CSCanada
Studies in Mathematical Sciences
Vol. 4, No. 1, 2012, pp. 18-21
DOI: 10.3968/j.sms.1923845220120401.1750

ISSN 1923-8444 [Print]
ISSN 1923-8452 [Online]
www.cscanada.net
www.cscanada.org

Research on the Role of Cache and Its Control Policies in Software Application Level Optimization

ZHENG Ying[1,*]

[1]Editorial Department of Journal, Inner Mongolia University for Nationalities, Tongliao, 028043, China
[*]Corresponding author.
Address: Editorial Department of Journal, Inner Mongolia University for Nationalities, Tongliao, 028043, China

Received 26 November, 2011; accepted 16 January, 2012

Abstract

Cache becomes very important in high-load computer application. In a web application, cache can improve the performance of application by several orders of magnitude generally. The article analyzes the role of cache in software application level optimization in detail and divides cache into local cache, local shared-memory cache, distributed memory cache, disk cache. Several cache control policies including alive time, invalid in writing, invalid in reading et.al are studied and the problems when using cache in applications are pointed out.

Key words

Software application level; Cache; Optimization; Cache control policies

ZHENG Ying (2012). Research on the Role of Cache and Its Control Policies in Software Application Level Optimization. *Studies in Mathematical Sciences, 4*(1), 18-21. Available from: URL: http://www.cscanada.net/index.php/sms/article/view/j.sms.1923845220120401.1750 DOI: http://dx.doi.org/10.3968/j.sms.1923845220120401.1750

INTRODUCTION

Cache becomes very important in high-load computer application. In a classical web application, using cache produces less content than directly providing service. Cache can improve the performance of application by several orders of magnitude generally [1]. When using cache, the following points should be paid attention to: which content needs to be cached; where to cache; the size of cache granularity and invalid cache policy, et.al.

Generally, a high-load application has many levels of cache. Cache not only appears in servers, but also includes users' web browsers [2]. In general, the closer to clients the cache is, the more resource is reduced and the higher the efficiency is. For example, a photo is read faster from a browser cache than the internal memory of a web server, and it is read better from the internal memory of the web server than directly from external memory [3].

Cache is divided into initiative cache and passive cache. Passive cache does nothing, except saving and returning data. When requesting content from passive cache, there are two results only: 1) returning the result you need; 2) there is no content you need. While initiative cache cannot find the requested data, it usually delivers your requests to certain parts of application. It can produce the requested content, and then initiative cache saves the content and returns to the clients [4].

1. THE CLASSIFICATION OF CACHE BASED ON APPLICATION LEVEL

Generally, classical cache based on application level puts data on the local memory, or another computer's memory on the internet. Generally, cache on application level has higher efficiency than lower levels of cache, because application can save parts of calculated results in the cache. Cache is helpful for two kinds of jobs: Reading the data and calculating the data. For example, for each block of HTML text, application can produce HTML paragraph such as headline news, and then cache them. Then the following web page can put the headline news cached on the web page directly. Generally speaking, the more the data is processed before cache, the more the work is saved after using cache.

The cache based on application level is classified below, taking MYSQL database application for example.

1.1 Local Cache

The cache is usually very little, and exists in the memory when requesting processing. They can be used for avoiding multiple requests for the same resource. It is usually a variable or hash table in application program code. The technology is very simple, but it can save much work. For example, if user names have to be shown, while we only know user ID, in order to design a function get_name_from_id , where cache function is, the php code is below:

```
<?php
Function get_name_from_id($use_id){
Static $name;//use static to modify to keep variables persistent
If (!$name){
//obtain user name from database
}
Return $name;
}
?>
```

1.2 Local Shared-Memory Cache

The size of local shared-memory cache is medium, about several GB generally and its visit is very fast, but keeping each machine in synchrony is very difficult. They apply to small and semi-static data storage. For example, hotels list, mapping table and the date that uses TTL (Time-to-live) policy. The largest advantage of local shared-memory cache is the fast visiting speed. Generally, it is much faster than any distant cache.

1.3 Distributed Memory Cache

Distributed memory cache is larger than local shared-memory cache and it is easy to expand. Each cached data is created only once. When the same data is cached in different places, there will not be the problem of data inconsistency. Distributed memory cache specializes in sequencing of shared objects, such as user information file, forum and HTML content clips.

The cache has higher delay than local shared-memory cache, so in order to effectively use them, multiple access operation has to be done, which means to read multiple data objects in one round. In addition, how to add more nodes should be also planned well, as well as how to do when a node breaks down. Under the two situations above, application has to decide how to distribute and re-distribute cache objects between each node.

1.4 Disk Cache

Disk has low speed, relative to the memory, therefore, it is the most suitable for persistent objects to become disk cache. Objects are often not suitable to be put in the memory.

To capture lost cache through 404 error processing procedure is the technology to effectively use disk cache and web server [5]. For example, joining web application has to show a customized picture on the head of web page. If the picture does not exist, it will produce a 404 errors and triggers error processing procedure. Then, error processing procedure produces the picture and it is saved in the disk. Then a redirection is started or the picture is filled in the browser only. The following visits can return the picture from the file directly. The technology can be used in many occasions, for example, HTML codes that is used to show headline news is not cached, but put them in a JavaScript file, and then insert an index that points to the js file in the head of web page.

Operation for invalid cache: deleting the file is ok. Through operating a periodic task, the file created before M minutes are deleted so as to realize TTL invalid policy. If limiting the size of cache, TRU (Least Recently Used) invalid policy can be used to delete contents, according to the creation time of cache content.

2. CACHE CONTROL POLICIES

The problem derivative from cache is the same as that the design of database violates basic paradigm. Both are because they contain repeat data, which means that updating data needs to update many different places, and also avoids overdue data [6]. Here are several important cache control policies below:

2.1 Alive Time

Each cache object has an invalid date. Take a deletion program to regularly check whether the invalid time of the data is up. If it is up, delete it immediately or pay no attention to it temporarily until you visit it next time. If the invalid time exceeds, the latest version is used to replace it. The invalid policy is most suitable for the data with little change or without refreshing.

2.2 Invalid in Writing

If the data in the cache is too obsolescent and cannot be accepted, the process of updating cache data will make the old version of data invalid immediately. The policy has two variant types: invalid in writing and update in writing. The policy of invalid in writing is very simple: directly mark the data as invalid or delete it from the cache. The policy of invalid in update does more because it needs the latest data to replace old cached data. The policy is useful, particularly when the cost of producing cache data is very high. After updating cache, it is unnecessary for the next requests to wait for the application to produce the data. If the invalid process is executed in backstage, for example, the invalid process based on TTL, the latest version of data can be produced in a process independent of any user requests to replace the data invalid in the cache.

2.3 Invalid in Reading

Relative to make corresponding old data invalid when changing metadata, the alternative method is to save some information to help judge whether the data read from the cache has been invalid. It has a more obvious advantage than invalid in writing: as time goes on, its cost is fixed. If an object is invalid, but there are 1 million objects that rely on it in the cache. If it becomes invalid in writing, the related 1 million objects in the cache have to be invalid. While the delay of 1 million times of reading is very little, this can reduce the time cost of the operation of cancellation and avoid long-term delay when reloading.

The simplest method to adopt the policy of invalid in writing is practicing object version management. In the method, when saving objects in the cache, the version number or time stamps the data relies on

have to be saved. For example, if the statistical information about the articles published in a user's blog is saved in the cache, the information includes the number of articles. When it is taken as blog stats objects for cache, current version number of the user should also be save at the same time, because the statistical information relies on specific users.

CONCLUSIONS

Cache is very important in high-load computer application. In a classical web application, using cache produces less content than directly providing service. Cache can improve the performance of application by several orders of magnitude generally. When using cache, the following points should be paid attention to: which content needs to be cached; where to cache; the size of cache granularity and invalid cache policy, et.al. The article analyzes the role of cache in software application level optimization in detail and divides cache into local cache, local shared-memory cache, distributed memory cache, disk cache. Several cache control policies including alive time, invalid in writing, invalid in reading et.al are studied and the problems when using cache in applications are pointed out.

REFERENCES

[1] LIN Song, WU Jianping, XU Ke (2010). Analysis and Evaluation of Internet Resource Distribution Models. *Journal of Tsinghua University (Science and Technology), 1*, 58-62.

[2] WEI Yan-shan, ZHANG Jian (2010). An Improved Cache Optimization Algorithm Based on Weighted Path Pattern Mining. *Microcomputer Information, 33*, 137-139.

[3] WANG Xin (2011). The Application of Cache Technology in Web. *Journal of Weifang University, 4*, 46-49.

[4] QIN Zheng, XU Ya-fei (2009). The Research of Design of Data Services Engine of Data Center. *Microcomputer Information, 30*, 16-18.

[5] XU Jin-long, JIANG Lie-hui, DONG Wei-yu, FANG Ming (2011). Research of Optimization On Parallel Multiple Thread Dynamic Binary Translation. *Computer Engineering and Design, 7*, 2370-2372, 2380.

[6] OU Yanglijun, LIU Yansong (2009). Streaming Media Caching Proxy Server Technology Research. *China New Technologies and Products, 7*, 16-17.