EFFICIENT EXTRACTION OF AN ARBITRARY SUBSET OF DATA FROM LARGE THREE-DIMENSION VOLUMES BY MANIPULATING OF PHYSICAL POSITION OF DATA

Reviakin A.V.

Scientific Supervisors: Chilingaryan S.A., Ph.D.; Fadeev A.S., Ph.D.

Tomsk Polytechnic University, Russia, Tomsk, Lenin str., 30, 634050

E-mail: presler@tpu.ru

Abstract

In this work, the method of organizing three-dimensional data on a storage drive for more efficient extraction of an arbitrary subset of data was considered.

ЭФФЕКТИВНОЕ ИЗВЛЕЧЕНИЕ СЛУЧАЙНОГО ПОДМНОЖЕСТВА ДАННЫХ ИЗ БОЛЬШОГО ТРЕХМЕРНОГО ОБЪЕМА ПУТЕМ ИЗМЕНЕНИЯ ФИЗИЧЕСКОГО РАСПОЛОЖЕНИЯ ДАННЫХ

Ревякин А.В.

Научные руководители: Чилингарян С.А., к.т.н; Фадеев А.С., к.т.н. доцент

Томский Политехнический Университет, Россия, г. Томск, ул. Ленина 30, 634050

E-mail: presler@tpu.ru

Аннотация

В данной работе рассмотрен способ организации трехмерных данных на диске для более эффективного извлечения случайного подмножества.

Introduction

With growing of data volumes, providing of fast random access to information becomes a challenge. There are several ways to make the process of getting information faster:

- by buying new more powerful hardware that leads to spending of money (scale up);
- by combining existing available set of machines into cluster (scale out). This way can be not so good, because in order to speed up access to big files it is necessary to split them to blocks, which led to dilemma between reliability (with decreasing of available space due to necessity of doubling data) and speed (with the same hardware there will be lack of redundancy) [1];

For some types of data, there is an opportunity to get boost in speed by manipulating of physical position of data. Three-dimension images come into that category. Such manipulations can be done on already available hardware, in this article it will be the main point.

Nowadays, there is an ongoing process of increasing of three-dimension data volumes, and with time, scientists want to see more details, which brings to demand for higher resolution that means higher amount of data, eventually. Amongst such fields where three-dimensional arrays are widespread in use are geology, seismology, meteorology, oceanography, biology, medicine, astrophysics and many others [2].

This work is based on needs of fast extraction of arbitrary data from large three-dimensional tomographic images, which were got at synchrotron facilities. However, results can be applied in wide range of applications.

Testing

The data of three-dimensional array, if it is not a special file format, is stored on a hard disk as a plain one-

dimensional array. If one wants to extract a part of such parallelepiped (in general case), one has to read information from different parts of hard disk, which led to active use of random access. For HDDs random reading speed is much slower than sequential reading speed. Therefore, there is an opportunity to change physical position of file on HDD according to its logical structure to turn random access to sequential access.

For experiments there was used Linux OpenSUSE 13.2, for writing of script tests Python 3.4.1 was used, also NumPy and scikit-image packages were used for some routine stuff. For hardware, there was used machine with core-i5 CPU, 16 GB of RAM, Western Digital RE4 HDD (2TB, 7200 RPM, 64 MB cache, 3GB/s SATA) with ext4 as a file system. As a test volume, TIFF three-dimensional image was used with resolution of 2016*2016*2016, which with 1 byte per pixel gives about 7.6 GB in total size.

Initially the work of extracting subvolumes from such array was made using memmap function of NumPy package. Memmap is used to mapping entire file into memory, but NumPy's memaps's are array-like objects and Python's mmap module uses file-like objects. Therefore, it is possible to open such TIFF image using memmap function and easily get random access to any part of file. Such method will be used as a reference method for comparison with other ways.

For comparison, it was written script to divide volume to small cubes (parallelepipeds) with the edge size of 32, 64, 100, 128, 150 pixels. During tests, cubes of the size 300*300*300 pixels (25.749 MB) were extracting from random parts of initial volume through 100 iterations. To the reference test, there were added two types of extracting data using these small cubes: the first one is when each next read data was concatenated to the previous one, the second one is when the memory for 300 pixels edge cube was preliminary reserved and the read data were copied to it. The reading was made using ordinary read function, it means more data than needed was read, due to that nominal speed of reading (for total reading data) and efficient speed of reading (for the useful information of 300 edge cube) were calculated.

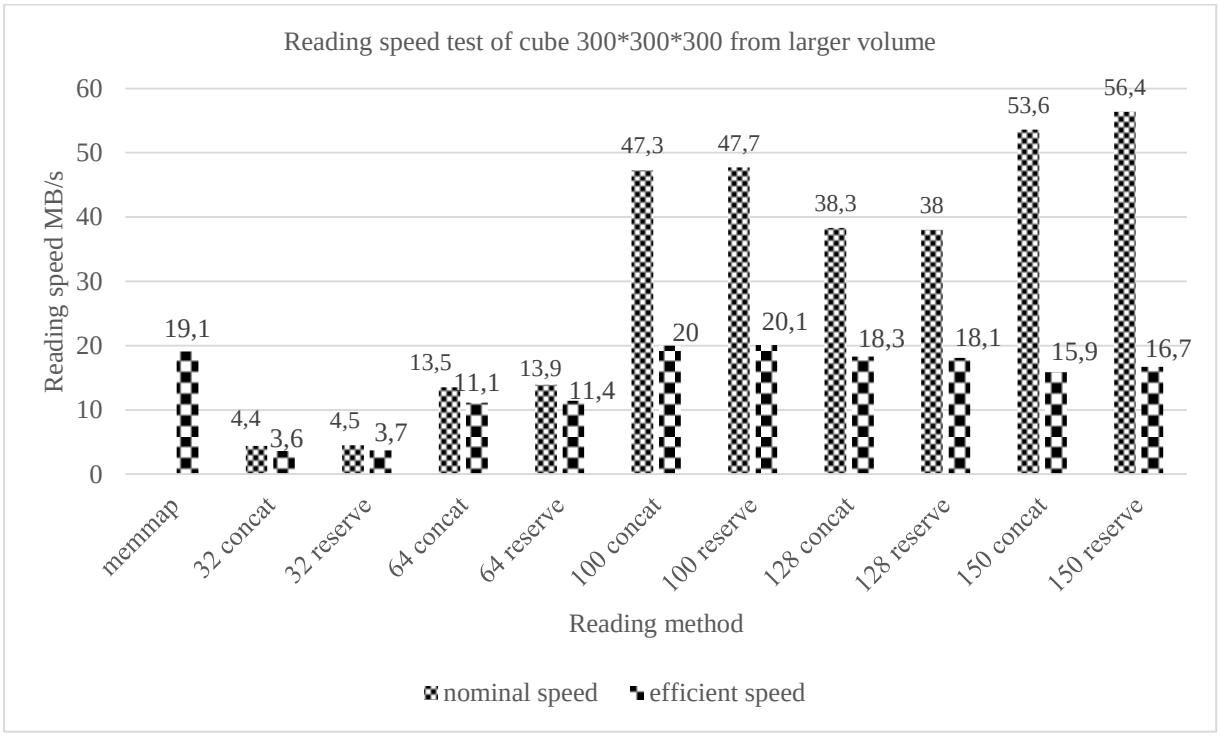

Figure 1. Extracting of cube with 300 pixels edge (25.749 MB)

Judging by results of this chart it can be concluded that splitting of big volume to small cubes (parallelepipeds) with different edge sizes almost does not give any speed advantage in efficient speed with

using of ordinary reading. It can be seen, that nominal speeds since the tests with edge size of 100 pixels are higher than speed of reference test (memmap), however most of efficient speeds much lower than that in reference test. Such results can be explained mostly by reading of overhead information, because only part of data in small cubes are required. In addition, there was one more overhead, because firstly the opened file should be copied to buffer and then from buffer part of it (or the whole buffer) to the target place. The only case when efficient speed exceeded the speed in reference test was when the size of edge was equal to 100 pixels.

As it was said before, during the previous test nominal speed appeared. It means there is an opportunity of optimizing this process by reading only necessary data – using of mapping every small cube to memory as it was done with initial volume. Moreover, two different ways of reading files can be compared in similar conditions. The memory for extracted subvolume was preliminary reserved as was made for previous tests.

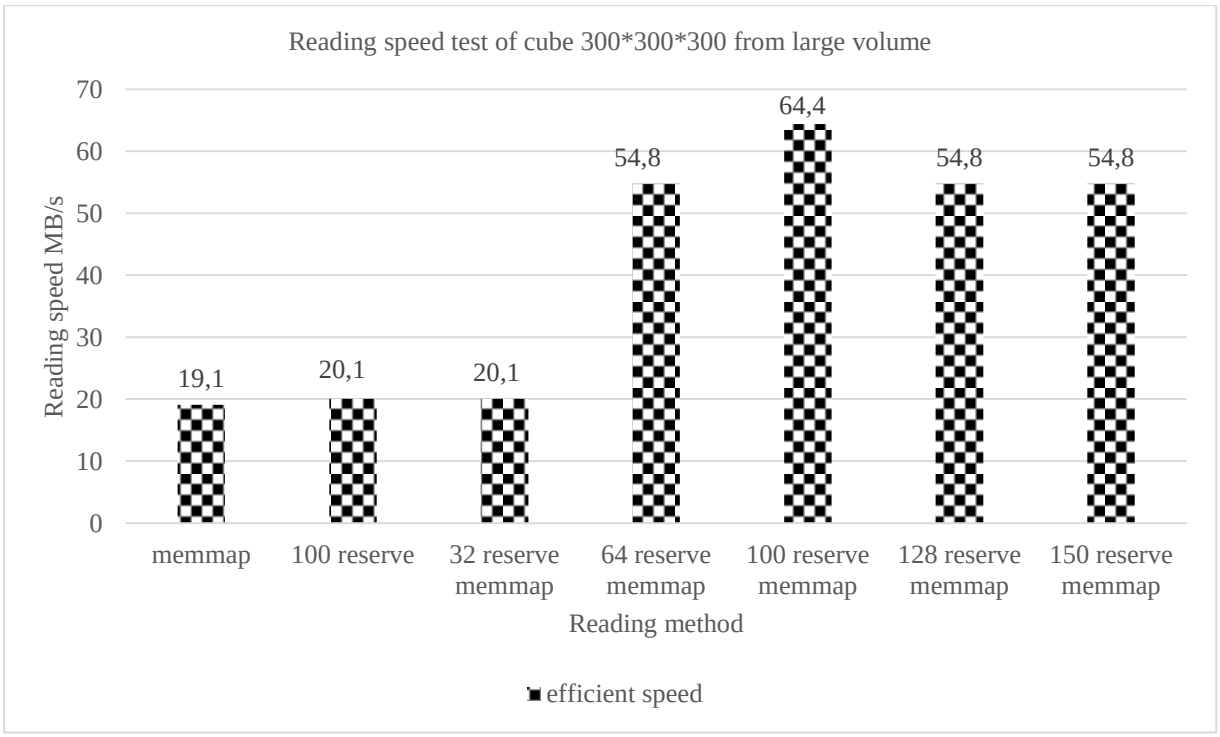

Figure 2. Extracting of cube with 300 pixels edge (25.749 MB)

Results showed huge improvement in reading speed with using of memmap function for every small cube, especially since edge sizes of 64 pixels. With edge size of 100 pixels the reading speed of 64.4 MB/s was achieved using memory mapping of each cube, what is about 3.2 times higher than using ordinary read function of the whole file and working with it in buffer.

Conclusion and future work

To conclude the results, it was proved that for tasks of extracting three-dimension subvolumes from big volumes can be optimized by changing of physical position of data on the storage drive according to logical structure of that information. This method is easy in use and can be applied in different fields where the workflow is similar to the described in this article. The main advantage that there is no need to buy any additional hardware or software, because similar algorithms can be implemented in house.

References

1. Reviakin A. Scalability of GlusterFS in a small cluster for providing fast access to large amount of data // Microsoft technologies in theory and practice of programming: VII festival Microsoft in TPU. – Tomsk, 2015.