Development of Remote Concatenation & Monitoring Tools for CDFII Production Farm

Solomon Mikael

The Collider Detector at Fermi Laboratory

Abstract

The CDF detector is a vital part of the physics program at Fermi Laboratory. Monitoring the farm is the web based Tikiwiki pages. The tiki pages provide vital information to the users allowing the farm to operate smoothly. The concatenation of reconstructed files depends on farm. Modifications were made to both the monitoring and concatenation system to make both systems perform efficiently. This paper constitutes the final project for he SIST program.

Introduction

Higher energy, higher intensity particle beams, and advanced detectors in high energy physics (HEP[1]) have led to the collection of larger volumes of data. The Collider Detector at Fermilab (CDF[2]) has improved its data acquisition capacity in the Run II program. The computing facility was also upgraded for processing larger volumes of data. The CDF II experiment began collecting data in 2000. The peak rate of data recording is 40MB/s. The total volume of raw data collected by the experiment is roughly 0.5 petabytes. This number is expected to increase dramatically in the future. As a result of this, the data management becomes a very important part of the experiment. This paper will describe the main hardware, software, monitoring, and control components of the CDF production farm.

The primary objective is to implement a method that will improve the disk write speed. Additionally modifications were made to the monitoring and concatenation systems of the farms.

CDF Experiment

The CDF (collider detector experiment at FermiLab) is an international collaboration involving many universities and national laboratories. The 100 ton CDF detector is a large general purpose solenoidal detector which combines precision charged particle tracking with fast projective calorimetry and fine grain muon detection. CDF must manage is made by the collisions or the protons and anti-protons. Of the collisions on a few are hard collisions, those which make energetic particles which go off at large angels from the beam allowing it to pass through the detector. A charged particle will be observed the rest can be tracked by taking into account the transverse momentum based on the conservation of momentum. Also particles that are not detected in the initial layers are detected in the calorimeter. Not all the energy can be accounted for some particles will inevitable escape detection.

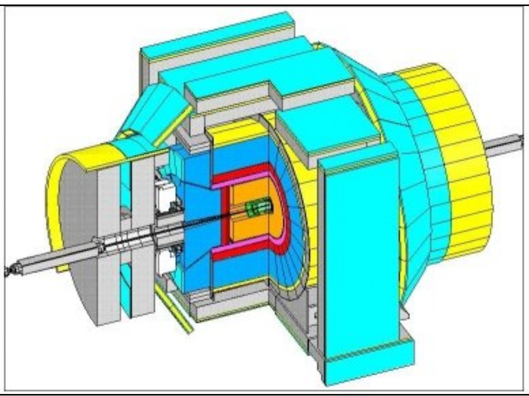


Figure 1 (this is a diagram of the CDF detector)

To observe the particles generated by the collision there are several detector systems. Each detector system is best at detecting or interacting with a certain particle. Of the millions of collisions per second only a few of them give large angles. The detectors record electronic information on the 1000000 channels. Based on a few of the channels preliminary information is provided. Triggers make a decision if a collision is interesting or not. As the precision and efficiency of the detector increases so is the amount of information it's able to draw from these collisions. As a result a data management system is imperative for CDF to operate.

The CDF Farm

The primary goal of CDF Production Farm is to perform computing and network intensive tasks in a cost effective manner. The data processing model for the CDF experiment is based on reconstruction of parallel data streams taken with different combinations of physics event triggers. Data processing consists of large clusters of Linux computers with data movement managed by CDF data handling system. The high energy physics experiments (HEP) being done at CDF generated enormous amounts of data and is expected to increase in the future. CDF's challenge was in finding a way to manage the large flow of data through the computing nodes. Condor is the software that manages, controls, and monitors the CDF production farm. Presently the production farm consists of 150 dual CPU PC's with a total computing power of about 800 GHz. The main software components of the farm architecture are SAM and CAF. SAM_3 stores and retrieves the metadata associated with files. CAF is control system used by all farms inside the experiment for batch job submission and access to the CDF data management system and databases. The new SAM data production system is suitable for job submission to any computing facility in the world that uses the CAF interface with direct access to the SAM data handling and database connection. The production farm is setup to take advantage to grid computing was a CAF headnode and SAM stations. The mass storage system has three major parts: the Enstore system which is used to access data on the tapes, dCache which provides a system for storing and retrieving huge amounts of data which is distributed among a large number of nodes under a single virtual filesystem, and PNFS which is a software used to map all files stored in the root to a Unix like namespace.

Software Products

For CDF farms to accomplish its goals it needs many individual programs to operate. Each program contributes a job to the entire system of the farm. In the end it is this system that programmers and users are familiar with.

I. Condor

The Condor Project[4] is one of the systems used in the farm. The goal of the Condor Project is to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of disruptively owned computing resources. This system enables scientists and engineers to increase their computing throughput. Condor achieves its goal by implementing a specialized work load management system for computer intensive jobs. The batch system in Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. After submission of a job, in series or in parallel, Condor places them into a queue and chooses when and where to run the jobs based upon a policy, monitors the jobs progress, and finally informs the user when the job is complete. Condor can be used to manage a cluster of dedicated computer nodes and harness wasted CPU power from idle desktop workstations. A big advantage of using Condor is that it does not require a shared file system across machines.

II. CVS

The concurrent version system also known as CVS[5] is another software tool used by the farms. CVS is a version control system. It allows one to record the history of source files. For example is there is a bug in a program after it has been modified. CVS allows for retrieval of the old version to see exactly what change caused the bug. CVS stores all the versions of a file in a single file which only stores the difference between versions. The software helps insulate different developers from each other so all developers work in their own directory avoiding overwriting others changes. The files are stored in a centralized repository which stores a complete copy of all the files and directories which are under version control.

III. SAM

For the CDF Farms to operate efficiently there is a need for a system that delivers files and keeps track of the files that have been analyzed. A system that does just this is called SAM or sequential data access via metadata. All the files accessible in a UNIX system area arranged in one big tree the file hierarchy is rooted at /. These files can be spread out over several devices. SAM mitigates the problem of one person hogging the tape drive and/or flooding the tape system. SAM is a data handling system organized as a set of servers which work together to store and retrieve associated metadata (data that is used to describe other data which may describe how, when, and by whom the data was received, created, accessed, and modified) including a complete record of the processing which has used the files. SAM is able: to track locations and comprehensive metadata for each file in the system, provide storage utilities to add a file to a permanent storage location, deliver files on request to systems that have deployed the file delivery components of the SAM system, provide methods for job submission which are coupled to the file delivery mechanism and can utilize location and system information for performance optimization, and track processing information permitting processing to be organized on the basis of the consumption history of a particular data set and allows the construction of processing jobs which use information which other files have processed successfully.

IV. Data Base

Structured Query Language (SQL) implements a database and interfaces on a computer called a database server. A database is a hierarchy of increasingly complex data structures. MySQL[7] has an ability to handle an unlimited number of simultaneous users and handle millions or records making it essential to the CDF Farms.

A few of the requirements of the CDF production farm are high throughput rate, I/O capability, easily configurable system, and efficient error handling. To make these happen the farm must monitor the hardware performance including status reporting and tuning the software of the production farm. The production farm web interface (PFWI) will parse, calculate, and display all major characteristics of the farm. The farm's performance indicators change in real time so the interface reports online results. All steps of processing heavily rely on information from the Oracle database. The worker nodes retrieve: calibration, geometric constants, and reconstructed files. The intermediate output on the worker nodes is declared to the database in order to avoid lost or duplicated events.

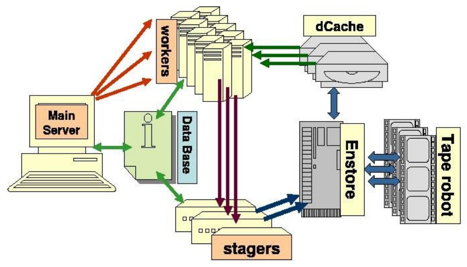


Figure 2 (the basic schematic of the CDF Farm)

The CDF farm begins raw data processing by accessing the data of the tape storage device. The collision information is stored on the tapes until this point. Then next step is reconstruction of the data. Reconstruction interprets the data that was collected to make it useful for physics analysis. The reconstructed data is then sent to the stagers. In some cases if the files are already 1-2 GB is size there is no need for concatenation. For the instances where this is not true the concatenation begins on the stagers. The stagers then send the reconstructed and concatenated data back to the tape drives for storage and later access by physicists.

Storage Speed

The issue of increasing the rate the data is written to the tape storage has many factors. In the configuration when the concatenated and the tape upload are running simultaneously the tape upload speed is affected dramatically.

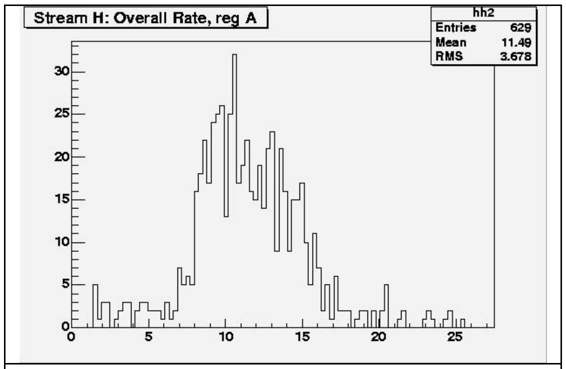


Figure 3 (the x –axis represents the MB/s while the y –axis represents the number of events that were sent at that particular speed the majority of the events seem to congregate around 12 MB/s)

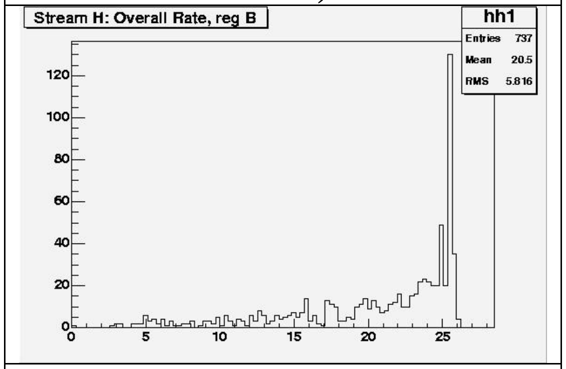


Figure 4 (a majority of the events congregate around 25 MB/s)

In the present scheme the concatenator and tape uploader are running at the same time resulting in limited I/O from the stagers. Disk access is directly related to the number of simultaneous I/O operations from the disk RAID 5. The two above figures demonstrate two different scenarios. The scenario in Figure 3 demonstrates when the stager is running concatenator as well as the tape uploader. The graphs x – axis represents the transfer rate in MB/s while thy y – axis represents the number of events that were transferred at a particular speed. The majority of the events that are being sent are on average being transferred at an average rate of about 12 MB/s. In Figure 4 the concatenation is removed from the stagers so the I/O to the disk RAID 5 is limited to the transfer of the data on the stagers. The difference is dramatic. The number of events that are send at a speed of 25 MB/s or higher constitute the majority of the events. This change nearly doubles the transfer rate of the information. The essence of my project at CDF entailed removing the concatenation to the workers to enable the stagers to achieve higher transfer rates to the disk storage.

Proposed Structure

To improve the tape write speed on the CDF farm it was imperative the concatenation be moved to another location. The proposed structure is displayed in Figure 5.

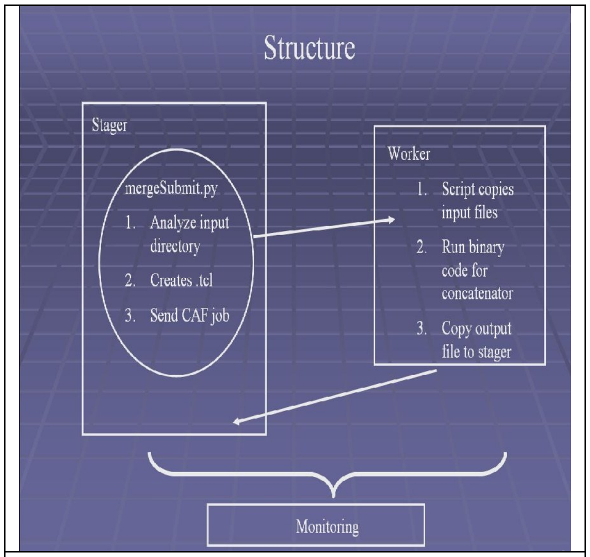

Structure
Stager
mergeSubmit.py
1. Analyze input directory
2. Creates .tcl
3. Send CAF job
Worker
1. Script copies input files
2. Run binary code for concatenator
3. Copy output file to stager
Monitoring

Figure 5 (the schematic of the proposed structure of the concatenation process)

The stagers are responsible for running the python script mergeSubmit.py. This code will analyze the input directory. The reason for this is so it can make a list of the input files needed for each of the production datasets. The list of input files is inserted in a file called a Tool Command Langue (.tcl) file which drives a binary code for concatenation. Once this file has been composed a job can be submitted to CAF. CAF divides the job into sessions and assigns to the workers. The workers will be held with the responsibility of concatenation. Once on the workers the script copies the input files. It knows where the files are because of the constructed .tcl file. Additionally when the submission to the worker is made certain parameters are established to specify certain parameters of the specific submission. The binary code for the concatenation is then run. The goal is creating files that are 1-2 GB in size. The reason for this is to optimize the speed that the data is being written to the tape drives. The output files are then written back to the stager. An advantage of this process is the output file generated is on the worker. This allows for it to be accessed by other mean instead of it being localized on a stager. For this configuration the output file is then written back to the stager. This system is intended to circumvent the problem of the RAID 5 getting numerous I/O calls which affect the rate data can be transmitted to the tape drives.

```
#!/bin/bash

name=`basename $0`

. ./cdfopr/scripts/common_procedures
. ./cdfopr/scripts/parse_parameters $* > temp_parse_log

echo $TCL_FILE
echo $PARAM_USER
echo $PARAM_HOST
echo $PARAM_PATH



cmd="fcp -c ${RCP}
${FCP_USER}@${FCP_HOST}:${PARAM_PATH}/${TCL_FILE}
.";
$cmd
STATUS=$?;

if[ STATUS -ne 0 ];then
   echo "$TCL_FILE was not able to be copied"
   exit 1;
fi

for file_loc in ` temp.awk –v num=$SEGMENT_NUMBER `;do

   cmd="fcp -c ${RCP}
${FCP_USER}@${FCP_HOST}:${file_loc} . ";
   $cmd
   STATUS=$?;

   if [ STATUS -ne 0 ];then
   echo "$file_loc was not able to be copide
   fi

done
```

Figure 6 (find_tcl.sh code used to extract the corresponding location of the input files and send the files to the appropriate worker)

```
#!/bin/awk

BEGIN {
  flag = 0;
}

/SEGMENT_NUMBER/ {
  if ($5 == seg) {
    flag = 1
  }
}
```

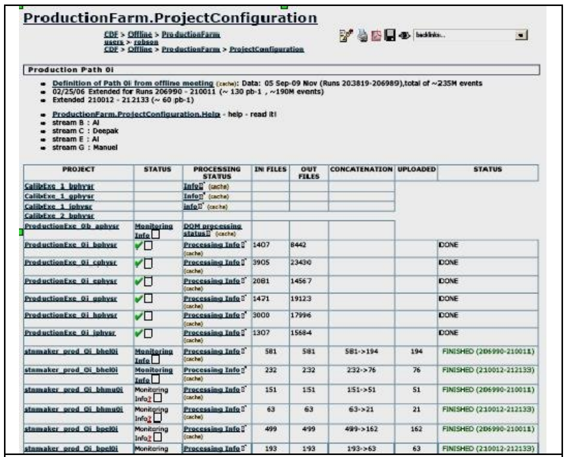
```
/include/ {
  if (flag == 1) {
    print $0
  }
}

{
  if ($1 == "}") {
     flag = 0
  }
#  if (($4 == "==") && (ENVIRON["SEGMENT_NUMBER"] ==
$5)) {print $0}
}
```

Figure 7 (temp.awk code that would extract the information needed from the .tcl file)

Figure 6 displays a short excerpt of the code used to perform the proposed task. The code is a combination of shell scripting and the awk editor. The awk editor helped in parsing the .tcl file to extract the location of the input files associated with a specific project. This code would be implemented right after step 2 on the stager in Figure 5. While this entire process is taking place the farm is being monitored. Monitoring is an essential part of the CDF farms. It enables the farm members to maintain the entire farm system in an efficient manner.

Farm Control & Monitoring

The CDF Farm is a huge system that requires the attention on many individuals. To take the most advantage of those individuals time there needs to be an intuitive interface. Farm maintenance is significantly simplified by introducing wiki-based farm projects. Tikiwiki[8] software is an extremely efficient tool for webbased documentation. Additionally its underlying database serverkeeps a history of all changes to the Farm Projects. The tiki pages with Project configurations give the possibility to: keep track of all existing projects, start or stop the projects, change resource sharing between projects. Several of the pages that the farms relied on needed slight modification to improve the overall presentation of the information on the tiki pages.



ProductionFarm.ProjectConfiguration

CDF > Offline > ProductionFarm
users > robson
CDF > Offline > ProductionFarm > ProjectConfiguration

backlinks...

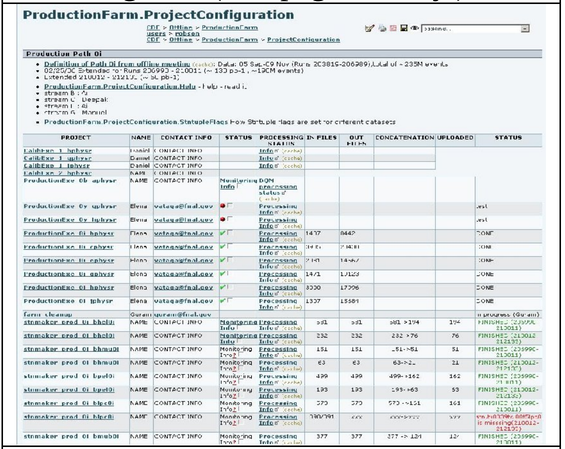
Production Path 0i

- Definition of Path 0i from offline meeting (cache): Data: 05 Sep-09 Nov (Runs 203819-206989),total of ~235M events
- 02/25/06 Extended for Runs 206990 - 210011 (~ 130 pb-1 , ~190M events)
- Extended 210012 - 212133 (~ 60 pb-1)

- ProductionFarm.ProjectConfiguration.Help - help - read it!
- stream B : Al
- stream C : Deepak
- stream E : Al
- stream G : Manuel

PROJECT	STATUS	PROCESSING STATUS	IN FILES	OUT FILES	CONCATENATION	UPLOADED	STATUS
CalibExe_1_bphysr		Info (cache)					
CalibExe_1_gphysr		Info (cache)					
CalibExe_1_iphysr		info (cache)					
CalibExe_2_bphysr							
ProductionExe_0b_aphysr	Monitoring Info ☐	DQM processing status (cache)					
ProductionExe_0i_bphysr	✓☐	Processing Info (cache)	1407	8442			DONE
ProductionExe_0i_cphysr	✓☐	Processing Info (cache)	3905	23430			DONE
ProductionExe_0i_ephysr	✓☐	Processing Info (cache)	2081	14567			DONE
ProductionExe_0i_gphysr	✓☐	Processing Info (cache)	1471	19123			DONE
ProductionExe_0i_hphysr	✓☐	Processing Info (cache)	3000	17996			DONE
ProductionExe_0i_iphysr	✓☐	Processing Info (cache)	1307	15684			DONE
stnmaker_prod_0i_bhel0i	Monitoring Info ☐	Processing Info (cache)	581	581	581->194	194	FINISHED (206990-210011)
stnmaker_prod_0i_bhel0i	Monitoring Info ☐	Processing Info (cache)	232	232	232->76	76	FINISHED (210012-212133)
stnmaker_prod_0i_bhmu0i	Monitoring Info? ☐	Processing Info (cache)	151	151	151->51	51	FINISHED (206990-210011)
stnmaker_prod_0i_bhmu0i	Monitoring Info? ☐	Processing Info (cache)	63	63	63->21	21	FINISHED (210012-212133)
stnmaker_prod_0i_bpel0i	Monitoring Info? ☐	Processing Info (cache)	499	499	499->162	162	FINISHED (206990-210011)
stnmaker_prod_0i_bpel0i	Monitoring	Processing Info	193	193	193->63	63	FINISHED (210012-212133)

Figure 8 (tiki page initially)



ProductionFarm.ProjectConfiguration

CDF > Offline > ProductionFarm
users > robson
CDF > Offline > ProductionFarm > ProjectConfiguration

backlinks...

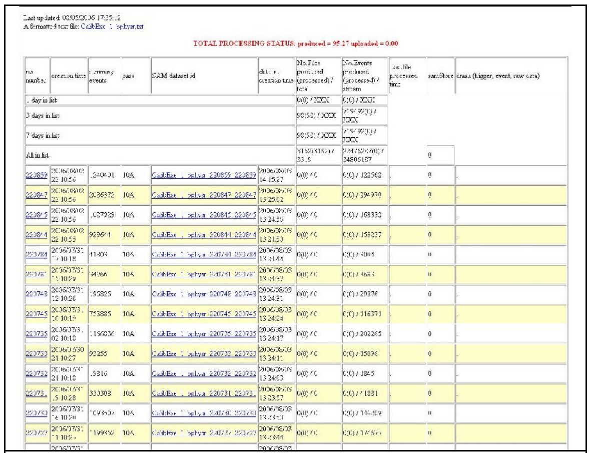
Production Path 0i

- Definition of Path 0i from offline meeting (cache): Data: 05 Sep-09 Nov (Runs 203819-206989),total of ~235M events
- 02/25/06 Extended for Runs 206990 - 210011 (~ 130 pb-1 , ~190M events)
- Extended 210012 - 212133 (~ 60 pb-1)
- ProductionFarm.ProjectConfiguration.Help - help - read it
- stream B : Al
- stream C : Deepak
- stream E : Al
- stream G : Manuel
- ProductionFarm.ProjectConfiguration.StntupleFlags How Stntuple flags are set for different datasets

PROJECT	NAME	CONTACT INFO	STATUS	PROCESSING STATUS	IN FILES	OUT FILES	CONCATENATION	UPLOADED	STATUS
CalibExe_1_bphysr	Daniel	CONTACT INFO		Info (cache)					
CalibExe_1_gphysr	Daniel	CONTACT INFO		Info (cache)					
CalibExe_1_iphysr	Daniel	CONTACT INFO		Info (cache)					
CalibExe_2_bphysr	NAME	CONTACT INFO							
ProductionExe_0b_aphysr	NAME	CONTACT INFO	Monitoring Info ☐	DQM processing status (cache)					
ProductionExe_0y_gphysr	Elena	vataga@fnal.gov	● ☐	Processing Info (cache)					test
ProductionExe_0y_hphysr	Elena	vataga@fnal.gov	● ☐	Processing Info (cache)					test
ProductionExe_0i_bphysr	Elena	vataga@fnal.gov	✓☐	Processing Info (cache)	1407	8442			DONE
ProductionExe_0i_cphysr	Elena	vataga@fnal.gov	✓☐	Processing Info (cache)	3905	23430			DONE
ProductionExe_0i_ephysr	Elena	vataga@fnal.gov	✓☐	Processing Info (cache)	2081	14567			DONE
ProductionExe_0i_gphysr	Elena	vataga@fnal.gov	✓☐	Processing Info (cache)	1471	19123			DONE
ProductionExe_0i_hphysr	Elena	vataga@fnal.gov	✓☐	Processing Info (cache)	3000	17996			DONE
ProductionExe_0i_iphysr	Elena	vataga@fnal.gov	✓☐	Processing Info (cache)	1307	15684			DONE
farm_cleanup	Guram	guram@fnal.gov							in progress (Guram)
stnmaker_prod_0i_bhel0i	NAME	CONTACT INFO	Monitoring Info ☐	Processing Info (cache)	581	581	581->194	194	FINISHED (206990-210011)
stnmaker_prod_0i_bhel0i	NAME	CONTACT INFO	Monitoring Info ☐	Processing Info (cache)	232	232	232->76	76	FINISHED (210012-212133)
stnmaker_prod_0i_bhmu0i	NAME	CONTACT INFO	Monitoring Info? ☐	Processing Info (cache)	151	151	151->51	51	FINISHED (206990-210011)
stnmaker_prod_0i_bhmu0i	NAME	CONTACT INFO	Monitoring Info? ☐	Processing Info (cache)	63	63	63->21	21	FINISHED (210012-212133)
stnmaker_prod_0i_bpel0i	NAME	CONTACT INFO	Monitoring Info? ☐	Processing Info (cache)	499	499	499->162	162	FINISHED (206990-210011)
stnmaker_prod_0i_bpel0i	NAME	CONTACT INFO	Monitoring Info? ☐	Processing Info (cache)	193	193	193->63	63	FINISHED (210012-212133)
stnmaker_prod_0i_blpc0i	NAME	CONTACT INFO	Monitoring Info? ☐	Processing Info (cache)	579	579	579->161	161	FINISHED (206990-210011)
stnmaker_prod_0i_blpc0i	NAME	CONTACT INFO	Monitoring Info? ☐	Processing Info (cache)	390/391	xxx	xxx->xxx	xxx	[illegible] is missing(210012-212133)
stnmaker_prod_0i_bmub0i	NAME	CONTACT INFO	Monitoring Info? ☐	Processing Info (cache)	377	377	377->124	124	FINISHED (206990-210011)

Figure 9 (tiki page after modifications)

Figure 8 shows the initial orientation of the page. This information is all presented in an unclear manner and there isn't a way to contact the individual running the particular project. To make these edits possible a few modifications needed to be made to include columns and remove others. This was all done using the online tiki editor. Another group of pages were modified to make the information more presentable. Instead of using the online tikiwiki editor python script was edited. The script itself was made to interface with html format. Also some adjustments were made including: removing columns, adding jobid information, and fixing hyperlinks. Figures 10 and 11 show the page before and after respectively.

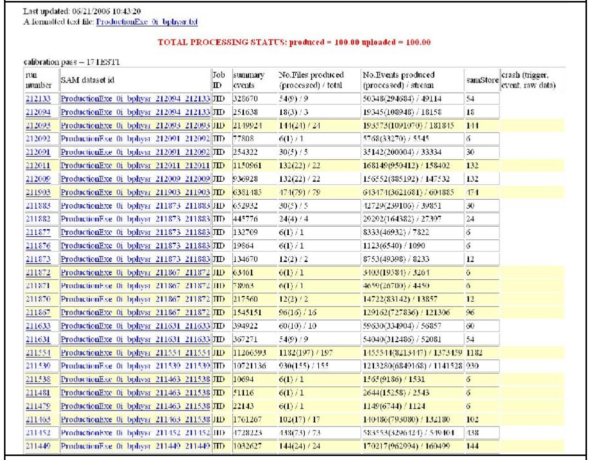


Last updated: 08/05/2006 17:35:12
A formatted text file: CalibExe 1 bphysr.txt

TOTAL PROCESSING STATUS: produced = 95.27 uploaded = 0.00

run number	creation time	summary events	pass	SAM dataset id	date creation time	No.Files produced (processed) / total	No.Events produced (processed) / stream	sam file processed time	samStore	crash (trigger, event, raw data)
1 day in list						0(0) / XXXX	0(0) / XXXX			
3 days in list						90(58) / XXXX	719492(0) / XXXX			
7 days in list						90(58) / XXXX	719492(0) / XXXX			
All in list						5152(5152) / 33.5	22475287(0) / 34805187		0	
220859	2006/08/02 22 10:56	1240401	10A	CalibExe 1 bphysr 220859 220859	2006/08/03 14 15:27	0(0) / 0	0(0) / 122562	.	0	.
220847	2006/08/02 22 10:56	2036372	10A	CalibExe 1 bphysr 220847 220847	2006/08/03 13 25:02	0(0) / 0	0(0) / 294970	.	0	.
220845	2006/08/02 22 10:56	1037929	10A	CalibExe 1 bphysr 220845 220845	2006/08/03 13 24:59	0(0) / 0	0(0) / 168332	.	0	.
220844	2006/08/02 22 10:55	929644	10A	CalibExe 1 bphysr 220844 220844	2006/08/03 13 21:50	0(0) / 0	0(0) / 153237	.	0	.
220784	2006/07/31 17 10:18	41403	10A	CalibExe 1 bphysr 220784 220784	2006/08/03 13 21:44	0(0) / 0	0(0) / 3004		0	
220781	2006/07/31 11 10:29	94966	10A	CalibExe 1 bphysr 220781 220781	2006/08/03 13 [illegible]	0(0) / 0	0(0) / 3683		0	
220748	2006/07/31 12 10:26	195825	10A	CalibExe 1 bphysr 220748 220748	2006/08/03 13 24:51	0(0) / 0	0(0) / 29376	.	0	.
220745	2006/07/31 10 10:19	753885	10A	CalibExe 1 bphysr 220745 220745	2006/08/03 13 24:24	0(0) / 0	0(0) / 116371	.	0	.
220735	2006/07/31 02 10:18	1156036	10A	CalibExe 1 bphysr 220735 220735	2006/08/03 13 24:17	0(0) / 0	0(0) / 202265	.	0	.
220733	2006/07/30 21 10:27	93355	10A	CalibExe 1 bphysr 220733 220733	2006/08/03 13 24:11	0(0) / 0	0(0) / 15096	.	0	.
220732	2006/07/31 21 10:18	19316	10A	CalibExe 1 bphysr 220732 220732	2006/08/03 13 24:03	0(0) / 0	0(0) / 1845	.	0	.
220731	2006/07/31 19 10:28	330303	10A	CalibExe 1 bphysr 220731 220731	2006/08/03 13 23:57	0(0) / 0	0(0) / [illegible]	.	0	.
220730	2006/07/31 16 10:20	1093507	10A	CalibExe 1 bphysr 220730 220730	2006/08/03 13 23:50	0(0) / 0	0(0) / [illegible]		0	
220727	2006/07/31 11 10:24	1199352	10A	CalibExe 1 bphysr 220727 220727	2006/08/03 13 23:44	0(0) / 0	0(0) / [illegible]		0	
	2006/07/31				2006/08/03					

Figure 10 (web page initially)



Last updated: 06/21/2006 10:43:20
A formatted text file: ProductionExe 0i bphysr.txt

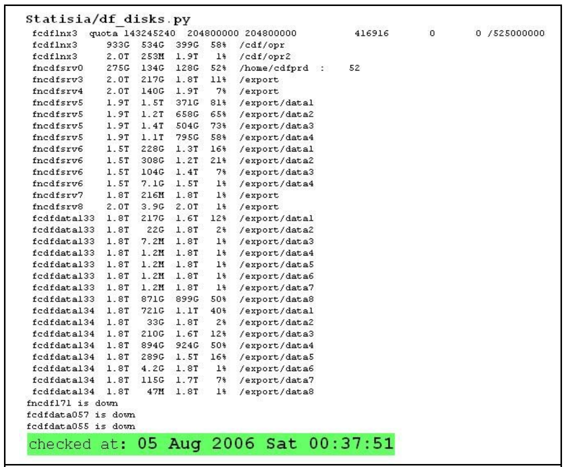
TOTAL PROCESSING STATUS: produced = 100.00 uploaded = 100.00

calibration pass – 17 TEST1

run number	SAM dataset id	Job ID	summary events	No.Files produced (processed) / total	No.Events produced (processed) / stream	samStore	crash (trigger, event, raw data)
212133	ProductionExe 0i bphysr 212094 212133	JID	328670	54(9) / 9	50348(294684) / 49114	54	
212094	ProductionExe 0i bphysr 212094 212133	JID	251638	18(3) / 3	19345(108948) / 18158	18	
212093	ProductionExe 0i bphysr 212093 212093	JID	2149924	144(24) / 24	193573(1091070) / 181845	144	
212092	ProductionExe 0i bphysr 212091 212092	JID	77808	6(1) / 1	5768(33270) / 5545	6	
212091	ProductionExe 0i bphysr 212091 212092	JID	254322	30(5) / 5	35142(200004) / 33334	30	
212011	ProductionExe 0i bphysr 212011 212011	JID	1150961	132(22) / 22	168149(950412) / 158402	132	
212009	ProductionExe 0i bphysr 212009 212009	JID	936928	132(22) / 22	156552(885192) / 147532	132	
211903	ProductionExe 0i bphysr 211903 211903	JID	6381485	474(79) / 79	643474(3621681) / 604885	474	
211883	ProductionExe 0i bphysr 211873 211883	JID	652932	30(5) / 5	42729(239106) / 39851	30	
211882	ProductionExe 0i bphysr 211873 211883	JID	445776	24(4) / 4	29292(164382) / 27397	24	
211877	ProductionExe 0i bphysr 211873 211883	JID	132709	6(1) / 1	8333(46932) / 7822	6	
211876	ProductionExe 0i bphysr 211873 211883	JID	19864	6(1) / 1	1123(6540) / 1090	6	
211873	ProductionExe 0i bphysr 211873 211883	JID	134670	12(2) / 2	8753(49398) / 8233	12	
211872	ProductionExe 0i bphysr 211867 211872	JID	63461	6(1) / 1	3403(19584) / 3264	6	
211871	ProductionExe 0i bphysr 211867 211872	JID	78963	6(1) / 1	4659(26700) / 4450	6	
211870	ProductionExe 0i bphysr 211867 211872	JID	217560	12(2) / 2	14722(83142) / 13857	12	
211867	ProductionExe 0i bphysr 211867 211872	JID	1545151	96(16) / 16	129162(727836) / 121306	96	
211633	ProductionExe 0i bphysr 211631 211633	JID	394922	60(10) / 10	59630(334904) / 56857	60	
211631	ProductionExe 0i bphysr 211631 211633	JID	367271	54(9) / 9	54040(312486) / 52081	54	
211554	ProductionExe 0i bphysr 211554 211554	JID	11266593	1182(197) / 197	1455544(8213447) / 1373459	1182	
211539	ProductionExe 0i bphysr 211539 211539	JID	10721136	930(155) / 155	1213280(6849168) / 1141528	930	
211538	ProductionExe 0i bphysr 211463 211538	JID	10694	6(1) / 1	1565(9186) / 1531	6	
211481	ProductionExe 0i bphysr 211463 211538	JID	51116	6(1) / 1	2644(15258) / 2543	6	
211479	ProductionExe 0i bphysr 211463 211538	JID	22143	6(1) / 1	1149(6744) / 1124	6	
211463	ProductionExe 0i bphysr 211463 211538	JID	1761267	102(17) / 17	140186(793080) / 132180	102	
211452	ProductionExe 0i bphysr 211452 211452	JID	4728223	438(73) / 73	583553(3296424) / 549404	438	
211449	ProductionExe 0i bphysr 211449 211449	JID	1032627	144(24) / 24	170217(962994) / 160499	144	

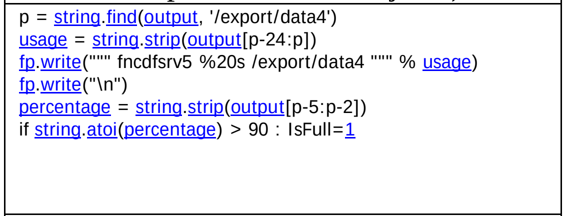
Figure 11 (web page after modifications)

Another python script that was edited is df_disk.py. The intention with this script was to display the disk space on the 32 partitions on the different servers. This expedited the process of obtaining the server information. The code and the respective result are displayed in Figures 12 and 13.



```
Statisia/df_disks.py
 fcdflnx3  quota 143245240  204800000 204800000         416916       0        0 /525000000
 fcdflnx3     933G  534G  399G  58%  /cdf/opr
 fcdflnx3     2.0T  253M  1.9T   1%  /cdf/opr2
 fncdfsrv0    275G  134G  128G  52%  /home/cdfprd  :    52
 fncdfsrv3    2.0T  217G  1.8T  11%  /export
 fncdfsrv4    2.0T  140G  1.9T   7%  /export
 fncdfsrv5    1.9T  1.5T  371G  81%  /export/data1
 fncdfsrv5    1.9T  1.2T  658G  65%  /export/data2
 fncdfsrv5    1.9T  1.4T  504G  73%  /export/data3
 fncdfsrv5    1.9T  1.1T  795G  58%  /export/data4
 fncdfsrv6    1.5T  228G  1.3T  16%  /export/data1
 fncdfsrv6    1.5T  308G  1.2T  21%  /export/data2
 fncdfsrv6    1.5T  104G  1.4T   7%  /export/data3
 fncdfsrv6    1.5T  7.1G  1.5T   1%  /export/data4
 fncdfsrv7    1.8T  216M  1.8T   1%  /export
 fncdfsrv8    2.0T  3.9G  2.0T   1%  /export
 fcdfdata133  1.8T  217G  1.6T  12%  /export/data1
 fcdfdata133  1.8T   22G  1.8T   2%  /export/data2
 fcdfdata133  1.8T  7.2M  1.8T   1%  /export/data3
 fcdfdata133  1.8T  1.2M  1.8T   1%  /export/data4
 fcdfdata133  1.8T  1.2M  1.8T   1%  /export/data5
 fcdfdata133  1.8T  1.2M  1.8T   1%  /export/data6
 fcdfdata133  1.8T  1.2M  1.8T   1%  /export/data7
 fcdfdata133  1.8T  871G  899G  50%  /export/data8
 fcdfdata134  1.8T  721G  1.1T  40%  /export/data1
 fcdfdata134  1.8T   33G  1.8T   2%  /export/data2
 fcdfdata134  1.8T  210G  1.6T  12%  /export/data3
 fcdfdata134  1.8T  894G  924G  50%  /export/data4
 fcdfdata134  1.8T  289G  1.5T  16%  /export/data5
 fcdfdata134  1.8T  4.2G  1.8T   1%  /export/data6
 fcdfdata134  1.8T  115G  1.7T   7%  /export/data7
 fcdfdata134  1.8T   47M  1.8T   1%  /export/data8
fncdf171 is down
fcdfdata057 is down
fcdfdata055 is down
checked at: 05 Aug 2006 Sat 00:37:51
```

Figure 12 (info on the partitions which are updated with cron jobs)



```
p = string.find(output, '/export/data4')
usage = string.strip(output[p-24:p])
fp.write(""" fncdfsrv5 %20s /export/data4 """ % usage)
fp.write("\n")
percentage = string.strip(output[p-5:p-2])
if string.atoi(percentage) > 90 : IsFull=1
```

Figure 13 (an excerpt of code that helps generate the partition information in Figure 12)

The information about the servers are continuously updated using the cron jobs. The information's previous update was made Aug 5 as can be seen at the bottom of figure 12. All of these systems are interconnected resulting in one large cohesive farm unit. Enabling users to access and monitor the farms in an effective and efficient manner.

Conclusion

The monitoring and concatenation systems at the CDF Farms are an essential part to its function. The monitoring system obtained the needed modifications. The concatenation aspect of the project received several changes.

This system is nearly operational. The benefits for operating in the new configuration are significantly larger. Removing the concatenation from the stager and to the workers will help improve the rate at which information is written to the tape drives. The Monte Carlo events as well as ntuples will be able to benefit from these modifications. In the future when this project is fully implemented its true benefits can be seen.

Acknowledgments

I would like to thank my supervisor Elena Vataga for showing an indispensable amount of patience. I cannot recall when she did not make time to address my questions. I would also like to thank Murat Pavel. Murat would always find time to squeeze me into his busy schedule. I learned many things from the both of them. Last but not least the SIST committee and the individuals affiliated, for giving me this opportunity to explore my interests.

References

1. High Energy Physics Information Ceter Feb 14, 2002 , http://www.hep.net/

2. Collidewr Detector at FermiLab Aug 2006 http://www-cdf.fnal.gov/about/index.html

3. SAM Home Page March 2 2006 http://d0db.fnal.gov/sam/

4. Condor Project Homepage August 2006 http://www.cs.wisc.edu/condor/

5. Concurrent Versions System August 2006 http://en.wikipedia.org/wiki/Concurrent_Versions_System

6. MySQL AB August 2006 http://www.mysql.com/

7. Tikiwiki: The Tikiwiki Community August 2006 http://tikiwiki.org/