

Analytics for NetApp E-Series AutoSupport Data Using Big Data Technologies

Jialiang Zhang



Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-23

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-23.html>

May 1, 2016

Copyright © 2016, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

**Analytics for NetApp E-Series AutoSupport Data
Using Big Data Technologies**

by

Jialiang Zhang

Masters Project Paper

Presented to the Faculty of the Graduate Division of

The University of California at Berkeley

in Partial Fulfillment

of the Requirements

for the Degree of

**Master of Engineering in
Electrical Engineering and Computer Sciences**

Faculty Committee

Professor Lee Fleming, Department of IEOR

Professor Michael Franklin, Department of EECS

The University of California at Berkeley

May 2014

DO NOT CIRCULATE

Acknowledgements

Many thanks to my faulty committee members Professor Lee Fleming and Professor Michael Franklin, my industry advisors Jordan Hambleton and Mittha Manjunath from NetApp, and my teammates Achal Pandey and Huisi Tong.

Abstract

Analytics for NetApp E-Series AutoSupport Data Using Big Data Technologies

Jialiang Zhang

University of California at Berkeley, 2014

Supervisor: Lee Fleming

Our capstone project, utilizing novel Big Data technology, was to help NetApp Inc. develop the AutoSupport (ASUP) Ecosystem for their E-series products [1]. With this software framework, NetApp Inc. was able to collect normalized data, perform predictive analytics and generate effective solutions for its E-series products customers. We used the Star Schema for the data warehousing structure and built seven dimension tables and two fact tables to handle the plethora of E-series ASUP data. To refine our decision and eliminate improper technologies, we made a comparison of many eligible Big Data technologies with respect to their technical strengths and weaknesses. We utilized the latest Spark/Shark Big Data technology developed by Berkeley AMPLab [2] to construct the software framework. Additionally, to perform the featured predictive analytics we used K-means Clustering and K-fold cross-validation machine learning techniques on the normalized data set.

My main contribution in this project was to develop a Python based script to convert the majority of the E-series product's daily/weekly and event-based ASUP logs

into the normalized data format. After performing multiple trials and the overall assessment of both the difficulty and feasibility of different data parsing approaches, I recommended the approach of parsing the text-based data in raw ASUP data set. Based on the normalized data I generated, we then successfully built a prototype. And we expected that with our ASUP framework and predictive data analysis function, NetApp would have more power and efficiency in resolving the E-series product issue for its customer. At the same time, our project on ASUP framework would revolutionize NetApp's data storage and customer support business and help the company exploit its niche market in the Big Data industry.

Table of Contents

List of Tables	vi
List of Figures.....	vii
Chapter 1 Introduction	1
1.1 Company and Products	1
1.2 Project Overview.....	2
1.3 My Contributions	3
Chapter 2 Literature Review.....	4
2.1 Competitors' Strategy	4
2.2 NetApp's Strategy	6
Chapter 3 Methodology.....	8
3.1 ASUP Environment	9
3.2 Dataset.....	10
3.3 Technology Comparison.....	11
3.4 Data Storage Mechanism	12
3.5 Data Parsing and Storing	12
3.6 Data Querying and Insights	13
Chapter 4 Discussion.....	14
4.1 Technology Comparison Matrix	14
4.2 Data Parsing	15
4.3 Star Schema Data Structure	19
Chapter 5 Conclusion	20
Appendix A Data Parser Presentation Slides.....	22
Appendix B Sample Data Parser Source Code	25
Bibliography	43

List of Tables

Table 1:	Existing Landscape of Data Storage and Analysis Market.....	4
Table 2:	Technology Comparison Matrix	14

List of Figures

Figure 1:	NetApp E2600 Storage System	1
Figure 2:	The Size of ASUP Data Set Generated in 24-Hour Period.....	2
Figure 3:	NetApp AutoSupport Infrastructure	9
Figure 4:	AutoSupport Data on ASUP Search Engine	10
Figure 5:	Details of a Sample AutoSupport Data Set	10
Figure 6:	ASUP Data Processing Using Binary -> XML -> Tabular Format Approach	16
Figure 7:	Star Schema Structure	19

Chapter 1: Introduction

1.1 COMPANY AND PRODUCTS

NetApp Inc. is a traditional computer storage and data management company. According to International Data Corporation (IDC), in the second quarter of 2013, NetApp Inc. achieved 13.3% of market share in external disk storage systems [3]. Its major competitors are EMC Corporation, International Business Machines Corporation (IBM), Seagate Technology PLC and Western Digital Corporation (WD).

E-series is NetApp's new product line of conventional storage arrays which receives many attentions in the storage market. E-series is composed of model E2600, E2700, E5400 and E5500, with storage capacity ranging from 768TB to 1536TB [1].

For each individual E-series product, NetApp Inc. integrates



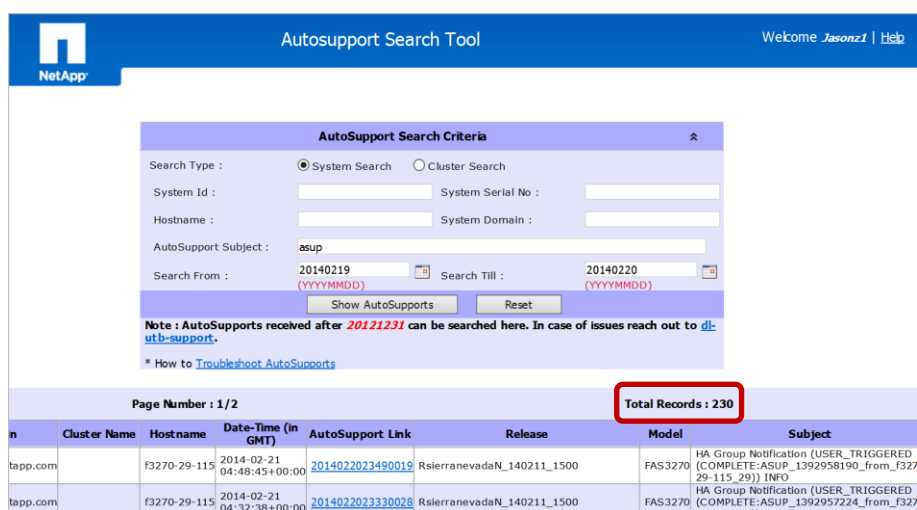
AutoSupport (ASUP) technology with it, in order to efficiently [Figure 1: NetApp E2600 Storage System \[1\]](#) check the health of the system “on a continual basis” [4].

Continual monitoring generated huge amount of AutoSupport data. In this project, our team focused on the NetApp's E-series AutoSupport raw data that were already collected on company's server in Sunnyvale, California.

1.2 PROJECT OVERVIEW

“Big Data” refers to the data that is “large or fast moving” and the current “conventional databases and technologies” are not sufficient enough to analyze them. The advent of Big Data technologies, such as distributed systems and in-memory computing, data repository with SQL compatibility and various machine learning algorithms have successfully “facilitated easier analysis of large amounts of data”.¹ Our capstone project, utilizing novel Big Data technology, is to help NetApp Inc. develop an AutoSupport (ASUP) Ecosystem for their E-series products.

At the customer end, plethora of daily/weekly E-series log files is generated worldwide every day.



n	Cluster Name	Hostname	Date-Time (in GMT)	AutoSupport Link	Release	Model	Subject
tapp.com		f3270-29-115	2014-02-21 04:48:45+00:00	2014022023490019	RserranevadaN_140211_1500	FAS3270	HA Group Notification (USER_TRIGGERED (COMPLETE:ASUP_1392958190_from_f327 29-115_29)) INFO
tapp.com		f3270-29-115	2014-02-21 04:32:38+00:00	2014022023330028	RserranevadaN_140211_1500	FAS3270	HA Group Notification (USER_TRIGGERED (COMPLETE:ASUP_1392957224_from_f327 29-115_29)) INFO

Figure 2: The Size of ASUP Data Set Generated in 24-Hour Period²

The screenshot above presents the size of a typical ASUP log dataset.² Within 24-hour period, there were totally 230 AutoSupport files reported to NetApp headquarter. What is more,

¹Referencing “NetApp Capstone Team Strategy Paper” in Jan., 2014

²Using NetApp internal AutoSupport data search engine

when the E-series storage system encounters an abnormal event, for example, a system level warning or a failure due to disk malfunction, an event-based log will be filed immediately. With this software framework, NetApp is able to capture the significant root cause from multiple warnings or failures reported, perform predictive analysis based on them and generate effective solutions for its customers.

1.3 MY CONTRIBUTION

While working with the other two Master of Engineering students together, my major contribution to the capstone project were as following:

- 1) Helped to investigate and understand the hardware configuration of NetApp's E-series product and how ASUP worked.
- 2) Participated in designing the evaluation matrix for different Big Data technologies.
- 3) Researched one of Big Data technologies – Phoenix from Salesforce.com Inc.
- 4) Participated in building the Star Schema data structure for ASUP data.
- 5) Accomplished ASUP raw log files data parsing and cleaning.
- 6) Generated tables containing necessary information in a normalized format for data repository, and had data cleaned for the team to analyze.

Chapter 2: Literature Review

Admittedly, there are many data storage service providers in the market who are advocate of Big Data technologies. Other than NetApp, EMC Corporation, Cisco Systems, International Business Machines Corporation (IBM), Seagate Technology PLC and Western Digital Corporation (WD) are all storage array solution companies who are potential competitors to NetApp. Table 1 below illustrates their key technologies, product trend, target and user group, and whether they are equipped with predictive ability or not.

Table 1: Existing Landscape of Data Storage and Analysis Market

Competitor	NetApp	EM C	Cisco	IBM	Seagate	WD
Product Name	AutoSupport (ASUP)	Fully Automated Storage Tiering (FAST)	Storage Area Networking (SAN)	Predictive Failure Analysis (PFA)	SimplyRAID™ technology	WD Smart Ware
Product Target	E-series Product	All Product Lines	Network Storage	Hard Drive Storage	NAS Storage	My Book Series
Core Process	Predictive Analysis & Solution	“SP Collect”	Receive – Confirm – Solve – Prevent	Diagnostics Indication	N/A	N/A
Product User	Engineers / Customers	Engineers / Customers	Engineers	Engineers	Customers	Customers
Predictive Ability	Yes	No	No	Yes	No	No

2.1 COMPETITORS’ STRATEGY

As the NetApp Project Strategy Paper emphasizes, huge amount of data requires fast-paced analysis and efficient management, especially in this Big Data Era.¹ To promote “Big Data

analytics”, EMC Corporation developed “Pivotal HD Solution”. In their marketing literature, “pivotal” solution referred to their utilization of Apache Hadoop distribution application, which was advertised as the revolutionary in “Hadoop analytics for unstructured Big Data” [5]. Similarly, as a worldwide leader in networking, Cisco IT chose Hadoop to deliver its commitment that “Enterprise Hadoop architecture, built on Cisco UCS (Unified Computing System) Common Platform Architecture (CPA) for Big Data, unlocks hidden business intelligence” [6]. What is more, in their promotional material, IBM emphasized “Big Data platform”, whose key capabilities included: “Hadoop-based analytics”, “Streaming Computing” and “Data Warehousing”, with prominence on analytic applications of “Business Intelligence” and “Predictive Analytics” [7]. Unwilling to lag behind, traditional storage solution companies were dedicatedly building their own Big Data technology. As Mike Crump, VP of Seagate and Harrie Netel, director of Seagate denoted, “Seagate puts Big Data in action” with the “automated ODT (Outgoing DPPM Test)” and eCube technologies based on its own “Seagate’s Enterprise Data Warehouse (EDW)” [8]. WD (Western Digital), another major disk drive manufacturer, announced that they used Hadoop and Hortonworks to “optimize manufacturing with longer retention of sensor data” [9]. It is predictable that this market will evolve rapidly, and in order to survive, our ASUP ecosystem development for NetApp needs to exploit a niche market in this industry.

2.2 NETAPP'S STRATEGY

For NetApp Inc. the proper use of Big Data technology in our project will have a positive impact on its future business, because the successful deployment of Big Data technology on E-series products will “necessitates secure, robust and low-cost solutions for data storage and management”, as emphasized in NetApp Strategy Paper.¹ When AutoSupport was first introduced in NetApp white paper in 2007, it was highlighted that NetApp would have a more than “65% chance of resolving a customer case in less than one day” instead of only “35% [chances] without AutoSupport data” [10].

On the other hand, as the database structure has become increasingly complex, our strategy for NetApp in the project is a radical evolution in the industry. MapReduce was the milestone in data mining, processing and management, like Dr. Jeff Ullman claimed in his book *Mining of Massive Datasets*, “Implementations of MapReduce enable many of the most common calculations on large-scale data to be performed on computing clusters efficiently” [11]. Later, the MapReduce methodology was integrated with Hadoop Hive, specifically, *HiveQL* “which are compiled into map-reduce jobs executed on Hadoop” as demonstrated by Ashish Thusoo et al. in the paper entitled *Hive - A Warehousing Solution Over a Map-Reduce Framework* in 2011 [12]. Since then, the tool was tailored to handle large data set and was very powerful, and many companies still relied on it. However, we chose to use Berkeley Shark, which was Spark on top of Hadoop Hive with SQL compatibility. One of the special features of Shark was the fact that Shark could implement MapReduce functions approximately a hundred times faster [2], which was an ideal choice for fast-paced big data analysis. As illustrated in Table 1 above, with the help of Berkeley Shark technology, our data analysis function which required the predictive

nature and real-time feature over large-scale data set became feasible. This was innovative and would dramatically improve the user experience of NetApp's customers.

Actually, for all the IT companies in this Big Data era, the key to the success is whether the company can master the advanced technology and seize the opportunity in a niche market. Our project on E-series ASUP framework will revolutionize NetApp's data storage and customer support business and help the company exploit its niche market in the Big Data industry.

Chapter 3: Methodology

One of our tasks in this project was to gain extensive knowledge by researching, analyzing and testing various Big Data technologies for the E-series ASUP framework. Initially, we made our technology selection list with Spark/Shark from Berkeley AMPLab [2], Impala [13] and Parquet [14] from Cloudera, Phoenix from Salesforce.com [15] and Clydesdale from Google and IBM [16]. We then set up various benchmarks to compare these technologies in order to narrow down our list. After we finalized the decision to utilize the latest Berkeley Spark/Shark as our key technology, we developed the data storage schema, constructed the data repository thereafter and parsed the ASUP raw log files into tabular format data for the repository. At the same time, we made progress on Berkeley Shark configuration based on NetApp's computing clusters, with which we could store the large-scale parsed data, perform analysis and offer predictive solutions using machine learning techniques. Since my work is majorly focused on data parsing, this paper will be centered on data processing accordingly.

3.1 ASUP ENVIRONMENT

Figure 3 on the right is a demonstration of ASUP infrastructure from NetApp's AutoSupport documents online [4]. NetApp developed this technology many years ago, and integrated it with several branded product lines in order to continuously and efficiently monitor the health of storage systems. It is achieved by constantly sending ASUP reports back to NetApp headquarter and "My AutoSupport" online platform. As an effective troubleshooting tool, AutoSupport targets both of the NetApp support engineers and product customers.

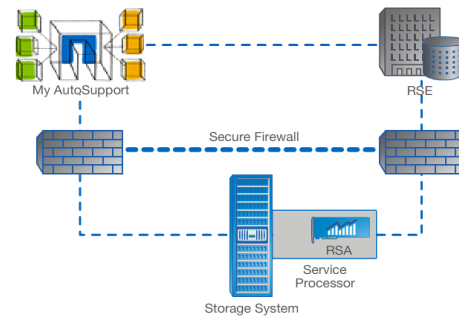


Figure 3: NetApp AutoSupport Infrastructure [4]

Although AutoSupport was already deployed in many other NetApp products, it had not been integrated with NetApp's E-series product line. Since E-series products are becoming one of NetApp's featured products, the company is desired to make this integration accomplished soon. And that is the goal of our capstone project.

3.2 DATASET

How to Troubleshoot AutoSupport

Page Number : 1/2				Total Records : 230			
Domain	Cluster Name	Hostname	Date-Time (in GMT)	AutoSupport Link	Release	Model	Subject
gdl.englab.netapp.com		f3270-29-115	2014-02-21 04:48:45+00:00	2014022023490019	RsierranevadaN_140211_1500	FAS3270	HA Group Notification (US (COMPLETE:ASUP_13929 29-115_29)) INFO
gdl.englab.netapp.com		f3270-29-115	2014-02-21 04:32:38+00:00	2014022023330028	RsierranevadaN_140211_1500	FAS3270	HA Group Notification (US (COMPLETE:ASUP_13929 29-115_52)) INFO
gdl.englab.netapp.com		f3270-29-115	2014-02-21 04:19:31+00:00	2014022023240019	RsierranevadaN_140211_1500	FAS3270	HA Group Notification (US (COMPLETE:ASUP_13929 29-115_30)) INFO
gdl.englab.netapp.com		f3270-29-115	2014-02-21 04:06:45+00:00	2014022023070050	RsierranevadaN_140211_1500	FAS3270	HA Group Notification (US (COMPLETE:ASUP_13929 29-115_75)) INFO
gdl.englab.netapp.com		f3270-29-115	2014-02-21 03:51:54+00:00	2014022022520031	RsierranevadaN_140211_1500	FAS3270	HA Group Notification (US (COMPLETE:ASUP_13929 29-115_49)) INFO
gdl.englab.netapp.com		f3270-29-115	2014-02-21 03:35:10+00:00	2014022022360007	RsierranevadaN_140211_1500	FAS3270	HA Group Notification (US (COMPLETE:ASUP_13929 29-115_66)) INFO
gdl.englab.netapp.com		f3270-29-115	2014-02-21 03:21:57+00:00	2014022022260053	RsierranevadaN_140211_1500	FAS3270	HA Group Notification (US (COMPLETE:ASUP_13929 29-115_37)) INFO
							HA Group Notification (US

Figure 4: AutoSupport Data on ASUP Search Engine

Now returning to the internal AutoSupport data search engine with details of AutoSupport data in Figure 4, it can be easily observed that one storage system can generate multiple AutoSupport reports continuously in just a short period of time. This is a pressure for us to do data cleaning and analysis. Likely, it is due to a hardware failure or a system warning occurred before. But within each of the AutoSupport report, most of the contents are duplicated. Therefore, how to efficiently extract the root cause of the problem becomes significant.

By clicking on one of AutoSupport report links, as demonstrated in Figure 5, one can see the details of this AutoSupport dataset and can download the whole dataset from the link. The size of such dataset varies greatly from a few megabytes to several hundreds of megabytes in total, depending on how large the storage system is and whether the AutoSupport data is a daily log or a system warning type.

NetApp

Autosupport Console

Welcome JasonZ | [Help](#)

System Dashboard

AutoSupport Id :
2014022023070050

System Version :
R8.2.2xN_140422_1823

System Id :
1574093258

Hostname :
f3270-29-115

System Serial No : 700000751320

AutoSupport Subject : HA Group Notification (USER_TRIGGERED (COMPLETE:ASUP_1392955670_from_f3270-29-115_75)) INFO

System Model : FAS3270

Recent AutoSupports

Surrounding AutoSupports

Top Sections

DE

FC-DEVICE-MAP

HEADERS

All Sections - Alpha Sort

Section Name : HEADERS

From UNKNOWN (HTTP PUT) Thu Feb 20 20:07:38 2014 (608377)
Received: from UNKNOWN via VMWUTBXTA01-STG (HTTP PUT) for spiderX.corp.netapp.com; 20 Feb 2014
X-Forwarded-For: 10.62.33.60
X-Netapp-asup-version: 3
X-Netapp-asup-content: complete
X-Netapp-asup-subject: HA Group Notification from f3270-29-115 (USER_TRIGGERED (COMPLETE:ASUP_1392955670_from_f3270-29-115_75))
X-Netapp-asup-system-id: 1574093258
X-Netapp-asup-os-version: NetApp Release RsierreanvedaN_140211_1500 7-Mode: Tue Feb 11 15:18:00 2014
X-Netapp-asup-hostname: f3270-29-115
X-Netapp-asup-generated-on: Fri Feb 21 04:06:45 GMT 2014
X-Netapp-asup-serial-num: 700000751320
X-Netapp-asup-partner-system-id: 1574093029
X-Netapp-asup-partner-hostname: f3270-29-114

Figure 5: Details of a Sample AutoSupport Data Set

These are the raw datasets that we used for our capstone project. With access to the NetApp’s repository of AutoSupport raw data, we can continuously collect these data globally. However, to process and integrate the huge dataset demands novel Big Data technologies rather than traditional database and data management solutions.

3.3 TECHNOLOGY COMPARISON

We made technical comparison of five eligible Big Data technologies, namely Berkeley Spark/Shark, Cloudera Impala and Parquet, Salesforce.com Phoenix and Google Clydesdale. They all have various advantages and disadvantages. And one of our tasks in this project was to narrow down this list, and made a final decision on which technology we were going to use to

construct the framework. In order to achieve that goal, we did research on their hardware limitations and computing constraints one by one, and list our evaluation standards and results to examine each single technology.

3.4 DATA STORAGE MECHANISM

In order to efficiently organize and store all of the normalized data, we utilized the Star Schema data structure. The Star Schema consisted of fact tables and dimension tables, in which fact tables stored the central metrics and information, whereas dimension tables were data warehouse linked to the fact tables. Based on our design, we decided to construct:

Two Fact Tables: one for the ASUP fact table containing all of the keys, and the other one was for all of the unstructured data

Seven Dimension Tables: for the Controller, Drive, Drive_Error, Storage_Array, Tray, Drawer and Major_Event_Log separately

3.5 DATA PARSING AND STORING

After choosing Berkeley Spark/Shark technology, it was important to install and configure it properly on the NetApp's company computing cluster. Our computing cluster consisted of one master node and three worker nodes. And we installed the Berkeley Spark/Shark with the latest release on February 2014 on all of the cluster nodes. With that

accomplished, I began to work on data parser, convert the ASUP raw data into tabular format to store in data repository.

3.6 DATA QUERYING AND INSIGHTS

Last but not least, we spent time and effort on identifying example use cases for NetApp's E-series products, and generating insightful data queries. Because this was one of our key tasks for the project, we wanted to offer valuable and predictive solutions for our customer.

A simple use case would be to collect any drive errors from one system, performing analysis on its system configuration, record of repairing, capacity usage and device running time etc., aggregating similar errors and identifying the root cause, and predicting what the next time that the potential failure would occur. We applied K-means clustering and K-fold cross-validation machine learning algorithms on our dataset and generated insightful conclusions accordingly.

Chapter 4: Discussion

4.1 TECHNOLOGY COMPARISON MATRIX

Table 2 below presents the technology comparison results we concluded for five major advanced Big Data technologies.

Table 2: Technology Comparison Matrix [17]

Name	Spark	Impala	Phoenix	Parquet	Clydesdale
Company	UCB/Apache	Cloudera	Salesforce	Cloudera/Twitter	Google/IBM
Ease of Setup	Easy ✓	Easy ✓	Easy ✓	Medium	Hard ✗
Compatibility	Hive ✓	Hive ✓	HBase ✓	Hadoop ✓	✗
SQL Like	✓	✓	✓	✗	✗
Star Schema	✓	✓	✗	✗	✗
Unstructured Data	✓	Not for Non-scalar Data	✓	✗	✗
Accelerated Storage Format	✗	Columnar ✓	✗	Columnar ✓	Columnar ✓
Bulk Data Load	✓	✗	✓	✓	✓
In-memory	✓	✗	✗	✗	✗
UDF	✓	✗	✗	✗	✗
Predictive Analytics	✓	✓	✗	✗	✗
Available APIs	Java, Python, Scala	Java	Java	Java	Java
Maturity	High ✓	Medium	Medium	Medium	Low ✗
Note	In-memory computing, faster data queries, ideally suited for machine learning	Best integration with Parquet	Table Join function not available in Phoenix Version 2.1.2	Requires extensive configuration, query dependent, not suitable for multiple queries	Still a research prototype. Performance varies depending on query type
Users	IBM, Yahoo!, Intel, Groupon	Cloudera	Salesforce	Salesforce, Cloudera, Twitter	Google, IBM
Our Choice	✓	✗	✗	✗	✗

The evaluation standards are:

1. Company: The entity who developed and supported such technology
2. Ease of Setup: To measure how easy it is for users to setup and configure such technology
3. Compatibility: To examine whether such technology is compatible with HIVE/HBase/Hadoop
4. SQL Like: To examine whether such technology has the SQL skin, which is easy to develop
5. Star Schema: To examine whether such technology supports “Star Schema”
6. Unstructured Data: To survey how well such technology handles the unstructured data like txt
7. Accelerated Storage Format: To identify if such technology utilizes Columnar data format
8. Bulk Data Load: To examine whether such technology supports large size data bulk loading
9. In-memory: To observe if such technology has the function of in-memory computation
10. UDF: To examine if such technology has the User Defined Function features
11. Predictive Analytics: To survey whether such technology has the predictive analytics function
12. Available APIs: To examine what APIs it supports, like Java, Python or Scala
13. Maturity: To measure how mature such technology is, in the level of High, Medium and Low
14. Extra Note: Other significant features, functions, releasing or development notes
15. Users: The example of companies/entities who utilize or deploy such technology

4.2 DATA PARSING

In company’s ASUP repository, all of the raw log files were in the format of .snappy file, which was a unique, but unstructured data format for ASUP log files. One of my major tasks in this capstone project was to parse the raw log files and extract valuable data from them. At the early stage of our project, we discovered that there was a binary file entitled *object-bundle.bin* in the log file jar. Utilizing an internal java-based parser, we could convert these binary files into semi-structured xml files for preliminary analysis. However, xml file was not valuable to us, because this type of data format was not compatible with databases and none of the machine learning algorithms could be applied upon. We needed to further normalize these data and

convert them into tabular format, then store them into our databases residing on powerful computing clusters, aggregate them further to perform the predictive analysis using modern machine learning algorithms and finally generate insightful solutions. To achieve these goals, we devoted our effort on creating a new parser based on Python, to convert these xml files into csv (comma-separated values) format with organized data in it.

Hence, in consideration of the efficiency of our AutoSupport ecosystem, we needed to explore an alternative approach. We found that there were many text files within the same ASUP log jar. Though they are all unstructured data, those text-based data contain almost equally sufficient and valuable

Figure 6: ASUP Data Processing Using Binary -> XML -> Tabular Format Approach

information as the *object-bundle.bin* binary files. So we decided to set aside the previous approach, and begin to develop a new parser aiming to parse these text-based data. This process is clearly illustrated in Appendix A.

I wrote a parser based on Python, which took in the text files in ASUP log file jar, and generated all the seven dimension tables and two fact tables automatically. The parser would extract all the key words in the text file, like *ASUP_ID*, *ASUP_GenDate*, *Storage Array_Cache Flushing Rate*, *Drive_Date of Manufacture*, *Drive Error_Total Command Count*, *Major Event Log_Sequence Number* etc. as column names in the table, and the associated value or description to those key words would be stored in a tabular format in a csv file, for example, *DIM_MAJOR_EVENT_LOG.csv*.

As discussed in section 3.2 and Figure 4, the same storage system could continuously generate multiple ASUP reports in a short period of time. These datasets were mostly alike to each other, so to simply bulk load those in the data repository without proper process might cause overwriting problems. To deal with such issue, I utilized the “Partitioning” function in Hive, as well as Shark (because Shark was Spark on top of Hive), to solve this issue. ASUP generation date was in the format of *yyyy/mm/dd/hh/mm/ss*, for example, 20130327002204. Since it had sufficient precision as a 14-digit integer, we decided to use it as the partition field to differentiate distinct ASUP data, or the data generated by the same ASUP but in different time period. In the parser specifically, I added:

```
# Generate partitioned field names date_partition, which is equal to ASUP_GenDate
g_majorEventLog['date_partition'] = g_asup['ASUP_GenDate']
```

And when defining a table in the data repository, the DDL (Data Definition Language) would be modified as:

```
CREATE TABLE DIM_ASUP (ASUP_Content STRING, ASUP_Subject STRING, ASUP_SystemID
STRING, ASUP_ID STRING, ASUP_ClientID STRING, ASUP_OSVersion STRING, ASUP_GenDate
STRING) PARTITIONED BY (date_partition STRING) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' STORED AS TEXTFILE;
```

Similarly, when loading data into the database, we simply needed to add an extra partition command:

```
LOAD DATA LOCAL INPATH '/home/3bears/src/table/DIM_DRIVE_ERROR.csv' INTO TABLE
DIM_DRIVE_ERROR PARTITION (date_partition = '20130204200101');
```

Finally, we could write a query against it to manipulate data and generate insights, for example:

```
SELECT Drive_ID, Log FROM DIM_DRIVE_ERROR f1 JOIN FACT_UNSTRUCTURED_LOG f2
ON (f1.ASUP_ID = f2.ASUP_ID) WHERE Log LIKE "%log3%";
```

4.3 STAR SCHEMA DATA STRUCTURE

I also participated in designing the data structure for the repository. Figure 7 below is the sample Star Schema we created for ASUP data warehousing:

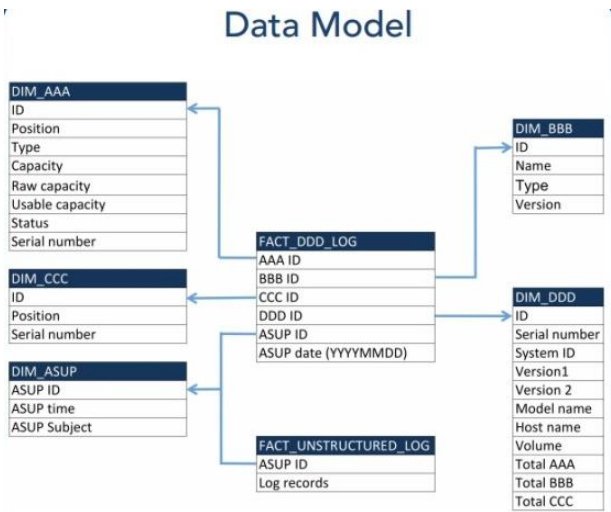


Figure 7: Star Schema Structure [17]

As discussed in previous sections, we designed seven dimension tables and two fact tables to implement the Star Schema structure for ASUP data warehousing. Large-scale ASUP data were all stored following this structure on computing cluster. For different dimension tables, we used either *Drive_ID*, *Tray_ID*, *Drawer_ID* or *Storage_Array_ID* as

the primary key to link to the two fact tables. And as claimed above, each table contained the column '*date_partition*' as the partition field when storing in data repository.

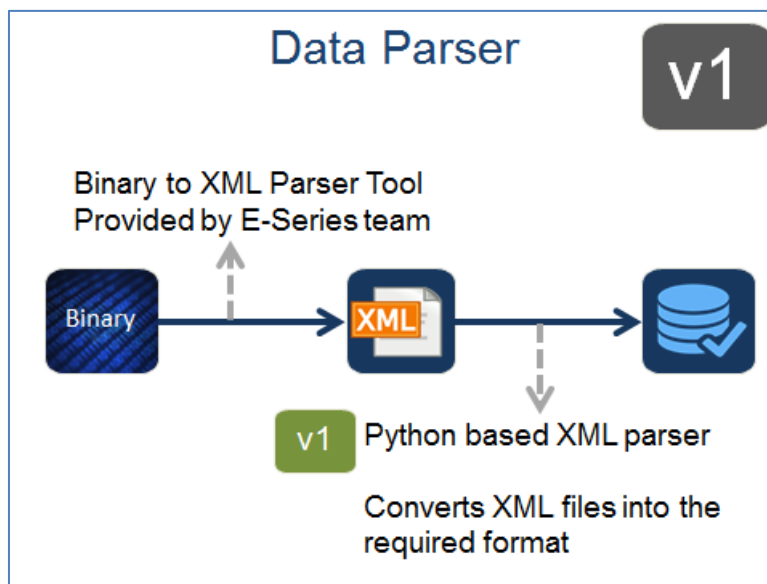
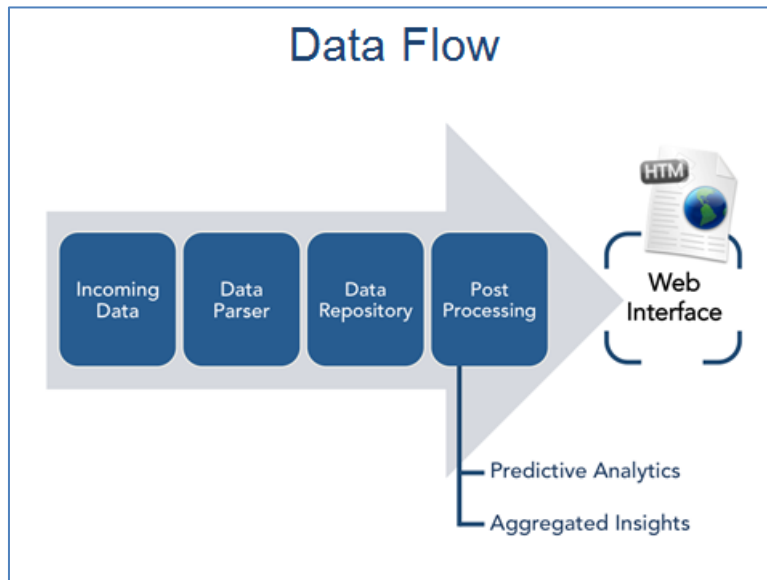
Chapter 5: Conclusion

At this time, we have successfully designed the data structure, configured Spark/Shark on computing clusters, had the majority of ASUP data parsed and cleaned for use, and generated several use cases insights based on machine learning algorithm already.

To review my part of work, the biggest difficulty I encountered was to parse the unstructured text-based AutoSupport data. Yet at the same time, I realized that it was important to have a clean and normalized dataset generated for machine learning application, user interface development and future predictive analysis. The difficulty lay in the fact that my parser might work well on one version of AutoSupport, but turn out to be a failure totally when testing on other AutoSupport versions, simply due to the new lines added in other AutoSupport versions. To overcome this difficulty, I had to run and test my parser on multiple AutoSupport versions one by one. Fortunately, I found out that all of the AutoSupport reports were in a “normalized” format to some degree. The total amount of information, or the column names in tabular data format after conversion, was set and fixed. The only difference was that some AutoSupport versions tended to omit certain hardware information, which might not be configured in the storage system. Therefore, I drawn the conclusion that to develop a parser for such unstructured text format data, it was important to aggregate all of the possible information first, no matter whether it existed in the current dataset or not. Furthermore, some parsing techniques, like the look-up table mechanism, should be used to parse the dataset completely, instead of parsing the dataset line by line or by searching key words in it.

In the future, continued work can be done in the areas of implementing more machine learning algorithms on the whole set of E-series AutoSupport data, constructing a friendly user interface for potential customers, and continuing working to make the E-series AutoSupport ecosystem more efficient and robust.

Appendix A: Data Parser Presentation Slides ³



³Referencing “NetApp Capstone Team Final Presentation” in May, 2014

Data Parser

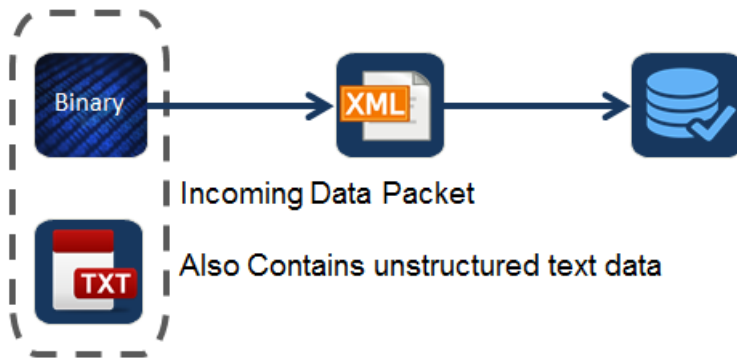
v1

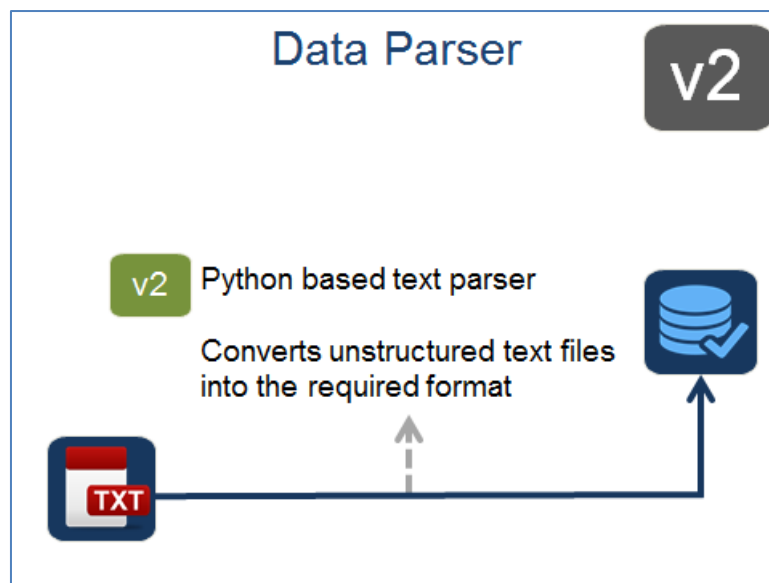
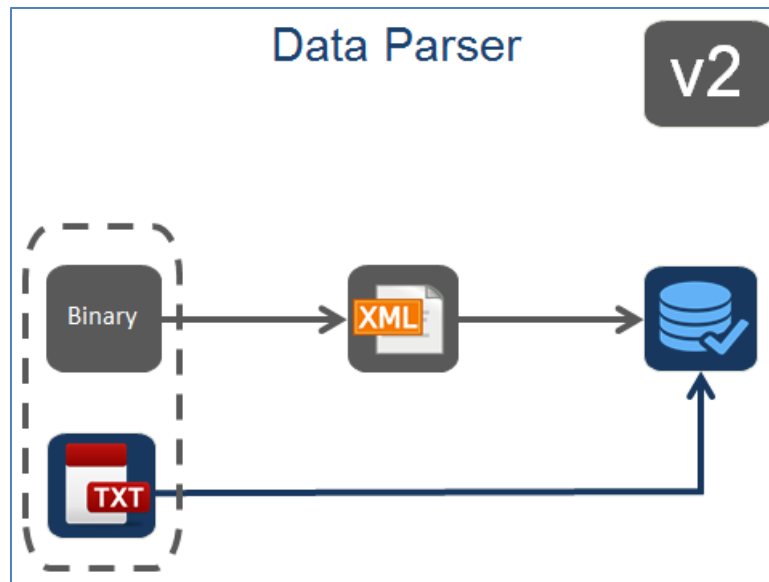
Binary to XML Parser Tool

Did **NOT** work for all ASUP versions!



Data Parser





Appendix B: Sample Data Parser Source Code

```
import sys
import re
import csv
from dateutil import parser as dateParser

nameIndex = 1

try:
    PATH_data_folder = sys.argv[1]
except:
    print "Please Specify the Correct Working Directory to the Data Files"
if PATH_data_folder[-1] != '/':
    PATH_data_folder = PATH_data_folder + '/'

## ~~~~~ header.txt Parser ~~~~~
asupMap={
    'X-Netapp-asup-content': 'ASUP_Content',
    'X-Netapp-asup-subject': 'ASUP_Subject',
    'X-Netapp-asup-system-id': 'ASUP_SystemID',
    'X-Netapp-asup-os-version': 'ASUP_OSVersion',
    'X-Netapp-asup-hostname': 'ASUP_Hostname',
    'X-Netapp-asup-generated-on': 'ASUP_GenDate',
    'X-Netapp-asup-sequence': 'ASUP_Sequence',
    'X-Netapp-asup-serial-num': 'ASUP_Serial',
    'X-Netapp-asup-model-name': 'ASUP_Model',
    'X-Netapp-asup-payload-checksum': 'ASUP_Payload',
    'X-Netapp-asup-contents-truncated': 'ASUP_ContentTruncated',
    'X-Netapp-asup-proxy-agent-id': 'ASUP_ProxyAgent',
    'X-Netapp-asup-client-id': 'ASUP_ClientID',
    'X-Netapp-asup-oem-id': 'ASUP_OEMID',
    'Received': 'ASUP_Received',
    'X-Forwarded-For': 'ASUP_Forwarded_For'}

g_asup = {}

for l in open(PATH_data_folder + 'header.txt'):
    #for l in open('/home/saasbook/ASUP/E-series_Data1/' + 'header.txt'):
        if 'From' in l: continue
        if 'Subject' in l: continue
        tmpL = l.split(":")
        try:
            g_asup[asupMap[tmpL[0]]]=str.strip(":".join(tmpL[1:]))
        except:
            print "Error parsing 'header.txt' in line [" + l + "]"
```

```

g_asup['ASUP_GenDate'] =
dateParser.parse(g_asup['ASUP_GenDate']).strftime("%Y%m%d%H%M%S")

### DIM_ASUP table

g_asup['ASUP_ID'] = g_asup['ASUP_GenDate'] + "_" + g_asup['ASUP_SystemID']
# Generate partitioned field names date_partition, which is equal to ASUP_GenDate
g_asup['date_partition']=g_asup['ASUP_GenDate']

DIM_ASUP_table = 'DIM_ASUP_{0}.csv'.format(nameIndex)

fileHeader = open(DIM_ASUP_table,'w')
dimAsup = csv.DictWriter(fileHeader,fieldnames = g_asup.keys())

dimAsup.writeheader()
dimAsup.writerow(g_asup)

fileHeader.close()

## ~~~~~ storage-array-profile.txt Parser ~~~~~

sapString = open(PATH_data_folder + 'storage-array-profile.txt').read();
#sapString = open('/home/saasbook/ASUP/E-series_Data1/' + 'storage-array-
profile.txt').read();
#sapString.split('\n')

### DIM_STORAGE_ARRAY table

tmpIndex = sapString.find("STORAGE ARRAY-----")

endIndex = re.search("Power Supplies\s\s\s", sapString).start()

saMap={
'Storage array world-wide identifier (ID)':'General_Storage array world-wide
identifier (ID)',
'Chassis Serial Number':'General_Chassis Serial Number',
'Event configuration data version':'General_Event configuration data version',
'Start cache flushing at':'Cache_Start cache flushing at',
'Start demand cache flushing at':'Cache_Start demand cache flushing at',
'Stop cache flushing at':'Cache_Stop cache flushing at',
'Cache block size':'Cache_Cache block size',
'Media scan frequency':'Cache_Media scan frequency',
'Failover alert delay':'Cache_Failover alert delay',
'Status':'AutoSupport_Summary_Status',
'Daily schedule':'AutoSupport_Summary_Daily schedule',
'Weekly schedule':'AutoSupport_Summary_Weekly schedule',
'Target name':'iSCSI_Target name',
'Target alias':'iSCSI_Target alias',
'Target authentication':'iSCSI_Target authentication',

```

'CHAP secret defined': 'iSCSI_Target authentication_CHAP secret defined',
 'iSNS server registration': 'iSCSI_Target discovery_iSNS server registration',
 'Registration state': 'iSCSI_Target discovery_Registration state',
 'IPv4 Configuration': 'iSCSI_Target discovery_IPv4 Configuration',
 'IPv4 IP address': 'iSCSI_Target discovery_IPv4 IP address',
 'IPv6 IP address': 'iSCSI_Target discovery_IPv6 IP address',
 'TCP listening port': 'iSCSI_Target discovery_TCP listening port',
 'Unnamed discovery sessions': 'iSCSI_Target discovery_Unnamed discovery sessions',
 'Tag': 'Core_Dump_Tag',
 'Time collected': 'Core_Dump_Time collected',
 'Retrieved': 'Core_Dump_Retrieved',
 'Disk pools': 'Storage_Disk pools',
 'Volumes on Disk Pools': 'Storage_Disk Pools_Volumes on Disk Pools',
 'Volume groups': 'Storage_Volume groups',
 'RAID 0 Volume Groups': 'Storage_Volume groups_RAID 0 Volume Groups',
 'RAID 1 Volume Groups': 'Storage_Volume groups_RAID 1 Volume Groups',
 'RAID 2 Volume Groups': 'Storage_Volume groups_RAID 2 Volume Groups',
 'RAID 3 Volume Groups': 'Storage_Volume groups_RAID 3 Volume Groups',
 'RAID 4 Volume Groups': 'Storage_Volume groups_RAID 4 Volume Groups',
 'RAID 5 Volume Groups': 'Storage_Volume groups_RAID 5 Volume Groups',
 'RAID 6 Volume Groups': 'Storage_Volume groups_RAID 6 Volume Groups',
 'Access volumes': 'Storage_Access volumes',
 'Standard Volumes (Used/Allowed)': 'Storage_Standard Volumes (Used/Allowed)',
 'Base': 'Storage_Standard Volumes_Base',
 'Repository': 'Storage_Standard Volumes_Repository',
 'Mirror repositories': 'Storage_Standard Volumes_Mirror repositories',
 'Thin Volumes (Used/Allowed)': 'Storage_Thin Volumes (Used/Allowed)',
 'Total volumes using SSD cache': 'SSD_Cache_Total volumes using SSD cache',
 'Maximum system-wide capacity allowed': 'SSD_Cache_Maximum system-wide capacity allowed',
 'Volume copies': 'Copy_Services_Volume copies',
 'Snapshot Groups': 'Copy_Services_Snapshot Groups',
 'Snapshot Images': 'Copy_Services_Snapshot Images',
 'Snapshot Volumes': 'Copy_Services_Snapshot Volumes',
 'Consistency Groups': 'Copy_Services_Consistency Groups',
 'Member Volumes': 'Copy_Services_Member Volumes',
 'Asynchronous mirror groups': 'Copy_Services_Asynchronous mirror groups',
 'Asynchronous mirrored pairs': 'Copy_Services_Asynchronous mirrored pairs',
 'Remote Volume Mirroring mirrored pairs': 'Copy_Services_Remote Volume Mirroring mirrored pairs',
 'Snapshot volumes(legacy)': 'Copy_Services_Snapshot volumes(legacy)',
 'Access volume': 'Host_Mapping_Access volume',
 'Default host OS': 'Host_Mapping_Default host OS',
 'Mapped volumes': 'Host_Mapping_Mapped volumes',
 'Unmapped volumes': 'Host_Mapping_Unmapped volumes',
 'Trays': 'Hardware_Number of Trays',
 'System configured to use batteries': 'Hardware_Trays_System configured to use batteries',
 'Controllers': 'Hardware_Number of Controllers',
 'Redundancy mode': 'Hardware_Controllers_Redundancy mode',
 'Drives': 'Hardware_Number of Drives',

```

'Current drive media types':'Hardware_Drives_Current drive media types',
'Current drive interface type(s)':'Hardware_Drives_Current drive interface type(s)',
'Total hot spare drives':'Hardware_Drives_Total hot spare drives',
'Standby':'Hardware_Drives_Total hot spare drives_Standby',
'In use':'Hardware_Drives_Total hot spare drives_In use',
'Drive security key identifier':'Hardware_Drive security key identifier',
'Feature enable identifier':'Features_Feature enable identifier',
'Feature pack':'Features_Feature pack',
'Feature pack submodel ID':'Features_Feature pack submodel ID',
'Snapshot groups allowed per base volume':'Features_Snapshot groups allowed per base
volume',
'Snapshots(legacy) allowed per base volume':'Features_Snapshots(legacy) allowed per
base volume',
'Volumes allowed per storage partition':'Features_Volumes allowed per storage
partition',
'AMW Version':'Firmware_SANtricity ES_AMW Version',
'Report Date':'Firmware_SANtricity ES_Report Date',
'Storage Array Name':'Firmware_Storage Array_Name',
'Current Package Version':'Firmware_Storage Array_Current Package Version',
'Current NVSRAM Version':'Firmware_Storage Array_Current NVSRAM Version',
'Staged Package Version':'Firmware_Storage Array_Staged Package Version',
'Staged NVSRAM Version':'Firmware_Storage Array_Staged NVSRAM Version',
'Location':'Firmware_Controllers_Location',
'Current Package Version':'Firmware_Controllers_Current Package Version',
'Current NVSRAM Version':'Firmware_Controllers_Current NVSRAM Version',
'Board ID':'Firmware_Controllers_Board ID',
'Sub-Model ID':'Firmware_Controllers_Sub-Model ID'}

```

```
DIM_STORAGE_ARRAY_sap = {}
```

```
stringStorageArray = sapString[tmpIndex:endIndex]
```

```

if 'Snapshot groups allowed per base volume (see note below)' in stringStorageArray:
    stringStorageArray = stringStorageArray.replace('Snapshot groups allowed per base
volume (see note below)', 'Snapshot groups allowed per base volume', 1)
elif 'Snapshot groups allowed per base volume (see note)' in stringStorageArray:
    stringStorageArray = stringStorageArray.replace('Snapshot groups allowed per base
volume (see note)', 'Snapshot groups allowed per base volume', 1)

```

```

for l in stringStorageArray.split('\n'):
#for l in open('/home/saasbook/ASUP/E-series_Data1/' + 'storage-array-profile.txt'):
    if 'PROFILE FOR STORAGE ARRAY' in l: continue
    if 'Note:' in l: continue
    if 'Location:' in l: continue
    if ':' not in l: continue
    tmpL = l.strip().split(":")
    try:
        DIM_STORAGE_ARRAY_sap[saMap[tmpL[0]]]=str.strip(":").join(tmpL[1:])
    except:
        print "Error parsing 'storage-array-profile.txt' in line [" + l + "]"

```

```

DIM_STORAGE_ARRAY_sap['ASUP_ID'] = g_asup['ASUP_ID']
# Generate partitioned field names date_partition, which is equal to ASUP_GenDate
DIM_STORAGE_ARRAY_sap['date_partition']=g_asup['ASUP_GenDate']

DIM_STORAGE_ARRAY_table = 'DIM_STORAGE_ARRAY_{0}.csv'.format(nameIndex)

fileStorageArray = open(DIM_STORAGE_ARRAY_table,'w')
dimStorageArray = csv.DictWriter(fileStorageArray,fieldnames =
DIM_STORAGE_ARRAY_sap.keys())

dimStorageArray.writeheader()
dimStorageArray.writerow(DIM_STORAGE_ARRAY_sap)

fileStorageArray.close()

# ~~~~~~

## ~~~~~~ drive-error.txt Parser ~~~~~~

erString = open(PATH_data_folder + 'controller-drive-error-event-log.txt').read();
#erString = open('/home/saasbook/ASUP/E-series_Data1/' + 'controller-drive-error-
event-log.txt').read();
erString = erString + '\n' + 'St '
#print erString[-100:], '#####'

DIM_DRIVE_ERROR = {}

recordOfExistingTrayNum = []
recordOfExistingDrawerNum = []

#DRIVEERRORfieldName = ['Devnum','Location','Role','Rank','Total CmdCount','Total
Error Count','Total Event Count']

### DIM_DRIVE table

DIM_DRIVE_sap = {}

tmpIndex = sapString.find("DRIVES-----")

endIndex = sapString.find("DRIVE CHANNELS-----")

TotalNumDrives = re.findall("Number of drives: (.*)\n",
sapString[tmpIndex:endIndex])[0].strip()
print "The total number of Drives in this system is: ", TotalNumDrives, '\n',
"*****"

tmpIndex = tmpIndex + sapString[tmpIndex:endIndex].find("DETAILS")
nextIndex = tmpIndex

```

```

DRIVEfieldName = ['Status','Mode','Raw capacity','Usable capacity','World-wide
identifier','Associated volume group','Associated disk pool','Media type','Interface
type','Drive path redundancy','Wear life monitoring','Average erase count','Spare
blocks remaining','Drive capabilities','Security Capable','Secure','Read/write
accessible','Drive security key identifier','Speed','Current data rate','Logical
sector size','Physical sector size','Product ID','Drive Firmware Version','FPGA
version','Serial number','Manufacturer','Date of manufacture']
DRIVEfieldName.sort()

DIM_DRIVE_table = 'DIM_DRIVE_{0}.csv'.format(nameIndex)
DIM_DRIVE_ERROR_table = 'DIM_DRIVE_ERROR_{0}.csv'.format(nameIndex)

fileDrive = open(DIM_DRIVE_table,'w')
fileDriveError = open(DIM_DRIVE_ERROR_table,'w')

checkIfDrawerMissing = sapString[tmpIndex:endIndex].find('Drawer')

## For system with NO Drawers configuration:
if checkIfDrawerMissing == -1:
    print "This system has NO Drawers configuration, the corresponding column in the
DIM_DRIVE table will be filled with 'na'. \n"
    for trayNum in [0,1,2,3,4,5,6,7,8,9,10,99]:
        for slotNum in range(36):
            slotNum += 1

            searchString = "Drive at Tray " + str(trayNum) + ", Slot " + str(slotNum)
+ "\n"

            checkIfExist = sapString[tmpIndex:endIndex].find(searchString)
            if checkIfExist == -1:
                continue
            else:
                recordOfExistingTrayNum.append(trayNum)

            # Generate partitioned field names date_partition, which is equal to
ASUP_GenDate
            DIM_DRIVE_sap['date_partition'] = g_asup['ASUP_GenDate']

            tmpIndex = tmpIndex + checkIfExist
            DIM_DRIVE_sap['Tray Number'] = trayNum
            #DIM_DRIVE_sap['Tray Number'] = re.findall("Tray (.?)*",
sapString[tmpIndex:endIndex])[0].strip()
            DIM_DRIVE_sap['Drawer Number'] = 'na'
            DIM_DRIVE_sap['Slot Number'] = slotNum
            for field in DRIVEfieldName:
                if field not in sapString[tmpIndex:endIndex]:
                    ###print '\n\n', "Cannot find field '{0}' in the 'storage-
array-profile.txt' file, \nin the corresponding column in DIM_DRIVE table, it will be
filled with 'na'. \n".format(field)
                    DIM_DRIVE_sap[field] = 'na'
                else:

```

```

        DIM_DRIVE_sap[field] = re.findall(field + ": (.*?)\n",
sapString[tmpIndex:endIndex])[0].strip()
        if 'Data Assurance (DA) capable' not in sapString[tmpIndex:endIndex]:
            print '\n\n', "Cannot find field 'Data Assurance (DA) capable' in
the 'storage-array-profile.txt' file. \nin the corresponding column in DIM_DRIVE table,
it will be filled with 'na'. \n"
            DIM_DRIVE_sap['Data Assurance (DA) capable'] = 'na'
        else:
            DIM_DRIVE_sap['Data Assurance (DA) capable'] = re.findall("Data
Assurance \ (DA\ ) capable: (.*?)\n", sapString[tmpIndex:endIndex])[0].strip()

        # Generate Drive ID
        DIM_DRIVE_sap['Drive ID'] = g_asup['ASUP_SystemID'] + '_' +
DIM_DRIVE_sap['Serial number']

        if trayNum == min(recordOfExistingTrayNum) and slotNum == 1:
            dimDrive =
csv.DictWriter(fileDrive,fieldnames=DIM_DRIVE_sap.keys())
            dimDrive.writeheader()
            dimDrive.writerow(DIM_DRIVE_sap)

### DIM_DRIVE_ERROR table

        # Generate partitioned field names date_partition, which is equal to
ASUP_GenDate
        DIM_DRIVE_ERROR['date_partition'] = g_asup['ASUP_GenDate']

        DIM_DRIVE_ERROR['Serial number'] = DIM_DRIVE_sap['Serial number']

        # Generate Drive ID
        DIM_DRIVE_ERROR['Drive ID'] = g_asup['ASUP_SystemID'] + '_' +
DIM_DRIVE_ERROR['Serial number']

        twoControllerIndexER = []

        # Deal With special cases: "Retrieved on Controller A/B at Not
Available"

        #ifControllerAExists = re.search("Retrieved on Controller A at
(.*?)\r\n\r\nError", erString)
        ifControllerAExists = re.search("Retrieved on Controller A at
(.*?)\n\nError", erString)
        if ifControllerAExists == None:
            locationindexER1 = erString.find(DIM_DRIVE_sap['Serial number'])
            twoControllerIndexER.append(locationindexER1)

        else:
            print '\n', "Retrieved on Controller A at 'Not Available', there
is an Error, and no DRIVE information on Controller A"

```



```

        #ifControllerBExists = re.search("Retrieved on Controller B at
(.*?)\r\n\r\nError", erString)
        ifControllerBExists = re.search("Retrieved on Controller B at
(.*?)\n\nError", erString)
        if ifControllerBExists == None:
            locationindexER2 = erString.rfind(DIM_DRIVE_sap['Serial number'])
            twoControllerIndexER.append(locationindexER2)

        else:
            print '\n', "Retrieved on Controller B at 'Not Available', there
is an Error, and no DRIVE information on Controller B"

    ###print twoControllerIndexER, '\n'

    for driveLocationIndexER in twoControllerIndexER:

        if ifControllerAExists == None:
            if driveLocationIndexER == locationindexER1:
DIM_DRIVE_ERROR['Controller Type'] = 'A'
            if ifControllerBExists == None:
                if driveLocationIndexER == locationindexER2:
DIM_DRIVE_ERROR['Controller Type'] = 'B'

            beginIndexER = erString.rindex('St ',0,driveLocationIndexER)
            endIndexER = erString.index('St ',driveLocationIndexER)

            cuttinglinelong = "-- -----"

            tmpIndexER = erString.find(cuttinglinelong,beginIndexER,endIndexER)
+ len(cuttinglinelong)
            tmpEndIndexER = erString.find('WWN',tmpIndexER,endIndexER) - 1

            erStringSplit = erString[tmpIndexER:tmpEndIndexER].split(' ')

            for itr in range(7):
                for element in erStringSplit:
                    if element == ':': erStringSplit.remove(element)
                    elif element == '\n': erStringSplit.remove(element)
                    elif element == '|': erStringSplit.remove(element)
                    else: continue

            DIM_DRIVE_ERROR['Devnum'] = erStringSplit[0]
            DIM_DRIVE_ERROR['Location'] = erStringSplit[1]
            DIM_DRIVE_ERROR['Role'] = erStringSplit[2]
            DIM_DRIVE_ERROR['Rank'] = erStringSplit[3]
            DIM_DRIVE_ERROR['Total CmdCount'] = erStringSplit[4]
            DIM_DRIVE_ERROR['Total Error Count'] = erStringSplit[5]
            DIM_DRIVE_ERROR['Total Event Count'] = erStringSplit[6]

```

```

        DIM_DRIVE_ERROR['WWN'] = re.findall("WWN: (.*?)\n",
erString[tmpIndexER:endIndexER])[0].strip()

        cuttinglineshort = "-----"
-----"

        tmpIndexER =
erString.find(cuttinglineshort,beginIndexER,endIndexER) + len(cuttinglineshort) + 1

        erStringSplitCmd = erString[tmpIndexER:endIndexER].split('\n')

        for itr in range(2):
            for element in erStringSplitCmd:
                if element == ':': erStringSplitCmd.remove(element)

            ###print "For the Drive (Serial Number): ",
DIM_DRIVE_ERROR['Serial number'], " in Controller ", DIM_DRIVE_ERROR['Controller
Type'], '\n', "-----"

        DIM_DRIVE_ERROR['Time since Boot'] = 'null'
        DIM_DRIVE_ERROR['CmdCount'] = 'null'
        DIM_DRIVE_ERROR['Command Type'] = 'null'
        DIM_DRIVE_ERROR['Idcmd'] = 'null'

        dimDriveError =
csv.DictWriter(fileDriveError,fieldnames=DIM_DRIVE_ERROR.keys())

        if ifControllerAExists == None:
            if trayNum == min(recordOfExistingTrayNum) and slotNum == 1
and driveLocationIndexER == locationindexER1:
                dimDriveError.writeheader()
            else:
                if trayNum == min(recordOfExistingTrayNum) and slotNum == 1
and driveLocationIndexER == locationindexER2:
                    dimDriveError.writeheader()

        for itrElement in erStringSplitCmd:
            if 'Event' in itrElement:
                EventString = itrElement.split()

                DIM_DRIVE_ERROR['Time since Boot'] = EventString[0]
                DIM_DRIVE_ERROR['CmdCount'] = EventString[1]
                DIM_DRIVE_ERROR['Command Type'] = EventString[2]
                DIM_DRIVE_ERROR['Idcmd'] = EventString[3]

            elif 'Error' in itrElement:
                ErrorString = itrElement.split()

                DIM_DRIVE_ERROR['Time since Boot'] = ErrorString[0]
                DIM_DRIVE_ERROR['CmdCount'] = ErrorString[1]

```

```

        DIM_DRIVE_ERROR['Command Type'] = ErrorString[2]
        #DIM_DRIVE_ERROR['Idcmd'] = ''.join(ErrorString[3:], ' ')
        DIM_DRIVE_ERROR['Idcmd'] = ErrorString[3] + ' ' +
ErrorString[4] + ' ' + ErrorString[5] + ' ' + ErrorString[6] + ' ' + ErrorString[7] +
' ' + ErrorString[8] + ' ' + ErrorString[9]

        else: continue

        dimDriveError.writerow(DIM_DRIVE_ERROR)

        #print
'\n','#####',DIM_DRIVE_ERROR

## For system with the Drawers configuration:
else:
    for trayNum in [0,1,2,3,4,5,6,7,8,9,10,99]:
        for drawerNum in range(12):
            drawerNum += 1
            for slotNum in range(36):
                slotNum += 1

                searchString = "Drive at Tray " + str(trayNum) + ", Drawer " +
str(drawerNum) + ", Slot " + str(slotNum) + "\n"
                checkIfExist = sapString[tmpIndex:endIndex].find(searchString)
                if checkIfExist == -1:
                    continue
                else:
                    recordOfExistingTrayNum.append(trayNum)
                    recordOfExistingDrawerNum.append(drawerNum)

                # Generate partitioned field names date_partition, which is equal
to ASUP_GenDate

                DIM_DRIVE_sap['date_partition'] = g_asup['ASUP_GenDate']

                tmpIndex = tmpIndex + checkIfExist
                DIM_DRIVE_sap['Tray Number'] = trayNum
                #DIM_DRIVE_sap['Tray Number'] = re.findall("Tray (.*)",
sapString[tmpIndex:endIndex])[0].strip()
                DIM_DRIVE_sap['Drawer Number'] = drawerNum
                DIM_DRIVE_sap['Slot Number'] = slotNum
                for field in DRIVEFieldName:
                    if field not in sapString[tmpIndex:endIndex]:
                        ###print '\n\n', "Cannot find field '{0}' in the 'storage-
array-profile.txt' file. \nin the corresponding column in DIM_DRIVE table, it will be
filled with 'na'. \n".format(field)
                        DIM_DRIVE_sap[field] = 'na'
                    else:
                        DIM_DRIVE_sap[field] = re.findall(field + ": (.*)\n",
sapString[tmpIndex:endIndex])[0].strip()

```

```

        if 'Data Assurance (DA) capable' not in
sapString[tmpIndex:endIndex]:
            print '\n\n', "Cannot find field 'Data Assurance (DA) capable'
in the 'storage-array-profile.txt' file. \nin the corresponding column in DIM_DRIVE
table, it will be filled with 'na'. \n"
            DIM_DRIVE_sap['Data Assurance (DA) capable'] = 'na'
        else:
            DIM_DRIVE_sap['Data Assurance (DA) capable'] =
re.findall("Data Assurance \ (DA\) capable: (.*)\n",
sapString[tmpIndex:endIndex])[0].strip()

        # Generate Drive ID
        DIM_DRIVE_sap['Drive ID'] = g_asup['ASUP_SystemID'] + '_' +
DIM_DRIVE_sap['Serial number']

        if trayNum == min(recordOfExistingTrayNum) and drawerNum ==
min(recordOfExistingDrawerNum) and slotNum == 1:
            dimDrive =
csv.DictWriter(fileDrive, fieldnames=DIM_DRIVE_sap.keys())
            dimDrive.writeheader()
            dimDrive.writerow(DIM_DRIVE_sap)

### DIM_DRIVE_ERROR table

        # Generate partitioned field names date_partition, which is equal
to ASUP_GenDate
        DIM_DRIVE_ERROR['date_partition'] = g_asup['ASUP_GenDate']

        DIM_DRIVE_ERROR['Serial number'] = DIM_DRIVE_sap['Serial number']

        # Generate Drive ID
        DIM_DRIVE_ERROR['Drive ID'] = g_asup['ASUP_SystemID'] + '_' +
DIM_DRIVE_ERROR['Serial number']

        twoControllerIndexER = []

        # Deal With special cases: "Retrieved on Controller A/B at Not
Available"

        #ifControllerAExists = re.search("Retrieved on Controller A at
(.*)\r\n\r\nError", erString)
        ifControllerAExists = re.search("Retrieved on Controller A at
(.*)\n\nError", erString)
        if ifControllerAExists == None:
            locationindexER1 = erString.find(DIM_DRIVE_sap['Serial
number'])
            twoControllerIndexER.append(locationindexER1)

        else:

```

```

        print '\n', "Retrieved on Controller A at 'Not Available',
there is an Error, and no DRIVE information on Controller A"

        #ifControllerBExists = re.search("Retrieved on Controller B at
(.*?)\r\n\r\nError", erString)
        ifControllerBExists = re.search("Retrieved on Controller B at
(.*?)\n\nError", erString)
        if ifControllerBExists == None:
            locationindexER2 = erString.rfind(DIM_DRIVE_sap['Serial
number'])

            twoControllerIndexER.append(locationindexER2)

        else:
            print '\n', "Retrieved on Controller B at 'Not Available',
there is an Error, and no DRIVE information on Controller B"

        ###print twoControllerIndexER, '\n'

        for driveLocationIndexER in twoControllerIndexER:

            if ifControllerAExists == None:
                if driveLocationIndexER == locationindexER1:
DIM_DRIVE_ERROR['Controller Type'] = 'A'
            if ifControllerBExists == None:
                if driveLocationIndexER == locationindexER2:
DIM_DRIVE_ERROR['Controller Type'] = 'B'

            beginIndexER = erString.rindex('St ',0,driveLocationIndexER)
            endIndexER = erString.index('St ',driveLocationIndexER)

            cuttinglinelong = "-- ----- -"

            tmpIndexER =
erString.find(cuttinglinelong,beginIndexER,endIndexER) + len(cuttinglinelong)
            tmpEndIndexER = erString.find('WWN',tmpIndexER,endIndexER) - 1

            erStringSplit = erString[tmpIndexER:tmpEndIndexER].split(' ')

            for itr in range(7):
                for element in erStringSplit:
                    if element == ':': erStringSplit.remove(element)
                    elif element == '\n': erStringSplit.remove(element)
                    elif element == '|': erStringSplit.remove(element)
                    else: continue

            DIM_DRIVE_ERROR['Devnum'] = erStringSplit[0]
            DIM_DRIVE_ERROR['Location'] = erStringSplit[1]
            DIM_DRIVE_ERROR['Role'] = erStringSplit[2]
            DIM_DRIVE_ERROR['Rank'] = erStringSplit[3]

```

```

DIM_DRIVE_ERROR['Total CmdCount'] = erStringSplit[4]
DIM_DRIVE_ERROR['Total Error Count'] = erStringSplit[5]
DIM_DRIVE_ERROR['Total Event Count'] = erStringSplit[6]

DIM_DRIVE_ERROR['WWN'] = re.findall("WWN: (.*?)\n",
erString[tmpIndexER:endIndexER])[0].strip()

cuttinglineshort = "-----"

tmpIndexER =
erString.find(cuttinglineshort,beginIndexER,endIndexER) + len(cuttinglineshort) + 1

erStringSplitCmd = erString[tmpIndexER:endIndexER].split('\n')

for itr in range(2):
    for element in erStringSplitCmd:
        if element == ':': erStringSplitCmd.remove(element)

    ###print "For the Drive (Serial Number): ",
DIM_DRIVE_ERROR['Serial number'], " in Controller ", DIM_DRIVE_ERROR['Controller
Type'], '\n', "-----"

DIM_DRIVE_ERROR['Time since Boot'] = 'null'
DIM_DRIVE_ERROR['CmdCount'] = 'null'
DIM_DRIVE_ERROR['Command Type'] = 'null'
DIM_DRIVE_ERROR['Idcmd'] = 'null'

dimDriveError =
csv.DictWriter(fileDriveError,fieldnames=DIM_DRIVE_ERROR.keys())

if ifControllerAExists == None:
    if trayNum == min(recordOfExistingTrayNum) and drawerNum
== min(recordOfExistingDrawerNum) and slotNum == 1 and driveLocationIndexER ==
locationindexER1:
        dimDriveError.writeheader()
else:
    if trayNum == min(recordOfExistingTrayNum) and drawerNum
== min(recordOfExistingDrawerNum) and slotNum == 1 and driveLocationIndexER ==
locationindexER2:
        dimDriveError.writeheader()

for itrElement in erStringSplitCmd:
    if 'Event' in itrElement:
        EventString = itrElement.split()

        DIM_DRIVE_ERROR['Time since Boot'] = EventString[0]
        DIM_DRIVE_ERROR['CmdCount'] = EventString[1]
        DIM_DRIVE_ERROR['Command Type'] = EventString[2]
        DIM_DRIVE_ERROR['Idcmd'] = EventString[3]

```

```

        elif 'Error' in itrElement:
            ErrorString = itrElement.split()

            DIM_DRIVE_ERROR['Time since Boot'] = ErrorString[0]
            DIM_DRIVE_ERROR['CmdCount'] = ErrorString[1]
            DIM_DRIVE_ERROR['Command Type'] = ErrorString[2]
            #DIM_DRIVE_ERROR['Idcmd'] = ''.join(ErrorString[3:], '
')

            DIM_DRIVE_ERROR['Idcmd'] = ErrorString[3] + ' ' +
ErrorString[4] + ' ' + ErrorString[5] + ' ' + ErrorString[6] + ' ' + ErrorString[7] +
' ' + ErrorString[8] + ' ' + ErrorString[9]

        else: continue

    dimDriveError.writerow(DIM_DRIVE_ERROR)

#print'\n','#####',DIM_DRIVE_
ERROR

fileDriveError.close()
fileDrive.close()

## ~~~~~ major-event-log.txt Parser ~~~~~
majorEventLogMap={
'Date/Time':'Major Event Date and Time',
'Sequence number':'Major Event Sequence number',
'Event type':'Major Event type',
'Event category':'Major Event category',
'Priority':'Major Event Priority',
'Event needs attention':'Major Event needs attention',
'Event send alert':'Major Event send alert',
'Event visibility':'Major Event visibility',
'Description':'Major Event Description',
'Event specific codes':'Major Event specific codes',
'Component type':'Major Event Component type',
'Component location':'Major Event Component location',
'Logged by':'Major Event Logged by'}

### DIM_MAJOR_EVENT_LOG table

g_majorEventLog = {}
i = 0

for l in open(PATH_data_folder + 'major-event-log.txt'):
#for l in open('/home/saasbook/ASUP/E-series_Data1/' + 'major-event-log.txt'):
    if i <= 15: i += 1 # 15 lines of useful information
    if 'Raw data:' not in l:
        if ':' not in l: continue

```

```

tmpL = l.split(":")
try:
    g_majorEventLog[majorEventLogMap[tmpL[0]]]=str.strip(":".join(tmpL[1:]))
except:
    print "Error parsing 'major-event-log.txt' in line [" + l + "]"

else:
    g_majorEventLog['Major Event Date and Time'] =
dateParser.parse(g_majorEventLog['Major Event Date and Time']).strftime("%Y%m%d%H%M%S")

    g_majorEventLog['ASUP_ID'] = g_asup['ASUP_ID']
    # Generate partitioned field names date_partition, which is equal to
ASUP_GenDate
    g_majorEventLog['date_partition']=g_asup['ASUP_GenDate']

    if i == 15:
        DIM_MAJOR_EVENT_LOG_table = 'DIM_MAJOR_EVENT_LOG_{0}.csv'.format(nameIndex)
        fileMajorEventLog = open(DIM_MAJOR_EVENT_LOG_table, 'w')
        dimMajorEventLog = csv.DictWriter(fileMajorEventLog, fieldnames =
g_majorEventLog.keys())
        dimMajorEventLog.writeheader()

        dimMajorEventLog.writerow(g_majorEventLog)

fileMajorEventLog.close()

## ~~~~~ storage-array-profile.txt Parser [DIM_TRAY table] ~~~~~
trayMap={
'Tray path redundancy':'Major Event Date and Time',
'Sequence number':'Major Event Sequence number',
'Event type':'Major Event type',
'Event category':'Major Event category',
'Priority':'Major Event Priority',
'Event needs attention':'Major Event needs attention',
'Event send alert':'Major Event send alert',
'Event visibility':'Major Event visibility',
'Description':'Major Event Description',
'Event specific codes':'Major Event specific codes',
'Component type':'Major Event Component type',
'Component location':'Major Event Component location',
'Logged by':'Major Event Logged by'}

### DIM_MAJOR_EVENT_LOG table

g_majorEventLog = {}
i = 0

for l in open(PATH_data_folder + 'major-event-log.txt'):
#for l in open('/home/saasbook/ASUP/E-series_Data1/' + 'major-event-log.txt'):
    if i <= 15: i += 1 # 15 lines of useful information
    if 'Raw data:' not in l:

```



```

    if ':' not in l: continue
    tmpL = l.split(":")
    try:
        g_majorEventLog[majorEventLogMap[tmpL[0]]]=str.strip(":".join(tmpL[1:]))
    except:
        print "Error parsing 'major-event-log.txt' in line [" + l + "]"

else:
    g_majorEventLog['Major Event Date and Time'] =
dateParser.parse(g_majorEventLog['Major Event Date and Time']).strftime("%Y%m%d%H%M%S")

    g_majorEventLog['ASUP_ID'] = g_asup['ASUP_ID']
    # Generate partitioned field names date_partition, which is equal to
ASUP_GenDate
    g_majorEventLog['date_partition']=g_asup['ASUP_GenDate']

    if i == 15:
        DIM_MAJOR_EVENT_LOG_table = 'DIM_MAJOR_EVENT_LOG_{0}.csv'.format(nameIndex)
        fileMajorEventLog = open(DIM_MAJOR_EVENT_LOG_table, 'w')
        dimMajorEventLog = csv.DictWriter(fileMajorEventLog, fieldnames =
g_majorEventLog.keys())
        dimMajorEventLog.writeheader()

        dimMajorEventLog.writerow(g_majorEventLog)

fileMajorEventLog.close()

```

Bibliography

- [1] [Online] <http://www.netapp.com/us/products/storage-systems/e5400/e5400-product-comparison.aspx>
- [2] [Online] <http://spark.apache.org/research.html>
- [3] [Online] <http://www.idc.com/getdoc.jsp?containerId=prUS24302513>
- [4] [Online] <http://www.netapp.com/us/services-support/autosupport.aspx>
- [5] [Online].<http://www.emc.com/microsites/big-data-explorer/index.htm#content/which/analytics/>, last access Jan. 30, 2014.
- [6] [Online].http://www.cisco.com/en/US/solutions/collateral/ns340/ns1176/data-center/BigData_Case_Study-1.html/, last access Jan. 31, 2014.
- [7] [Online]. <http://www-01.ibm.com/software/data/bigdata/>, last access Jan. 31, 2014.
- [8] [Online].<http://storageeffect.media.seagate.com/2013/04/storage-effect/seagate-puts-big-data-in-action-a-case-study/>, last access Feb. 1, 2014.
- [9] [Online].<http://hortonworks.com/customer/western-digital/>, last access Feb. 2, 2014.
- [10] "Proactive Health Management with AutoSupport" *NetApp White Paper*. Network Appliance, Inc. Technical Report. WP-7027-0907. Sept. 2007.
- [11] J. Ullman, "Mining of Massive Datasets" pp.19, Cambridge University Press, December 30, 2011.
- [12] A. Thusoo, J. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff and R. Murthy, "Hive - A Warehousing Solution Over a Map-Reduce Framework". *Proceedings of the VLDB Endowment*. Vol. 2 Issue 2. Pp. 1626-1629. August 2009.
- [13][Online]<http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html>
- [14] [Online] <http://parquet.io/>
- [15] [Online] <https://github.com/forcedotcom/phoenix>
- [16] T. Kaldewey, E. Shekita, S. Tata, "Clydesdale: Structured Data Processing on MapReduce" IBM Almaden Research Center, Google. http://www.timkaldewey.de/pubs/Clydesdale__EDBT12.pdf
- [17] A. Pandey, H. Tong and J. Zhang, "Analytics for E-Series AutoSupport Using Big Data Technologies." Poster, NetApp Inc., Sunnyvale, CA, Tech. Rep., Nov., 2013.