Improving Image Processing Systems by Using Software Simulated LRU Cache Algorithms

Cosmin CIORANU[1], Marius CIOCA[2], Lucian-Ionel CIOCA[2]
[1]UEFISCDI, Bucharest
[2]"Lucian Blaga" University of Sibiu, Sibiu
cosmin.cioranu@gmail.com, marius.cioca@ulbsibiu.ro, lucian.cioca@ulbsibiu.ro

Today's scientific progress is closely related with data processing, a process is implemented using algorithms, but in order to have a result, algorithms need data, and data are generated by sensors, particularly satellite imagery or collaborative GIS platforms. The progress has made those imaging capturing sensors more and more accurate therefore the generated data are becoming larger and larger. The problem is mostly related to the operating system and sometimes software design's inability to manage contiguous spaces of memory. In an ironic turn of events, those data sometimes cannot be held all at once in a computer system to be analyzed. A solution needed to be devised to overcome this easy problem at first, but complex in implementation. The answer is somehow hidden, but is has been around since the birth of computer science, and is called a memory cache, which is basically at its origins a fast memory. We can adjust this concept in software programming by identifying the problem and coming up with an implementation. The data cache can be implemented in many various ways but here we will present one based on LRU (least recently used) algorithm mostly to handle three dimension arrays, called 3dCache which is widely compatible with software packages that supports external tools such as Matlab or a programming environment like C++.
Keywords: *Cache, LRU, Matlab, C++, Three Dimension Array, Satellite Imagery*

1 Introduction

This paper describes a solution to a problem that somehow it is hidden to the average and even expert user of a computer system. When it comes to operating large volumes of data, most computer systems have a problem allocation large contiguous space in memory [1], [14] even if it has 4, 8, 16 or more GB of RAM at their disposal. The problem is mostly related to the operating system and sometimes software design's [10] inability to manage contiguous spaces of memory or model representation [4]. The problem becomes even more troubling when a processing algorithm (written in Matlab, C++ or other environment) behaves perfectly running on small amount of data but when it comes to large volumes it simply brakes down. The need for a solution came obvious where sophisticated processing in the field of satellite imagery. For those kind of data is common place to have a resolution than 10000x10000 with 20 bands staked [6], having a precision of 32 bit in depth, which means roughly 8 GB of data and for the simplest operation we needed memory space of at least twice, more than 16 GB of RAM. So, having this hypothesis in place we created the *3dCache package* which can be implemented and integrated into C++ and all software packages which have the ability to use compiled code, including Matlab.

2 General Overview

The current implementation is based on LRU (Least Recently Used) algorithm, which has been around since the birth of computer science, in various implementations [3], [9], [11], web solutions (proxies, web-servers, etc.), generic improvements of CPU execution time. In a computer system, as a hole, enhancements are added in order to overcome the never ending need of processing power. One of those enhancements is the addition a specialized memory between the CPU and main system memory called a cache memory [8]. The cache is much faster in terms of access but is limited in size, therefore there is a need of an algorithm to bring data information from the main memory into the cache and to evict already present data from to cache to the main memory [5], [8], [15]. The performance as a hole depends mainly upon the size of the cache, the internal organization, the algorithm used to manage the cache and of course the, nature of the data that the CPU needs to execute. Currently there are a number of tools, which can handle large data volumes of information, such as BOOST Libraries, represented by std::map class [11], but the main problem with most of them is the inability to predict the actual size the structure used, and given the problem stated before, we encountered the same problem with contiguous space allocation. Also, we should take into consideration that

most satellite imagery processing needs more than two result allocation (one as source and one as destination) so we need a solution that can be used in 100 % error free problems related with memory allocation. The main point of view taken into consideration had to be data itself and the types of processing. Usually most satellite imagery (Visible, Infrared, Water Vapors) [6], [16] are grouped into so called products. Those products have some metadata attached but invariably are composed by a stack of images usually the same in width and height, each image from this stack is called a band and represents in some form the date taken from the sensor.

3 Technical Approach

The usual approach of a system that processes data, in general and satellite products particularly is described in figure 1. First a software package detects the format in which the data are stored, allocated the necessary memory for the selected data, and then everything is loaded into internal memory to be analyzed by an algorithm. This process can be optimized in some ways but the general approach remains the same.

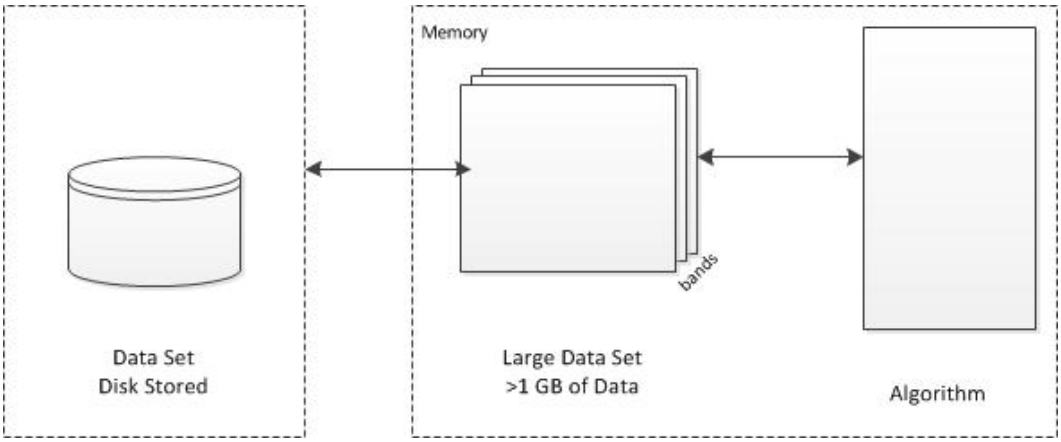

Fig. 1. Usual image loading process

As it can be seen, to be effective, the data has to be available, and for this purpose, the data has to load into main memory. The main advantage is speed. If the data is relatively small, there are no special issues, but if the data is large, let's say more than 1 GB of data, some systems have really big problems managing data internal structures.

Our Approach is a little different. We follow the same reasoning as before, but instead of putting the data into memory we are putting it in a special structure that works with a self-owned swap space. From outside, the algorithm side, it may see the hole data is stored in memory, but in reality everything is swapped between the 3d designated swap space and an internal structure (Figure 2).

In our technical approach we had to take into consideration the following elements:

a) Types of processing in terms of operations over stack layers;
b) Type of data contained in the fore mentioned bands;
c) Operating system;
d) Concurrency, how many processing instances are running in the same time.

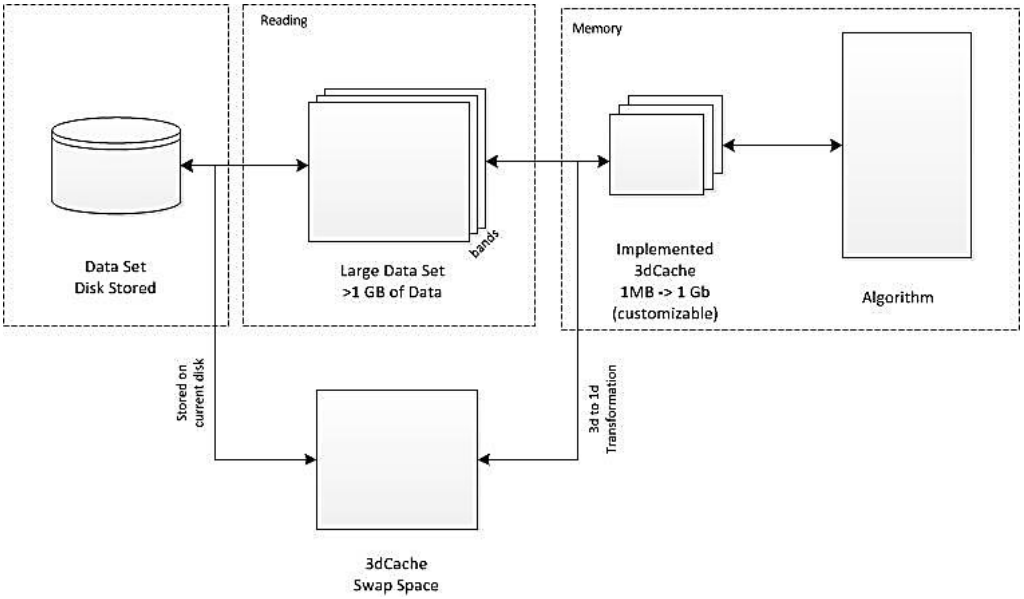

Fig. 2. Basic 3DCache Architecture

Usually most algorithms in this area, satellite imagery, use all the bands when a process decision is been made, so in our architecture we had to take into account this fact. The decision taken was that we need to structure our cache system in something called *tiles*. Usually tiles are 2d parts of an image, described by width and height, but given the stated issue, our tiles had to be parallelepiped in form (figure 3). The size of the tile is specified by a set of parameters used upon the initialization of the cache.

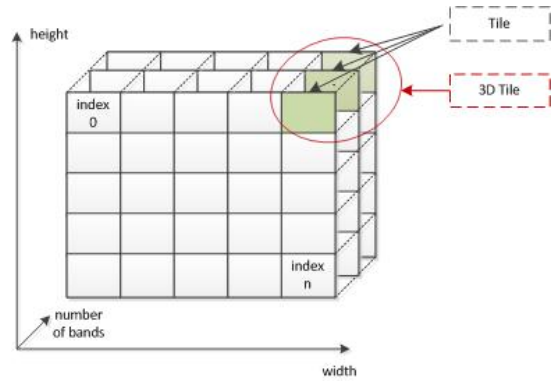

Fig. 3. Internal 3dCache Structure

The list of the parameters is as following:
a) *Width,* representing Maximum width of the stack images;
b) *Height*, representing Maximum height of the stack images;
c) *NoBands,* representing number of bands in the stack images;
d) *MaxCacheUnitSize,* this is the maximum cache that can be used for a tile of one depth. It is possible that de actual size will be smaller.
e) *CacheSize,* Number of cached elements.
f) *bandResolution,* representing the size in bytes of the internal representation resolution. It can be long, or double.
Total memory foot print of an internal structure can be calculated using these simple operations:

$$FootPrint = cacheSize * actualCacheUnitSize, \\ where\ actualCacheUnitSize < maxCacheUnitSize \quad (1)$$

The *actualCacheUnitSize* describes the actual size in memory of the unit of cache and is calculated using the following algorithm:
a) Calculate the maximum area of a 2d tile (width and height) that can be used for a one band tile:

$$maximumArea = \sqrt{\frac{maxCacheUnitSize}{noBands * bandResolution}} \quad (2)$$

b) Calculate maximum tile width using the *maximumArea*
c) Calculate maximum tile height using the *maximumArea*
d) Calculating *actualCacheUnitSize*

$$actualCacheUnitSize = tileWidth * tileHeight * noBands \\ * bandResolution \quad (3)$$

Each cache unit is identified by an *index*, which describes a portion of the large dataset into main memory. The translation of a point P(x,y,z) is determined by the internal address translation of the 3dCache. From the cache time point of view, the

current implementation is using direct mapping, which is faster in terms of internal selection, but it may be insufficient in case of complex processing, as stated in [11].
Given the fact that most processing data are build using various types of data, we built our system in two generic systems, one for integer types, and one for double types
The operating system was an issue, so we choose to build our system using C++ programming language. First it was developed under windows then ported over Linux.
One of the most important questions we had to answer was if our system will be part of a data processing server. Having this in mind we built the entire cache system fully parametric (size of working cache, size of one single tile, etc.)
Advantages of this approach:
a). Controllable foot print of each instance of this specialized structure
b). No need or little alteration of preexisting algorithms
Disadvantages of this approach
a). The problem is shifted from the main memory to the disk space in terms of amount of data stored
b). Speed may be an issue, but it depends of the internal management of the 3d Cache

4 Integration and Usage

In Matlab one can import C/C++ (or other kind) code by using the available installed compiler by creating a MEX file [12]. A MEX file is a built-in utility that enables you to call C, C++, or FORTRAN code in MATLAB by compiling your code into a MATLAB Executable called a MEX-file. MEX-files are dynamically linked subroutines that are called as regular MATLAB functions. This requires you to replace your application's main() with a special gateway function - called "mexFunction" - to pass inputs and outputs to and from MATLAB. The operation requires some skills regarding the compilation stages but it can accomplish by using the following steps:
a) On the host system it needs to be installed a compiler [13] A full list can be found on the bibliography
b) Use the following command to compile 3dCache main file, let's say Matrix.cpp

mex Matrix.cpp

c) Call from the Matlab console Matrix.cpp with the documentation provided parameters.
To gain access from an IDE, Visual Studio or otherwise to the enhancements provided by de 3dCache system you have to follow a few steps:
a) Import the necessary libraries
b) Instantiate the class using:
Cache3dDouble matrix(width,height,noBands, maximulAllowedCacheUnitSize, noOfUnits)
c) Get a value *double value=matrix(x,y,z)*
d) Set a value *matrix(x,y,z)=value*

5 Results

The nature of the proposed algorithm and implementation makes hard or impossible to compare it with other similar solution mainly due to:
a) It has been built to serve a specific kind of applications
b) Limited uses in terms of multidimensional data structure
c) Can hold only certain types of data, now are available only long and double
Given the condition(s) above we will display some results collected in contiguous mode set/get of data and in random access of the structure. We will choose for our demonstration the following configuration:

- Image size: Width: 256, Height: 256, Number of bands: 3, resulting in ~1.5 MB of data;
- Size of cache location (automatic) with a maximum of 800 bytes, resulting in 768 bytes.

i) Contiguous Mode

In contiguous mode, we asked our structure, upon initialization to retrieve the requested data, in case of various allotted internal locations (Table 1) and (Figure 4).

Table 1. Analysis of hit rate in case of contiguous access

No.	Cache Locations	Hit %
1	96	97.917
2	112	97.917
3	128	97.917
4	144	98.161
5	160	98.405
6	176	98.469
7	192	98.893
8	208	99.137

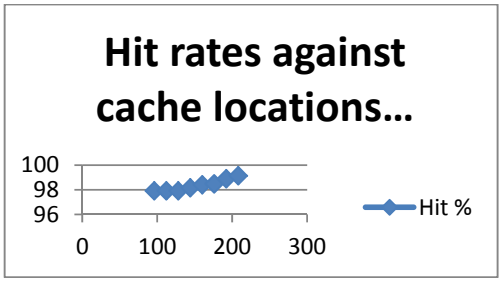

Fig. 4. Graphical representation of hit rates on various allotted number of cache allocation

ii). Random Mode

For the random experiments we have chosen 4096 points distributed randomly over the space generated by the maximum range of the tested product staked images.

The results are as following (Table 2).

Table 2. Results upon testing with random values from entire space of possible images coordinates

No.	Cache Locations	Hit %
1	96	36.093
2	112	42.271
3	128	47.829
4	144	53.865
5	160	58.802
6	176	66.078
7	192	71.35
8	208	76.718

As it can be seen the results are not very promising, but we should take into consideration that we kept in memory only about **10%** at 208 cache locations of the entire image.

But, with 20% at 208 cache location, the efficiency of the structure increased to 99% in case of random values (Table 3).

Table 3. Efficiency of the structure in case of 20% image stored in main memory

No.	Cache Locations	Hit %
1	96	71.517
2	112	83.492
3	128	94.012
4	144	96.827
5	160	96.827
6	176	96.827
7	192	96.827
8	208	96.827

6 Conclusions

This architecture has proven its uses but there is room for improvement, however it offers an alternative in case of large amount of processing data such as collaborative GIS (Geographical Information System) platforms [7] or collaborative DSS (Decision Support Systems) [2]. Among the benefits of using the presented system is its flexibility, it can be integrated with various professional software platforms and the proven viability of concept, by using an well know algorithm, such as direct mapped cached, with LRU as a way to evict non used information from the main system memory [8].

Feature developments include:

- Extending the dimensional limit, now we are using only three;
- Implementing N-Associative cache, direct-mapping has some issues regarding conflicts of mapped memory [8], which may force the 3dCache to evict cache locations that may be used in a future iteration.

References

[1] R.G. Belu, C. Oancea, A. Belu, L.I. Cioca, "A 2-d indoor radio propagation modeling by using MATLAB for classroom instruction" in 33 rd Annual Conference Frontiers in Education, FIE 2003, ISSN 0190-5848, DOI: 10.1109/FIE.2003.1263385

[2] M. Cioca, L.I. Cioca, "Decision Support Systems used in Disaster Management", in Decision Support Systems, Chiang S. Jao (Ed.), ISBN: 978-953-7619-64-0, 2010.

[3] E J. O'Neil, P. E. O'Neil, G. Weikum, "The LRU-K page replacement algorithm for database disk buffering", Volume 22 Issue 2, June 1, 1993, Pages 297 - 306

[4] F.G, Filip, "Opti.mization models with sparse matrices and relatively constant parameters" in Systems Analysis Modelling Simulation. Vol. 33, no. 4, pp. 407-430. 1998.

[5] J. B. Sartor, S. Venkiteswaran, K. S. McKinley, and Z. Wang, "Cooperative Caching with Keep-Me and Evict-Me", INTERACT '05 Proceedings of the 9th Annual Workshop on Interaction between Compilers and Computer Architectures Pages 46 – 57, IEEE Computer Society Washington, DC, USA 2005

[6] T. M. Lillesand; R. W. Kiefer; J. W. Chipman, "Remote sensing and image interpretation" in book "Remote sensing and image interpretation 2004" Edited by Lillesand, T. M.;Kiefer, R. W.;Chipman, J. W, ISBN 0-471-45152-5

[7] M. Cioca, S.C. Buraga, C. Cioranu, "Disaster prevention integrated into commonly used web rendered systems with GIS capabilities", INT J COMPUT COMMUN, ISSN 1841-9836, 2012
[8] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache prefetch buffers", IS-CA '98, Pages 388-397
[9] Rachid El Abdouni Khayari, RaminSadre, Boudewijn R. Haverkort, "A Class-Based Least-Recently Used Caching Algorithm" for WWW Proxies, 2012
[10] Image Acquisition Toolbox, Memory Limitation http://www.mathworks.com/help/imaq/imaqmem.html (accessed in 15.09.2012)
[11] LRU cache implementation in C++, BOOST, std:map http://timday.bitbucket.org/lru.html (Accessed in 01.11.2012)
[12] Matlab C++ Integration http://www.mathworks.com/support/solutions/en/data/1-GQC9 NF/index.html (Accessed in 25.10.2012)
[13] Matlab list of commercial and non-commercial compilers http://www.mathworks.com /support/compilers/R2011b/win32.html (Accessed in 25.10.2012)
[14] Matlab Memory limitations http://www.mathworks.com/help/matlab/matlab_prog/ resolving-out-of-memory-errors.html (accessed in 25.10.2012)
[15] Organizarea si proiectarea microarhitecturilor de calcul http://webspace.ulbsibiu.ro/lucia n.vintan/html/Organizarea.pdf (Accessed in 02.10.2012)
[16] Types of satellite imagery http://www.srh.noaa.gov/mrx/sattype.php (Accessed in 30.10.2012)

Cosmin CIORANU is working as Software Coordinator at UEFICDI (Executive Unit for Higher Education, Research, Development and Innovation Funding) closely involved in managing National Plan funded projects. Also, the satellite imagery systems are researched at ASRC (Advanced Studies and Research Center) in various projects, some financed by ESA (European Space Agency). He has a PhD from "Politehnica" University of Bucharest, Faculty of Power Engineering. Cosmin Cioranu has extensive knowledge and experience in software design, stand-alone and web..

Marius CIOCA – Marius CIOCA is a faculty member at the "Lucian Blaga" University of Sibiu (Romania). He earned his PhD from "Politehnica" University of Bucharest, the Faculty for Automation and Computers. Dr. Cioca published numerous scientific articles at prestigious international conferences under the aegis of Elsevier and IEEE. The scientific training as young researcher also came from his position as director of numerous research contracts awarded through national competition. His domains of interest are: reference architecture, industrial information systems, modeling languages, web engineering and technologies and decision support systems.

Lucian-Ionel CIOCA is a faculty member at the "Lucian Blaga" University of Sibiu. Full professor (since 2007), has extensive experience in Fuzzy technology and Matlab environment. Subsequently becomes expert in Healthcare and Occupational Safety. He has published numerous scientific papers (IEEE, IFAC, Elsevier etc.) and won the competition numerous research contracts.