SMASH: A Cloud-based Architecture for Big Data Processing and Visualization of Traffic Data

Siqi Wu, Luca Morandini, Richard O. Sinnott
Department of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
rsinnott@unimelb.edu.au

***Abstract* – In recent times, big data has become a popular research topic and brought about a range of new challenges that must be tackled to support many commercial and research demands. The transport arena is one example that has much to benefit from big data capabilities in allowing to process voluminous amounts of data that is created in real time and in vast quantities. Tackling these big data issues requires capabilities not typically found in common Cloud platforms. This includes a distributed file system for capturing and storing data; a high performance computing engine able to process such large quantities of data; a reliable database system able to optimize the indexing and querying of the data, and geospatial capabilities to visualize the resultant analyzed data. In this paper we present SMASH, a generic and highly scalable Cloud-based architecture and its implementation that meets these many demands. We focus here specifically on the utilization of the SMASH software stack to process large scale traffic data for Adelaide and Victoria although we note that the solution can be applied to other big data processing areas. We provide performance results on SMASH and compare it with other big data solutions that have been developed.**

Keywords – *Cloud, big data, data analytics, traffic analysis.*

I. Introduction

A staggering amount of data across all walks of life is being generated: social media data, scientific data, data collected by government, by industry and across all research endeavours. This is often referred to as the data deluge [1] and big data is now a term in common parlance [2]. In order to process, analyse and derive knowledge from this ever-growing amount of data, large-scale and above all scalable infrastructures are essential. For many domains, the challenges in developing such infrastructures are one of scale, e.g. the volume of data is the challenge that must be tackled; for other domains it is the speed (velocity) at which data itself is produced that is the issue to be tackled; for other domains it is the accuracy, authenticity and provenance of data that is important (veracity). Some disciplines demand all of these big data capabilities. Transport and traffic flows more generally are areas that demand the ability to cope with large-scale data sets that are produced in near real-time and have important demands on the accuracy and validation of information.

Tackling the issues of transport requires the ability to cope with large volumes of traffic that can be bursty in nature, e.g. morning rush hour in major cities. The volume of data over time can be substantial, especially if real time information is captured on the location and speed of cars and vehicles more generally. Infrastructures supporting the analysis of such data must be able to cope with such big data aspects. Historically the predominant form of scalable infrastructures have been based upon cluster-based technologies to support high performance computing at a given and typically centralised location, e.g. a given data centre, however more recently Cloud computing has been the predominant technology for big data processing and analysis [3]. There have been many Cloud initiatives that have established a range of capabilities, e.g. Amazon Elastic Compute Cloud (EC2); Amazon Simple Storage Solution (S3) Microsoft Azure, through to a range of widely adopted data storage services such as Dropbox and iCloud.

Across Australia the federally funded National eResearch Collaboration Tools and Resources (NeCTAR – www.nectar.org.au) project has been established to realise a national Cloud facility for all researchers across Australia - irrespective of research discipline. NeCTAR utilises the OpenStack Cloud middleware and has multiple Cloud nodes (also known as availability zones) established across Australia. At present almost 30,000 servers are available across availability zones in Melbourne, Monash, Brisbane, Canberra, Adelaide and Tasmania. The primary focus of NeCTAR has been to establish an Australia-wide Infrastructure-as-a-Service (IaaS) service. In this model, researchers are able to dynamically create a range of VMs from small VMs offering single core 4GB machines with 30GB storage, through to extra large machines offering 16 cores with 64GB memory and 480GB storage. In addition to this, the federally funded Research Data Services that builds upon the previous Research Data Storage Infrastructure (RDSI) project (www.rdsi.uq.edu.au) has rolled out major storage capabilities across Australia. This is expected to provide over 100 Petabyte data storage for Australian researchers.

In principle the capabilities exist for processing large volumes of transport data however access to virtual machines and storage does not in itself support this activity. Instead, targeted solutions are required that meet the demands of big data that are specific to the transport domain. In this paper we described the SMASH software solution that has been developed to meet these specific needs. We describe the technologies that are involved in realising the SMASH architecture: geoServer (http://geoserver.org), geoMesa (http://www.geomesa.org/), Apache Accumulo (http://accumulo.apache.org), apache Spark (http://spark.apache.org/) and Hadoop

978-1-5090-0214-6/15 $31.00 © 2015 IEEE
DOI 10.1109/DSDIS.2015.35

IEEE computer society

(http://hadoop.apache.org)). We describe their key capabilities that support the specific needs and demands in dealing with high velocity, high volume traffic data with demands for highly accurate data processing. We show how this solution provides the capabilities for the scalable analysis of disaggregated traffic data, i.e. down to traffic flows at specific junctions at specific time periods, with case studies and benchmarking of traffic data for Victoria. The data that we utilize is based on the Sydney Coordinated Adaptive Traffic System (SCATS) data from the Department of Transport in Victoria [4].

II. Related Work

It is the case that all research disciplines are becoming increasingly driven by the volume of data that can now be created and/or is available in various forms on the Internet. Many research endeavours in many disparate disciplines are now limited by the ability to discover, access and optimally use data. The work on e-Infrastructures allowing seamless access to distributed data resources has been on-going for decades: from the original efforts in large-scale data distributed systems, Grid efforts typified by [5-8] through to more recent Cloud-based efforts typified by Amazon EC2.

Transport is one area that many researchers have explored from a variety of perspectives. One common approach underpinning transport research is through use of transport/traffic modelling tools based on a variety of transport simulators. [9-14] are typical examples of such approaches. One key issue is that the accuracy of information of such systems is not guaranteed to reflect the true transport issues of cities and their associated daily road networks usage statistics, i.e. they are simulators. Thus for example, the day-day commuting patterns can fluctuate greatly for a given city based on a variety of factors such as real-time transport incidents/accidents, sporting events, flooding, bush fires amongst many other emerging issues that represent real use of the road network. Modelling tools do not capture this information using the actual data from the road network.

Other approaches have explored use of social media for event detection on the transport network, e.g. Twitter data. [15] use natural language processing to identify targeted keywords in tweets and correlates them with transport events, e.g. accidents that may have taken place. They also analyze the confidence level of traffic information.

Similar to the above study, [16] extracts traffic information from tweets using syntactic analysis and then classifies the results into two categories: Point and Line. Point data refers to a single accident/event on the road, e.g. a car crash, whilst Line data refers to a traffic status along certain streets including a start point and end point on a map over a given time period.

[17] provides a novel solution for use of social media for near real-time transport information. The novelty of this work lies in the scale of data that is captured and the fact that the algorithms only capture data (tweets) from the road network itself. This work has shown that formation of traffic jams can be identified, but the challenge is that not all road users tweet and hence there is not the depth of information on all users of the road network system.

Government agencies can and do collect information on the use of the road networks. In Australia, many transport agencies collect data provided by the SCATS (Sydney Coordinated Adaptive Traffic System) system to capture road traffic flows [18-20]. A typical example of a SCATS enabled sensor network is shown in Figure 1. This sensor network is the SCATS network for the Adelaide CBD and major junctions north of the city centre.

Figure 1: SCATS sensors installed in Adelaide CBD and North Adelaide

This (Adelaide) sensor network and the data that was collected from these sensors were used for evaluating the SMASH architecture. This system was then used to explore larger (big data) challenges arising from the SCATS sensor network from all of Victoria. The locations of the SCATS sensors for all of Victoria are shown in Figure 2.

Figure 2: SCATS sensors installed in Victoria

SCATS, is an intelligent transportation system developed in Sydney, Australia [18]. SCATS supports the dynamic timing of signal phases at traffic signals. The system uses sensors at each traffic signal to detect vehicle presence in each lane. The vehicle sensors are typically installed within the roads themselves (as strips across the roads). Information collected from vehicle sensors can allow SCATS (in principle) to be used to calculate and adapt the timing of traffic signals in the network to improve traffic throughput. In Australia, the majority of signalized intersections are SCATS operated (around 11,000). However a challenge with use of SCATS data is that they can be extremely large, e.g. data from a single intersection can include results on millions of cars that have passed over it. To tackle this, such data is often aggregated by time to provide statistics on the number of cars/lorries over given intersections, but such solutions do not lend themselves to real-time transport analysis and congestion identification. The real-time data processing demands scalable infrastructure and software to cope with the volume, velocity and veracity aspects of big traffic data.

Figure 3 and Table 1 demonstrate how SCATS sensors work in one particular intersection. The sensors are at traffic signals and detect vehicle presence in each lane. The vehicle sensors are generally inductive loops installed within the road pavement. Vehicles that traverse these road sections have information that is automatically captured and stored in central databases within the Department of Transport. The ability for real-time data analytics using all of these data sets is a challenge that has not yet been satisfactorily addressed.

Figure 3: Sensors in each lane

site_no	street name	date	movement	HFID	vehicle_count
4700	McNaughton rd N of Centre rd	2008-01-02 00:00:00	Thru	9508	{vc1-vc96}
487	Forester rd S of Waverley rd	2008-01-02 00:00:00	Left	15088	{vc1-vc96}
4905	Park st E of Clarendon st	2008-01-02 00:00:00	Right and Thru	19928	{vc1-vc96}

Table 1. Examples of SCATS Adelaide traffic data, {vc1-vc96} stands for vehicle count number in every 15-minute interval

This information shows a small example of the kinds of data that are captured through SCATS sensors. There are more fields in the metadata, however, only a small subset of them is listed here. With the increasing population growth of Australia and increasing urbanization of the cities – where it is predicted that 50% of the population will live in two cities: Melbourne and Sydney, solutions that allow for traffic analysis and improved (optimized) use of the road network are urgently demanded.

III. SMASH Architecture

In essence, the SMASH is a software stack: a collection of software components that work together to form a complete framework for the developing of applications in a given domain. The original research on big data processing techniques can be traced back to early 2000s and efforts of Google through MapReduce programming model, BigTable DBMS and Google File System [21,22].

There is no definitive description of big data, but generally it refers to datasets of such size (volume) that they cannot be stored on a single machine (server) and the analysis cannot be undertaken on a single machine (server). These issues place numerous other demands on the underlying big data infrastructure: fetching the data, searching the data, storing the data and validating the data are just some of the challenges that must be tackled.

The SMASH architecture was developed to tackle these issues directly. Firstly it was recognized that a distributed file store was needed. Getting large volumes of data that is real-time in nature demands that distributed data storage is in place. To tackle this, the SMASH stack utilizes the Hadoop Distributed File System (HDFS) [23]. Secondly there is a need for performing computation on traffic data, such as aggregation, statistics, or machine learning. Since the computations are complex in nature, we think Apache Spark would be more performing than the native Hadoop MapReduce component. Traffic data are geospatial and temporal in nature, hence it is of the essence to be able to index them spatio-temporally, and visualize them on maps; therefore, GeoMesa and GeoServer were added to the stack. GeoMesa adds spatial indexing to the Accumulo BigTable DBMS, while GeoServer can serve maps and vector data to geospatial clients.

The overall architecture of SMASH and the comparison with the traditional Hadoop ecosystem [24] is shown in Figure 4.

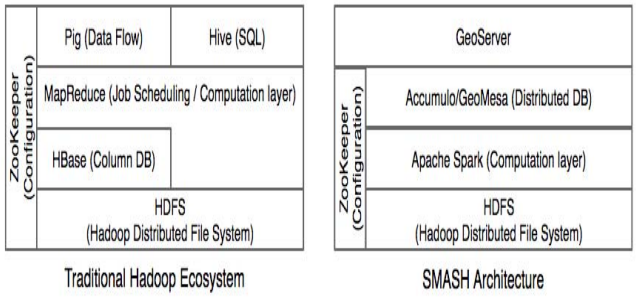

Figure 4: SMASH Architecture

We described each of these capabilities of the stack in turn and the role they perform within the SMASH architecture.

3.1 Hadoop Distributed File System (HDFS)

At the lowest level of the SMASH architecture is a distributed file system: HDFS [23]. HDFS offers a reliable, scalable and essentially a distributed file system. With the volume and velocity of traffic data, there is an absolute need to distribute the storage and access to the data. A single server cannot cope with large volumes of high velocity data hence spreading the ingestion of data and the subsequent storage across multiple servers is essential. The composition of HDFS system within the SMASH system is illustrated in Figure 5.

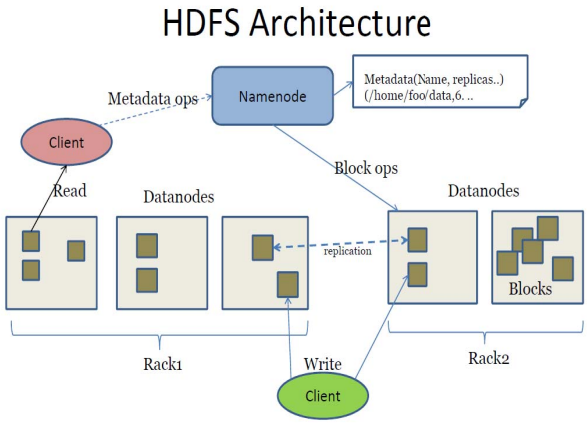

Figure 5: HDFS architecture

In the HDFS architecture, there normally exists a namenode with a set of datanodes. A namenode stores the metadata information to indicate which datanode hosts particular data. HDFS chunks large files into small blocks (typically 64Mb) then spreads them across multiple machines. Datanodes can communicate with each other via TCP/IP sockets. Replicas are important mechanisms in HDFS to guarantee partial failure. Thus if a given datanode fails or the server upon which this data is stored fails, then HDFS will automatically provide replicas of this data on other datanodes. To increase the robustness of the system, a secondary namenode can also be introduced in case of failure of namenode resources.

A core data processing element of HDFS is MapReduce [21]. MapReduce is a common way to process large amounts of data in a distributed manner. A MapReduce job typically involves splitting an input dataset into independent chunks, which can then be processed by the map tasks in parallel, i.e. using distributed servers. This ability to distribute the processing activities to separate servers means that it is very scalable and ideally suited to Cloud-like environments where more servers can be added on demand. To achieve this the MapReduce framework passes the outputs of map tasks to reduce tasks. Normally temporary values and output files are sorted. The MapReduce framework normally consists of a master JobTracker and one slave TaskTracker per node. The master is responsible for scheduling and monitoring slave tasks. A typical dataflow in MapReduce is shown in Figure 6.

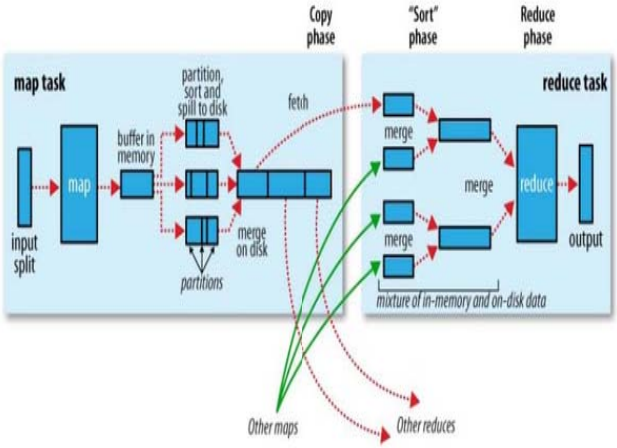

Figure 6: MapReduce working model

Hadoop MapReduce is a computing framework that allows processing large amounts of data in parallel across a cluster of servers. It has been developed and applied as a mature technique for many years. More recently Apache Spark is a new-generation, fast engine for large-scale data processing, it promises 100x faster performance than MapReduce in memory [26]. We have included Apache Spark within the SMASH architecture and undertaken benchmarking of its performance for large-scale data analysis.

3.2 Apache Spark

Apache Spark [25] has many more concepts and features than Hadoop MapReduce. Some of the most important of these that support the SMASH architecture are described below.

Firstly Apache Spark offers Resilient Distributed Dataset (RDD) capabilities. This provides a fault-tolerant abstraction for in memory cluster computing. All RDD operations are categorized into transformation or actions, with all jobs associated with and operating on an RDD. Importantly a job can be divided into tasks and stages sent to different executors, allowing decomposition of processing. Similarly, data within the same RDD can be divided into many partitions. The dependencies between stages, jobs and the data upon which they process can be represented as a directed acyclic graph. In this model a variety of dependencies can exist. Thus a child RDD can depend on certain data partitions to exist within a parent RDD (narrow dependency) or a child RDD may depend on all data partitions of a parent RDD (wide dependency).

Apache Spark also offers advanced caching management capabilities that allow speeding up data processing – especially important when dealing with large-scale distributed data sets. Apache Spark also offers failure recovery mechanisms, e.g. by automatically re-computing processes that have failed. Figure 7 shows the typical execution of an Apache Spark application within the SMASH architecture.

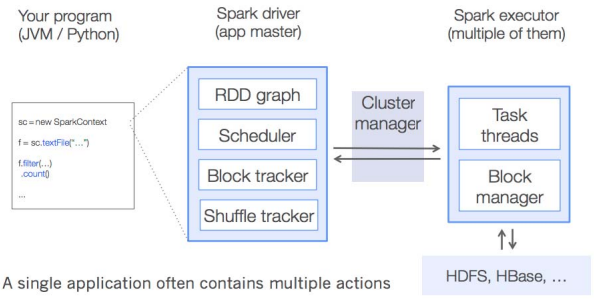

Figure 7: Apache Spark application execution

To justify the use of Apache Spark compared to HDFS MapReduce a series of benchmarking activities were undertaken. The results of these are shown in Table 2.

	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

Table 2: Benchmarking Hadoop and Apache Spark

There are a variety of factors for this speed-up. Firstly MapReduce stores data on the HDFS system, while Spark leverages in-memory RDDs and "spills" data on disk only when it exceeds memory limitations. Thus the general execution of Spark is much faster than MapReduce by reducing the amount of I/O operations. Secondly, RDDs allow recovery of failed nodes by re-computation using the DAG predecessor. In some ways this supports a similar recovery to MapReduce by way of checkpointing. However to reduce the dependencies of lineage further, it is possible to combine those two methods together. As a result if an iterative computation program needs to be executed, e.g. a machine learning training stage, Spark is far more performant than MapReduce since the data is always in memory. One point to make however is that Spark has stricter memory requirements. Thus in a Spark cluster the memory should be at least large as the amount of data we need to process, because the data has to fit into the memory for further operation. Block sizes are thus important: processing big data sets requires larger memory, but if only smaller data sets are to be processed then they will waste block space (as they will only use a fraction of the block size).

3.3 ZooKeeper

In setting up the SMASH software stack, capturing the detailed configuration information for the tools is essential. Configuration, especially in distributed environments involving multiple tools and services, can require a lot of work to set up, change and update. A wrong configuration can adversely impact upon performance and lead to unreliable systems. ZooKeeper is the most commonly used tool used to manage configuration information in the SMASH software stack [27].

ZooKeeper offers replicated synchronisation services with eventual consistency offered for the tools and their configurations. It provides degrees of robustness by migrating multiple nodes as ensembles where clients can connect to any of them to get the associated configuration information required. As long as the majority of nodes are working, the ensemble of ZooKeeper nodes is alive. This means that a master node can be dynamically chosen by consensus within the ensemble. If the master node fails, the role of the master can automatically migrate to another node.

3.4 Apache Accumulo/GeoMesa

Accumulo is a sorted, distributed key/value store that provides robust, scalable, high performance service [28]. It was inspired by the BigTable technology from Google, but has improved over time with the introduction of cell-based access control and server-side programming mechanisms that can modify key/value pairs at various points in the data management process. Accumulo is one of the most popular NoSQL databases currently available.

GeoMesa is an open-source, distributed, spatio-temporal index built on top of the Accumulo column family store [29]. Leveraging a highly parallelized indexing strategy, GeoMesa aims to provide high levels of spatial querying and data manipulation to Accumulo. GeoMesa is capable of ingesting, indexing and querying billions of geometrical features. It is integrated with the GeoTools open-source Java library and can act as a data source for GeoServer by the use of bespoke plug-in.

GeoMesa implements an indexing strategy using a geo-hashing algorithm, which combines three dimensions of traffic information (latitude, longitude and time) into a single dimension lexicographic key space in Accumulo.

3.5 GeoServer

GeoServer is an open-source server based on JEE technology that allows users to share, process and edit geospatial data [30]. Designed for interoperability, it allows users to publish data from many major spatial data sources using open standards – typically based upon the Open Geospatial Consortium (OGC). GeoServer implements a range of web-based administration interfaces and associated Rest-based API for easy configuration. When publishing data, e.g. a map layer, a time dimension can be specified so that it is possible to create visualisations with temporal dimensions. Such capabilities are important, e.g. to be able to visualize and process traffic flows at different parts of the day.

IV. SMASH Case Studies and Benchmarking

The SMASH architecture was tested with a range of large-scale data sets from Adelaide and Victoria. The first experiment comprised the processing of 181 million rows of

SCATS data containing all the sensor readings between 2008 and 2014. A cluster of 4 nodes was set up to process those data. Data reliability was provided by HDFS with a replica factor of 3, i.e. 3 copies of every block are held on nodes across the cluster. The virtual machines were all similar and comprised 4-core, 12GB RAM, 30GB Disk – all of which were provided by NeCTAR. The basic SMASH cluster was capable of processing a 15GB dataset without exhausting its resources. The resulting heat map showing traffic flows through Adelaide is shown in Figure 8.

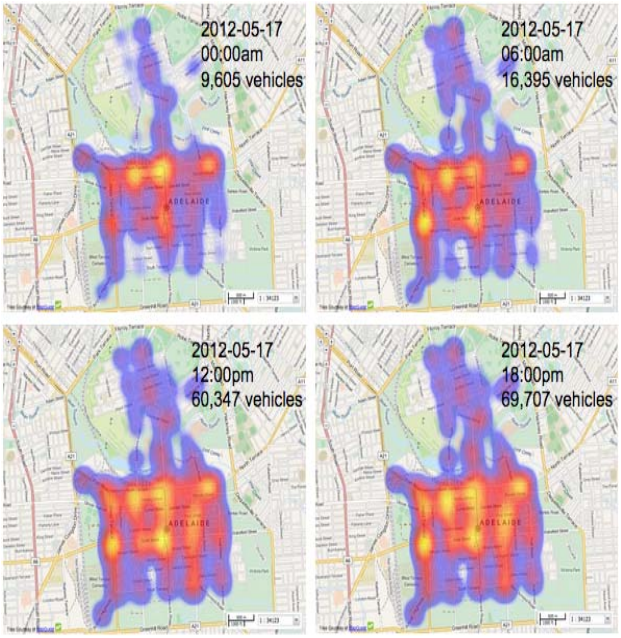

Figure 8: Traffic flows in Adelaide visualised as a heatmap

It is noted that by building an Accumulo store in GeoServer, it is possible to access and visualise aggregated SCATS data directly with different rendering styles. The original idea was to create an animation to illustrate how the traffic changes throughout the day in Adelaide. However GeoServer heatmaps do not support rendering with absolute values acting as the weight. Instead GeoServer publishes maps based upon HTTP requests, hence every time when a user zooms in or zooms out the map, a new request will be sent again to render a new map, which leads to a redistribution of weight percentage. Therefore the change in animation is often not noticeable. Nevertheless, the methods leveraging the OpenLayers API to return a heatmap to the browser can be used to see high-level traffic flow information at different temporal dimensions.

In order to compare the performance of Spark and the traditional Hadoop MapReduce program, two Java programs performing the same aggregation were written for these two environments. The program written to run under Spark took advantage of the Lambda expressions available in Java 8. The performance of both programs is displayed in Table 3.

Running mode, 11GB 22,342,350 rows	Standalone 1 node, 4-core, 12GB RAM	Cluster 4 nodes, 16-core, 48GB RAM
MapReduce, by day	1 h 42 min	2.3 min
Spark, by day	59.8 min	1.3 min

Table 3. Performance comparison of SCATS data aggregated at daily intervals on Standalone and Cluster configurations

From the results given in Table 3, we can observe that the SMASH stack significantly outperforms the traditional Hadoop MapReduce engine. Spark is even more performant when numerous shuffle or iterated operations are involved, since it greatly reduces the time spent on I/O operations.

The original dataset of 77GB spread was over 181,740,678 rows. A filtering operation was applied to reduce invalid or empty readings. The filtering reduces the dataset to 10GB and 22,342,350 rows, which is subsequently used as input to the analytical routines. The actual implementation aggregates vehicle counts from all lanes of each intersection by day, however this can in principle be at any level of aggregation. Intersection locations are then joined to the aggregated data for visualisation of traffic volumes on a map.

An example of the processing results in Accumulo feature store is shown in Table 4. The meaning of this SCATS data row in plain English would read as: *"on July 1st 2014, there were 37,308 vehicle passing over intersection 2580, located at 145.219837East, -37.622730 North").*

siteNo	2580
timestamp	2014-07-01
geom	POINT (145.219837 -37.622730)
vehicleCount	37,308

Table 4. Individual computation result in Accumulo feature store

With such information, a range of detailed analytics using SMASH was undertaken. Figures 9 and 10 illustrate traffic distribution at different times across Victoria. From the results, we observe an 18% increase of traffic across Victoria from 2008 to 2014. Moreover, this increase is not uniformly distributed across the State, as some areas show higher traffic volumes in 2014 than in 2008 (peaks are shown as "warm" colours on the heat-map).

We note that these peaks may be explained by a variety of factors: the population migration and/or economic growth. For example, Melbourne's population is growing at approximately 100,000 per year. Understanding the impact on the road network and the issues of where increased levels of congestion are to arise is an important aspect of SCATS data processing. Similarly, an increase in motor vehicles might imply increased wealth (or poorer public transport systems).

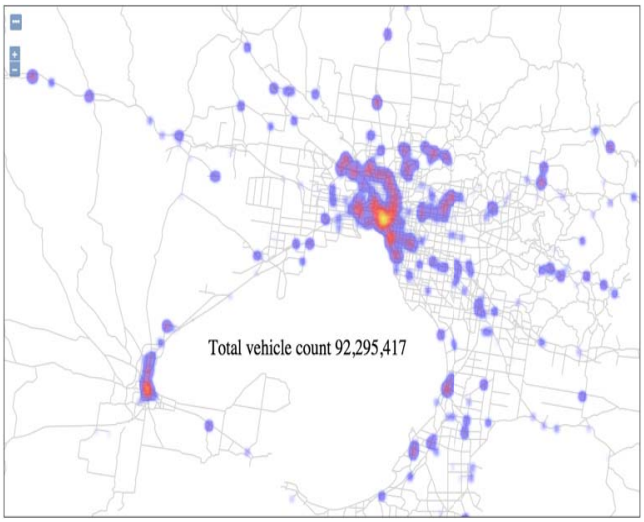

Figure 9: Victoria traffic distribution on July 2008

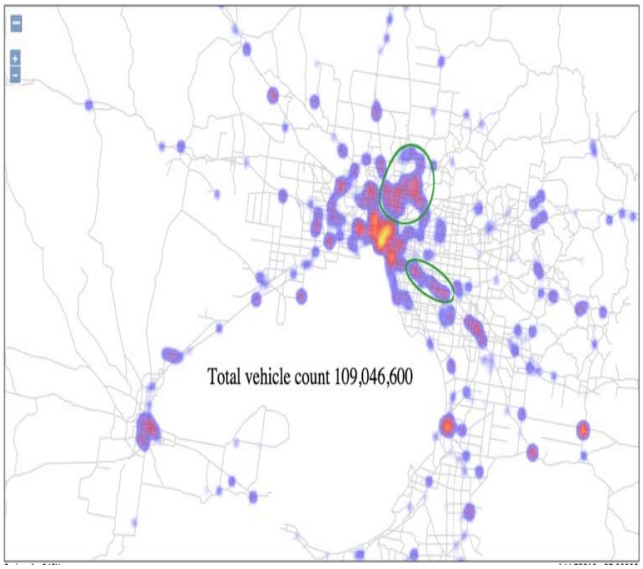

Figure 10: Victorian traffic distribution on July 2014

As seen there are approximately 17 million more vehicles reported through the SCATS data. As shown in Figure 10, new areas of increased volumes of traffic are shown within the green ellipses. These might represent new transport corridors.

It is noted that smaller scale and more localised data analytics is also possible through the SCATS data analytics capabilities offered through SMASH.

Future work will allow researchers to select the areas of interest, e.g. specific regions/suburbs and visualise/analyse the traffic flows specific to that region. Similarly, the SMASH system has been explicitly designed to be scalable and performant. It is directly possible to extend the infrastructure to support larger scale analyses covering all of Australia for example – provided the data exists for such analysis. Future work will also focus on novel ways of presenting changes in traffic volumes and distribution over time by generating animations to show the change of continuous traffic data.

V. Conclusions

In this paper we have presented a big data analytics platform suited for the processing of large-volume data. The focus here was on road traffic data, but the system is quite generic and can be applied with other large-scale data sets. This platform, named SMASH, is composed of: GeoServer, GeoMesa, Accumulo, Spark, Hadoop. We observe that the use of Spark as a processing platform allows significantly faster processing than Hadoop MapReduce, while Accumulo and GeoMesa make the storing of large spatio-temporal datasets highly efficient.

The visualisation of data using maps of different types is handled by GeoServer and is highly flexible with other data (overlays) possible to include. Further research will focus on extending the cluster size to allow real-time data aggregation and area selection; on more advanced visualization, and on the filtering of traffic flows both in space and in time. Access to the live SCATS data resources will also be pursued, however this is very much dependent upon the willingness of the agencies involved, e.g. VicRoads. At present live access to SCATS data is not offered and the data used here was collected using non-web based models (a hard disk was used for data collection/delivery). We continue to work with agencies such as VicRoads (and many others through the AURIN activities) to improve the way in which data can be programmatically accessed and used. This is increasingly happening through open data government initiatives [31-33] and activities such as www.data.gov.au.

Acknowledgments

The authors would like to thank the National eResearch Collaboration Tools and Resources (NeCTAR) project for the Cloud computing resources used here. We thank the VicNode (www.vicnode.org.au) for the data storage systems that were used. We also acknowledge VicRoads for providing the Victorian data and the Australian Urban Research Infrastructure Network (AURIN – www.aurin.org.au) for providing the SCATS data on Adelaide (belonging to the Government of South Australia - Department of Planning, Transport and Infrastructure) that were used as the original basis for this work.

References

[1] A. Jacobs, (2009). "The Pathologies of Big Data". ACMQueue. Available: http://queue.acm.org/detail.cfm?id=1563874

[2] The Economist (2010). "Data, data everywhere". Available: http://www.economist.com/node/15557443

[3] D. Agrawal, D. Sudipto, and A. El Abbadi. "Big data and cloud computing: current state and future opportunities." In Proceedings of the 14th International Conference on Extending Database Technology, pp. 530-533. ACM, 2011.

[4] SCATS: The benchmark in urban traffic control. Available: http://www.scats.com.au/

[5] L. Clifford. Big data: How do your data grow? Nature

455.7209 (2008): 28-29.
[6] T. Hey, A. Trefethen, The Data Deluge: An e-Science Perspective, Grid Computing: Making the Global Infrastructure a Reality (eds F. Berman, G. Fox and T. Hey), (doi: 10.1002/0470867167.ch36)
[7] S. Madden. From databases to big data. Internet Computing, IEEE 16.3 (2012).
[8] M. Nino-Ruiz, et al, Elastic scaling of e-Infrastructures to Support Data-intensive Research Collaborations, International Conference on e-Science 2014, Sao Paolo, Brazil, October 2014.
[9] J.C. Aydos A., O'Brien SCATS Ramp Metering: Strategies, Arterial Integration and Results, Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems, Qingdao, 2014.
[10] S. Clement and J. Anderson, "Traffic signal timing determination: the Cabal model," 1997.
[11] M. Hall and L. Willumsen, "SATURN-a simulation-assignment model for the evaluation of traffic management schemes," Traffic Engineering \& Control, vol. 21, no. 4, 1980.
[12] D. Yin and T. Z. Qiu, "Compatibility analysis of macroscopic and microscopic traffic simulation modeling," Canadian Journal of Civil Engineering, vol. 40, no. 7, pp. 613-622, 2013.
[13] J. Monteil, A. Nantes, R. Billot, J. Sau and others, "Microscopic cooperative traffic flow: calibration and simulation based on a next generation simulation dataset," IET Intelligent Transport Systems, vol. 8, no. 6, pp. 519-525, 2014.
[14] J. Brugmann, M. Schreckenberg and W. Luther, "A verifiable simulation model for real-world microscopic traffic simulations," Simulation Modelling Practice and Theory, vol. 48, pp. 58-92, 2014.
[15] N. Wanichayapong, W. Pruthipunyaskul, W. Pattara-Atikom and P. Chaovalit, "Social-based traffic information extraction and classification," in ITS Telecommunications (ITST), 2011 11th International Conference on, 2011.
[16] A. Ishino, S. Odawara, H. Nanba and T. Takezawa, "Extracting transportation information and traffic problems from tweets during a disaster," Proc. IMMM, pp. 91-96, 2012.
[17] Y. Gong, F. Deng, R.O. Sinnott, Identification of (near) Real-time Traffic Congestion in the Cities of Australia through Twitter, Understanding the City with Urban Informatics, CIKM 2015, Melbourne, Australia, October 2015.
[18] P. Lowrie, "Scats, Sydney Co-ordinated Adaptive Traffic System: A traffic responsive method of controlling urban traffic," 1990.
[19] S. Clement and J. Anderson, "Traffic signal timing determination: the Cabal model," 1997.
[20] E. Mazloumi, G. Rose, G. Currie and S. Moridpour, "Prediction intervals to account for uncertainties in neural network predictions: Methodology and application in bus travel time prediction," Engineering Applications of Artificial Intelligence, vol. 24, no. 3, pp. 534-542, 2011.
[21] J. Dean, and G. Sanjay. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.
[22] G. Sanjay, H. Gobioff, and S.T. Leung. "The Google file system." ACM SIGOPS operating systems review. Vol. 37. No. 5. ACM, 2003.
[23] D. Borthakur. "HDFS architecture guide." Hadoop Apache Project (2008): 53.
[24] T. White, Hadoop: the definitive guide: the definitive guide. " O'Reilly Media, Inc.", 2009.
[25] Databricks. “Advanced Spark”. Available: https://databricks-training.s3.amazonaws.com/slides/advanced-spark-training.pdf
[26] R. Xin. (2014). “Spark officially sets a new record in large-scale sorting”. Available: http://databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html
[27] Apache ZooKeeper. Available: https://zookeeper.apache.org/
[28] Apache Accumulo. Available: https://accumulo.apache.org/
[29] GeoMesa. Available: http://www.geomesa.org/
[30] GeoServer. Available: http://geoserver.org/
[31] M. Janssen, et al, Benefits, adoption barriers and myths of open data and open government. Information Systems Management, 29(4), 2012.
[32] A.S. Hellberg, et al, The story of the sixth myth of open data and open government. Transforming Government: People, Process and Policy, 9(1), 35-51, 2015
[33] E. G. Salvador, et al, A Cloud-based Exploration of Open Data: Promoting Transparency and Accountability of the Federal Government of Australia, Symposium on Information Management and Big Data, Cusco, Peru, September 2014.