

## Redundancy Management in Very Large Datasets

Sorin-Lucian PAVEL

Dauphine University, Paris / Academy of Economic Studies, Bucharest  
pavelsorin@gmail.com

**Abstract:** *Very large data sets (VLDS) are defined and it is illustrated their use in distributed collaborative systems. VLDS quality characteristics are described. It defines data redundancy and shows its effect on quality characteristics. Redundancy management is dealt with from two perspectives: the required minimum redundancy and the maximum accepted redundancy. Quantitative expressions are given and redundancy intervals are set.*

**Keywords:** *datasets, redundancy, quality, replication, distributed collaborative systems.*

### 1. VLDS and collaborative distributed systems

The modern collaborative systems are facing significant transformations of principles, structure and content with the transition to "cloud computing." This transformation addresses two aspects that characterize the entire behavior of the systems:

- adoption of both data and processes distribution using the communication facilities and the quasi-permanent presence of the computer networks; giving up the centralized treatment of data (acquisition, processing, storage) and moving to distributed systems, that appear to user as a single system called "cloud"; inside it, there are actually multiple systems that collaborate sharing physical and logical resources, for requirements resolution;
- treatment of very large collections of data using common resources that exist in a distributed system; since the systems are accessible to users, the input and circulating data grow exponentially, reaching very large data collections ( $10^7 \div 10^{10}$  datasets).

The importance of these two aspects makes the combined treatment of them essential, meaning the scientific research of distributed collaborative systems that work with very large data sets (VLDS).

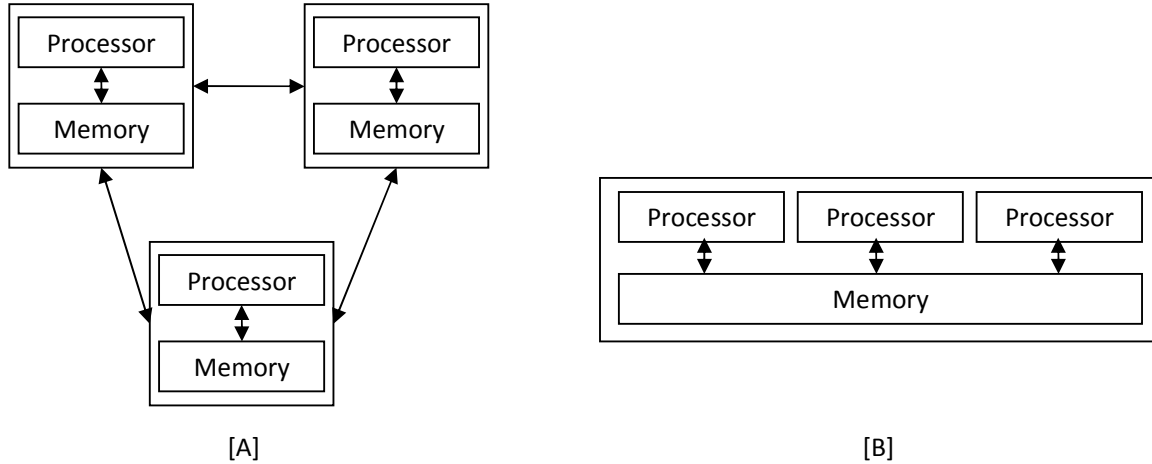
Throughout time, distributed systems have been defined in several, more or less scientific ways such as [10]:

- "you know you are using a distributed system when failure of a computer which you did not even know that existed retain you from doing your job" [8];
- a collection of computers that do not share common clock or common physical memory but which communicates through messages sent over a network; computers have their own memory and operating systems and work to solve a common problem [14];
- a collection of computers that appears to users as a single coherent system [15];
- a wide range of computers, from poorly connected systems, such as very large networks, to strong connected systems such as local area networks, and to strongly coupled systems such as multiprocessor systems [6];

Since there are many definitions of a distributed system [5] some properties are widely encountered:

- multiple autonomous processing entities, each with its own local memory [1], [3], [5], [12], [7], [16]; the system consists of several sequential processes that are independent and have their own resources; processes must have different address spaces for the

system to be able to be called "distributed"; multiple processing units with shared memory are not distributed systems, but parallel systems (Figure 1);



**Fig. 1.** Difference between collaborative distributed systems [A] and parallel systems [B]

- communication via messages between entities [1], [5], [12], [7]; processes communicate with each other through messages which arrive in a finite time; the order of messages depends strictly on physical characteristics of the communication channels;
- common purpose of processes [7], [12]; processes must interact with each other to achieve a goal; if two processes  $P$  and  $Q$  are considered in a network of processes,  $P$  calculates  $f(x) = x^2$  for some values of  $x$  and  $Q$  a set of values multiplied by  $\pi$  ( $\pi$ ); the two processes do not form a distributed system, since there is no interaction between  $P$  and  $Q$ ; if the two processes cooperate to calculate areas of circles of radius  $x$ , then  $P$  and  $Q$  is a good example of distributed system.

Collaborative systems choose distribution because they are often required to handle very large data collections. The cost of processing them in uniform (centralized) mode is very high and it includes important risks compared with the costs and risks of the distributed approach. Large data collections cover, but are not limited to: databases, collections of text files/xml files/multimedia files, data warehouses, or any combination thereof. Their management requires specialized tools to harmonize the specific hardware and software aspects of large data sets.

Examples of collaborative distributed systems and applications that work with VLDS include [1], [5], [12]:

- telecommunications networks:
  - o fixed and mobile networks;
  - o computer networks (Internet);
  - o networks of wireless devices;
  - o routing algorithms and applications;
- networking applications:
  - o www and peer-to-peer networks;
  - o communities of virtual reality and online games in multiplayer mode;
  - o distributed databases and distributed database management systems;
  - o file systems;
  - o distributed processing systems (banking, airlines);
- real-time control of processing:
  - o traffic control systems;

- industrial control systems;
- parallel processing:
  - scientific processes (cluster computing, grid computing)
  - 3D graphics.

The best example of distributed collaborative system working with VLDS is *Seti@Home* Project, initiated by the Berkeley University of California. This project embedded in distributed processing not less than 5.2 million computers worldwide, who voluntarily took part in the process. The computing power of this virtual supercomputer is 769 teraFLOPS using 2-300.000 available computers at a time [13]. It is the most widely spread distributed project and it aims to use computational power of computers in standby to analyze radio signals collected by a telescope in search of alien life forms.

## 2. Redundancy and the quality characteristics

In order for VLDS from distributed collaborative systems to generate results that are used in decision making and requirement solving, the data must meet certain conditions known as quality characteristics [11].

VLDS quality characteristics are measurable indices that describe the status of datasets or data collection and express their suitability for inclusion in operating processes. Each feature is aimed at a specific state aspect of VLDS. The software quality characteristics system includes:

- *completeness* as number of elements and number of describing characteristics; in quantitative terms, the dataset must include all its components and must capture all of the description characteristics, so there are no blank or null elements;
- *accuracy* of value; sets must register content in accordance with reality; testing methods for checking correctness involves both data acquisition verification and cross validation of the recorded values;
- *homogeneity* both in terms of structure – the data set form, and value - the dataset content; homogeneity is important for determining other quality characteristics;
- *comparability* so that datasets are ready for mutual analysis and processing; the datasets comparison is only acceptable in terms of homogeneity, because in certain situations a number of factors affect the evolution of characteristics, making them incomparable.
- *availability* requires that whether or not the nodes are connected and the resources are active, VLDS are present in the system for processing; availability is affected by the dynamic nature of nodes which are part of the collaborative in the sense that connecting/ disconnecting them doesn't influence the processing system but determine the presence/absence of data residing on that nodes;
- *reliability* of data collection requires that data should not contain errors or morphological syntactic nature which may cause system failure;
- *maintainability* characterize the probability of a wrong data set to be restored to specified conditions within a timeframe in which maintenance is performed according to certain procedures; maintainability measures the ability to isolate and fix an error in a dataset in a given time.

The VLDS quality characteristics are influenced by the behavior of the distributed collaborative system to which they belong [11]. The way it approaches the acquisition, storage and maintenance of data processing has a major impact.

Data redundancy in VLDS represents a way to manage the level of certain quality characteristics. Data redundancy consists of storing the same data in multiple instances at the same node or at different nodes in the system.

The influence of redundancy on the quality of large data collections should be viewed from two seemingly opposing perspectives:

- positive outlook – redundancy improves VLDS availability because if one system node becomes unavailable, the copied data from the system is still available for processing, or the parts available are used to reconstruct the original data;
- negative outlook: redundancy affects data processing results because the same data is taken several times in the calculation of indices, indicators etc.; such results are inconsistent with reality.

The way in which a high level of redundancy influences other quality characteristics is described in Table 1.

**Table 1.** Influence of redundancy on the quality characteristics

Quality index	Redundancy		
	Positive influence	Negative influence	No influence
Completeness	*		
Accuracy			*
Homogeneity			*
Comparability		*	
Availability	*		
Reliability			*
Maintainability		*	

Considering both positive and negative influences that redundancy has on distributed collaborative system working with VLDS, its level should be managed by setting the levels of two limits:

- minimum limit of redundancy (MiR) that describes how much redundancy must be created for maximum availability of the VLDS in the distributed system;
- maximum limit of redundancy (MaR) that describes how much redundancy should allow for not having data sets that negatively influence the outcome of VLDS processors.

These two limit sizes are analyzed, measured and applied in the present research.

### 3. The minimum required redundancy

Redundancy is added to the systems in order to have the data built even when parts of them are not available (are on unavailable nodes). Redundancy introduced from the beginning to data sets handles temporary disconnections, because even if some nodes are not connected, the data available on offline nodes is enough to rebuild the legacy files. There are situations where some nodes are permanently disconnected which leads to permanent loss of data and to decrease redundancy. For these cases it is necessary to perform maintenance and data repair.

There are many different techniques to increase data redundancy, also known redundancy schemes. Each plan determines:

- how to create data redundancy;
- how to rebuild the datasets when data is lost.

The most common redundancy schemes are: replication and erasure coding [2], [4].

Analysis of a redundancy scheme consists in evaluating the reliability of data storage along with the associated costs. Ability or success of a redundancy scheme consists in the ease with which a stored object is reconstructed in the event of data loss. However, this measure is not absolute but is conditioned by the behavior of nodes: one of the most important factors is the probability of concurrent failures. Therefore, measuring the reliability of a redundancy scheme is to determine the number of concurrent losses that occur without compromising the ability of reconstructing the original data.

Redundancy cost is the amount of storage space that is consumed. The redundancy factor ( $RF$ ) is defined as ratio of the data storage space with redundant data and the original size.

$$RF = \frac{data_{red}}{data}$$

where:

$data_{red}$  – space occupied by VLDS with redundancy;

$data$  – original dimension of VLDS.

During the whole lifetime of the distributed storage system, there are many disconnections. Whenever there is a disconnection, parts of redundant data are lost which increases the chances that the original data is lost. Maintenance consists of remanufacturing redundant data in their time of loss. The operation takes place by reading the available data and create new ones. To assess the contributions of redundancy scheme, the amount of data read is measured in relation to the amount of redundant data created. The degree of repair measures how many bits should therefore be read to create a new redundant data bit.

$$DR = \frac{data_{read}}{data_{created}}$$

where:

$data_{read}$  - data read for the construction of new data (bits);

$data_{creat}$  - data created after the reading (bits).

These two metrics are applied to the two redundancy schemes: replication and erasure coding.

*Replication* is the simplest method of adding redundancy. Basic version is to create multiple copies on different nodes of a stored object. If it is considered that for an original object  $x$  are stored  $N_{rep}^x$  copies on different nodes, number of losses that the system supports is  $N_{rep}^x - 1$ . The probability of losing an object on the condition of concurrent loss probability is:

$$P(loss | c) = \begin{cases} 0, & c < N_{rep}^x \\ 1, & c = N_{rep}^x \end{cases}$$

The redundancy factor is:

$$RF = N_{rep}^x$$

The repair degree is:

$$RD=1$$

because the reconstruction of a single element corresponds to the production of a single copy of the object.

For *erasure coding*, method proceeds as follows: the object (file) to be stored is divided into  $k$  parts. These are processed to produce  $k+h$  parity fragments so that any  $k$  fragments are sufficient to reconstruct the original fragments. The number of losses that the scheme supports is  $h$ .

The probability of losing an object conditioned by the probability of concurrent loss is:

$$P(\text{loss} | c) = \begin{cases} 0, & c < h \\ 1, & h < c \leq k + h \end{cases}$$

The redundancy factor is:

$$RF = \frac{k + h}{k}$$

which corresponds with a higher storage capacity and efficiency. For a replication scheme with  $N_{rep}^x=3$  and an erasure coding scheme with  $k=3$  and  $h=2$ , in both cases the system supports up to two losses without losing the original data. Replication consumes storage space suitable for three original objects, while erasure coding consumes only 5/3 of space.

The degree of repair is

$$DR=k$$

Replication is thus considered as a special case of erasure coding in which  $k=1$  and  $h=N_{rep}^x-1$ .

The analyzed distributed system consists of a number of  $N_n$  nodes which create a collaborative system for storing a quantity of data with the original size (before applying the redundancy scheme) of  $|data|$ . The data consist of  $N_f$  files. Each file – whose size is denoted by  $|file|$  is administered separately and when the need for reconstitution is detected, a new parity fragment is rebuilt using existing data. In this research are not taken into account exogenous factors: the organization of nodes, management structure, the existing infrastructure etc.

Each node is connected to the network through a link that enables bandwidth  $lb$ . Unlimited download capacity is considered (because the nodes are connected in asymmetric networks, the download speed is faster than upload). Using the same approach as in [4] is considered a system in which the length of life of the nodes is infinite which means that data may not be lost. Thus durability is not a problem and availability is assured by implementing an optimal redundancy scheme. The objective is to store for each file a number of  $k+h$  parity fragments so as at any time  $k$  fragments are available. Also it is assumed that the average activity time of a node and the average time of inactivity follow an exponential distribution. It also defines the percentage of activity, which describes how much of node's life is spent online.

$$PA = \frac{T_a}{T_a + T_i}$$

where:

$T_a$  – time of activity (the node is present in system);

$T_i$  – time of inactivity (the node is absent from the system).

Designed in such a way, the model corresponds to continuous time Markov chain from which can be analytically derived the probability  $\pi_r$ , in which  $r$  nodes out of  $k+h$  are online (available), and the other  $k+h-r$  are offline (unavailable) depending on  $k$ ,  $RF$  and  $PA$ .

$$\pi_r = \begin{cases} \left[ 1 + \sum_{j=1}^n \prod_{i=0}^{j+1} \frac{RFk - i}{i + 1} \frac{PA}{1 - PA} \right]^{-1} & r = 0 \\ \pi_{r-1} \frac{RFk - r + 1}{r} \frac{PA}{1 - PA} & 0 < r \leq n \end{cases}$$

A file is considered available, meaning it can be reconstructed, when at least  $k$  nodes are online. The probability of this condition is denoted by  $a$  and is expressed as:

$$a = \sum_{r=k}^n \pi_r$$

This means that  $a$  depends on  $k$ ,  $RF$  and  $PA$ .

If  $\hat{a}$  is the desired availability, knowing  $k$  and  $PA$ , it is determined by finding the lowest  $RF$  for which:

$$a_{k,RF,PA} \geq \hat{a}$$

In [4] it is considered  $\hat{a}=99\%$ , which corresponds to  $a=0.99$  and the following values are obtained (Figure 2) function of  $k$  for different values of  $PA$ .

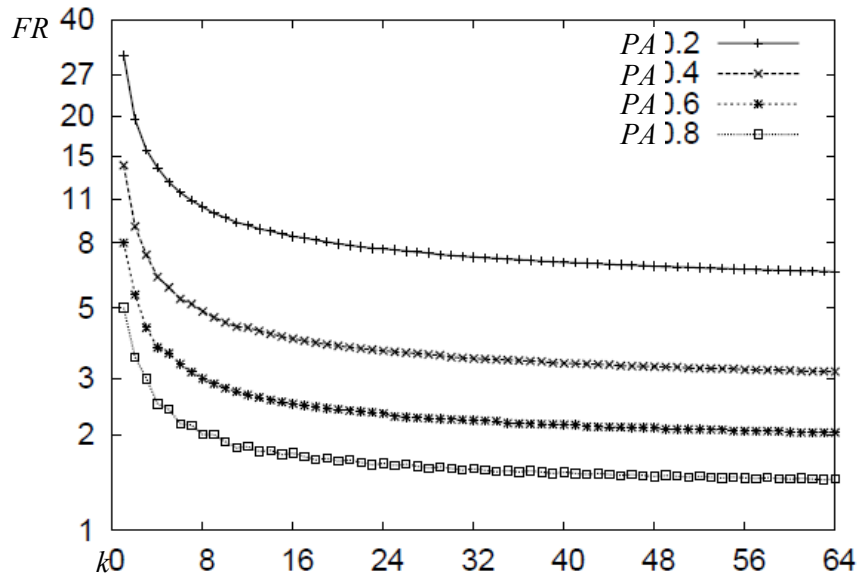


Fig. 2. Redundancy factor determined by PA and k [4]

Thus,  $RF$  redundancy factor is determined for a given level of availability as a function of  $PA$  and the number of fragments  $k$  dataset is divided. It sets a minimum limit of redundancy in a distributed collaborative system with VLDS for ensuring a certain level of availability and depending on the parameters of that system.

For the very large matrix management application are considered (arrays) present in a distributed system with  $M=62$  nodes. The sets are divided by a coefficient  $k=4$  (every matrix in symmetrical blocks) in a network where  $PA=0.6$ . Considering the availability of desired = 99%, the calculated value of  $RF$  is 4 which means that:

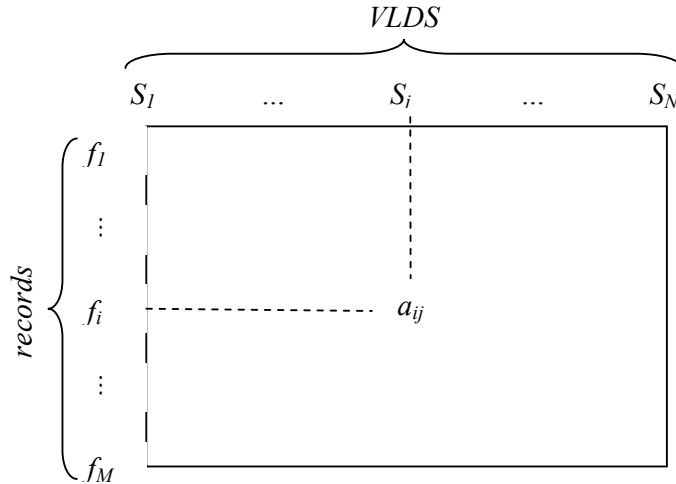
- if redundancy scheme based on replication is applied, 4 identical replicas of each matrix have to be created to ensure the desired availability;
- if redundancy scheme based on erasure coding is applied, 16 parity fragments will be created which will occupy only 6/4 of the space.

Given the results, the erasure coding schema is applied.

#### 4. The maximum accepted redundancy

Redundancy in the negative approach is developed during operation of the system and refers to the uncontrolled duplication of entered or stored data, which increases system costs, produces inconsistent results and increases system response time.

Uncontrolled redundancy should be minimized in all stages of the development and operation of the system. For each new set of data, the value-added information is verified and the extent to which the data set is already present or not in the database. Method of minimizing redundancy in fact refers to its avoidance before VLDS reach the collection. After the dataset is completed and before it is deployed, it is subject to tests to determine the extent to which dataset is in the same form found in the collection. For database systems, this is in some measure already executed by the management module, integrity constraints, etc. The method should be adopted for the collection of files so that the results are qualitative.



**Fig. 3.** The structure of VLDS

Assume data is organized as in [9] a matrix in which rows are records and columns are sets of data as in Figure 3.

A matrix structure  $A$  is associated for which:

$$a_{ij} = \begin{cases} 1 & \text{if the record } i \text{ is in the file } j \\ 0 & \text{otherwise} \end{cases}$$

with  $i=1,2,\dots,M$  and  $j=1,2,\dots,N$

The physical memory space is  $F$ , being an array of records:

$$F = \{f_1, f_2, \dots, f_M\}$$

On which it is defined the family:

$$\Phi = \{S_1, S_2, \dots, S_N\}$$

Of datasets  $S_j \in F$ , where  $S_j$  it's a dataset and

$$S_i \cap S_j \neq \emptyset$$

The content of the  $S_j$  dataset is described by the column  $j$  of matrix  $A$ .

Let  $c_j$  be the number of records of the  $S_j$  dataset, also known as the length of the set.

Because of the sequential access to data, the search time is determined by the length of the dataset.

Let  $Q'$  be a query for  $\alpha$  units of data. The query identifies a subset of  $F$  noted  $F'$ . From matrix  $A$  a family of sets is determined

$$F': \Phi' \subset \Phi, \text{ cu } \Phi' = \{S_{j_1}, S_{j_2}, \dots, S_{j_k}\}, \bigcup_i S_{j_i} = F'$$

But  $F'$  is covered by many families  $\Phi_1', \Phi_2', \dots, \Phi_n'$ . Each dataset family is an array in which the information from  $Q'$  should be looked for. From  $\Phi_1', \Phi_2', \dots, \Phi_n'$  a family  $\Phi_{min}$  needs to be extracted which minimizes the search time for the records in  $F$ . The problem that must be solved is that of the minimum coverage of the lines in the matrix  $A$  by its columns.

The solution should not be sought in the entire matrix  $A$ , but in sub-matrix defined by  $Q'$ . The matrix rows are separated corresponding to records contained in the query and the columns (data sets) containing such records are selected. Minimum coverage problem is solved by transforming the obtained matrix  $A'$  as the matrix  $A$  remains unchanged.

The transformation matrix  $A'$  consists of the following steps:

*i.* let  $a_i$  be row number  $i$  from matrix  $A$ , and  $v$  the unit column vector and  $v_j$  vector with the  $j$  component 1 and all others set to 0; if

$$a_i \otimes v = v_j, \text{ meaning } \exists f_i \in S_j, \text{ with } f_i \notin S_l, l \neq j,$$

then

$$S_j \in \Phi'_{\min} \text{ and } a_i \notin A';$$

*ii.* if  $\exists a_p$  and  $a_q$  and  $a_p \otimes a_q = a_p$  and  $a_q \oplus a_p = a_q$  where  $\otimes$  and  $\oplus$  are the logical operations of multiplication and addition, then  $f_q$  is pulled off from  $F$  and from  $A$  an  $A'$ , because  $\Phi'$  that covers  $f_p$  also covers  $f_q$ .

*iii.* if  $k$  rows from  $A$  simultaneous satisfy the conditions:

$$\sum_k a_{ik} = v$$

$$\prod_k a_{ik} = v_j$$

$$c_j \geq \sum_k c_k$$

then  $S_j$  is not included in  $A'$ ,  $S_j \notin \Phi'_{\min}$ .

The resultant matrix  $A'$  is generated by the query  $Q'$  and resolves the minimum cover for finding the family  $\Phi'_{\min}$ .

If  $Q' \in A'$  then the dataset is redundant for the given level of  $c_j$ . Thus sets the maximum level of accepted redundancy for VLDS to generate coherent and correct results.

## 5. Conclusions

Distributed collaborative systems are subject to changes that concern on one hand the manner of distribution and on the other hand the VLDS operation. The quality characteristics of the VLDS and of the systems that use them are affected (positively or negatively) by the level of redundancy present in the content of datasets. The two effects of redundancy should be kept in balance in order for the VLDS to optimize their quality.

The minimum level of required redundancy is calculated based on the availability of VLDS, desired number of dividing fragments and the nodes' time of activity. Maximum acceptable level of redundancy is determined depending on the VLDS content, their length and the importance of the fields. The level of redundancy must be maintained at the optimum value, and future research should investigate cases in which the maximum level allowed is less than the minimum required.

## References

- [1] G. R. Andrews, *Foundations of Multithreaded, Parallel, and Distributed Programming*, Addison-Wesley, 2000.
- [2] A. G. Dimakis, B. Godfrey, Y. Wu, M. J. Wainwright and K. Ramchandran, "Network – Coding for distributed storage systems," *Computer Research Repository (CoRR)*, 2008, arXiv:0803.0632v1, Available at: <http://arxiv.org/abs/0803.0632>.

- [3] S. Dolev, *Self-Stabilization*, MIT Press, 2000.
- [4] A. Duminuco, *Data Redundancy and Maintenance for Peer-to-Peer File Backup Systems*, Phd Thesis, University TELECOM ParisTech, Oct. 2009.
- [5] S. Ghosh, *Distributed Systems – An Algorithmic Approach*, Chapman & Hall/CRC, 2007.
- [6] A. Goscinski, *Distributed Operating Systems: The Logical Design*, Reading, MA, Addison-Wesley, 1991, pg. 942.
- [7] A. D. Kshemkalyani, M. Singhal, *Distributed Computing, Principles, Algorithms and Systems*, Cambridge University Press, 2008, pg. 756.
- [8] L. Lamport, *Distribution email*, May 28, 1987, Available at: [http://research.microsoft.com/users/lamport/pubs/distributed\\_systems.txt](http://research.microsoft.com/users/lamport/pubs/distributed_systems.txt).
- [9] L. M. Novozhiiova, „Minimal covering problem and redundancy reduction in databases,” *Journal of Mathematical Sciences*, Vol. 66, No. 3, September 1993.
- [10] S. Pavel, „Developing reliable distributed applications oriented on large datasets,” *Journal of Applied Business Information Systems*, Vol. 1, No. 1, December 2010, pg. 67-76.
- [11] I. Ivan, C. Ciurea, S. Pavel and M. Doinea, „Security of Collaborative Processes in Large Data Sets Applications,” *The 5th International Conference on Applied Statistics*, November 19-20, 2010, NIS Publishing House, Bucharest, Romania.
- [12] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, SIAM, 2000, 343 pg.
- [13] <http://setiathome.berkeley.edu/>, Accessed March 2011.
- [14] M. Singhal, N. Shivaratri, *Advanced Concepts in Operating Systems*, New York, McGraw Hill, 1994, pp. 448.
- [15] A. Tanenbaum, M. Van Steen, *Distributed Systems: Principles and Paradigms*, 2nd edition, Upper Saddle River, NJ, Prentice-Hall, 2006, pp.704.
- [16] D. Prakash Vidyarthi, B. Kumer Sarker, A. Kumar Tripathi, L. Tianruo Yang, *Scheduling in distributed computing systems, Analysis, Design and Models*, Springer Science-Business Media, 2009.

#### Author



**Sorin PAVEL** has graduated the Faculty of Economic Cybernetics, Statistics and Informatics from the Bucharest Academy of Economic Studies in 2008. He is currently following Master's in Software Project Management and the Doctoral School in Economic Informatics, both at the Academy of Economic Studies.