D4M and Large Array Databases for Management and Analysis of Large Biomedical Imaging Data

Siddharth Samsi, Laura Brattain, Vijay Gadepally and Jeremy Kepner
MIT Lincoln Laboratory
Lexington, MA

***Abstract*—Advances in medical imaging technologies have enabled the acquisition of increasingly large datasets. Current state-of-the-art confocal or multi-photon imaging technology can produce biomedical datasets in excess of 1 TB per dataset. Typical approaches for analyzing large datasets rely on downsampling the original datasets or leveraging distributed computing resources where small subsets of images are processed independently. These approaches require significant overhead on the part of the programmer to load the desired sub-volume from an array of image files into memory. Databases are well suited for indexing and retrieving components of very large datasets and show significant promise for the analysis of 3D volumetric images. In particular, array-based databases such as SciDB utilize an architecture that supports massive parallel processing while also providing database services such as data management and fast parallel queries.**

In this paper, we will present a new set of tools that leverage the D4M (Dynamic Distributed Dimensional Data Model) toolbox for analyzing giga-voxel biomedical datasets. By combining SciDB and the D4M toolbox, we demonstrate that we can access large volumetric data and perform large-scale bioinformatics analytics efficiently and interactively. We show that it is possible to achieve an ingest rate of 2.8 million entries per second for importing large datasets into SciDB. These tools provide more efficient ways to access random sub-volumes of massive datasets and to process the information that typically cannot be loaded into memory. This work describes the D4M and SciDB tools that we developed and presents the initial performance results.

I. Introduction

Advances in imaging technologies are enabling the acquisition of larger and higher resolution biomedical datasets. New imaging techniques such as CLARITY [1] have the potential to significantly advance our understanding of brain function by enabling molecular and optical interrogation of brain tissue. However, a significant challenge in this area is the management and analysis of 3D volumetric data generated using such techniques. Several commercial [2], [3] and open source tools [4], [5], [6] are available for the analysis and visualization of biomedical imaging data. Each tool has its set of advantages and disadvantages, but a common limitation in many of these systems is the inability to analyze and/or visualize data sets that are significantly larger than the total amount of memory available on the system. Additionally, many of these packages are focused on single-client visualization and may not have a programmatic way to serve image data to a variety of clients. Traditional parallel computing approaches to the analysis of large image datasets involves the use of multiple processors to analyze subsets of images. In this article, we describe a new approach for data management that leverages the array-based database SciDB [7] and the data analysis tool, Dynamic Distributed Dimensional Data Model (D4M) [8].

SciDB is an open-source database that uses an array data model [9], [10]. The array-based data model provides support for parallel processing, efficient sparse storage, and in-database linear algebra operations that are well suited for the storage and analysis of biomedical imaging data. D4M is an open source software package that provides a uniform framework based on the mathematics of associative arrays [11] that can be applied to diverse domains such as cyber, bioinformatics, free text and social media data. D4M can also be used to perform linear algebraic operations inside a database [12]. In this article, we extend D4M to encompass multi-dimensional arrays in SciDB using imaging data management as an example application.

This work is sponsored by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the authors and are not necessarily endorsed by the United States Government.

II. D4M Extensions

The overall goal is to enable a workflow where volumetric data is inserted into a multi-dimensional array in SciDB and accessed from MATLAB using D4M as shown in Figure 1.

SciDB gives us the ability to efficiently extract random, multi-dimensional data using appropriate queries. Consider a three-dimensional array in SciDB with the following schema:

```
vol3d<val:double> [row=1:4096,4096,0,col
    =1:4096,4096,0,slice=1:1000,1,0]
```

Using the SciDB AFL query language, a sub-volume can be extracted as shown below:

```
between(vol3d, 100, 100, 10, 300, 500, 100);
```

This query will extract all values in the cube bounded by rows=100:300, cols=100:500 and slice=10:100. This is a very powerful capability that we extend to D4M so that end users can extract sub-volumes from SciDB using standard MATLAB indexing syntax as shown in Listing 1.

Listing 1: D4M example to extract sub-volume

```
DB = DBsetupSciDB('txg-testdb');
T = DB('vol3d<gray:uint8>row=1:4096,4096,0,col
    =1:4096,4096,0,slice=1:1000,1,0]');
v = getVolume(T, 100:300, 100:500, 10:100);
```

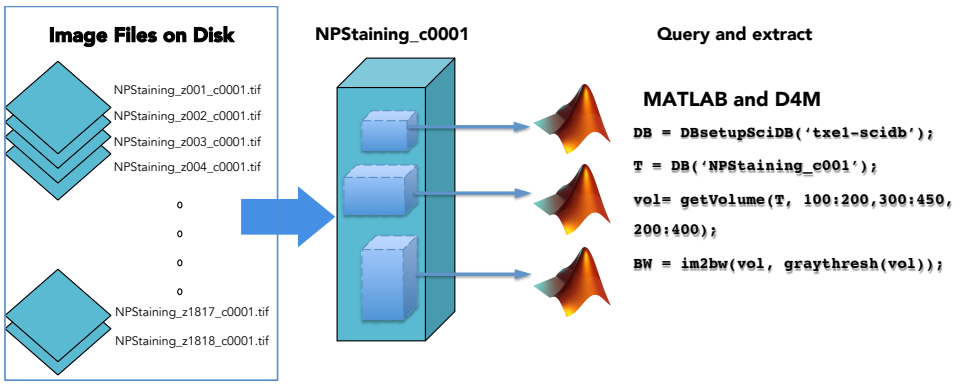


Fig. 1: Importing and accessing 3D volumetric data from SciDB using MATLAB and D4M

Similarly, data is ingested into SciDB using D4M as shown in Listing 2. In this example, slice number 15 of a 1000 slice 3D volume is ingested using D4M.

Listing 2: D4M example to extract sub-volume

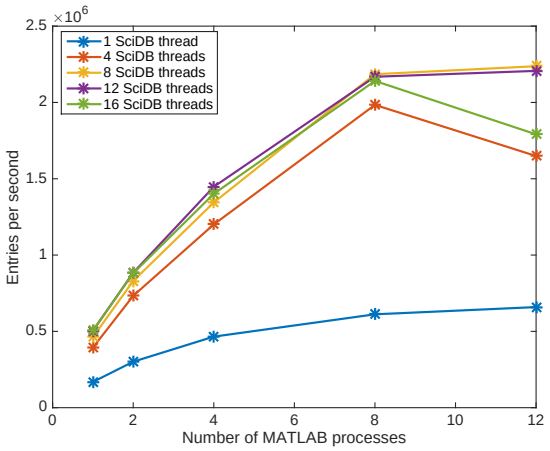
```
DB = DBsetupSciDB('txg-testdb');
T = DB('vol3d<gray:uint8>row=1:4096,4096,0,col
    =1:4096,4096,0,slice=1:1000,1,0]');
im = imread('test-image.tif');
[nr, nc] = size(im);
[rowids, colids] = ind2sub([nr nc], [1:nr*nc]');
slicenum = 15*ones(size(ir));
T = putTriple(T, [rowids colids slicenum], im(:));
```

III. Benchmarking Data Ingest

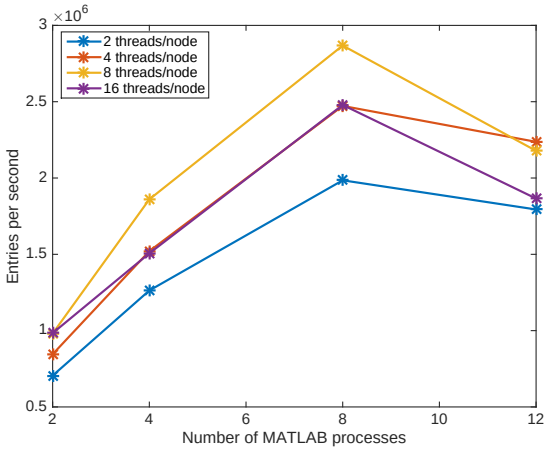
Due to the size of datasets, one of our goals was to benchmark the data ingest process. For this test, we used randomly generated imaging data to simulate a volume of size 5120 x 5120 x 1000 pixels. The data was ingested using a range of SciDB configurations. For a single node instance of SciDB configurations with 1, 4, 8, 12 and 16 SciDB worker processes were used. A two-node SciDB configuration using 2, 4, 8 and 16 SciDB worker threads per node was also used for testing. Data was imported using 2, 4, 8 and 12 processes running in parallel on the same node as the database. Data was imported using the SciDB Shim interface from MATLAB and D4M. Figure 2 shows the ingest rates achieved. When importing data into a single node instance of SciDB, we achieved a maximum ingest rate of 2.23 million entries/second by using 8 parallel MATLAB processes as shown in Figure 2a. A maximum ingest rate of 2.876 million entries/second was observed when using 8 parallel MATLAB processes importing into a two node SciDB instance. In each of these configurations, the use of more than 8 parallel processes for importing data resulted in a degradation of performance. Similarly, SciDB configurations using more than 8 SciDB instances per node did not result in an appreciable increase in ingest rates and actually results in a slower ingest in the case of the two-node SciDB instance as shown in Figure 2b.

IV. Conclusion

D4M and SciDB offer a new approach to the storage and analysis of large biomedical imaging datasets. By levaraging



(a) Ingest rate with single node SciDB instance. Maximum ingest rate: 2.23 million entries/sec.



(b) Ingest rate with two node SciDB instance. Maximum ingest rate: 2.876 million entries/sec.

Fig. 2: Timing results for data ingest in SciDB: Data was ingested using multiple parallel processes running on the same compute node as the database. Parallelization was achieved using pMATLAB [13].

D4M, it is possible to access large volumetric imaging data stored in SciDB using high-level languages such as MATLAB. The ability to efficiently access any 3D sub-volume from such a dataset gives us a capability that is not easily available using traditional approaches to image data management. Future work in this area includes the implementation of image analysis routines as linear algebra operations that can be run directly in SciDB, thus removing the need to retrieve data from the database. Other areas of research include the development of a more efficient method to import images into SciDB by exploiting sparsity and statistical distribution of the data.

V. Acknowledgement

The authors would like to thank Prof. Kwanghun Chung for providing imaging data used in the development of D4M extensions.

References

[1] K. Chung and K. Deisseroth, "Clarity for mapping the nervous system," *Nat Meth*, vol. 10, no. 6, pp. 508–513, 06 2013. [Online]. Available: http://dx.doi.org/10.1038/nmeth.2481

[2] BITPlane, "Scientific Data Visualization & Analysis Software for Microscopy," http://www.bitplane.com/imaris, [Online; accessed 16-Dec-2015].

[3] Molecular Devices, "MetaMorph Microscopy Automation and Image Analysis Software," http://www.moleculardevices.com/systems/metamorph-research-imaging/metamorph-microscopy-automation-and-image-analysis-software, [Online; accessed 16-Dec-2015].

[4] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "Nih image to imagej: 25 years of image analysis," *Nat Meth*, vol. 9, no. 7, pp. 671–675, 07 2012. [Online]. Available: http://dx.doi.org/10.1038/nmeth.2089

[5] J. R. Swedlow, I. Goldberg, E. Brauner, and P. K. Sorger, "Informatics and quantitative analysis in biological imaging," *Science*, vol. 300, no. 5616, pp. 100–102, 2003.

[6] H. Peng, Z. Ruan, F. Long, J. H. Simpson, and E. W. Myers, "V3d enables real-time 3d visualization and quantitative analysis of large-scale biological image data sets," *Nat Biotech*, vol. 28, no. 4, pp. 348–353, 04 2010. [Online]. Available: http://dx.doi.org/10.1038/nbt.1612

[7] M. Stonebraker, P. Brown, D. Zhang, and J. Becla, "Scidb: A database management system for applications with complex analytics," *Computing in Science Engineering*, vol. 15, no. 3, pp. 54–62, May 2013.

[8] J. Kepner, W. Arcand, W. Bergeron, N. Bliss, R. Bond, C. Byun, G. Condon, K. Gregson, M. Hubbell, J. Kurz, A. McCabe, P. Michaleas, A. Prout, A. Reuther, A. Rosa, and C. Yee, "Dynamic distributed dimensional data model (d4m) database and computation system," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, March 2012, pp. 5349–5352.

[9] J. Bayard Cushing, J. French, S. Bowers, M. Stonebraker, P. Brown, A. Poliakov, and S. Raman, *The Architecture of SciDB*. Springer Berlin Heidelberg, 2011, vol. 6809, pp. 1–16.

[10] P. G. Brown, "Overview of scidb: Large scale array storage, processing and analysis," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 963–968.

[11] J. Kepner, C. Anderson, W. Arcand, D. Bestor, B. Bergeron, C. Byun, M. Hubbell, P. Michaleas, J. Mullen, D. O'Gwynn, A. Prout, A. Reuther, A. Rosa, and C. Yee, "D4m 2.0 schema: A general purpose high performance schema for the accumulo database," in *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*, Sept 2013, pp. 1–6.

[12] V. Gadepally, J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, L. Edwards, M. Hubbell, P. Michaleas, J. Mullen, A. Prout, A. Rosa, C. Yee, and A. Reuther, "D4m: Bringing associative arrays to database engines," in *High Performance Extreme Computing Conference (HPEC), 2015 IEEE*, Sept 2015, pp. 1–6.

[13] N. Travinin Bliss and J. Kepner, "pmatlab parallel matlab library," *Int. J. High Perform. Comput. Appl.*, vol. 21, no. 3, pp. 336–359, Aug. 2007.