빅데이터 관리를 위한 데이터베이스 선정분석

박승범[*] · 이상원[**] · 안현섭[***] · 정인환[****]
[*]한국정보화진흥원 경영기획부
[**]원광대학교 정보전자상거래학부(융복합창의연구소)
[***]브라운슈바이크공과대학 경영정보학과
[****]한성대학교 컴퓨터공학과

Selection Analysis of Databases to Manage Big Data

Sungbum Park[*] · Sangwon Lee[**] · Hyunsup Ahn[***] · In-Hwan Jung[****]
[*]Department of Management Planning, National Information Society Agency
[**]D. of Information & Electronic Commerce (I. of Convergence & Creativity), Wonkwang University
[***]Department of Wirtschaftsinformatik, Technische Universität Braunschweig
[****]Department of Computer Engineering, Hansung University
E-mail: parksb@nia.or.kr, sangwonlee@wku.ac.kr, hs.ahn@tu-bs.de, ihjung@hansung.ac.kr

요 약

빅데이터를 관리하기 위해서 사용하는 NoSQL은 두 가지 중요한 요소를 가지고 있다. 애플리케이션 프로그래머의 생산성을 높이는 것과 데이터 접근 성능을 높이는 것이다. 그러나, 많은 기업 현장에서는 이러한 바람직한 계획들이 철저히 고려되고 있지 않다. 빅데이터의 효과적이고 효율적인 관리와 분석을 위해서는 NoSQL 기술을 사용할지 말아야할지를 결정하기 전에, 애플리케이션 프로그래머의 생산성과 성능에 대한 기대를 충족시키기 위한 테스트를 필수적으로 수행해야한다. 본 논문에서는 프로그래머 생산성, 데이터 접근 성능, 위험 관리 등에 대해 연구하고자 한다.

ABSTRACT

There are two major factors to use NoSQL in order to manage Big Data; to increase productivity of an application programmer and to increase data access performance. But, in many business fields, this hopeful plan lacks careful consideration. For efficient and effective management and analysis of Big Data, it is necessary to perform a test with the expectation for productivity and performance of the application programmer before deciding whether NoSQL technique is used or not. In this paper, we research on programmer productivity, data access performance, risk distribution, and so forth..

키워드
Big Data, Selection Analysis, Databases, Program Productivity

I. Introduction

In many business fields, this hopeful plan lacks careful consideration. For efficient and effective management and analysis of Big Data, it is necessary to perform a test with the expectation for productivity and performance of the application programmer before deciding whether NoSQL technique is used or not. In this paper, we research on programmer productivity, data access performance, using Relational Databases and risk management (Figure 1).

II. Considerations in Selecting Databases for Big Data

The first consideration in selecting databases for Big Data is programmer productivity. It is generally accepted that a definition of programmer productivity needs to be established and agreed upon. Appropriate metrics need to be established. Productivity needs to be viewed over the lifetime of code. Programming productivity means a variety of software development issues and methodologies affecting the quantity and quality of code produced by an individual or team. A key topic in productivity discussions is amount of code that can be created or maintained per programmer. The amount of code is often measured in source lines of code per day. The second topic is to detect and avoid errors. The detection is performed by techniques like Agile Software Development, six sigma management, zero defects coding, and Total Quality Management. The last topic is software cost estimation. Of course, the cost is a direct consequence of productivity. The importance of programming productivity has improved along with other industry factors, such as the relative costs of manpower versus. In the field of unstructured or semi-structured data, NOSQL systems are more efficient and effective in managing and handling Big Data.

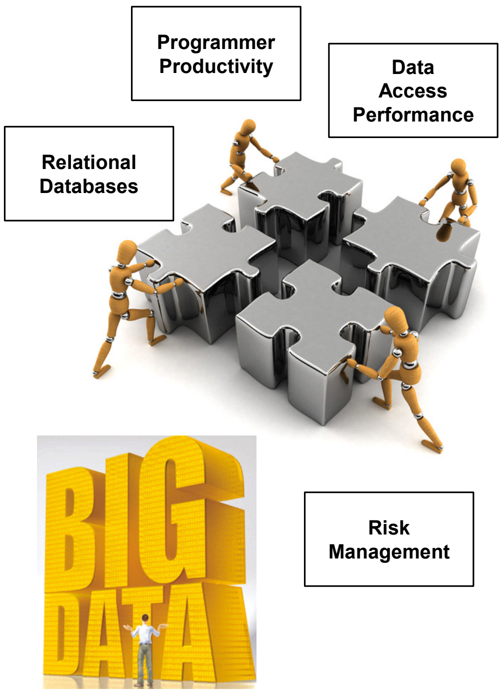

Figure 1. Four Decisions for Big Data

The second consideration in selecting databases for Big Data is data access performance. Data access typically refers to software and activities. The activities are storing, defining, retrieving, or acting on data housed in a database or other repository. Sequential access and random access are two types of data access. Since data access can help distinguish the abilities of administrators and users, data access is simply the authorization you have to access different data files. With including each different database, file system, and many of these repositories stored their content in different and incompatible formats, every repository requires different methods and languages. It is very important to perform a test with valuable scenarios for NoSQL databases.

The third consideration in selecting databases for Big Data is to use Relational Databases with NoSQL. Even though NoSQL is a best solution for Big Data, it is not a panacea for Big Data. It is meaningful to optionally use Relational Databases with NoSQL.

The last consideration in selecting databases for Big Data is risk management. Risk management is a process of identification, analysis and either acceptance or mitigation of uncertainty in investment decision-making when using databases for Big Data. Whenever a user or data operator analyzes and attempts to quantify the potential for losses in managing database, risk management would occur anytime. Especially, risk management would do so when taking appropriate actions or inaction ones, given their investment objectives and risk tolerance. Inadequate risk management would certainly cause severe consequences for companies as well as individuals in handling and analyzing Big Data. Traditional approaches such as Data Mapper and Repository (Fowler PoEAA) would be helpful in encapsulating the process of selecting databases,

III. Conclusions

In sum, there are two major factors to use NoSQL in order to manage Big Data; to increase productivity of an application programmer and to increase data access performance. For effective and efficient management of Big Data, it is necessary to perform a test for programmer productivity and data access performance before choosing databases. If not strategic, most of applications may use Relational Databases techniques.

References

[1] A B M Moniruzzaman and Syed Akhter Hossain, "NoSQL Database: New Era of Databases for Big data Analytics - Classification and Characteristics and Comparison," International Journal of Database Theory and Application, Vol. 6, No. 4, 2013.

[2] Burton H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," Communications of the ACM, Vol. 13, No. 1, pp. 422-426, 1970.

[3] Christof Strauch, NoSQL Databases, Stuttgart Media University Press, 2012.

[4] Edgar F. Codd, "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, Vol. 13, No. 6, pp. 377-387, 1970.

[5] John Ousterhout, Parag Agrawal, David Erickson, Christos Kozyrakis , Jacob Leverich, David Mazières, Subhasish Mitra, Aravind Narayanan, Guru Parulkar, Mendel Rosenblum, Stephen M. Rumble, Eric Stratmann and Ryan Stutsman, "The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM," SIGOPS Operating Systems Review, Vol. 43, No. 1, pp. 92-105, 2010.

[6] Joshua Bloch, Effective Java - Programming Language Guide, Addison-Wesley Longman, 2001.

[7] Marcos K. Aguilera, Wojciech Golab and Mehul A. Shah "A Practical Scalable Distributed B-Tree," Proceedings of the VLDB Endowment, Vol. 2008, No. 1, pp. 598-609, 2008.

[8] Patrick O' Neil, Edward Cheng, Dieter Gawlick and Elizabeth O' Neil, "The Logstructured Merge-Tree," Acta Informatica, Vol. 33, No. 4, pp. 351-385, 1996.

[9] Pramod Sadalage and Martin Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Addison-Wesley, 2012.

[10] Stavros Harizopoulos, Daniel J. Abadi, Samuel Madden and Michael Stonebraker, "OLTP through the Looking Glass, and What We Found Fhere," ACM SIGMOD International Conference on Management of Data, Vol. 2008, No.1, pp. 981-992, 2008.