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

A. Baranovski*[1], V. Bartsch[2], D. Benjamin[4], K. Genser[1]
E. Lipeles[3], A. Lyon[1], I. Sfiligoi[1],
[1]Fermi National Accelerator Laboratory, [2]University College London,
[3]University of California, [4]Duke 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 data handling 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 or hardware failures of the worker nodes 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 Data Acquisition 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

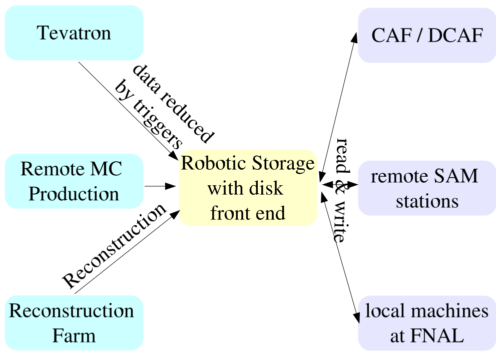

Figure 1: *CDF Data Acquisition system and Analysis Flow*

runs at a crossing rate of 1.7 MHz, after the trigger data about 75 Hz are accepted. The reconstruction farm reconstructs the raw Tevatron data. All the experiment data are stored on tapes using dCache [3]. Overall, for CDF, about 1 PB is stored on tapes and about 170 TB on dCache disks. dCache uses disks to improve analysis times for the most frequently accessed files.

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

For example, the central analysis farm (CAF) [4] is located at Fermilab. In addtion , around the world, there are about 10 decentralized analysis farms (DCAF), which all are built using same principals as the CAF. Both CAF and dCAFs data handling needs are served 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. In addition to CPU , DCAFs have also contribited 78 TB of storage. Also , computing facilities at Karlsruhe and Oxford chose not to use CAF infrastructure, but nevertheless have been leveraging SAM service independently.

Typically , the analysis farms are used to submit jobs requiring lots of CPU resources and are been run in 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 focuses on ease of recovery from the non systematic failures while leaving the reliability issues of the execution environment (CAF) to its respective developers.

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. He provides information about the job which needs to be executed, the parallelization 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 and physical content of the data. In case of jobs that do not need input - no data handling system needs to be chosen.

The jobs at the CAF start by "start" section (see Fig. 2) that sets the common environment for the job. In case the SAM data handling system is chosen, SAM project is started and input data sets are translated into a list of files to deliver. The SAM project is monitored by the SAM database throughout the job execution. SAM project is also 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.

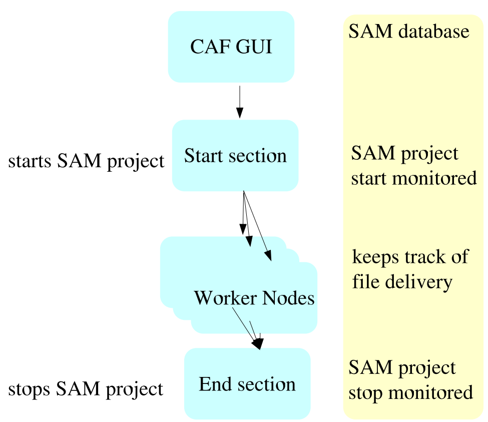

Figure 2: *Job handling at the CAF and monitoring of the SAM database connected to the job before automatic recovery. The interfaces inform the SAM database when a consumer process has failed due to a hardware or software problem.*

ERROR CASES AT THE CAF

During operation of the CAF several error cases have been observed. Generally, error cases can be divided into problems with file delivery , problems with the job sections on the worker nodes and problems at the end section of files. Typical issue with CAF job sections is thier restarts. The restart can happen due to reboot of a worker node or due to any other hardware failure. The restarts of job sections cause loss of data files because the files are already registered as consumed with the SAM database but the corresponding output files are lost.The SAM DB does not easily track section failures, and so it does not know that previously successfully consumed files need to be reprocessed to reproduce the lost output file. The 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 none 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 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 a SAM project. The SAM project is an entity that describes and registers computational request to SAM data handling. This request typically contains such parameters as dataset name, name of the station (SAM data handling domain), name and the version of the analysis application. 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.

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. Similar to the SAM project the process object aggregates states of all the it's files.

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 of information that allows to recover any portion of the user job.

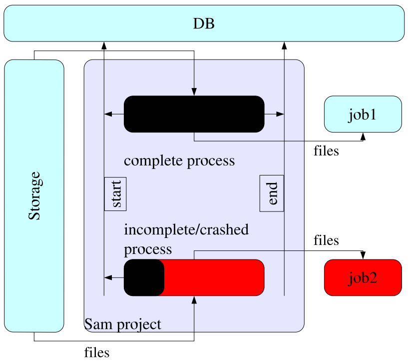

Figure 3: *SAM Project diagram*

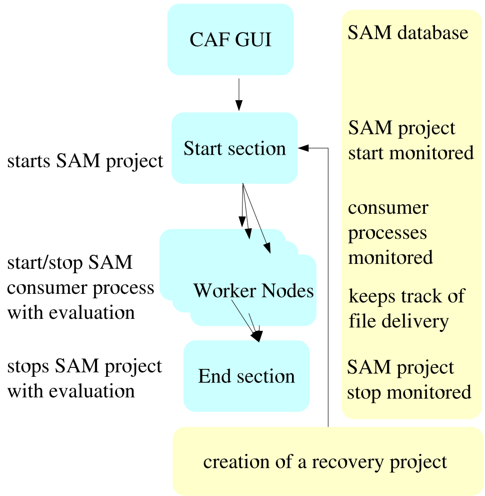

Figure 4: *Job handling of the CAF and interfaces to the SAM database for the automatic recovery.*

INTERFACE TO SAM AND BOOKKEEPING

The jobs have a common section which initializes the environment shared by all subsequent sections (parallelized jobs). If a user selects the SAM data handling, the CAF start section starts a SAM project. In the CAF environment, SAM project is the portal to the data handling 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 defines itself to SAM by registering the SAM 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 (see Fig. 3). In the article we refer this registration step as the start of the data handling transaction. The transaction that tracks statuses of participating file delivery and file processing events. At the same time, processing events are independently reflected in the SAM project file context object.

RECOVERY

Both execution and data handling environments are the two major contributing factors to the success rate of the computational request. Therefore, both has 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 data handling transactions that can encompass the scope of the data intensive jobs. By integrating data handling 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 track-able by SAM data handling transactions:

- Staging job executable
- Staging multiple input files
- Analysis
- Verification
- 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 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 important piece of information to answer the question of what files have not been consumed at all.

The success status of the transaction is set by the execution environment. Now, in conjunction with the SAM project file consumption history it is possible to formulate a recovery job that complements the user's initial computational request without duplication in the output.

The monitoring of the experiment specific software generates more information that helps to build more precise recovery requests. This approach is depicted in Fig. 4.

AUTOMATIC RECOVERY AT THE CAF

Use of SAM bookkeeping to formalize job work flow allows user to observe success status of a job immediately after completion and perform recovery without much effort or knowledge of the underlying execution environment. Compared to the parsing of output files the new approach is less error-prone, much faster and independent of the output format.

One of the advantages of the suggested model is the ability to automate user requests to full completion. The jobs is complete when all the input data is processed or can not be processed due systematic, non recoverable errors. The latter condition is easily detected by the algorithm that flags the unprocessable requests after learning that its recovery attempts do not reduce the size of their input datasets. In CAF environment, it is the start section that is responsible for implementing the automated recovery for job failures. After the end section a recovery project can be generated (see Fig. refrecovery figure) and submitted again. The automatic resubmission based on SAM bookkeeping is in a testing phase. Therefore , not all the issues are well understood.

CONCLUSION

The described automatic recovery incorporates the recording and monitoring of the states of the experiment specific software using SAM data handling system. The bookkeeping is completely transparent to the user and effective tool that improves any user implemented recovery.

REFERENCES

[1] I. Terekhov et al., ”Meta-Computing at D0”, Nuclear Instruments and Methods in Physics Research, Section A, NIMA14225, vol. 502/2-3 pp 402 - 406.

[2] http://d0db-prd.fnal.gov/sam/

[3] http://dcache.desy.de/

[4] http://www.cdfcaf.fnal.gov