Optimization of HDF5 Performance for Virtual Reality Objects Enhanced by Implicit Features

Alexander Tzokev
Faculty of Industrial Technologies
Technical University of Sofia
Sofia, Bulgaria
alextz@tu-sofia.bg

Angel Bachvarov, Stoyan Maleshkov
Faculty of German Engineering Education and Industrial Management (FDIBA)
Technical University of Sofia
Sofia, Bulgaria
a_bachvarov@tu-sofia.bg, maleshkov@tu-sofia.bg

***Abstract*—The amount of data used in Virtual Reality environments can be very large especially when working on real-world engineering problems. The interaction between the environment and the user or enhancement the objects therein with implicit features require integration of a high performance data management system which allows an efficient data handling. The deployment of specialized scientific databases such as Hierarchical Data Format 5 (HDF5) offers certain advantages over more business-oriented relational database management systems. This paper presents results from a study for optimization of the storage efficiency of the HDF5 data base trough chunked datasets enabling effective handling of the large data amounts produced within and for virtual reality environments. Further, a method for predicting the optimal chunk size or arrays of native data types with rank 1 and 2 is discussed.**

Keywords – Virtual Reality; HDF5; Data Chunking; Chunk Size Prediction; Databases.

I. INTRODUCTION

Recently a concept of so called implicit features of the objects for enhancement of their immersive representation within multimodal virtual reality (VR) environments was proposed in [1]. The implicit features represent "hidden" properties of an object which normally are not part of the object model and cannot be perceived directly by the observer through his/her senses, such as magnetization, radiation, humidity, toxicity, surface roughness etc.

The VR paradigm assumes that the virtual objects are fully functional replicas of the physical originals. However, the commonly used approaches for building the virtual objects are related to some limitations in their presentation within the virtual world. Only the so called *explicit* features of the objects, meaning the (geometrical, structural and topological) characteristics which could be perceived directly by an observer are covered. Furthermore, the mechanism of integration the newly created objects in the VR environment imposes some additional restrictions due to the need for simplification of the data structure describing the model. All this leads to significant reduction of the informational content and the use of VR technologies for presentation of the objects does not give any substantial advantages in comparison to the much affordable conventional techniques, which depends mainly on the observer's visual channel. Normally, he/she can explore the object model trough a set of commands which modify its spatial position, size and attributes.

The new approach using the implicit features extends the information content and deepens the immersive representation of the object model through its enhancement with a set of already mentioned implicit features. An example for this is shown in Fig. 1 where beside the visually perceived attributes the observer can get additional information concerning the functional behavior and operating modes of the object. Depending on the specific needs the properties could be presented only visually or in combination with sonification, which means by aural and/or tactile channel for perceptualizing data. This approach extends the underlying general scene-graph structure of the VR software system incorporating additional effects nodes for implicit features representation. It is flexible and can be easily implemented on different hardware configurations on a single computer or on a computer cluster for immersive presentation. The use of implicit features not only enhances the informational content of the virtual objects, but also enables their exploration in different contexts enabling representation of different properties sets according, e.g. narrative, informative, marketing, technical, instructional etc.

The assignment (mapping) of the implicit features to the CAD model of the object and further manipulation of the relevant datasets are discussed in detail in [2]. In the general case, this is done using simulation input and configuration files from simulation solvers. The reason for this is the fact that the most of the researchers are still looking at VR mainly as a CAD/CAE post-processing tool only for viewing and assessing the design and simulations results in real time.

Within this context the deployment of a high performance parallel hierarchical data management and storing system, such as HDF5 [3] is significant precondition for successful implementation of the concept.

 ISBN: 978-1-61208-319-3

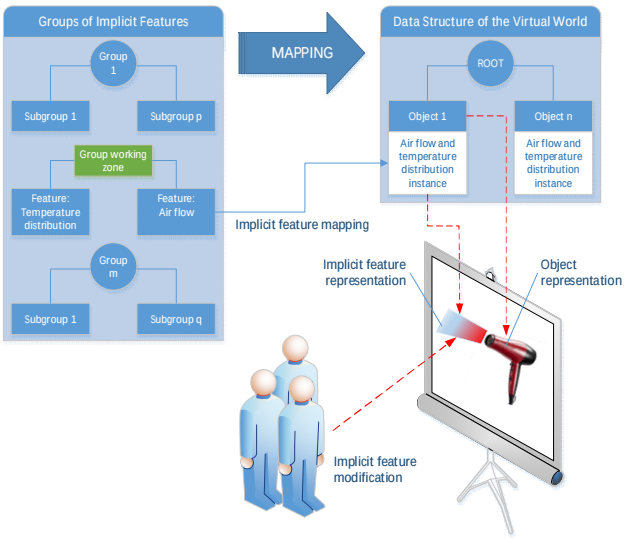

Figure 1. Implicit features mapping in VR

A preliminary study [4] analyzes the possibilities for using NoSQL database management systems (DBMSs) when storing and retreiving large ammount of scientific information in form of 2D arrays storing image infromation. The I/O performance of MySQL is compared to HDF5 and the results proofs that the HDF technology is faster in terms of reading and writing data when managing large datasets. In some cases the information query is faster with MySQL due to lack of builtin indexing in HDF5.

The computing power used at the scientific and engineering simulations and the generated ammount of data require a high degree of parallelism both within the software applications and at all levels of the underlying architectural platforms and storage systems [5]. The HDF model was design for efficient parallel usage and storage of very large multidimensional datasets of complex data types. To achieve optimal performance the HDF5 chunk and cache parameters must be fine-tuned and an algorithm for prediciting their values will be useful.

Further this paper contains overview of the HDF5, representing the software model and the implementation of the technology in VR objects enhanced by implicit features, a discussion of experiments, and results for the HDF5 performance with different chunk sizes is given. The next part of the paper presents a model based on the experimental data for predicting the optimal chunk size for 2D arrays of double data type. Finnaly, the conclusions and acknowedgments are given.

II. HDF5 TECHNOLOGY

In order to implement the mapping of the implicit features in VR as presented in Fig.1, a high performance database management system is required for managing a large amount of numerical data of different type (scalar values, multidimensional arrays, vectors, tensors, etc.). An easy and convenient way to define the object hierarchy is use of HDF5 data model or other type of specialized scientific hierarchical data files. Using relational DBMSs shall result in complex data and links descriptions and shall cause possible performance bottlenecks.

The HDF5 consists of a specialized data model, software libraries and a file format for data management, which supports different data types. Such a model is very flexible and efficient when working with high-volume and complex scientific datasets. An HDF5 file contains variables, arrays, groups and types which, based on the software model are known as Datasets, Group and Datatype. The model also defines simple and extending link mechanism for creating associations between information items in the file [6]. Fig. 2 shows the HDF5 models and their implementations between software objects [7].

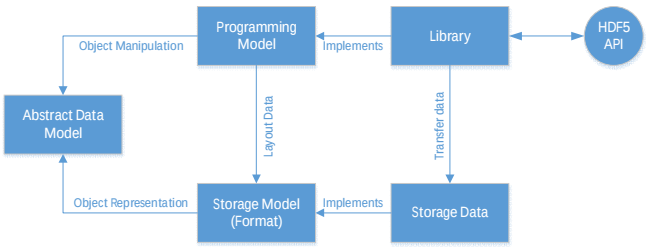

Figure 2. General HDF5 model and implementation

All these features determine the HDF5 as a favorable solution for data management system in VR environments.

The HDF5 datasets are array variables whose data elements are described and shaped as multidimensional arrays (up to rank of 32). A dataset can shrink and grow its limits up to predefined maximum extent. Depending on the storage layout strategy, this maximum extend can be virtually unlimited [6]. The latest version of the HDF model supports the following storage layouts:

- *Contiguous* – array elements are laid out as a single sequence in the file;
- *Chunked* – array elements are stored as a collection of sub-arrays with preliminary defined fixed-size, called chunks;
- *Compact* – used for small datasets with size less than 64 KB. All elements can be read as a part of variable's metadata or header retrieval.

The performance and some other capabilities of HDF5 depend mainly on the used storage layout. For partial data manipulation the so called hyperslab HDF5 object is required and the storage layout should be chunked.

An important task is to define the optimal chunk size and shape for each dataset, since this parameter is related to the overall I/O performance and as stated before to the additional features (i.e. hyperslabs, data compression, encryption, etc.). A general metric for the storage efficiency is the expected number of chunks retrieved by queries under access workload. In [8], Otoo et al. have presented mathematical models based on geometrical programming and steepest descent optimization for defining the chunk parameters.

 ISBN: 978-1-61208-319-3

The prediction of the optimal chunk size of HDF5 dataset based on the size and rank of the input array may improve the performance of the data storage system and shall lead to more fluent communication and workflow. Hence, one of the main purposes of this study is to define a model for adaptive prediction of the chunk size for HDF5 datasets based on the size and shape of the data.

Another possibility to improve the I/O performance of the HDF5 data set is to adapt the chunk cache size by its automatically resizing as needed for each operation. The cache should be able to detect when the cache should be skipped or when it needs to be enlarged based on the pattern of I/O operations. At a minimum, it should be able to detect when the cache would severely hurt performance for a single operation and disable the cache for that operation [7]. However, the chunk cache is relevant to the chunk size and defining the optimal size for the chunk will lead also to possible optimization of the cache parameters.

III. EXPERIMENTAL WORK

Four experiment series have been carried out under the same laboratory conditions (hardware, operating system, installed software modules and libraries and running processes):

1. Parsing input and storing data;
2. Sequential reading entire HDF5 dataset to memory object;
3. Reading partial information from the HDF5 dataset;
4. Partial modification of HDF5 dataset.

The experiment variables are described in Table 1.

TABLE I. EXPERIMENT VARIABLES

	Experiment Variables		
	Variable	***Description***	***Domain of possible values***
1.	Data type	Type of data stored in the array	integer or double
2.	Size of array	Number of elements in the array	**Rank 1:** 1 to 99 with step 1 and from 100 to 100100 with step of 500. **Rank 2**: 1 to 99 with step 1 and from 100 to 500 with step of 10.
3.	Rank of array	Rank of array with data	1 and 2
4.	Chunk size	Size of chunk	1 to 100
5.	Chunk shape	Rectangular or square	**Rectangular:** [1,x], where x=[1..100] **Square:** [x,x], where x=[1…100]

For testing purposes, a dedicated Linux application has been developed in C++ language and compiled as 64-bit executable with the latest available HDF5 static libraries (hdf-1.8.11). All tests were automated and with no user interaction.

The */usr/bin/time* Linux command was used to measure the execution time. This command runs the specified program with the given arguments. When the execution finishes, *time* writes a message to the standard output giving time statistics about this program run. These statistics consist of the elapsed real time between invocation and termination, the user CPU time (the sum of the *tms_utime* and *tms_cutime* values in a *struct tms* as returned by *times()*), and the system CPU time (the sum of the *tms_stime* and *tms_cstime* values in a *struct tms* as returned by *times()*) [9]. The study can be further extended with direct analysis of the IO operations performed by the HDF5, but this requires complex modification and new build of the HDF5 library. The input data for the tests represents results from a real-world mechanical and heat transfer simulations.

Fig. 3 shows the workflow for analyzing the test result data and observations.

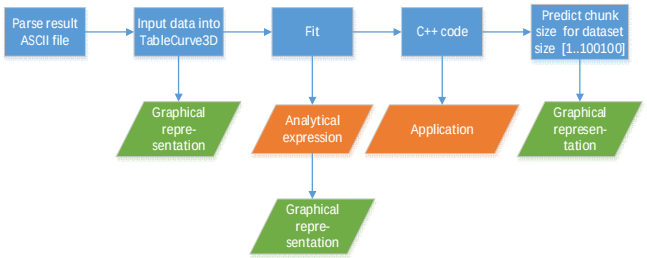

Figure 3. Analysis of the test result data

The two most important deliverables from the analysis are the analytical expression for predicting the optimal chunk size and the derived software application or libraries based on this model.

IV. DISCUSSION OF EXPERIMENTAL REULTS

A. One Dimensional Integer Array

Fig. 4 shows the results from automated writing test (data storage) with one dimensional array of integer data type with varying number of elements according to Table I.

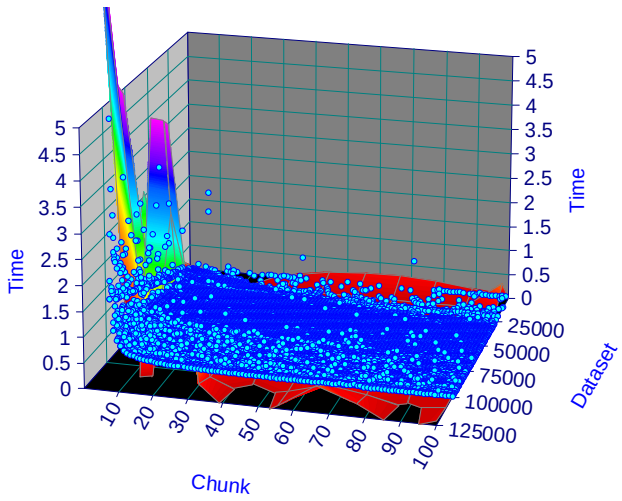

Figure 4. Results from writing one dimensional integer array (x – chunk size [number of elements], y – real time [s], z - size of dataset [number of records])

The resulting data represent a surface with higher values of analyzed parameter (real time) close to a small chunk size regarding the size of the dataset. This result is expected and the chunk size must be kept small in order to optimize the performance of the software modules and the cache that are used to store the data into HDF5 file. This will lead to smaller memory footprint but shall increase the number of I/O operations for large dataset or frequently modified data.

When using the HDF5 libraries there is a certain amount of overhead associated with finding chunks. When chunks are made smaller, there are more of them in the dataset. When performing I/O on a dataset, if there are many chunks in the selection, it will take extra time to look up each chunk. Moreover, since the chunks are stored independently, more chunks results in more I/O operations. The additional metadata necessary to locate the chunks also increases the size of the file as chunks are made smaller. In the most cases, making chunks larger shall result in fewer chunk lookups, smaller file size, and fewer I/O operations [6]. Here should be mentioned that the chunk size of 1 and the chunk size equal to the size of the dataset shall not be recommended by the HDF5 developers [7].

The distribution of the test data in Fig. 4 confirms the hypothesis that the chunk size have to be adaptive and the optimal value for the corresponding size of the dataset should be calculated before imitating the writing functions in the HDF5.

The results from the sequential read test for one dimensional integer array are presented in Fig. 5.

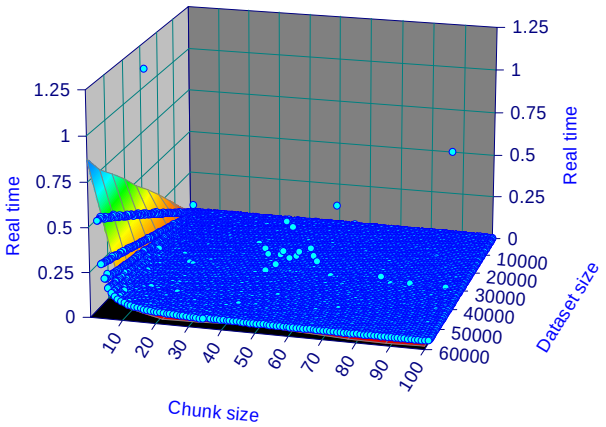

Figure 5. Results from sequentialy reading one dimensional integer array (x – chunk size [number of elements], y – real time [s], z - size of dataset [number of records])

Within the tests for partial reading (Fig. 6) and modifying (Fig. 7) 1% of the dataset is retrieved and altered. If the dataset size is less than 1, only one record is used.

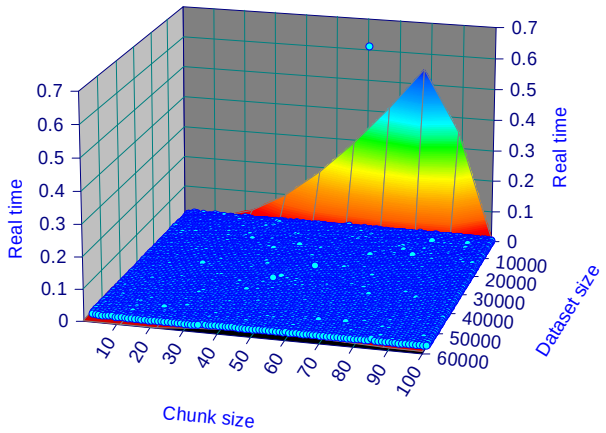

Figure 6. Results from partial reading of 1% of the size of one dimensional integer array (x – chunk size [number of elements], y – real time [s], z - size of dataset [number of records])

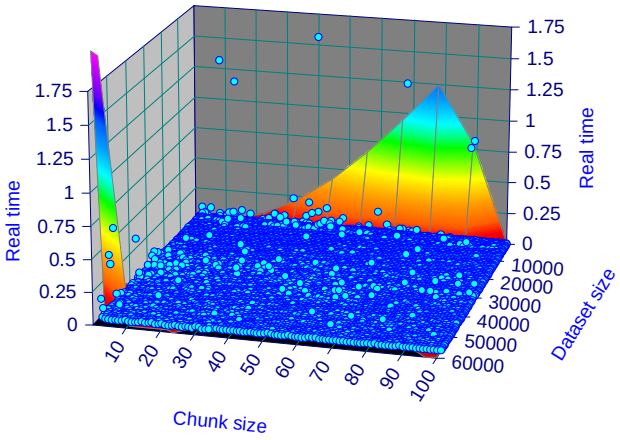

Figure 7. Results from modify test for one dimensional integer array (x – chunk size [number of elements], y – real time [s], z - size of dataset [number of records])

As can be seen from Fig. 4, Fig. 5, Fig. 6, and Fig. 7 the write test is the slowest test (in terms of real time parameter).

Another observation is that the chunk size has greater influence on the data storage than the reading and modifying tests.

B. Two dimensional double array

The same four experiments have been performed for two dimensional array of double data type. The results from the write test are shown in Fig. 8.

 ISBN: 978-1-61208-319-3

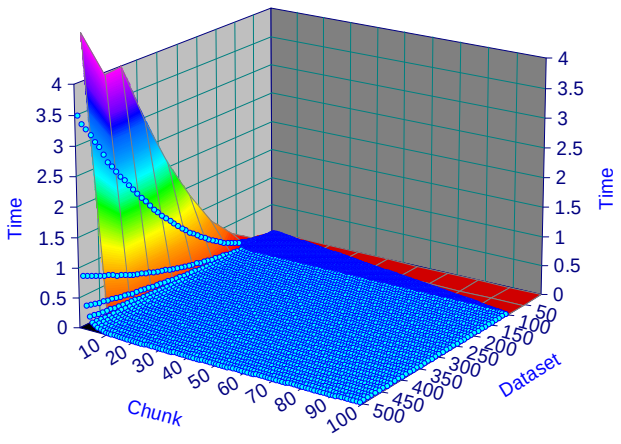

Figure 8. Results from writing a two dimensional double array (x – chunk size [number of elements], y – real time [s], z - size of dataset [number of records])

The tests for sequential and partial reading and the test for modifying show simmilar results in comparison to the resulst obtained for one dimensional array. Here, the slowest test again is the test for writing the datasets into the HDF5 file.

V. PREDICTING CHUNK SIZE BY THE EXPECTED AMOUNT OF STORED INFORMATION

The experimental results show that optimizing the chunk size for writing datasets in HDF5 file shall have positive influence on the overall performance of the data storage system integrated in a VR application.

The observed data from the automated tests is analyzed and approximated using Sigmaplot TableCurve3D software package. Several approximation functions were tested. For further use the Chebyshev series X, lnY Bivariate Order 5, which gives very good results for all analyzed data within the test cases.

$$
\begin{aligned}
&T_n(x') = \cos(na\cos(x')) \\
&z = a + bT_1(x') + cT_1(y') + dT_2(x') + \\
&+eT_1(x')T_1(y') + fT_2(y') + gT_3(x') + hT_2(y')T_1(y') + \\
&+iT_1(x')T_2(y') + jT_3(y') + kT_4(x') + lT_3(x')T_1(y') + \\
&+mT_2(x')T_2(y') + nT_1(x')T3(y') + \\
&+oT_4(y') + pT_5(x') + qT_4(x')T_1(y') + \\
&+rT_3(x')T_2(y') + sT_2(x')T_3(y') + tT_1(x')T_4(y') + uT_5(y')
\end{aligned}
\tag{1}
$$

A. *Approximating One Dimensional Integer Array*

The approximation with the Chebyshev series for the write experimental data is presented in Fig. 9.

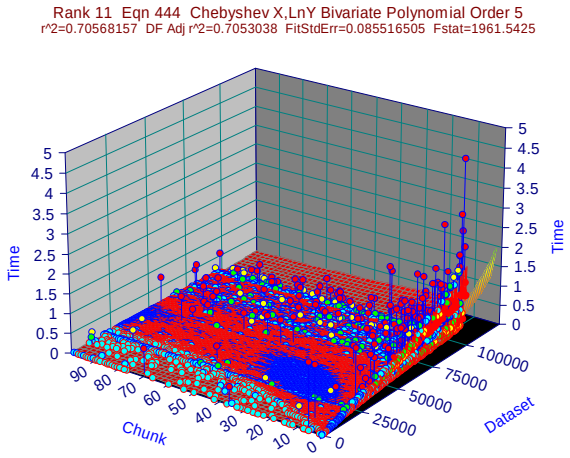

Figure 9. Results from approximation of experimental data for writing a one dimensional integer array (x – chunk size [number of elements], y – real time [s], z - size of dataset [number of records])

The resulting polynomial from (1) and the calculated parameters are used to create a C++ function for estimating the optimal chunk size based on the expected number of records in the dataset. The calculated values from calling the chunk size prediction function for dataset with size from 1 to 100100 is given in Fig. 10.

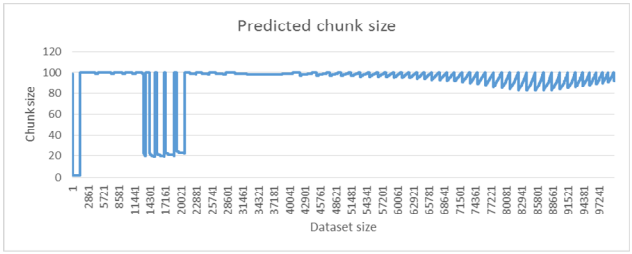

Figure 10. Predicted chunk size for optimal writing of one dimensional integer array (x – size of the input data, y – predicted size of the chunk)

The distribution of the suggested chunk size within range {1-100} is shown in Fig. 11.

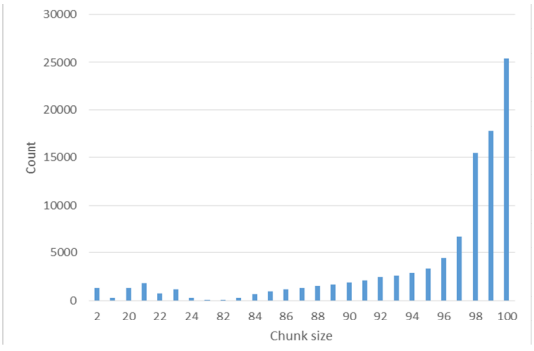

Figure 11. Distribution of the predicted chunk size for optimal writing of one dimensional integer array (x – predicted size of the chunk, y - count)

B. *Approximating Two Dimensional Double Array*

The same approach for results analysis used for one dimensional array is applied. The surface fit with Chebyshev series is shown in Fig. 12.

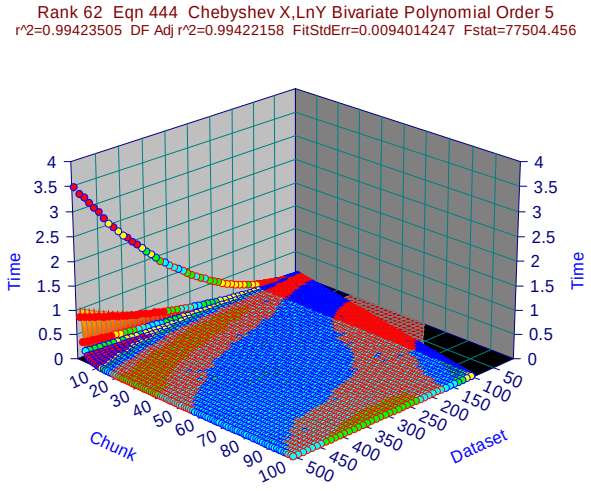

Figure 12. Results from approximation of experimental data for writing a two dimensional array of double data type (x – chunk size [number of elements], y – real time [s], z - size of dataset [number of records])

The predicted value for chunk size for dataset with 1 to 500 records for the two dimensional array is summarized in Fig. 13.

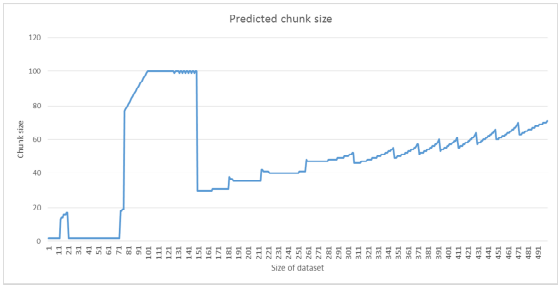

Figure 13. Predicted chunk size for optimal writing of two dimensional double array (x – size of the input data, y – predicted size of the chunk)

The distribution of the predicted chunk size values can be obtained from Fig. 14.

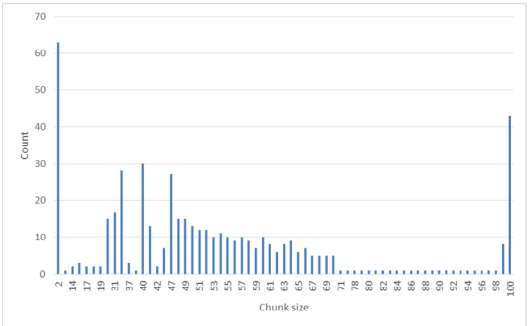

Figure 14. Distribution of the predicted chunk size for optimal writing of tow dimensional integer array (x – predicted size of the chunk, y - count)

A separate experiment has been carried out Based on the function for predicting the chunk size for two dimensional double array.

The results from 1022 executions of HDF5 dataset (two dimensional double array containing random values) write function with random sizes of the chunk and the dataset are compared to the same number of write operations with predicted chunk size. The graphical representation of the results can be obtained from Fig. 15.

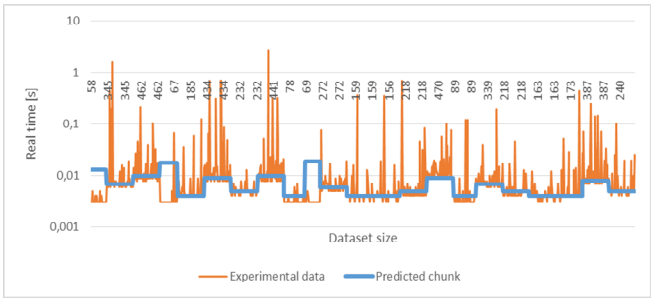

Figure 15. Comparing the real time for 1022 writes of two dimensional dataset with random values for chunk and dataset sizes with real time observed with predicted chunk size

The real time needed to finish all 1022 write cycles with random parameter values within the test is 19.15 s.

The same experiment but using the predicted chunk size took time of 7.14 s.

VI. CONCLUSIONS AND OUTLOOK

The chunk size has a significant influence on the performance of the HDF5 operations and since chunking is obligatory for data compression and for using hyperslabs finding the optimal chunk size for specific dataset is important task.

Based on the analysis of the experimental data an analytical model for surface and function approximation has been derived. This model is based on Chebyshev series X, lnY Bivariate Order 5.

e analytical equations have been used for nt of software functions for predicting the chunk size based on expected dataset size, reading the entire dataset in sequential and partial orders and for modification of the records in a dataset.

The proposed approach for chunk size prediction has been successfully validated by synthetic tests with network and local data storage. The overall performance improvement is of 65,56% (based on most important task – data storage).

A good practice is to predict the chunk size based on the amount of data before creating the dataset. Each dataset in HDF5 database should have the appropriate size of the chunks.

Using the HDF5 for data storage and retrieval has several benefits like very high performance when working

 ISBN: 978-1-61208-319-3

with large datasets of complex data types, parallel access to the data (requires MPI and parallel file system) and the predicted optimal chunk size based on the expected amount of data can significantly improve the performance. All these makes HDF preferred over other SQL and NoSQL DBMSs when developing VR applications, especially if the objects are enhanced with implicit features.

The study can be extended to analyze the relation between the chunk size, chunk cache options and the overall I/O performance. The results could be used to derive a model for predicting their optimal values based on the size of the input data set.

Another important further study is to perform experiments with dataset dimension larger than 2 and with complex data types.

VII. ACKNOWLEDGMENT:

The authors wish to thank for the support to:

- EC Research Executive Agency received through FP7 IAPP grant 285782;
- Project No BG051PO001-3.3.06-0046 "Development support of PhD students, postdoctoral researchers and young scientists in the field of virtual engineering and industrial technologies". The second project is implemented with the financial support of the Operational Programme Human Resources Development, co-financed by the European Union through the European Social Fund.

REFERENCES

[1] A. Bachvarov, S. Maleshkov, D. Chotrov, and J. Katicic, "Immersive Representation of Objects in Virtual Reality Environment Implementing Implicit Properties", 4-th International Conference on Developments in eSystems Engineering - DeSE 2011, Springer, 2011, pp. 587-592, ISBN 978-0-7695-4593-6.

[2] A. Bachvarov: S. Maleshkov, D. Chotrov, J. Ovtcharova, and J. Katicic, "Using implicit features for enhancing the immersive object representation in multimodal virtual reality environments", Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2012 IEEE International Conference, 2012, pp. 91-96.

[3] "The HDF Group", http://www.hdfgroup.org/, last access on 1.1.2014.

[4] A. Tzokev, "Приложение на не-SQL бази данни за съхранение на информацията при автоматизиран анализ на металографски изображения", Машиностроене и машинознание 19, 2013, pp. 128-131, ISSN 1312-8612.

[5] A. Shoshani, D. Rotem, "Scientific Data Management – Challenges, Technology and Deployment", CRC Press, ISBN 978-1-4200-6980-8, 2009.

[6] M. Folk, G. Heber, Q. Keziol, E. Pourman, and D. Robinson, "An Overview of HDF5 Technology Suite and Applications", Proc. AD'11 EDBT/ICDT 2011 Workshop on Array Databases, ACM, NY, 2011, pp. 36-47, doi:10.1145/1966895.1966900.

[7] "HDF5 User's Guide", http://www.hdfgroup.org/HDF5/doc/UG/UG_frame03DataModel.html, last access on 12.8.2013.

[8] E. Otoo, D. Rotem, and S. Seshadri, "Optimal Chunking of Large Multidimensional Arrays for a Data Warehousing", Proc. DOLAP '07, NY, 2007, pp. 25-32, doi:10.1145.1317331.1317337.

[9] "AIX 7.1 Information" - http://pic.dhe.ibm.com/infocenter/aix/v7r1/index.jsp?topic=%2Fcom.ibm.aix.prftungd%2Fdoc%2Fprftungd%2Fuse_time_measure_cpu_use.htm, last access on 14.8.2013.