Distributed Storage System Approach for Imagery Data in Content Delivery Networks

Andrew Kokoulin, Alexander Yuzhakov
Perm national research polytechnic university
Perm, Russian Federation
{a.n.kokoulin, yuzhakov}@at.pstu.ru

***Abstract*—This paper describes an approach that improves the functionality of LH*RS - high-availability scalable distributed data structure. The core concept behind this approach is the particular implementation of LH* schema that allows efficient imagery data storage for content delivery networks (CDN) to be created. We propose the novel wavelet-based method of buckets composition and distribution among storage nodes to improve the basic characteristics of schema. This method is less redundant than image pyramid structure (MIP maps). Also we propose some ideas for multilevel encoding algorithms and multilevel privacy protection in this storage schema.**

I. INTRODUCTION

The Internet plays a central role in our society – work and business, education, entertainment, social life. The vast majority of interactions relate to content access [11]

- P2P overlays (e.g. BitTorrent, eMule, live streaming)
- Media aggregators (e.g. YouTube, GoogleVideo)
- Over-the-top video (e.g. Hulu, iPlayer)
- Content Delivery Networks (e.g. Akamai, Limelight)
- Social Networks (e.g. Facebook, MySpace)
- Photo sharing sites (e.g. Picasa, Flickr)

Online Internet-resources started with host-centric communication model with file sharing and evolved to multicast-streaming real-time video through overlay nodes with information-centric networking model.

There are some issues that appear when processing user's imagery data in online resources. In our opinion, all these problems exist due to imperfect image file format.

Historically images were stored and processed on local computer or server and this was efficient for local infrequent users' requests. Most local image processing software assumes you can load the entire image into memory and then perform an image processing. Assuming the typical web resource structure, the original image (static content) can be presented as a thumbnail, low-quality preview image and in its original quality on different pages of the same web-site. If we use the original image in all the HTTP-responses this will lead to an excess traffic and network overload. To avoid this we have to support and optimize device-specific content to construct a web page. The basic idea is to store multiple image copies in different scales rather than single image in original resolution using such techniques as Data pyramiding (Fig.1).

Data pyramiding is an image scaling technique that has been used in terrain rendering to reduce computation and eliminate aliasing problems [1]. For a large input dataset it is important to be able to view the data at different scales, from a perspective that overlooks the entire dataset down to a small area for a close look of the terrain. An input data pyramid is built by repeatedly dividing the input image by a factor of 2 in each dimension. A multi-level input pyramid requires about 1/3 more memory than the original image, which could be significant when trying to fit large datasets to any particular machine.

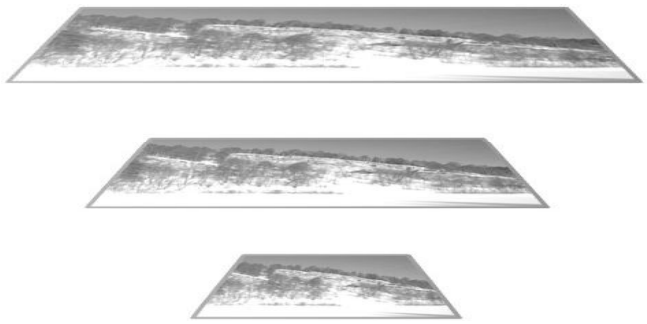

Fig. 1. Data pyramiding (MIP maps)

Other method is dynamic optimization of the image size and resolution to deliver faster page load times on mobile devices. Web-site creates the appearance of a full page allowing the user to start interacting with the site immediately, then quietly fills in the rest of the page without disrupting the user experience or sends reduced-size images first and full resolutions images later.

Both approaches increase redundancy of data storage and the number of objects to be stored and synchronized. And it is even more important when using a content delivery networks (CDN) to cache and synchronize the static imagery content.

As it was noticed in [12] the most significant internet delivery challenges include:

- Peering point congestion. Capacity at peering points where networks exchange traffic typically lags demand, due in large part to the economic structure of the Internet;
- Inefficient routing protocols. Although it has managed admirably for scaling a best-effort Internet, BGP has a number of well-documented limitations;
- Unreliable networks. Across the Internet, outages are happening all the time, caused by a wide variety of reasons— cable cuts, misconfigured routers, DDoS

attacks, power outages, even earthquakes and other natural disasters;
- Inefficient communications protocols: Although it was designed for reliability and congestion-avoidance, TCP carries significant overhead and can have suboptimal performance for links with high latency or packet loss, both of which are common across the wide-area Internet. The effect of distance affects the throughput and download time and causes the network latency or round trip time (RTT) to increase from milliseconds in local applications to hundreds of milliseconds in multi-continent telecommunications (Table I).

TABLE I. EFFECT OF DISTANCE ON THROUGHPUT AND DOWNLOAD TIME

Distance (Server to User)	Network RTT	Typical Packet Loss	Throughput
Local: < 100 mi.	1.6 ms	0.6%	44 Mbps
Regional: 500–1,000 mi	16 ms	0.7%	4 Mbps
Cross-continent: ~3,000 mi	48 ms	1.0%	1 Mbps
Multi-continent: ~6,000 mi	96 ms	1.4%	0.4 Mbps

In our previous paper we described the distributed storage structure based on LH*RS schema optimized for imagery data [6,7]. In this paper we introduce the technique that reduces resource consumption for static imagery data storage. Applying this technique for CDNs could significantly reduce traffic and optimize the performance.

II. DISTRIBUTED STORAGE SCHEMA

A. LH*RS concepts

LH*RS is a high-availability scalable distributed data structure (SDDS) [7]. The similar concept of Alexander G. Tormasov is invented in project of TorFS filesystem [10]. TorFS is fault-tolerant distributed storage method and controller using (N,K) schema. An LH*RS file is hash partitioned over the distributed RAM of a multicomputer, e.g., a network of PCs, and supports the unavailability of any $k \geq 1$ of its server nodes. A *k*-availability scheme preserves the availability of all records despite up to *k* bucket failures (Fig.2).

Advantages of LH*RS:

- The parity storage overhead is about the lowest possible.
- LH*RS is positioned as the general-purpose method.
- Efficiency does not depend on the data file type.

As file grows, 1-availability or even *k*-availability for any static *k* is not sufficient to prevent the reliability decrease. So *k* is being dynamically adjusted. The result is scalable availability schemes such as LH*RS [7,8]. It retains the LH* generic efficiency of searches and scans. Each record belongs to one group only, but with *k* or *k+1* (generalized) parity records. This provides the (scalable) *k*-availability to the file. The parity calculus uses the Reed Solomon Codes (RS-codes). A LH*RS file consists of an LH*-file called data file with the data records generated by the application in data buckets *0,1...M-1*. The LH* data file is augmented for the high-availability with parity buckets to store the parity records at separate servers. LH*RS coordinator initiates the creation of parity buckets for new groups and the scaling up of the availability. It also manages the record and bucket recovery.

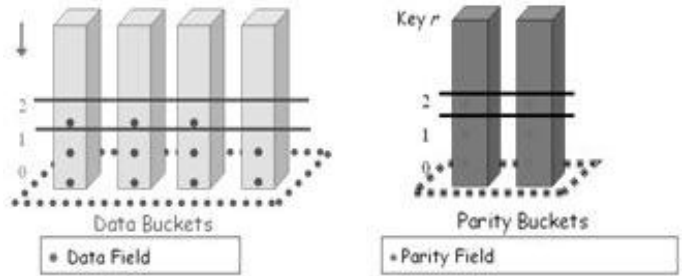

Fig. 2. LH*RS bucket structure

The LH*RS is positioned as the general-purpose method, and its efficiency does not depend on the data file type. We can analyse the way to optimize its performance characteristics, in particular for the case of imagery data.

B. Imagery Data Decomposition and Distribution among Storage Nodes

The basic idea of multilevel block structure is to construct independent blocks from correlated imagery data for each level [6]. We can consider the uncompressed image as multidimensional array. This 2-D (or 3-D) array represents cells filled with the vectors (color components, e.g. 24-bit RGB). These vectors are correlated. Usually the closer are cells, the stronger is the relation. We can apply filter to this array to mark cells belonging to different levels of pyramid. Basic blocks correspond to the top of pyramid and we can assemble the image in lowest resolution of them. Values for basic blocks data are aggregated or interpolated by the color values of neighborhood area. Next level blocks use the basic blocks data as the initial values, which forms the palette. We can form image of some quality using blocks of both basic and first levels. We can continue assembling the pyramid until we reach the original resolution of the image file, but image data is not considered as the redundant multiresolution stacked representation of the image. Data-driven algorithm of imagery data decomposition and distribution makes it possible to acquire the imagery resources this way: ** - parameters are included in request and user receives only necessary data to assemble image.

For example, we have the imagery data containing RGB color components. Each pixel contains three independent values - *R, G, B*, so we can store array of *R* values in one block, *G* values in other, *B* values in third and store them in different storage nodes. Data in blocks of any level *i* depends on data in blocks of lower level *i-1*. This can be performed in different ways and the simplest way is shown on Fig.3.

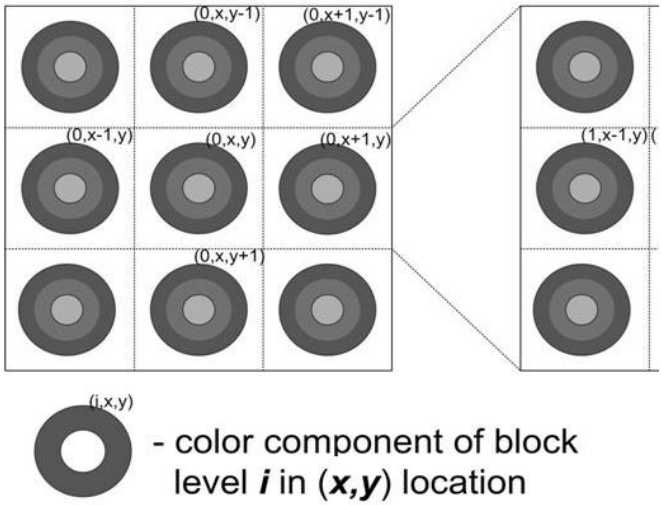

Fig. 3. Multilevel block structure

The lowest (basic) levels contain color values which can be calculated as average of the corresponding subsets of higher level data (e.g.: *R(0,x+1,y)* is an average value of subset *{R(1,x-1,y-1),R(1,x-1,y), R(1,x-1,y+1), ... R(1,x,y) ...R(1,x+1,y+1)}*). After the block assembling we get the multilevel array structure which corresponds to the entire original data. Using this structure we can serve any request with different data quality demands, constructing the answer from several levels of blocks instead of sending the entire (original) array, without processor resources consumption. For example, if client is the PDA application with QVGA display of 64K colors, connected by GPRS, it can request only necessary image blocks and construct desired low-quality image, and another client, residing on PC can require the full-quality image using all blocks; all of these clients work with the same physical distributed storage URL.

Moreover, we can produce additional levels, containing the extended values for imagery data. For example, we are processing the RAW image from camera or PNG file. Color depth of these images is better than of RGB schema, used in standard pictures and JPEG images. We can split the color component values to different blocks as shown on Fig.4.

Considering the image decomposed into blocks of three layers (Fig.5) we can describe the blocks (buckets) structure as following:

- Basic level blocks are shown under label (c). One block for each color component, block size is 4Kb. This layer corresponds with the thumbnail of original image;
- The intermediate level under (b) contains data calculated using K-means filter for compression purposes. It has the doubled dimensions of thumbnail data. Also we divide these arrays into 4 Tiles. We use the basic blocks data to initialize this level;
- Third level marked as (a) contains the remaining data which is necessary to build the double-sized image. Each of four tiles is decomposed into 4K Blocks of the 8-bit color components (R,G,B) and extension blocks for 32 or 64-bit color scheme. We use the previous level data as the initial.

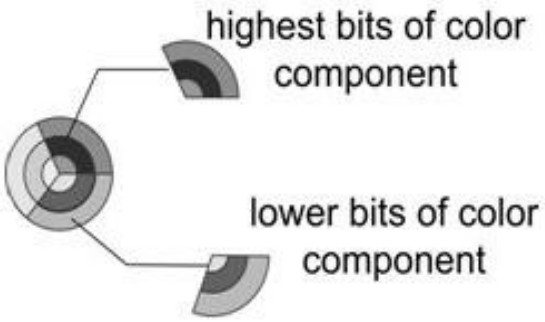

Fig. 4. Splitting the color channel into several parts

The higher level blocks are decomposed and tiled in accordance to this schema and we can complete the request by assembling image of any size and color depth using only the necessary blocks. Also we do not add the redundancy as the pyramiding does. Storage schema contains the sole copy of imagery data at highest resolution and scale suitable for all client's request.

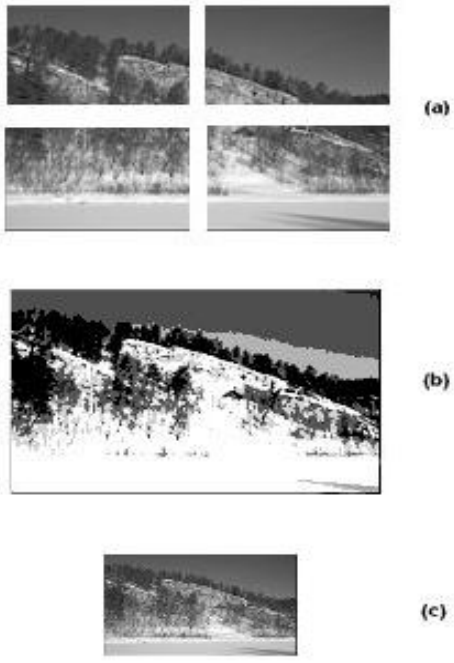

Fig. 5. Decomposed image

C. Fault-Protection

The high-availability management is based on Reed-Salomon erasure correcting coding. We can expand these correcting techniques with new capabilities: we can change the code redundancy among blocks of different levels and we can protect only the significant bits (digits) of imagery data. According to this, the basic level blocks are protected with

higher redundancy, because they store initial data for all the image assembling procedures. The higher is the level of the block, less significant is the content, so less redundant is the encoding. The highest level blocks can be even unprotected, or be protected using only significant digits.

The good example of codes having encoding and decoding algorithms with less computational complexity in comparison with reed-Solomon codes are the simple Hamming codes.

Several years ago we made a research on undetected error probability and erasure correction for the Hamming codes with d_{min}=3..4 and concluded that the erasure correcting ability of codes is not limited by Hamming border ($e=d_{min}-1$) but we can correct more erasures, up to 4,5 and even 6 erasures in one code group.

TABLE II. AMOUNT OF UNCORRECTED ERASURES FOR HAMMING CODE (18,12,4)

Erasure length	6	5	4
Simple decoder ($e > d_{min}-1$)	18564	8568	3060
Exhaustive correction algorithm	**12292**	**2072**	**148**

These results are compatible with those of reed-Solomon codes of same size and this means that we can use the simplified codes instead of sophisticated and efficiently use the computational resources of storage center.

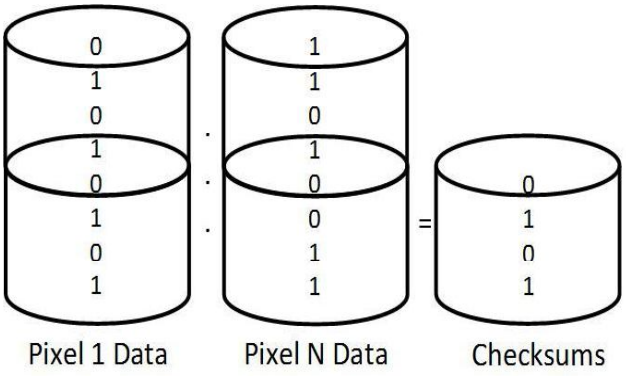

Fig. 6. Significant bits encoding schema

We can change the code redundancy among blocks of different levels and we can protect only the significant bits of imagery data as shown on Fig.6. This schema can be used for partial protection of high-level blocks if the partial loss of data is acceptable.

D. Compression

We can apply different methods of compression for blocks of different levels, for example, lossless image compression on the basic-level blocks, and lossy methods as JPEG2000-like compression for the higher levels (on Fig.7). This helps to dispense the logical redundancy but prevents from data loss on the basic levels.

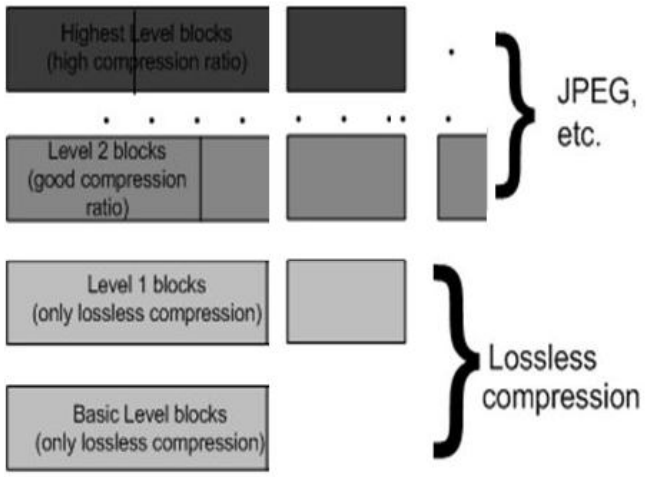

Fig. 7. Compression methods for blocks of different levels

E. Security permissions

We can apply different access permissions for different levels of blocks. The main benefit is that it is unnecessary to create different copies of images for each permission. For example, if we allow users of social network to see preview of the image, and the high-resolution image has restricted access, than we can encrypt the higher-level blocks and leave the basic ones opened. This can be used widely to apply efficient access control policy without storage redundancy in the similar way with "fine-grained access control" from the world of Database control systems: we do not protect the whole file object, but only the set of blocks.

III. DECOMPOSITION ALGORITHMS

Here we present the experimental results for the proposed approach. Currently this project is a "work in progress" and the results could be more accurate due to code optimization. We have considered wavelets for imagery data processing:

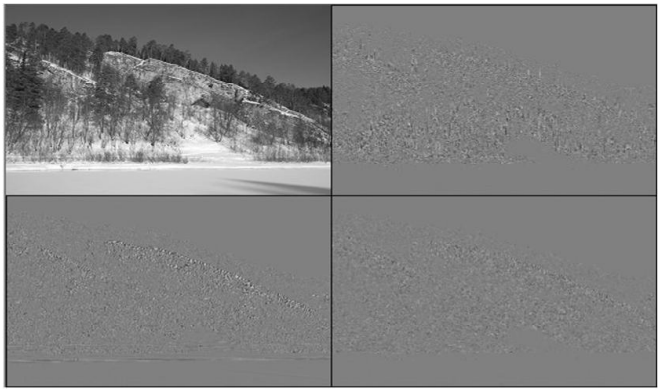

Fig. 8. Wavelet transform results

Wavelet Transform has several attractive properties that make it natural for representing signals and images in scalable colour image indexing methods [1]. Discrete wavelet transform (DWT) decomposes an image into a pyramidal

structure of subimages (subbands) with various resolutions corresponding to different scales. Output of the low-pass column DWT-filter is the thumbnail of the original image and is used for indexing. The derivation from this wavelet decomposition technique can be used for image decomposition and distribution.

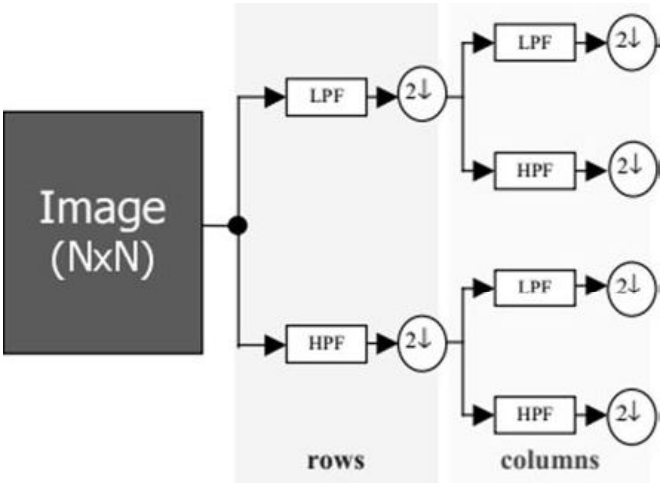

Fig. 9. 2-Dimensional pyramid decomposition

The main features of wavelet transform being used are:

- Multiresolution: Wavelet atoms compress and dilate to analyze at a nested set of scales (Fig.8,9). This allows the transform to match both short duration and long-duration signal structures [2];
- Compression: The wavelet transforms of real-world signals and images tend to be sparse. As a result, the wavelet coefficient distributions of real-world signals and images can be compressed.

The WT processing can be significantly accelerated in comparison to the usual 2-D transform by replacement of floating-point multiplications with shift and superposition operations [4,5]. In addition all four reconstructed data sequences (LL-, LH-, HL-, HH-filers of DWT) are processed simultaneously. We designed the DWT application for LH*RS storage for multilevel block structure and it produces the output as follows (Fig.10):

Original image (RGB24)	42 099 570
WT Haar compressed + gzip + EXIF	3 777 182
WT Haar: expanding to 2^x size, 3 WT levels + gzip each level, 4K block division + EXIF	3 801 203

Fig. 10. Compression of the original image with Wavelet transform in multilevel block structure.

The sum of the block sizes (second row) is slightly greater than the compressed file size (first row), but this traffic overload is leveraged for the parameterized users requests (Fig.11).

	LH*RS	LH*RS*WT
Full image request	3 777 182	3 801 203
320*240 RGB24	3 777 182	**189 300**

Fig. 11. Traffic amounts for simple LH*RS schema and for multilevel block structure serving different request types.

We have used two request types: request for full image and for image of specified quality and size (parameterized request). Original storage system should always send all the file blocks, even if we need the thumbnail image, and the LH*RS descendant approach, being developed by author is data-driven, and saves traffic.

IV. CDN MODEL AND SIMULATION

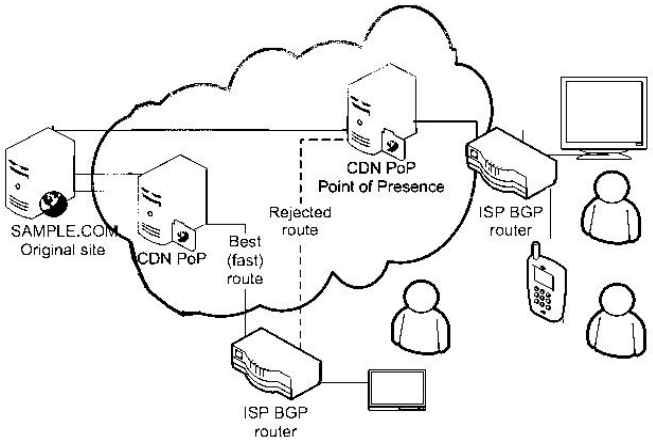

Fig. 12. Simplified CDN structure used for modeling.

CDNs are used to solve such problems as reducing latency (the amount of time it takes for the host to deliver the requested page resources) improving global availability and reducing bandwidth. This network structure can be used as the base for LH*RS distributed image storage.

The static imagery resources of the original page are cached with a certain probability p_k at points of presence, but not as the entire image sets (Image Pyramid for each original image), but as the buckets of original image. According to our concept and to CDN caching algorithm these buckets are cached in accordance to their relevance among users and the bucket sets could contain the entire image or only low-resolution image. For example if all the web-page users requested only previews and thumbnails of corresponding image, the buckets set will be enough to build the preview image without redirecting requests to the original site (S_i); if any user requested the image with the original quality and resolution, the rest buckets will be cached on P_i - Points of Presence (PoP) (see Fig.12).

This figure demonstrates that the whole set of blocks is cached only after the corresponding requests (user Client 1D requested the Hi-Res image (Fig.13,a) and Client 2D requested only low-quality image (Fig.13,b)).

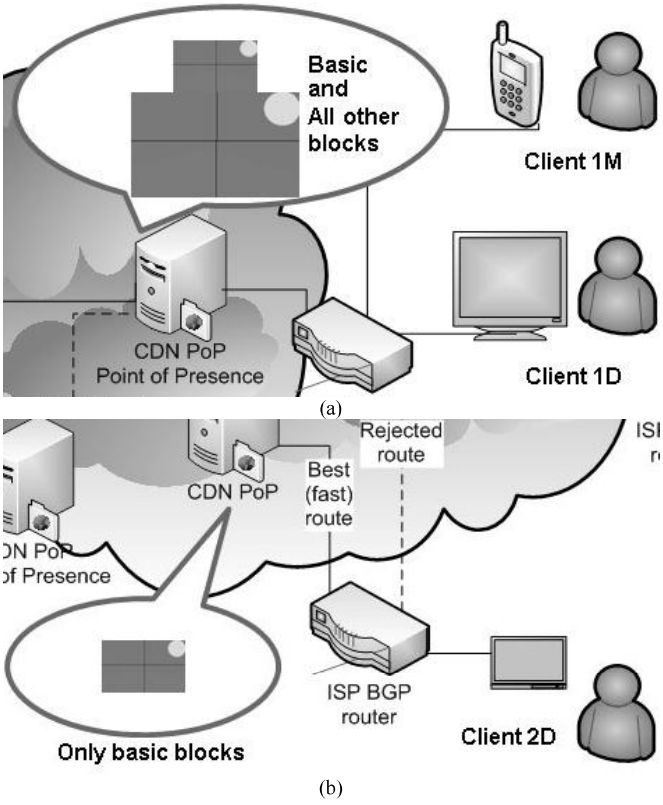

Fig. 13. Image blocks, cached at different CDN Points of presence according to user requests.

We tried to consider this system as queuing system M/M/1, simulating the single image downloading process. Each image contains *1..S* of buckets. Buckets are distributed among PoPs with probability p_i. Latency of server S is t_S, latency of P is t_P. Probability of request for bucket j is d_j. Intensity of the load for each PoP is estimated as

$$\lambda_{Pi} = p_i \cdot \lambda_{\Sigma} \cdot p_K. \quad (1)$$

Intensity of the load for each S is estimated as

$$\lambda_{Si} = d_i \cdot (1 - p_K) \cdot \lambda_{\Sigma}. \quad (2)$$

According to Little's equation

$$T_{SMO} = t/(1 - \lambda \cdot t). \quad (3)$$

And total latency T_{cp} equation is:

$$T_{cp} = p_K \cdot \sum_{i=1}^{P} (t_{P-K} + T_{Pi}) \cdot p_i + (1 - p_K) \cdot \sum_{i=1}^{S} (t_{P-K} + T_{Pi} + t_{S-P} + T_{Si}) \cdot d_i. \quad (4)$$

Varying the probability p_k we obtain the following results (Fig.13): the higher is p_k, the lower is latency. The plot (b) corresponds to T_P=0.002s and T_S=0.02s. The plot (c) corresponds to T_P=0.02s and T_S=0.02s and we can notice the almost linear latency decrease when the CDN is used.

V. Conclusion and Future Work

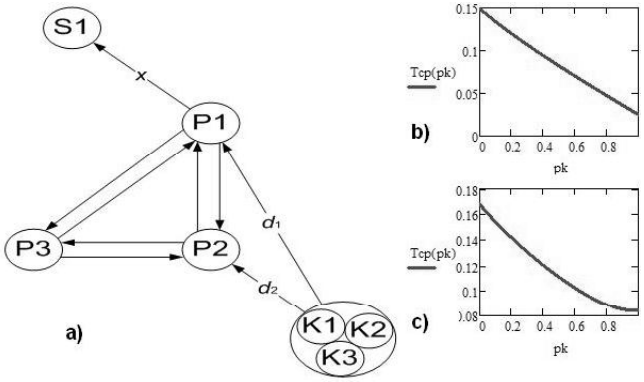

Fig. 14. CDN Markov's model (a) and simulation results (b,c).

A new result of the paper is suggested approach to build distributed fault-tolerant storage system of imagery data for online resources. Developed storage schema contains the sole copy of imagery data at highest resolution and scale. Image is decomposed into data blocks of several levels. The image with the required scale and resolution is reconstructed from the corresponding set of downloaded blocks on client's side. The developed method allows adopting the encoding and compression algorithms usage and algorithm redundancy on the block level. Thereby we can flexibly manage the redundancy, security permissions and resource consumption.

Our future direction is implementation of the proposed ideas in multimedia (audio and video) streaming systems in social networks and CDN. The main idea of this application is to share single multimedia stream between users with different quality and resolution requests to reduce traffic.

References

[1] Elif Albuz, Erturk Dogan Kocalar, Ashfaq A. Khokhar: Scalable Color Image Indexing and Retrieval Using Vector Wavelets. IEEE Trans. Knowl. Data Eng. 13(5): 851-861 (2001)

[2] L. Demaret, M. Fedrigo, F. Friedrich, H. Führ. Wedgelet Partitions and Image Processing. Workshop on Algorithms in Complex Systems EURANDOM, Eindhoven, The Netherlands, 2007

[3] W. Gansterer, G. Niederbrucker, H. Strakova, S. Schulze-Grotthoff. Scalable and Fault Tolerant Orthogonalization Based on Randomized Distributed Data Aggregation. Journal of Computational Science, 4 (6). pp. 480-488 ISSN 1877-7503 (In Press) (2013)

[4] Holschneider, M. A real-time algorithm for signal analysis with help of the wavelet transform / M. Holschneider [and others] // Wavelets, Time-Frequency Methods and Phase Space, Chapter A. - Berlin: Springer-Verlag, 1989. P. 289-297.

[5] HyungJun Kim. A parallel algorithm for the biorthogonal wavelet transform without multiplication. CIS'04 Proceedings of the First international conference on Computational and Information Science pp. 207-212, Springer-Verlag Berlin, Heidelberg ©2004

[6] Kokoulin A. Methods for large image distributed processing and storage. EUROCON 2013, pp.1606-1610. DOI: 10.1109/EUROCON.2013.6625191

[7] Witold Litwin, Rim Moussa, Thomas J. E. Schwarz. LH*RS: a highly available distributed data storage. [Online] Available: http://portal.acm.org/citation.cfm?id=1316816.

[8] Witold Litwin, Hanafi Yakouben, Thomas Shwarz LH* RS: A Scalable Distributed Data Structure for P2P Environment ACM-TODS, Sept 2005.

[9] D. Stanfill, Using Image Pyramids for the Visualization of Large Terrain Data Sets, International Journal of Imaging Systems and Technology, vol. 3, 1991.

[10] Patent № 6,961,868 The USA. Fault tolerant storage system and method using a network ofservers. Tormasov A., Khassine M., Beloussov S., Protassov S. ; 2005.

[11] G. Pavlou, N. Wang, W.K. Chai, I. Psaras, Internet-Scale Content Mediation in Information-Centric Networks, invited paper in the Annals of Telecommunications, special issue on Networked Digital Media, Vol. 68, No. 3-4, pp. 153-165, Springer, April 2012

[12] E.Nygren, R. Sitarman, J.Sun, The Akamai Network: A Platform for High-Performance Internet Applications. IEEE INFOCOM 2003, Apr. 2003

[13] A.Kokoulin, S. Dadenkov, Distributed storage system for imagery data in online social networks. Application of Information and Communication Technologies (AICT), 2015 9th International Conference on Year: 2015 Pages: 17 - 21, DOI: 10.1109/ICAICT.2015.7338507