Scalable Storage of Whole Slide Images and Fast Retrieval of Tiles Using Apache Spark

Daniel E. Lopez Barron[a], Dig Vijay Kumar Yarlagadda[b], Praveen Rao[c], Ossama Tawfik[d], and Deepthi Rao[e]

[a]School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, USA
[b]School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, USA
[c]School of Computing and Engineering, University of Missouri-Kansas City, Kansas City, USA
[d]Department of Pathology and Lab Medicine, University of Kansas Medical Center, Kansas City, USA
[e]ProPath, Dallas, USA

ABSTRACT

Whole slide images (WSIs) can greatly improve the workflow of pathologists through the development of software for automatic detection and analysis of cellular and morphological features. However, the gigabyte size of a WSI poses serious challenge for scalable storage and fast retrieval, which is essential for next-generation image analytics. In this paper, we propose a system for scalable storage of WSIs and fast retrieval of image tiles using Apache Spark, a space-filling curve, and popular data storage formats. We investigate two schemes for storing the tiles of WSIs. In the first scheme, all the WSIs were stored in a single table (partitioned by certain table attributes for fast retrieval). In the second scheme, each WSI is stored in a separate table. The records in each table are sorted using the index values assigned by the space-filling curve. We also study two data storage formats for storing WSIs: Parquet and ORC (Optimized Row Columnar). Through performance evaluation on a 16-node cluster in CloudLab, we observed that ORC enables faster retrieval of tiles than Parquet and requires 6 times less storage space. We also observed that the two schemes for storing WSIs achieved comparable performance. On an average, our system took 2 secs to retrieve a single tile and less than 6 seconds for 8 tiles on up to 80 WSIs. We also report the tile retrieval performance of our system on Microsoft Azure to gain insight on how the underlying computing platform can affect the performance of our system.

Keywords: Whole slide imaging, scalable storage, fast retrieval, Apache Spark, space-filling curve

1. INTRODUCTION AND OBJECTIVE

Using optical microscopy, a pathologist can view histopathology glass slides at very high resolutions (e.g., 400X magnification) to observe individual cells and intra-cellular features. Such high resolutions are necessary, for example, in visualizing precise morphological details that aid in grading and subtyping the tumor or predicting the aggressiveness of tumor cells, which can be critical for the diagnosis, prognosis, and treatment. Some of the tasks performed by pathologists such as manually counting the mitotic figures in a region of interest using a microscope can be very tedious and time consuming. Through advances in digital imaging, a glass slide can be scanned in a few minutes or less to produce a whole slide image (WSI) of near-optical resolution.[1] WSIs can therefore, greatly improve the workflow of pathologists through the development of new software for automatic detection and analysis of cellular and morphological features.

Further author information:
Daniel E. Lopez Barron: Email: dl544@mail.umkc.edu,
Dig Vijay Kumar Yarlagadda: Email: dy5kc@mail.umkc.edu
Praveen Rao (corresponding author): E-mail: raopr@umkc.edu,
Ossama Tawfik: Email: otawfik@kumc.edu,
Deepthi Rao: Email: deepthi.rao@propath.com

Citation: Daniel E. Lopez Barron, Dig Vijay Kumar Yarlagadda, Praveen Rao, Ossama Tawfik, and Deepthi Rao, "Scalable Storage of Whole Slide Images and Fast Retrieval of Tiles Using Apache Spark," Proc. SPIE 10581, Medical Imaging 2018: Digital Pathology, 10581-38 (2018), DOI:http://dx.doi.org/10.1117/12.2290380

While there is growing interest in using WSIs for primary diagnosis and consultation,[2] several technical and regulatory barriers must to be overcome before WSI systems can be adopted by pathologists. One technical issue is the gigabyte size of a WSI. WSIs cannot be stored and processed efficiently using traditional database systems. Recent advances in big data systems (e.g., Apache Spark[3]) provide new opportunities for storing and processing large number of WSIs. To the best of our knowledge, *ours is the first work to investigate the potential of Apache Spark[3]–a popular in-memory cluster computing framework–for fast retrieval of tiles in WSIs.* This is essential to enable next-generation image analytics on WSIs (e.g., deep learning on histopathology images). Recently, Yildirim et.al.[4] studied the use of parallel file systems and Amazon Web Services (AWS) cloud storage to enable scalable data access of WSIs. Their focus, however, was on retrieving the full (tiled) image of a particular resolution in a WSI.

2. OUR METHOD AND SYSTEM

Our proposed system relies on Spark's data structures and processing primitives, a space-filling curve for partitioning the WSIs into tiles and ordering them, Spark SQL[5] for querying, and popular data storage/file formats. Figure 1 shows the overall system architecture.

2.1 Apache Spark

Apache Spark is an in-memory cluster computing framework for analyzing large datasets. Spark's basic abstraction for storing data is a Resilient Distributed Dataset (RDD), which is an immutable distributed collection of elements. An RDD is partitioned across nodes in a cluster and can be operated in parallel. Spark provides another abstraction called DataFrame, which is also an immutable distributed collection of data, but organizes data into named columns like a relational database system. This enables users to pose queries on structured datasets using Spark SQL, which is a module of Spark that provides a SQL interface for querying.

2.2 Storing WSIs Using Spark SQL

A typical WSI contains an image pyramid with different resolutions. Level 0 image represents the highest resolution (e.g., 400X magnification) and is the largest in size compared to lower resolution images at higher levels. As Level 0 (or Level 1) image in a WSI can be several GBs in size, our system partitions it into tiles using a space-filling curve (e.g., Z-order curve).[6] A space-filling curve provides a mathematical function to map points in a higher dimensional space (e.g., 2D) to 1D, while preserving spatial proximity of points in the higher dimension. Our system is designed for image analytics where rectangular regions of WSIs (one or more tiles) are randomly accessed. Thus, based on the space-filling curve, each tile in a Level 0 (or Level 1) image is assigned an index from 1 to 2^q, where q is a power of 2. Neighboring tiles tend be closer in terms of their index values, which is ideal when nearby tiles are to be retrieved. The number of tiles a Level 0 (or Level 1) image is partitioned into will depend on the desired maximum size of a tile. A tile is extracted from a WSI using OpenSlide.[7] Each tile is denoted by a variable-length record [`fileID`, `imageLevel`, `tileIndex`, `tileWidth`, `tileHeight`, `tileBytes`], where `fileID` is the name of the WSI and `tileBytes` denotes the actual tile content.

Once all the records are created, our system can store the records in Spark SQL using two schemes: In the first scheme, called *Singletable*, all the records (i.e., all the tiles from all the WSIs) are stored in a single table partitioned by `fileID` and `imageLevel`, to enable fast retrieval of tiles. The tiles from the same level are sorted (using Spark's primitives) by `tileIndex` so that range searches are efficient for fetching neighboring tiles from a small number of cluster nodes. In the second scheme, called *Multitable*, we create a separate table for each WSI. The table name for a WSI is based on `fileID` as it facilitates easy lookup during retrieval. The records in each table are sorted (using Spark's primitives) first by `imageLevel` and then by `tileIndex`.

Each table can be stored in the Hadoop Distributed File System (HDFS)[8] using two formats: Parquet[9] or ORC.[10] Parquet is a column-based storage format suited for data with many columns or queries that need to read most of the column values. ORC is also an optimized column-based storage format but has built in indexes to speed up query processing. Both formats employ compression to reduce the size of files.

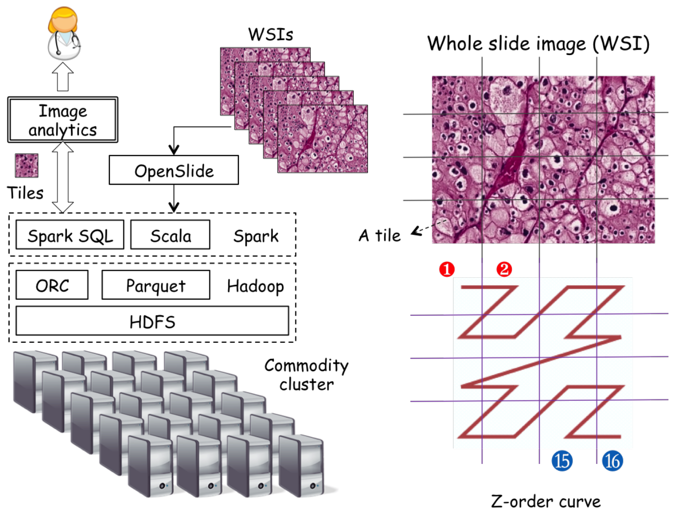


Figure 1. System architecture. (Image sources: http://en.wikipedia.org/wiki/File:Renal_Cell_Carcinoma.jpg, http://en.wikipedia.org/wiki/File:Four-level_Z.svg)

Scheme	Key Spark SQL statements
Singletable	sqlContext.read.orc('hdfs://ctl:9000/home/user/singletable.orc').createOrReplaceTempView(bigtable)
	sqlContext.sql('SELECT imageBytes FROM bigtable WHERE fileID='2.svs' AND imageLevel=0 AND (tileIndex >= 1 AND tileIndex <= 4)').map(_.getAs[Array[Byte]](0)).rdd.zipWithIndex.map{case (bytes, index) => (bytes, index, s"out$index.JPEG")}.collect.map(writeToLocal)
Multitable	sqlContext.read.orc('hdfs://ctl:9000/home/user/multitable.orc/fileID=2.svs'). createOrReplaceTempView(smalltable)
	sqlContext.sql('SELECT imageBytes FROM smalltable WHERE imageLevel=0 AND (tileIndex >= 1 AND tileIndex <= 4)').map(_.getAs[Array[Byte]](0)).rdd.zipWithIndex.map{case (bytes, index) => (bytes, index, s"out$index.JPEG")}.collect.map(writeToLocal)

Table 1. Singletable and Multitable schemes: Spark SQL statements for retrieving tiles

2.3 Querying Tiles of a WSI Using Spark SQL

Tiles of a WSI are retrieved by posing queries in Spark SQL. For Singletable, the main table (containing all the WSIs) is loaded just once before executing all the queries. For Multitable, the table for a specific WSI is loaded first before executing one or more queries to retrieve tiles in that WSI. Table 1 shows examples of how we can retrieve 4 tiles from a level 0 image of a WSI using ORC. The query output is processed using Spark's primitives and written to local storage.

3. RESULTS

We implemented our system using Spark 2.0.1, Hadoop 2.7, and Scala 2.11 and conducted its performance evaluation on a 16-node cluster in CloudLab,[11] a testbed for cloud architectures and distributed computing applications. We used 7 WSIs from the Department of Pathology and Lab Medicine at the University of Kansas Medical Center. These WSIs were produced by an Aperio scanner and were in the SVS format. Each WSI had 4 levels. The size of each WSI is shown in Table 2 along with the number of tiles constructed and the total size of the tiles (in JPEG).

One may wonder if WSIs can be stored directly in HDFS and fetched locally when needed instead of using our system. However, we observed that copying a single WSI from HDFS to local storage required 12 seconds. Additional time would be needed to retrieve one or more tiles using OpenSlide.

Filename	SVS file size	# of tiles	Total size of tiles
1.svs	3.4 GB	274	6.1 GB
2.svs	2.6 GB	70	4.0 GB
3.svs	4.0 GB	274	6.6 GB
4.svs	2.6 GB	70	4.5 GB
5.svs	3.0 GB	274	5.3 GB
6.svs	2.0 GB	70	3.7 GB
7.svs	3.9 GB	274	6.7 GB
Total	21.5 GB	1,306	36.9 GB

Table 2. Dataset

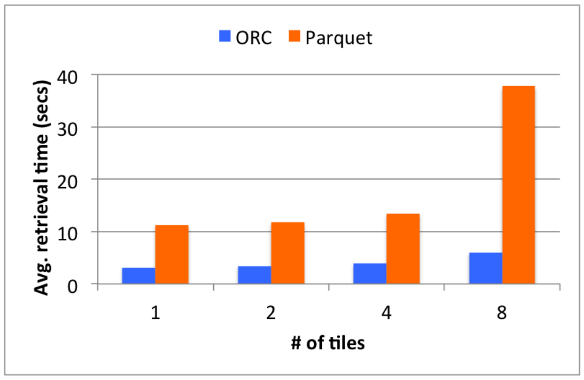
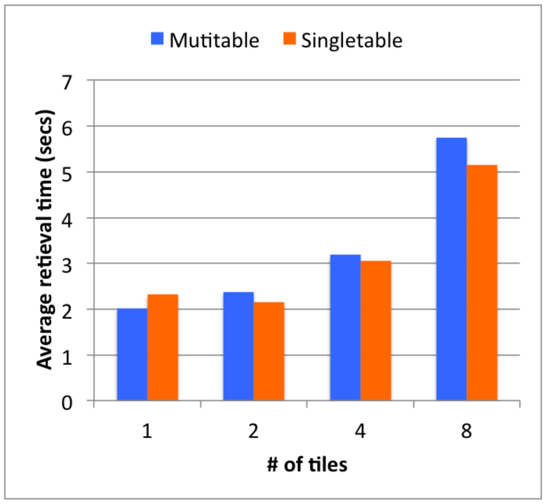


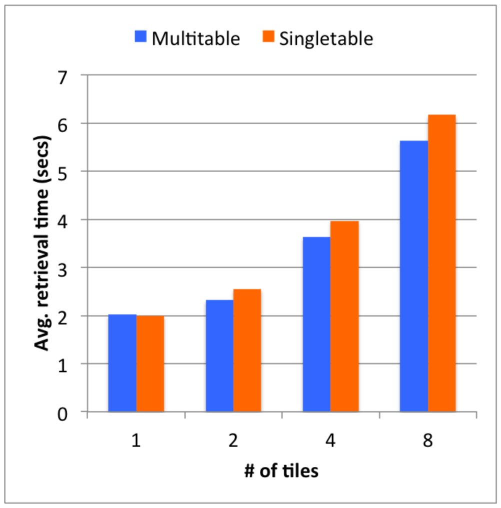
Table 3. ORC vs Parquet

3.1 ORC vs Parquet

We first investigated whether ORC or Parquet is a better choice for WSIs. We tested with Singletable on 7 WSIs. While ORC consumed 24 GB of disk storage, Parquet required about 6 times the storage space (125 GB). We generated 100 random queries (for level 0) to retrieve 1, 2, 4, and 8 tiles, respectively. Figure 3 shows the average retrieval time for these query workloads. Clearly, our system is significantly faster and more space efficient when ORC is used for managing WSIs.



(a) 20 WSIs



(b) 40 WSIs

Figure 2. Tile retrieval performance of our system for datasets with 20 and 40 WSIs

3.2 Tile Retrieval Performance

To test the scalability of our system, we replicated the WSIs in Table 2 to create larger datasets with 20, 40, and 80 WSIs, respectively. The dataset with 80 WSIs was 246 GB in size. We used ORC as the storage format. The size of the ORC file for this dataset was 413 GB. For each dataset, we generated 100 random queries on the WSIs to retrieve 1, 2, 4, and 8 tiles, respectively. Figure 2 shows the average retrieval time for the datasets with 20 and 40 WSIs with increasing number of tiles. Figure 3 shows the average retrieval time for the dataset with 80 WSIs with increasing number of tiles. Both Singletable and Multitable achieved comparable performance as

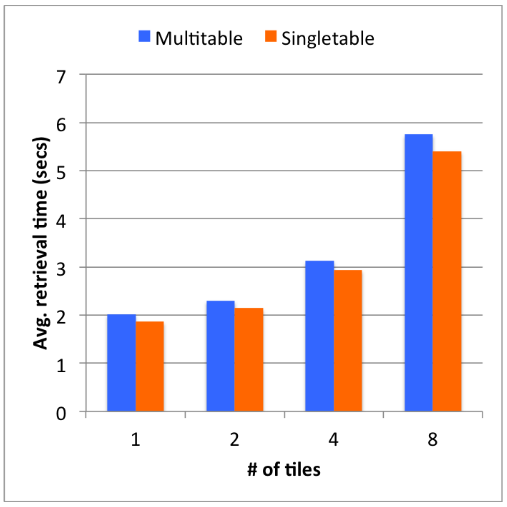


Figure 3. Tile retrieval performance of our system for the dataset with 80 WSIs

the dataset size was increased. Our system required an average of 2 secs to retrieve a single tile and less than 6 secs (on an average) for 8 tiles on up to 80 WSIs. Thus, our system enables fast retrieval of tiles, which is essential for image analytics on large number of WSIs.

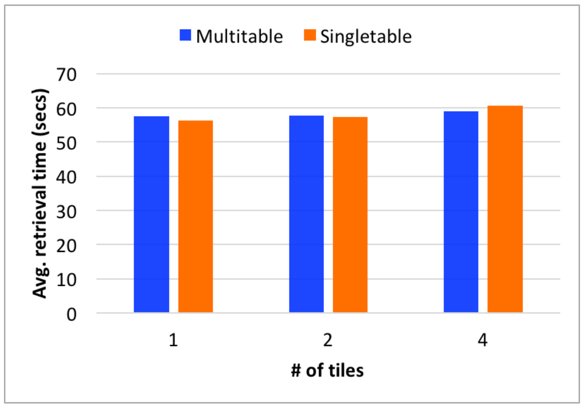


Figure 4. Tile retrieval performance of our system on Microsoft Azure for the dataset with 80 WSIs

4. PERFORMANCE ON MICROSOFT AZURE

We were curious to understand if also the performance of our system on Microsoft Azure, a public cloud computing platform.[12] We set up a cluster using virtual machines with 2 head nodes and 4 worker nodes to run Apache Spark available as part of Azure HDInsight. (This was the maximum number of nodes we could acquire through Microsoft Azure for Research Program.) We stored and processed 80 WSIs with Singletable and Multitable schemes using ORC as the storage format.

Figure 4 shows the tile retrieval performance of our system on Azure. We observed that the average tile retrieval time was much higher than what was observed on CloudLab. The average retrieval time was close to 60 secs for both Singletable and Multitable. We attribute this significant slowdown in performance to two reasons: First, unlike CloudLab, where we used physical machines, we could only use Linux virtual machines on Azure. Second, the size of the cluster on Azure was much smaller than what we used on CloudLab. This shows that the underlying computing platform can have a significant impact on the performance of our system when dealing with large number of WSIs.

5. CONCLUSIONS AND FUTURE WORK

We presented a system for scalable storage of WSIs and fast retrieval of tiles, which is essential for next-generation image analytics on WSIs. Our system relies on Spark's data structures and primitives, a space-filling curve, Spark SQL, and popular storage formats. We investigated two popular storage formats and the Singletable and Multitable schemes to store the tiles. Through performance evaluation on a 16-node cluster in CloudLab, we observed that ORC is a better choice than Parquet for storing WSIs in terms of tile retrieval time and storage cost. We also observed comparable performance between the Singletable and Multitable schemes. On an average, our system required 2 secs to retrieve a single tile and less than 6 secs for 8 tiles on up to 80 WSIs. We also reported the tile retrieval performance of our system on Microsoft Azure. We observed that using Azure, our system could not achieve the same tile retrieval performance observed on CloudLab due to the use of virtual machines and a smaller cluster. Thus, the underlying computing platform can significantly impact the performance of our system. More investigation is necessary to truly understand the full potential of public cloud computing platforms for large-scale storage and retrieval of WSIs.

In the future, we would like to investigate adaptive partitioning of tiles in WSIs using space-filling curves. This is because different regions of a WSI can contain varying granularity of medical information of interest to pathologists. Creating smaller partitions of regions containing higher density of cells will enable better tile retrieval performance when applying next-generation image analytics. Finally, we would like to extend our system by integrating with Spark's Deep Learning Pipelines[13] for enabling users to test their deep learning algorithms for cancer diagnosis on large number of WSIs.

ACKNOWLEDGMENTS

The first author was supported by the Mexican Council for Science and Technology (CONACyT) under scholarship 264829. The third author would like to acknowledge the partial support received from a University of Missouri Research Board grant. We thank Vinutha Nuchimaniyanda for her assistance. Cloud computing resources were provided by a Microsoft Azure for Research Award and by CloudLab.

REFERENCES

[1] Ghaznavi, F., Evans, A., Madabhushi, A., and Feldman, M., "Digital Imaging in Pathology: Whole-Slide Imaging and Beyond," *Annual Review of Pathology: Mechanisms of Disease* **8**, 339–351 (2013).

[2] Pantanowitz, L., Sinard, J. H., Henricks, W. H., Fatheree, L. A., Carter, A. B., Contis, L., Beckwith, B. A., Evans, A. J., Lal, A., and Parwani, A. V., "Validating whole slide imaging for diagnostic purposes in pathology: guideline from the College of American Pathologists Pathology and Laboratory Quality Center," *Archives of Pathology and Laboratory Medicine* **137**(12), 1710–1722 (2013).

[3] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I., "Spark: Cluster Computing with Working Sets," in [*Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*], *HotCloud'10*, 10–10 (2010).

[4] Yildirim, E. and Foran, D. J., "Parallel Versus Distributed Data Access for Gigapixel-Resolution Histology Images: Challenges and Opportunities," *IEEE Journal of Biomedical and Health Informatics* **21**, 1049–1057 (July 2017).

[5] "Spark SQL." http://spark.apache.org/sql (2017).

[6] Sagan, H., [*Space-Filling Curves*], Springer Verlag (1994).

[7] Goode, A., Gilbert, B., Harkes, J., Jukic, D., and Satyanarayanan, M., "OpenSlide: A Vendor-Neutral Software Foundation for Digital Pathology," *Journal of Pathology Informatics* **4**, 1049–1057 (July 2013).

[8] "HDFS Architecture Guide." http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (2017).

[9] "Apache Parquet." http://parquet.apache.org (2017).

[10] "Apache ORC." http://orc.apache.org (2017).

[11] "CloudLab." http://cloudlab.us (2017).

[12] "Microsoft Azure." http://azure.microsoft.com (2017).

[13] "Deep Learning Pipelines for Apache Spark." https://github.com/databricks/spark-deep-learning (2017).