Dependencies of File System Cluster Size in Correlation with Database Management System

Tin Kramberger, Davor Cafuta, Ivica Dodig
Polytechnic of Zagreb
Department of informatics and computing
Vrbik 8, 10000 Zagreb, Croatia
{tin.kramberger, davor.cafuta, ivica.dodig}@tvz.hr

Abstract. *Databases represent core in many software solutions. Features required by software applications for database management systems include fast response, reliability and fault tolerance. It could be suspected that the file system as primary layer of a database management system can affect database management system performance on every computer system, especially on computer systems with an inferior hardware. Database performance does not depend primarily on the type of file system, but also on preferences like cluster size, implemented cache strategy and applied fault tolerance. It can be assumed that these dependences are proportional on different types of file systems. Several measurements are presented showing correlation of the cluster size on database performance in different file system type surroundings using PostgreSQL. With results obtained by measurements, one should be able to determine which of the measured file systems in correlation with cluster size should be implemented in simple computing solutions like embedded systems due to their inferior computing power.*

Keywords. Database, File system, performance, cluster size

1 Introduction

File system is a mechanism which describes how files are named and where they are placed logically for storage and retrieval. The DOS, Windows, OS/2, Macintosh, and UNIX-based operating systems all have file systems in which files are placed somewhere in a hierarchical tree structure. A file is placed in a directory or subdirectory at the desired place in the tree structure. The file system enables organization of the data and some features of the operating system like fast search, security of the files and other.

In a computer system information is considered as the most valued data. The data placement is often done as a last phase in every process. Due to every phase of the process specific duration, performance upgrade of data placement can provide overall system performance boost. In large computer systems with significant computing power and large memory consumption this upgrade can make a small notice. When the computer system is limited to lower computing power, slower hard disk drives and smaller amount of RAM available, upgrades made with only changing the file system and its cluster size can make significant performance boost from user perspective.

Main reasons for using large database systems are data protection in event of failure and I/O bottlenecks. Worst case scenario without adequate data protection could be reparation of data using database backups. Backups should be created in short time periods to full proof system in case of database management system error. Some other protection methods include RAID mirror systems and load balancing which can be expensive, hard to configure and difficult to maintain. Using RAID is the best option due to performance increase, but in case of hard disk failure it can take significant amount of time to recover from the failure. [1]

Fixing issues without investing in new hardware requires knowledge in areas specific to database management system design and file system features. This process can be time consuming, costly and risky due to potential of data loss, and unplanned server downtimes. Choosing a different file system for database storage can decrease I/O bottleneck, reduce possibility of write error and generally resolve the problem without the need for investment into new hardware and even introduce new features which can results in increased reliability of the database server [2].

Linux operating system is commonly used as a database server due to its scalability, code openness and ease of administration. Today, a Linux kernel supports large variety of file systems. Kernels are constantly upgraded with new features including support for newly developed file systems and file system features. A newer file system sometimes does not boost performance in I/O operation time due to its improvement on reliability which requires additional operations on disk [3].

To simplify the measurements, standard Linux operating system with the cache option disabled was used. The measurements were made using different cluster sizes with different file systems to determine if a file system choice can make a difference. Additionally, sets of tests were performed to determine correlation of cluster size on the file system performance. [4]

Measurements were taken by analysing the database management system speed in reading, writing and deleting data on the database.

In this article the focus is to show a file system cluster size correlation to the database performance using system with a lack of computing power and without predetermined operating system. Such system could be implemented in future on embedded system. Chapter 2 describes the file systems storage. Chapter 3 correlates the file system with the database management system. Testing methodology is presented in chapter 4. Results are shown in chapter 5. Chapter 6 summarizes this paper.

2 File systems

The file system is the facility of the operating system that organizes files. For example, on DOS and older Windows PCs, there is a file allocation table (FAT) that consists of a linked list of clusters where each cluster consists of a fixed number of sectors, varying with the overall size of the disk. When the operating system has to access a file, it can go through the table and find the clusters belonging to that file, read the data and send it to the requesting application. Modern file systems further organize files into groups called folders or directories, which can be nested several layers deep. Such hierarchical file system makes it easier for users to organize the dozens of applications and thousands of files found on today's PCs. For example, a folder called "White-paper" might have a subfolder for each chapter, which in turn contains folders for the text and illustrations relating to that chapter. Besides storing and retrieving files, the modern file system sets characteristics or attributes for each file. Typical attributes include write (the file can be changed), read (the file can be accessed but not changed), and archive (which determines whether the file needs to be included in the next backup). In multi-user operating systems such as UNIX there are also attributes that indicate ownership (that is, who has certain rights with regard to the file). Thus a file may be executable (run as a program) by anyone, but writeable (changeable) only by someone who has a "superuser" status. The current generation of file systems for PCs includes additional features that promote efficiency and particularly data integrity. [5][6]

Versions of Windows starting with NT, 2000, and XP come standard with NTFS, the "New Technology File System," which includes journaling, or keeping records of all transactions affecting the system (such as deleting or adding a file). In the event of a mishap such as a power failure, the transactions can be restored from the journal, ensuring that the file system reflects the actual current status of all files. NTFS also uses metadata that describes each file or directory. Database principles can thus be applied to organizing and retrieving files at a higher level.

File systems are a necessity in modern operating systems. Operating system improvement implicates file system development. Features like fast indexed search, better security options are a requirement in modern operating system implicating a development of new version or completely new file system. As an example, FAT32 file system does not enable security tab option on file property. Windows operating systems require usage of NTFS file system to enable security tab on file properties. This is due to insufficient storage capacity in File Attribute Table of FAT32 file system.

There are several aspects of file system which can distinguish their usability. One of the most important issues is space management. Other features include file naming, directories limitation, metadata, utilities, security permissions and maintaining integrity. Some of these features are important and can provide large boosts in computer systems with a shortage of computing power and memory resources. This research is intended to provide the data for a file system implementation usage on such systems.

Space management defines how a physical unit is allocated on the disk by the file system. Usually a file system uses multiple physical units on a device. This type of allocation results in unused space for a large percentage of files. For example, saving 1 byte in 512 bytes allocation causes 511 bytes unused space. Larger allocation units cause increase of unused space in files and as a consequence poor space management. Space management is also responsible for organizing data on a physical disk. Some file systems permit or require specifying an initial space allocation and subsequent incremental allocations as the file grows. As files are deleted, the space they were allocating is eventually considered available for use by other files. This creates irregular used and unused areas with various sizes which is called free space fragmentation. When a file is created and there is not an area of continuous space available for its initial allocation, the space must be assigned in fragments. When a file is modified such that it becomes larger, it could exceed the space initially allocated to it. Due to exceeding of space, another allocation must be assigned in a different position and because of that the file becomes fragmented.

Reading and writing data is a complex procedure. Data stored on a hard disk passes through disk cache and kernel cache to become available for application. Cache is computer memory with very short access time used for storage of frequently or recently used instructions or data. Also it is increasing performance of the file system and physical disk. There is a significant disadvantage of the cache. In case of failure or power loss system cache memory can be lost. Incomplete operations may result in loss of operation data or other data already written on the drive.

One significant responsibility of a file system is to ensure that, regardless of actions by programs accessing the data, the structure remains consistent. It includes actions taken if a program modifying data terminates abnormally or neglects to inform the file system that it had completed its activities. This includes updating the metadata, the directory entry and handling any data that was buffered, but not yet updated on the physical storage device.

Other failures which the file system must deal with include device failures or loss of connection to remote systems. In the event of an operating system failure or power failure, special routines in the file system must be invoked similar to failure of an individual program. The file system must also be able to correct damaged structures. These damages might occur as a result of an operating system failure for which the operating system was unable to notify the file system. The file system must also record events to allow analysis of system issues as well as problems with specific files or directories. Most modern file systems protect file system integrity for possible power failures or crashes via journaling, which groups operations into transactions that commit automatically.

There are numerous file systems on Linux operating system: EXT2, EXT3, EXT4, FAT32 and NTFS. The file systems selected, were targeted according to availability on various operating systems and Linux kernels.

EXT2 (Extended File System) is the oldest file system on Linux released in January 1993 as a successor of the EXT file system. EXT2 has been Linux file system for many years and is still used as RAM disk file system and when non-journaling file system is required. EXT2 has proven to be stable and quite lightweight in terms of overhead. The downside is lack of journal. [7][8].

EXT3 is the successor of EXT2 implementing journal option. There are three different mount modes: journal, ordered and write back. Journal mode logs all file system data and metadata changes. It is the slowest of the three EXT3 modes but also the most secure mode to enable reconstruction of disk in case of failure. Ordered mode logs only changes in file system metadata but only writes these changes if file data write into disk is confirmed. Write back mode is the fastest EXT3 mode as it only writes metadata changes into journal before data disk file write is confirmed. In case of failure the file can be reconstructed but this mode does not guarantee that data in the file is correct. Ordered mode is the default file system mode. EXT2 mode is compatible with EXT3 mode, meaning that you can mount an EXT3 file system as an EXT2 because the layout on disk is exactly the same. This enables the existing file system repair tool and tuning of the system. EXT2 partition can be easily switched to EXT3 partition without copying files. EXT3 is based on binary trees to enable fast indexation of files. [9][10][11]

EXT4 is the newest of all mentioned file systems developed in October, 2006 as a compatible improvement of EXT3, featuring support for larger volumes and support for extend. Extend is a continuous area of storage reserved for a file. When starting to write a file, a whole extend is allocated at start, disabling file fragmentation in a file system. These improve speed of read operation on disk. EXT4 has the same modes as EXT3 depending on usage of journal: journal, ordered or write-back. [12]

File Allocation Table (FAT) is the name of computer file system architecture and a family of industry standard file systems utilizing it. The FAT file system is technically relatively simple yet robust. The name of the file system originates from the file system's prominent usage of an index table, the FAT, statically allocated at the time of formatting. The table contains entries for each cluster, a contiguous area of disk storage. Each entry contains either the number of the next cluster in the file, or a marker indicating end of file, unused disk space, or special reserved areas of the disk. The root file directory of the disk contains the number of the first cluster; the operating system can then traverse the FAT table, looking up the cluster number of each successive part of the disk file as a cluster chain until the end of the file is reached. As disk drives have evolved, the maximum number of clusters has significantly increased, and so the number of bits used to identify each cluster has grown. The successive major versions of the FAT format are named after the number of table element bits: 12 (FAT12), 16 (FAT16), and 32 (FAT32). Each of these variants is still in use. The FAT standard has also been expanded in other ways while generally preserving backward compatibility with existing software. Reliability is based on relocation of the root folder and usage of the backup copy of the file allocation table instead of the default copy. In addition, the boot record on FAT32 drives is expanded to include a backup copy of critical data structures [13].

NTFS supersedes the FAT file system as the preferred file system for Microsoft’s Windows operating systems. NTFS has several improvements over FAT such as improved support for metadata and the use of advanced data structures to improve performance, reliability, and disk space utilization, plus additional extensions such as security access control lists (ACL) and file system journaling. The NTFS on-disk format has five released versions. The latest version 3.1 was released in autumn 2001.

The structure of an NTFS volume is considerably different that of the FAT32 file system. NTFS uses relational database called the master file table (MFT) to manage contents of a volume. The MFT serves much the same purpose in the NTFS file system that FAT serves in the FAT file systems. The MFT stores a record for each file and directory, including the MFT itself. Each entry includes the name, security descriptor, and other attributes. The MFT is an array of data with rows representing file records, and columns representing attribute fields for each record, as shown in figure 1[14].

NTFS is a robust, self-healing file system that offers several customizable features that affect how well NTFS performs in a given environment. Some of these parameters are global and others are specific to individual NTFS volumes. By examining specific storage needs and then tailoring NTFS volumes accordingly, some significant increases in systems disk performance can be achieved [15][16].

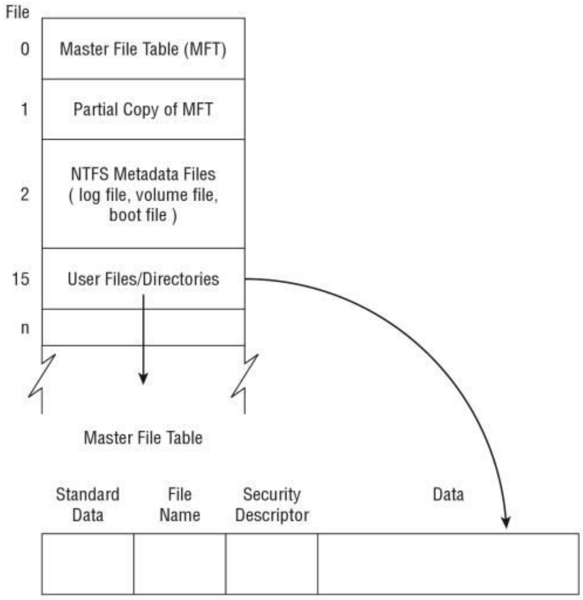

Figure 1. MFT structure

Described file systems are available for implementation in numerous systems. As a file system and its cluster size options take significant part in system performance, a study was performed to measure their speed.

Larger cluster size increases unused space on disk as described above. This can be a problem in some systems where memory consumption can be a significant issue. In our case due to small number of files that need to be stored this is not the issue. The goal of this paper is to detect if difference in cluster size can make significant changes in performance on different file systems. Due to lack of computational power of some used systems on our polytechnic, small differences can make large imprint on the performance of database management systems [17].

3 Database

A database is a well-organized collection of data, which is related in a meaningful way, which can be accessed in different logical orders. Database systems are systems in which the interpretation and storage of information are of primary importance. The database should contain all the data needed by the organization. As a result of that necessity, a huge volume of data, the need for long-term storage of the data, and access of the data by a large number of users has to be provided by the database management system. [18] Due to these requirements the file system should be prompt, durable, reliable and fault-tolerant.

There are small differences in open source database server features like automatic conversion of code pages, object-relational extension, XML support and user defined data types, but both servers are compatible with SQL'92 standard and ACID standard. ACID standard describes critical features of database engines to protect data integrity: Atomic, Consistent, Isolated and Durable. ACID essentially means that when a transaction is performed within a database, either the whole transaction is successful and the information is written to the database or nothing is written.

PostgreSQL uses only one storage mechanism named PostgreSQL storage system. To further increase performance level, oversized attributes in tables are stored out-of-line storage in separate file. This technique is called TOAST (The Oversized-Attribute Storage Technique) [19].

There are certainly situations where other database solutions will perform better. PostgreSQL is missing features needed to perform well on some of the more difficult queries. It's correspondingly less suitable for running large data warehouse applications than many of the commercial databases. If queries like some of the very heavy ones are needed, other databases such as Oracle, DB2, and SQL Server have better performance. There are also several PostgreSQL-derived databases that include features, making them more appropriate for data warehouses and similar larger systems. But unlike them PostgreSQL has smaller memory and power consumption [3].

Due to its benefits, PostgreSQL open source database management system was used to measure performances of most common file systems using different cluster sizes.

4 Testing methodology

Measurements were run on a Dual CPU Intel U7300@1.3GHz, 4GB RAM@800MHz, and a Seagate ST332060AS 320GB 7200RPM SATA2 hard disk drive. Tests used Linux 3.5.0-17 kernel with the Ubuntu v12.10 distribution. Measurements were made on exactly the same partition using 0.57% of the hard disk drive. Partition was formatted to tested file system. Every measurement was performed according to algorithm shown in figure 2.

One hundred measurements were taken for every file system using one of three actions: insert (write), select (read) or delete. Between measurements pause was made precisely 10 minutes to allow the operating system to perform other activities. Every measure consisted of 1000 iterations.

Iteration was one database transaction. Database transaction was timed. One database transaction commits 1000 database operations. Sum of transactions time was taken for 100 iterations (100000 rows), 500 iterations (500000 rows) and 1000 iterations (1000000 rows).

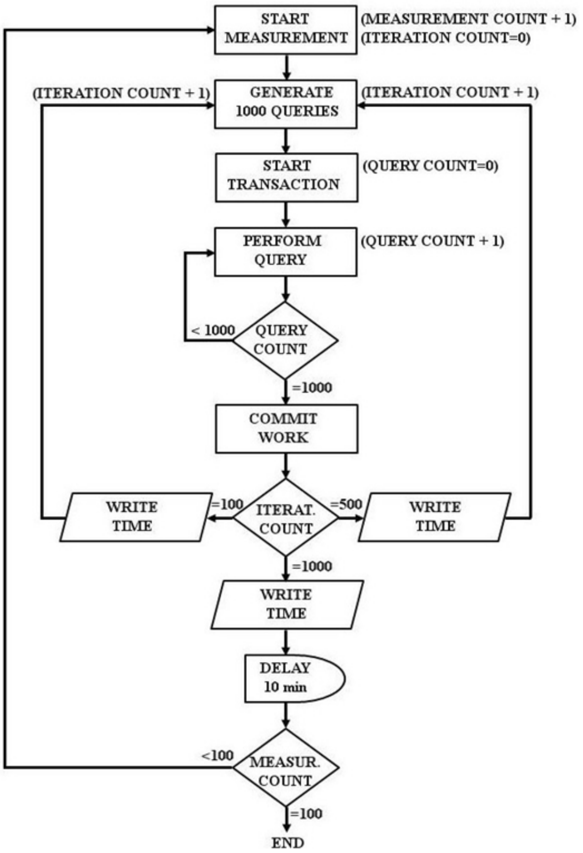

Figure 2. Testing metodology algorithm

Average size of all measurements was calculated with isolated maximal deviations. To measure file system performance data was sent to database management system via queries. All queries were randomized with unstructured flat objects of one kind in bulk mode. Inserts were done on one table with the same structure that had testing columns: “ID” type of INT which is also primary key using BTREE, “testNumber” type of INT and “testString” type of varchar(100). Index choice was made on the assumption that the BTREE is the most preferred index type due to its data corruption protection unlike hash index. Commits were made after each 1000 randomized queries. Queries were executing write, read and delete action through data manipulation statements. Write action executes INSERT query according to SQL standard. As index on ID column needs to be rebuilt, write action should be the slowest action on database. Read action executes SELECT query according to SQL standard. Read action should be the fastest query. Delete action represent DELETE query. Database management system used to perform measurements was PostgreSQL 9.1.7. [20]

5 Results

Result sets were obtained using write, read and delete actions on Ubuntu 12.10 operating system using PostgreSQL 9.1.7. database management system. File systems that were measured include: NTFS, FAT32, EXT2, EXT3 and EXT4. Due to file system limits for cluster size option, measurements were performed on file system cluster sizes shown in Table 1.

Table 1. File system cluster size options

Cluster size	**File system**				
	EXT2	EXT3	EXT4	FAT32	NTFS
512B	No	No	No	Yes	Yes
1024B	Yes	Yes	Yes	Yes	Yes
2048B	Yes	Yes	Yes	Yes	Yes
4096B	Yes	Yes	Yes	Yes	Yes
8K	No	No	No	Yes	Yes
16KB	No	No	No	Yes	Yes
32KB	No	No	No	No	Yes
64KB	No	No	No	No	Yes

Due to detection of large differences in performance during read and delete actions, additional graphs are shown. The graphs on figures 3 – 7 are showing additional data on execution time according to number of queries on different file systems with different cluster sizes.

Figure 3 shows reading performance on EXT2 file system with cluster sizes 1024B, 2048B and 4096B. It’s evident that the fastest results are achieved with a cluster size of 2048B, while the results for 1024B and 4096B are practically the same. The same measurements were repeated on EXT3, EXT4 file systems, as it’s shown on figures 4 and 5.

On figure 4 it’s evident that the EXT3 file system is the fastest using 1024B cluster size. The performance of reading action on EXT3 file system using 1024B cluster size is almost the same as it’s using a 2048B cluster size. Results are similar to reading performance on EXT2 file system.

Measurements of EXT4 file system which are presented on figure 5 are showing that the cluster size of 4096B is the best cluster size for reading performance using a PostgreSQL database.

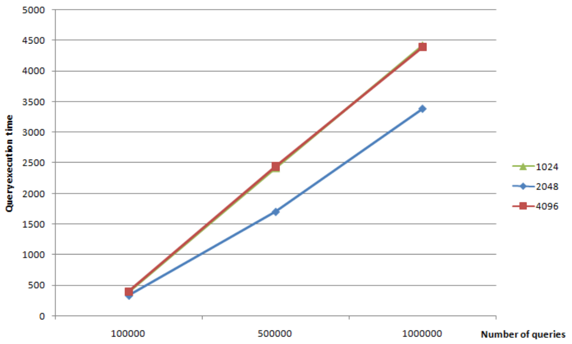

Figure 3. Read action (execution time according to number of queries on EXT2 file system)

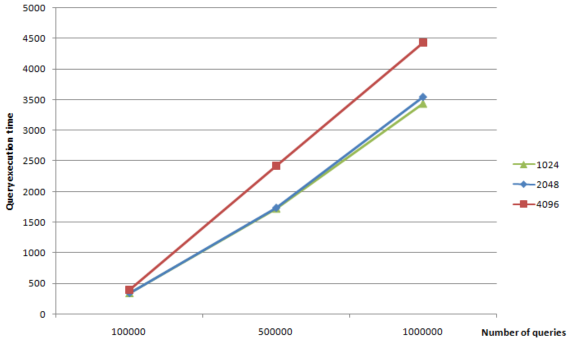

Figure 4. Read action (execution time according to number of queries on EXT3 file system)

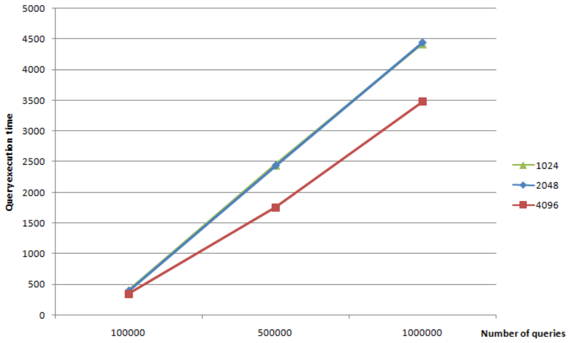

Figure 5. Read action (execution time according to number of queries on EXT4 file system)

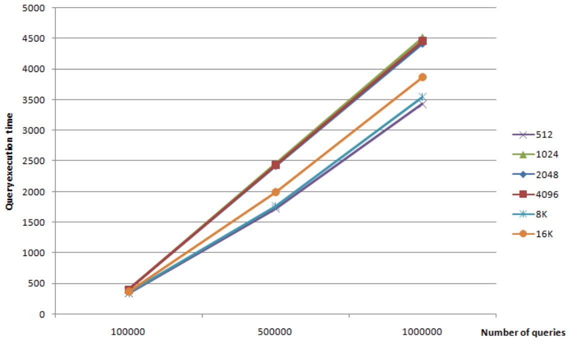

Figure 6. Read action (execution time according to number of queries on FAT32 file system)

Due to different features of the measured file systems, on FAT32 and NTFS the cluster sizes differ. Consequently, FAT32 has clusters sizes of 512B, 1024B, 2048B, 4096B, 8kB and 16kB. NTFS has the same cluster sizes as FAT32, but adds cluster sizes of 32kB and 64kB.

On figure 6 the performance of FAT32 using read action is shown. It is evident that the FAT32 file system is the fastest using a 512B cluster size. The performance of reading action on FAT32 file system using a 512B cluster size is almost the same as it's using an 8kB cluster size.

NTFS file system read action performance is presented on figure 7 and the results suggest that it is the fastest using a cluster size of 4096B. The performance of reading action on the NTFS file system using a 512B cluster size is almost the same as it's using a 4096B cluster size.

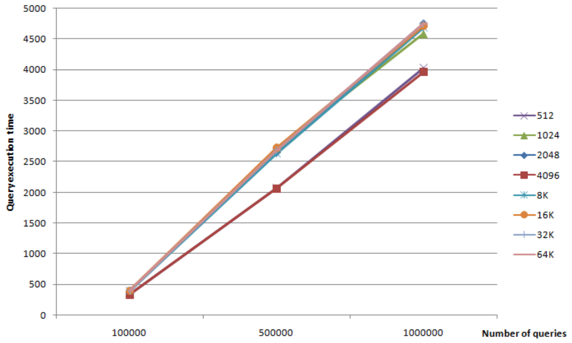

Figure 7. Read action (execution time according to number of queries on NTFS file system)

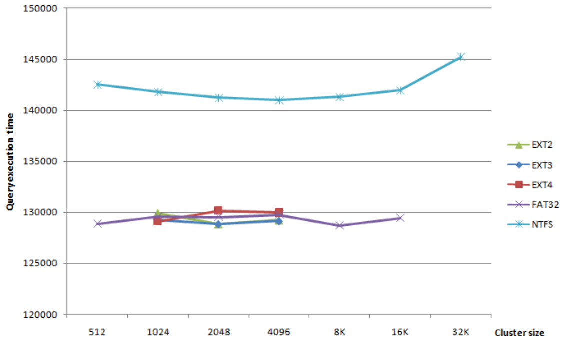

Figure 8. Write action (execution time according to diferent file systems with diferent cluster sizes)

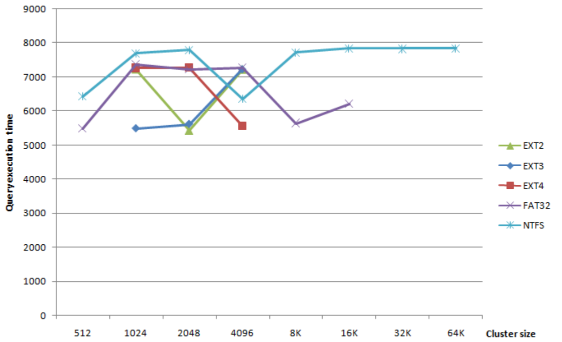

Figure 9. Read action (execution time according to diferent file systems with diferent cluster sizes)

On Figures 8-10 results of actions write, read and delete are shown using calculated average sizes of all measures with isolated maximal deviations.

Figure 8 is showing write action on all measured file systems using different cluster size options as was presented in table 1. It can be concluded that all file systems have similar performances on all cluster sizes except NTFS which is slower due to the MFT database. The fastest file system measured was FAT32 using 8kB cluster size. Similar performance measurements were obtained for EXT2 and EXT3 using a 2048B cluster size. EXT2 and EXT3 using a 2048B cluster size are under 0.1% slower than FAT32 using an 8kB cluster.

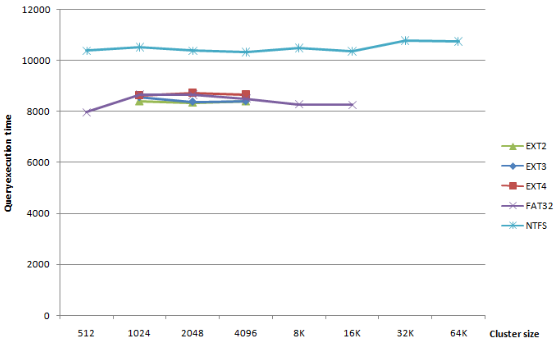

Figure 10. Delete action (execution time according to diferent file systems with diferent cluster sizes)

Measurements of read action presented in figure 9 show that the EXT2 file system using a 2048B cluster size is the fastest file system measured. The performance of reading action on FAT32 using 512b and 8kB, EXT3 using 1024B and EXT4 using 4096B have similar performances. Due to the MFT database, the NTFS file system is generally slower than other tested file systems.

Figure 10 shows that delete action measurements have similar results, except NTFS which is again slower than other tested file systems due to reasons previously mentioned. The fastest file system measured was FAT32 using 512B.

6 Summary

Databases are widely spread in many application even in very specific surroundings ex. embedded system devices. Different surroundings require different approaches in hardware. In some cases file system performance could make differences in device viewed from user perspective. The main goal of this paper is establishing major differences between common file system storage cluster size options for database management systems.

In large and expensive hardware operating conditions this difference can be small, but in systems with inferior hardware the difference can be significant. This papers goal was to try to detect differences of various file systems and clustering size options. Depending on the measurements it can be concluded which file system is the best for implementation to maximize performance of the system.

The EXT2 file system shows the best performance on medium cluster sizes (2048B). Following is FAT32 using cluster sizes 512B or 8kB. FAT32 cluster sizes 2048B and 4096B show slower performance than edge cluster sizes. Other file systems are slower in most of the cases. It can be concluded that the choice of file system is EXT2 or FAT32, depending on simplicity of the implementation on the system or choice of the installed operating system. The NTFS file system is the slowest due to MFT database described in chapter V. It should be used only in case where security is primary concern which is lacking in other file systems.

In the future, it would be interesting to measure transaction performance using different database cluster sizes in correlation with different file system cluster sizes on most used open source database management systems.

References

[1] Derek Vadala, “Managing RAID on LINUX”, O’Reilley, pp. 11-32

[2] J. B. Chen , Y. Endo , K. Chan , D. Mazieres , A. Dias , M. Seltzer , M. D. Smith, The measured performance of personal computer operating systems, Proceedings of the fifteenth ACM symposium on Operating systems principles, p.299-313, December 03-06, 1995, Copper Mountain, Colorado, United States

[3] Gregory Smith, “PostgreSQL 9.0 High Performance”, Packt Publishing 2010, pp.69-97

[4] D.P. Bovet and M. Cesati. Understanding the Linux Kernel (2.nd edition) O’Reilley, 2003

[5] Arnold Robbins, Unix in a nutshell, (Third edition) O’Reilley, 1999

[6] L. Budin, M. Golub, D. Jakobović, L. Jelenković Operacijski sustavi, Element, 2010, pp.241-280

[7] Rémy Card, Theodore Ts'o, and Stephen Tweedie, “Design and Implementation of the Second Extended Filesystem,” Proceedings of the First Dutch International Symposium on Linux, 1994.

[8] D. Gibson, Measuring Parameters of the EXT2 File System, University of Wisconsin-Madison Department of Electrical and Computer Engineering

[9] Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Analysis and Evolution of Journaling File Systems, USENIX Annual Technical Conference (USENIX-2005), Anaheim, CA, April, 2005.

[10] R. Card, T. Ts'o, and S. Tweedie. Design and Implementation of the Second Extended Filesystem. In Proceedings of the FirstDutch International Symposium on Linux, 1994.

[11] C. Frost, M. Mammarella, E. Kohler, A. Rayes, S. Howsepian, A. Matsuoka, L. Zhang, Generalized File System Dependencies, UCLA, 2011.

[12] M. T. Jones, Anatomy of ext4 - Get to know the fourth extended file system, IBM-Developer works, URL: http://www.ibm.com/developerworks/linux/library/l-anatomy-ext4/, 2010.

[13] Silberschatz, Galvin, Gagne. Operating System Concepts, Sixth Edition. John Wiley & Sons, Inc.

[14] Jeffrey R. Shapiro, Windows Server 2008 Bible, Wiley Publishing Inc., 2008, pp. 447-486

[15] Claybrook, Billy G. File Management Techniques. John Wiley & Sons,

[16] Digit-life.com, "NTFS file system" URL: http://www.digit-life.com/articles/ntfs/, 2003.

[17] Andrew S. Tanenbaum, Modern operating systems (Third edition), Pearson, 2009, pp. 290-310

[18] S. Sumathi, S Esakkirajan, Fundamentals of Relational Database Management Systems, Springer-Verlag Berlin Heidelberg 2007, pp. 2-30

[19] PostgreSQL documentation, TOAST, URL: http://www.postgresql.org/docs/8.3/static/storage-toast.html, 2011.

[20] Gregory Smith, "PostgreSQL 9.0 High Performance", Packt Publishing 2010, pp. 209-231