

Automated recovery of data-intensive jobs in D0 and CDF using SAM

A. Baranovski¹, V. Bartsch², D. Benjamin⁴, K. Genser¹
E. Lipeles³, A. Lyon¹, I. Sfiligoi¹,

¹Fermi National Accelerator Laboratory, ²University College London,
³University of California, ⁴Duke University

Abstract

SAM [1] is a data handling system that provides the Fermilab HEP experiments D0, CDF and MINOS with the means to catalog, distribute and track the usage of their collected and analyzed data. SAM serves petabytes of data per year to physics groups performing data analysis, data reconstruction and simulation at various computing centers across the world.

Reliability issues in the user analysis or production are important concerns to address in the SAM project. The SAM datahandling and its interface have been used to detect failures at any level of the data intensive jobs.

In this paper the flow of a typical analysis job and possible error cases at the example of the CDF experiment are presented and a method to automatically recover from this failures is described. We present an automated method that uses SAM data handling to formalize distributed data analysis by defining a transaction based model of the physics analysis job work cycle to enable robust recovery of the unprocessed data.

INTRODUCTION

Given the volume of the detector data, a typical physics analysis job consumes terabytes of information during several days of running at a job execution site. At any stage of that process, non systematic failures e.g. failures of file delivery may occur, leaving a fraction of the original dataset unprocessed. To ensure convergence to completion of the computation request, a facility user has to employ a procedure to identify pieces of data that need to be re-analyzed in a manner that guarantees completeness without duplication in the final result.

It is common that these issues are addressed by analyzing the output of the job. Such an approach is fragile, since it depends critically on the (changeable) output file format, and time-consuming. The approach that is reported in this article saves the users time and ensures consistency of the results.

ANALYSIS FLOW

The features of the automated recovery system are introduced using the analysis flow of CDF as an example. The typical Data Aquisition and Analysis Flow of the experiment CDF is shown in Fig. 1.

The Tevatron, the reconstruction farms and the remote Monte Carlo production are producing data. The Tevatron runs at a crossing rate of 1.7MHz, after the trigger about

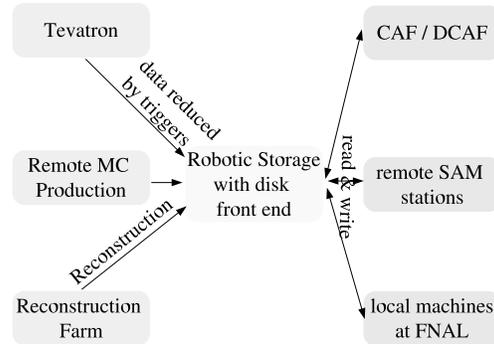


Figure 1: CDF Data Acquisition system and Analysis Flow

75 Hz are accepted. The reconstruction farm reconstructs the Tevatron data. The data are stored on tape into the en-store system which has a disk frontend using dCache [2]. In total about 1 PB user data are on tape for CDF and about 170 TB of disks of the dCache system are used, so that data used frequently can be accessed from disk.

The stored data is read by users using the SAM data handling system. The analysis of the data can be performed either at the Fermilab analysis farms or at the remote computing facilities. Both ways can benefit from distributed SAM datahandling services independently deployed at Fermilab or at participating resource providers.

For example, the central analysis farm (CAF) [3] is located at Fermilab. In addition, around the world, there are about 10 decentralized analysis farms (DCAF) that are using the same mechanisms as the CAF. Both CAF and dCAFs datahandling needs are successfully satisfied by SAM. As of the time of writing the CAF/dCAFs have a CPU power of 2.6 M SPECInt2k at Fermilab and 2.6 M SPECInt2k offsite. The dCAFs have 78 TB of storage. Several remote computing facilities e.g. at Karlsruhe and Oxford have not installed a DCAF, but nevertheless have been leveraging SAM service independently. Such facilities are typically used by the user community local to the SAM deployments.

Typically, the analysis farms are used to submit jobs requiring lots of CPU resources and are the batch (non interactive) environments. In the batch environment, it is often a challenge to track and recover from failures due to massive amount of user activity that may hinder the problem diagnosis and recovery. Our approach is to focus on ease of recovery from the non systematic failures while leaving the reliability issues of the execution environment (CAF)

to its respective service providers. jobs.

JOB HANDLING AT THE CAF

In order to explain the recovery mechanism more details about the job handling at the CAF need to be given. The left hand side of Fig. 2 shows the job handling of the CAF. User submit jobs to the CAF using a graphical or command line interface. They provide information about the job which needs to be executed, the parallization of the job, the data handling system used and the data needed for the job. The data needed for the job is grouped in data sets which match requirements on the data e.g. run numbers, time period, physical content of the data. In case of jobs which do not need input data no data handling system needs to be chosen. The jobs at the CAF start with a start section (see Fig. 2) which sets the environment for the job. In case the SAM data handling system is chosen a SAM project is started and the data sets are translated into a list of files. The SAM project is monitored by the SAM database throughout the job execution and is unique for the job. The CAF start section sends the job to worker nodes depending on the number of parallel executions specified by the user. Throughout the execution of the job on the worker nodes, the SAM database monitors the file delivery. The end section closes the SAM project and delivers the output files.

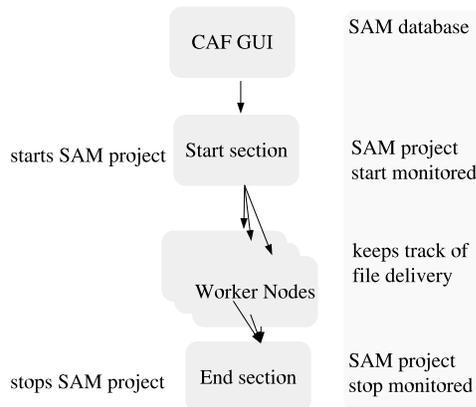


Figure 2: Job handling of the CAF and monitoring of the SAM database connected to the job before automatic recovery.

ERROR CASES

During operation of the CAF several error cases have been observed. Those can be divided into problems of file delivery (which is covered by the recovery based on consumption of files), problems with the job sections on the worker nodes and problems at the end section of files. Typically the problems with job sections are CAF section restarts, for example due to reboot of a worker node or due to any other hardware failure. The restarts of job

sections cause a loss of data files because the files are already registered as consumed with the SAM database. This loss is very likely to be unnoticed by common users. The problems of the end section are most likely output retrieval failures, which could be due to full disks, authentication problems or wrongly specified output destinations. Except for the file delivery non of these issues are tracked by the SAM database, however they are monitored by the CAF software. Statistically about 3% of all jobs have a problem with files retrievals. 0.4% of all sections get restarted. The jobs have about 70 sections on average. 3% of the jobs fail during the end section.

SAM BOOKKEEPING MODEL

SAM is a comprehensive solution that enables variety of data management and data access use cases tailored for the HEP application. In addition to that, SAM provides means to monitor the consumption and enable mining of data that has been requested and subsequently served to the user. Data management, data bookkeeping and mining use-cases are implemented using a unified model that covers most sophisticated needs of the end user application. The basic component of that model is SAM project. SAM project is an entity that describes and registers computational request to SAM datahandling. This request typically contains such parameters as dataset name, name of the station (SAM datahandling domain), name and the version of the analysis application. That latter is important to bind requested data to the type/version of the analysis been used. The lifetime of the SAM project is defined by how long the computational request persists in the execution environment. On the course of its lifetime this state changes in response to user analysis interface calls to SAM as well as to external factors such as file delivery or error notifications. At the same time, the state of the SAM project is recorded in the persistent storage. The principal ability to persistently store the history of the SAM project allows to subsequently mine and recover processed or unprocessed data.

There are two types of objects that provide persistency within SAM project state. These are “the process” and “the file”. Picture here: SAM project contains processes. Process has node. Files belong to processes.

The process object declares SAM application constraints that define set of storages that application wants to receive data from. File arrival events from these storages are recorded in the context of the respective “process”. And, similarly to the SAM project, in addition to its own state, the process object aggregates states of the all the files that it happens to own.

“the file” object aggregates the state of the file with respect to the application. Presently, SAM supports 3 different states for this type of the object: delivered, opened and consumed.

The premise of the article is based on ability of the user

application running in the experiment specific environments to manipulate persistent states of the SAM project, process and file to record sufficient amount information that allows to recover any portion of the user job.

EXECUTION ENVIRONMENT. INTERFACE TO SAM DATAHANDLING AND BOOKKEEPING

As was explained in the previous paragraphs, CAF start section is first user code that is been executed. The code bears responsibility to initialize necessary environment common for all subsequent sections (parallelized jobs). If user selects SAM datahandling, the CAF start section is the section start starts SAM project. In CAF environment, SAM project is the portal to the datahandling universe that helps managing data delivery to parallel applications (sections) in a manner files are received in random yet non overlapping sets. The exact distribution of data among jobs is computed based on the most optimal delivery time.

Each job, in turn, defines itself to SAM by registering "the process". The registration step is permanent and is used to enable data streaming to the application as well as bind the success status of the job (and the CAF section herein) with status of the SAM project process. In the article we refer this registration step as the start of the datahandling transaction. The transaction that tracks statuses of participating file delivery and file processing events. The events that at the same time are independently reflected in the SAM project file context object.

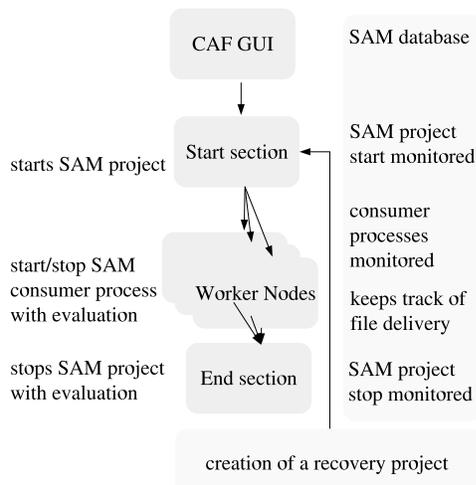


Figure 3: Job handling of the CAF and interfaces to the SAM database for the automatic recovery.

RECOVERY

Both execution and datahandling environments are the two major contributing factors to the success rate of the computational request. Therefore, both have to work tight

to enable sufficient input of the information to detect and recover from failures on all levels. SAM records file delivery and consumption history and provides means to create and end datahandling transactions that can encompass the scope of the data intensive jobs. By integrating datahandling and jobs handling environments it is possible to define the scope of the application beyond the analysis itself. For example in CAF environment, these steps are all parts of the job scope trackable by SAM datahandling transactions: 1) Staging job executable. 2) Staging multiple input files. 3) Analysis. 4) Verification. 5) Output storage. These steps are independent and equally important components in the overall success of the job. Which is why, irrespective of how much data has been analyzed by the failed application, such data has to be rolled back and re-analyzed again. The SAM process transaction mechanism is an efficient way to roll back files that were delivered and consumed but have not contributed to the final results. The data that would be mistakenly rejected based on sole consumption history. Nevertheless, file delivery/consumption history is still an important piece of information to answer the question of what files have not been consumed at all.

The success status of the transaction as set by the execution environment joined with the SAM project file consumption history allows to formulate recovery jobs that always complement user's initial computational request without duplication in the output.

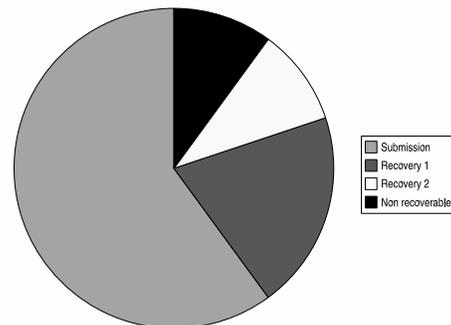


Figure 4: Job handling of the CAF and interfaces to the SAM database for the automatic recovery.

As pointed out it is reasonable to incorporate the monitoring of the experiment specific software into the recovery. This approach is depicted in Fig. 3.

The advantage for the user is that the success of a job is immediately visible after completion and a recovery is accessible without much user interaction. Compared to the parsing of output files the new approach is less error-prone, much faster and independent of the output format.

AUTOMATIC RECOVERY

CONCLUSION

The described automatic recovery incorporates the monitoring of the experiment specific software, in this case the

analysis farms of CDF, and the monitoring of the SAM data handling system. The bookkeeping is completely transparent to the user and more effective than any user implemented recovery.

REFERENCES

- [1] <http://d0db-prd.fnal.gov/sam/>
- [2] <http://dcache.desy.de/>
- [3] <http://www.cdfcaf.fnal.gov>