Simulation Modelling Practice and Theory 42 (2014) 129–146

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

The performance improvements of highly-concurrent grid-based server

Tinghuai Ma [a,*], Chenghui Wu [b], Wei Tian [b], Wenhai Shen [c]

[a] Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing, Jiangsu Province 210044, PR China
[b] School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing 210044, PR China
[c] National Meteorological Information Center, Beijing 100080, PR China

ARTICLE INFO

Article history:
Received 21 November 2012
Received in revised form 9 September 2013
Accepted 4 December 2013
Available online 24 January 2014

Keywords:
Data grid
I/O schedule
Zero copy
Event-driven

ABSTRACT

Server performance is one of the critical activities in the data grid environment. A large number of applications require access to huge volumes of data from grid servers. In this case, efficient, scalable and robust grid server which can deal with large file transfer concurrent is needed. In this paper, we analyze the bottleneck of our grid servers and introduce user-space I/O scheduling, zero copy and event-driven architecture in our grid server to improve the servers' performance. The user-space I/O scheduling can save almost 50% I/O time in a huge number of small files transfer. Grid servers can elimination CPU consumptions between kernel and user space by zero copy and cut 63% times for context switches. Event-driven architecture will save 30% CPU usage to reach the best performance by thread-driven architecture. Optimization method combination of these three above are used in our grid servers, the full-load throughput of our system is 30% more than traditional solutions and only 60% CPU consumed compared with traditional solutions.

1. Introduction

In recent years, data grid has been widely used in the scientific community. The performance of grid server has become more and more important. Thousands of concurrent requests need to be handled by grid server at the same time. How to keep the delay within the range of user's acceptance is a challenge. As time going, the users' request data become very larger, how to keep the hard disk working in an efficient way is also very important.

Today, there are many resources such as disks, memory, and CPU in modern computers which based on Von Neumann Organization. Data flows as a circle: data read from disk or received from network card, and then copied to memory, processed by CPU, at last sent to network card or write to the disk again. It seems simple when the number of data flows is small. As the number of data flows grows, it will become very complex, because different stages of the data flows intertwined at the same time. So optimization and scheduling of grid server which has a high concurrent data flows are needed.

We usually from two aspects to improve the performance of grid servers showed in Fig. 1: System Efficiency and High Concurrent Requests Handling Architecture (HCRHA). In System Efficiency, Disk I/O management, memory and CPU usage are important, they are all in a pipeline. If we want to get the best performance, none of them should be the bottleneck of the system. Because of the physical differences between them, we must schedule them to do different works at the same time. In the target of higher throughput in the whole system, additional target is higher throughput of disk, lower memory and CPU usage to perform an efficient and stable system. And in HCRHA, three key points are contained: stability,

* Corresponding author. Tel.: +86 25 5873 1267.
E-mail addresses: thma@nuist.edu.cn (T. Ma), chwu1203@163.com (C. Wu), tw@nuist.edu.cn (W. Tian), shenwh@cma.gov.cn (W. Shen).

1569-190X/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2013.12.008

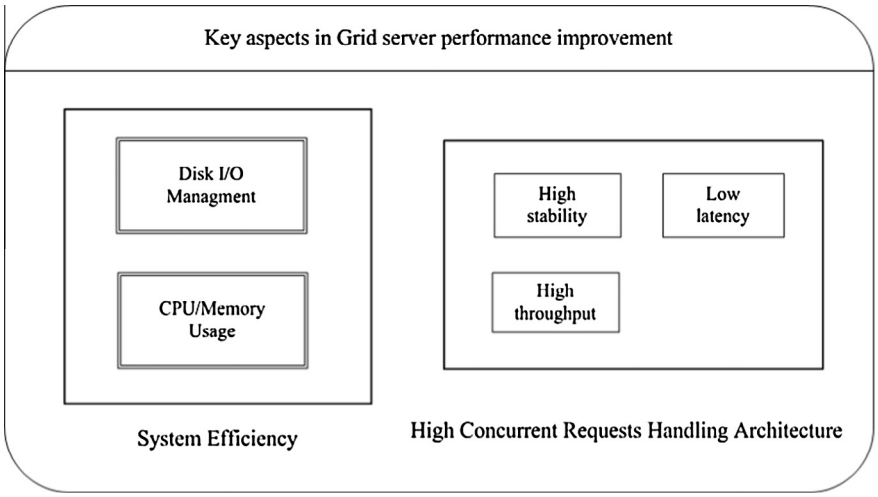

Fig. 1. Key aspects in grid server performance improvement.

throughput, and latency. The architecture is presented as the way how to schedule grid servers' resources to process the requests in the whole system. Grid server must achieve many targets such as clients' request must be responded within a strict time and get the stability of the processing speed and high data transmission speed.

In grid server, with the widening performance gap between memory and disk, the disk is become the bottleneck of most grid servers. How to maximize the disk throughput is becoming an important topic. Some specific file system such as parallelism file system [1] or flash based file system [2] is created to enhance the disk performance, but it is not general and cannot be widely used. Today, many tasks are running in the Linux kernel for different purposes. For example, video application has a strict requirement for latency and bandwidth, a web page must be returned within a short time. While other tasks like batch jobs do not have any special requirements, but all of them should share disk resources fairly. So we need to use more RAM, CPU cycles and a better scheduler to reach the less latency and higher throughput of disk I/O. There are five components that influence the performance of disk I/O:

I. Applications: running in the user space, initiate I/O operations through system call like read/write. Different I/O characteristics of applications may cause a different performance even using the same system. This is important, and user-space I/O scheduler was introduced for optimize it. Application can sort, merge requests before send them to the kernel, specially for accessing a large number of small files.
II. File system: translating applications read/write requests into one or more block I/O requests. Block allocation strategies are implemented in this level which determine the block size, physical placement and potential fragmentation on the disk. Here, we have much more choices like EXT3, EXT4 and NTFS.
III. Page cache: caching disk blocks which likely to be used again. Many articles used to implement new cache strategies to achieve a better performance. After DIRECT I/O was introduced into the Linux kernel, applications can access the disk files without page cache. This feature make it possible that applications can implement their own cache strategies in user space without page cache.
IV. Kernel I/O scheduler: sorting, merging and prioritizing the I/O requests and dispatching them to disk controller. Four common schedulers have been implemented in the kernel (CFQ/DEADLINE/AS/NOOP), at least 5 arguments can be optimized by applications for each scheduler like queue length, expire time and so on. But how to change and when to change them for different environment is a big problem.
V. Disk: servicing the requests and providing another layer of catching. The behavior is vendor-specific.

With the growth of requested data size, huge requests for data transfer will cause a large number of memory copies and system calls between user and kernel space which are useless and time consuming for grid server. Frequent context switch prevent CPU resources from been allocated to other processes. But in most cases, the data which copied from kernel memory do not need to be processed and handled in user space. This feature of the operating system (OS) waste a lot of valuable computer resources. The overhead of Networking software can be broken up into two parts: per-packet, which is constant for a given protocol, like max transfer unit (MTU) has been limited to 1500 Byte in TCP/IP stack. Per-byte, the cost of this is determined by data coping and check summing. So per-byte costs is the main cause for our grid servers.

A method called zero-copy has been designed for moving data between application domains and network interface without CPU intervention. Several zero-copy schemes have been proposed as follow: