Web Services, Client/Server and n-tiered distributed applications, broad experience in all parts of project life cycle including the analysis, design, development, implementation testing, debugging/profiling and support. He is a full-time faculty member at The Higher Education Technical School of Professional Studies, Novi Sad, Serbia.