Automated Finite State Workflow for Distributed Data Production

L Hajdu, L Didenko, J Lauret, J Amol, W Betts, H J Jang, and S Y Noh

1 Software and Computing Group, RHIC/ STAR experiment, Brookhaven National Lab, PO Box 5000, Upton, NY 11973-5000, USA
2 Korea Institute of Science and Technology Information, 245 Daehangno, Yuseong, Daejeon 305-806, KOREA
3 Korea University of Science and Technology, Yuseong, Dajeon 305-350, KOREA

* Corresponding Author: Mr. Levente B. Hajdu, LBHajdu@BNL.gov

Abstract. In statistically hungry science domains, data deluges can be both a blessing and a curse. They allow the narrowing of statistical errors from known measurements, and open the door to new scientific opportunities as research programs mature. They are also a testament to the efficiency of experimental operations. However, growing data samples may need to be processed with little or no opportunity for huge increases in computing capacity. A standard strategy has thus been to share resources across multiple experiments at a given facility. Another has been to use middleware that “glues” resources across the world so they are able to locally run the experimental software stack (either natively or virtually). We describe a framework STAR has successfully used to reconstruct a ~400 TB dataset consisting of over 100,000 jobs submitted to a remote site in Korea from STAR’s Tier 0 facility at the Brookhaven National Laboratory. The framework automates the full workflow, taking raw data files from tape and writing Physics-ready output back to tape without operator or remote site intervention. Through hardening we have demonstrated 97(±2)% efficiency, over a period of 7 months of operation. The high efficiency is attributed to finite state checking with retries to encourage resilience in the system over capricious and fallible infrastructure.

1. Introduction
The STAR experiment, located in one of the interaction regions of the Relativistic Heavy Ion Collider (RHIC), started taking physics events in year 2000. Since then it took approximately two billion events per year until 2012, when upgrades skyrocketed the number of recorded events past six billion events a year. In 2016 the number of events recorded by the STAR detector is expected to climb again. With the rising number of events, the reconstruction time is also increasing. Hence there is considerable demand to distribute some of the reconstruction load from the RCF (RHIC Computing Facility) to STAR Tier 1 sites in order to increase parallelization and thus decrease the total time between data taking and analysis.
STAR has been offloading simulation workload to Tier 1 sites from the very beginning of its operation, even before the inception of the Open Science Grid (OSG). Simulation was preferentially selected as a target for offloading because relative to real captured datasets, simulated datasets are orders of magnitude smaller. The smaller dataset size also means the whole output of the dataset can be stored under one path on centralized disk. For simulation input there are no input events, only non-unique input parameters and a random seed (initializer), hence there is no need in the workflow to deliver a unique input file to each job. A dataset of arbitrary size can be produced. Because of simulation jobs’ non-unique input, inefficiencies are dealt with by over requesting data to account for the number lost due to inefficiencies. In short, simulation is an easily streamlined, failure tolerant target to offload to satellite grid connected computing sites, and hence it is always considered first.

In contrast, reconstruction of real data has an input file of significant size. Each input file is unique. The number of input files is limited to what was taken during the run and may represent the bare minimum to reach a physics goal. Thus it is important to achieve efficiency close to 100 percent. The trick of over submission as done with simulation is not an option. And in the event of failure of reconstruction jobs, there must be recovery, preferably automated. These requirements make reconstruction of real data a more complicated task.

Local reconstruction has a simpler path routing input to the job and copying output files back, as the whole path is limited to within a single site. Whereas offloaded reconstruction has to use a national or international backbone, which is less reliable because the number of routing points is greater. Besides the added complexity of the network, there is also added complexity in the software stack as a grid framework must be leveraged, in order to handle additional requirements such as user account mapping, authentication, grid file transfer, and job transmission to the remote batch system. The question is can a grid job processing framework be designed to overcome inefficiencies brought by the added complexity to achieve efficiency approaching the reconstruction efficiency of the local batch system? To answer this question STAR developed a framework and offloaded one of its datasets to its Tier 1 site at KISTI (Korea Institute of Science and Technology Information).

2. **KISTI**
The Korea Institute of Science and Technology Information (KISTI), a STAR Tier 1 site, provided the worker node resources for this production, specifically about 1024 slots. All sites are different in terms of the level of effort they provide, additional (grid) software stack, and user access rights internally and through gateways to the outside world. Though computing power for the production run was plentiful, manpower was limited along with central storage space. Taking this profile into account with our implementation we wanted to avoid deploying new software at KISTI that would require administrative intervention, such as delegated file transfer services such as BeStMan SRM [1]. These services allow caching the output files at the local site so that the job slot can be freed up right after reconstruction without needing to deal with the copy within the run time of the job. However in this instance, without deploying delegated file transfer services, the copy would be handled within the runtime of the job. Optimizing a dedicated network path is always key to the success of large productions, but even more important in this case because it adds to the jobs run time, so STAR reached out to ESnet [2] and KREONET in order to configure a static routing path. The other feature of delegated file transfer services is that the in-job-runtime copy has to replicate the fault tolerance for the transfer. A production that takes many months will see many failures such as crashed file transfer gatekeepers and network outages. To replicate this fault tolerance, a copy script was designed to detect all the failure states for the globus-url-copy [3] command, such as freezes and unexpected return states, and a set of re-tries was implemented. Each time the job re-tries it would try one of three randomly selected gatekeepers and each time the copy failed it adds one minute to the delay before the next retry. The job will retry for up to 24 hours, which is approximately half of the average job’s run time. Small network interruptions will not cause the loss of output files, nor will the crash of one gatekeeper because the random selection emulates load balancing with fail-over. This method was found to be practical for production. The time spent copying both input and output was typically less
than 1% of the total run time of the job. The total input for this production is 397.3 TB, and the total output is 213.7 TB. For the movement of data in both directions the methods employed with KISTI worked very well with little measurable loss.

3. Stages of production

Figure 1 shows the stages of STAR’s grid production framework. Many of the components are used for other functions and have been reused here, because they are well debugged, actively maintained, proven to work over a decade, and cut down on development time. These components include the STAR file catalog [4], STAR data carousel [5] [6], STAR Unified Meta Scheduler [7], HTCondor [8] and, Globus [9].

![Stages of STAR’s production system](image)

**Figure 1.** Stages of STAR’s production system – each box and stages are explained in details as subsections of section 3.

3.1. Input File Staging

The first task illustrated in box one, is staging input files. The input files are initially stored on tape. They have the extension *.daq which stands for data acquisition, meaning it is data read out of STAR’s detectors. The STAR file catalog holds the list of files belonging to the dataset that needs to be reconstructed. It holds LFNs (logical file names), PFNs (physical file names) as well as a breadth of MetaData (MD) users may key on. The file list from the STAR file catalog is passed via a script which is part of the grid production framework to the STAR data carousel and an entry stub for each file is made in the production DB. Once the files are restored from tape to central disk the entry in the production database is marked as restored and input staging is completed.

3.2. Job Submission

Once the input is staged we move on to submission. With the stub in the production database and the input file restored, a script will check the production database for restored files and compose a list of...
the restored files. The list is passed to SUMS, the STAR Unified Meta Scheduler. SUMS provides a uniform interface to STAR collaborators across different sites, batch systems, and grid interfaces. For the purpose of this framework SUMS transforms the input file into an actual job by generating a script which runs the reconstruction program, passes parameters, initiates transfer of input and output files, handles message passing for the purpose of monitoring, and does other wrapper housekeeping functions such as clean-up of temporary disk space. Each of these generated job scripts are submitted by SUMS to the HTCondor batch system. The HTCondor batch system in turn via an underlying level of Globus grid libraries passes the job to the KISTI site, or another grid site as specified in the job description file, also generated by SUMS at the time of submission. Globus handles mapping of user accounts via grid certificates [10]. The production database is then updated to reflect the state of the job as submitted.

3.3. Job Runtime Monitoring
Once the job is submitted we move on to the next phase in box number three. A script will run a condor_q command and parse the output recovered, it will also iterate over all of the jobs in the production database marked as submitted but not done and try to correlate them and update their states, which can be idle, run, held, and done. The job also sends back messages in the form of small files that are copied back the same way as output files that provide fine detail about what the job is doing internally. Though using file transfer to pass messages may seem inelegant it is guaranteed to be available because it is used for transferring the output files and the same script that retries failed copies can be reused to harden message passing. These messages are useful because if jobs run over their expected run times, there is information in the database as to what step the job is working on and for how long. The internal job states ideally progress from none, to input copy started, input copy stopped, reconstruction started, output copy started, and output copy stopped.

3.4. Job Output Scanning
When the monitor sees the job has exited the queue, the job passes to the next phase which is scanning of the output files and logs. The reconstruction and batch system logs are scanned for known error messages. If the output files or logs do not exist, the job is marked as failed. If the binary output files exist, their sizes are compared with the stated output file sizes from the log. File checksum comparison between source and destination of the binary output files is not necessary as this is done internally by the copy itself – only a check for truncation due to early termination of the copy is needed. If any of the above failures are noted the job is marked for resubmission, which the system will try to do up to a fixed number of times in case of failure. There needs to be a limit on the number of attempts to resubmit the job because some jobs are irrevocably broken regardless of the number of times the job is resubmitted such as a job receiving a corrupted input file. We have noted 6 of these jobs out of 105,632 or 0.0056%. After these seven tries the system gives up.

3.5. Output File Placement
If the job finishes and is marked as successful after one or more submissions it can pass on to the next phase illustrated in box number five, placement. At this point the reconstructed file with extension *.MuDst, which holds the reconstructed events, is sitting in the output buffer. It is moved to another central disk location for general analysis by users, from there a copy is sunk back to HPSS and it can even be moved to distributed disk if the analysis is to be immediate. The production database is once again updated, to indicate a copy has been pushed back to HPSS tape. If this full path is done, reconstruction is considered a success.

3.6. Output File Cataloguing
The STAR data management system spiders will eventually find the output file and register it in the STAR file catalog so users can request it. This is illustrated in box six, cataloguing. This process is fully automated.
4. Dataflow
The diagram in Figure 2 shows STAR’s grid production system dataflow. Starting from the left, the input files are originally located on HPSS tapes with the list of files to be restored coming from the STAR file catalog. The carousel will recover as many files as can possibly be fit into the grid production system’s DAQ buffer on NFS disk at any one time. This will only be a small fraction of the total dataset. SUMS picks up the list of files and submits one file per job to the remote site’s batch system where they eventually execute on the worker nodes. While the job is running it passes back messages. Once the job is done it copies the output MuDst file back to the local Tier 0 site (BNL). The file is placed into the output buffer. Once the scanner passes over the file it is copied to both centralized disk, where it can be used for physics analysis, and to HPSS tape for backup. At multiple stages along the way the scanner is checking the throughput for validity, advancing the dataflow with its movement scripts, and retrying steps where possible and necessary.

![Figure 2. STAR’s production framework data flow](image)

5. Finite State Checking
Because of the added complexity of grid production, stability can sometimes be poor. A high degree of efficiency could be gained by using multiple passes always resubmitting the failed portion. However,
this is a poor solution as it wastes resources having to stage and resubmit each failed file, whereas in a finite state checking model, the workflow only repeats the smallest number of steps required to achieve successful reconstruction. In order to isolate the finite states, the states must be identified, and there must be checks inside the framework, such as those shown in Figure 3, to determine if the state finished successfully, or whether a state needs to be reattempted. These checks include file system-based checks for file existence and size, queue state polling, and log file parsing.

6. Outcomes
The first production campaign to which the STAR Grid data production framework was assigned consisted of proton on proton collisions at 200 GeV. The dataset consisted of 105,632 *.daq input files totalling 397.3 TB transferred to KISTI for reconstruction. Each of these input files in turn produced a MuDst output file and a log file, both of which had to be transferred back from KISTI to BNL. The sum of the MuDst files which were transferred back to BNL is 213.7 TB. The average run time for one job is about 48 hours, not including time spent idle in the queue. This production was allocated 1,025 slots which were fully saturated with jobs, excluding normal unavoidable batch system overhead. And the total production time was 9 months consuming 6,179,544 wall CPU hours. This length of time allowed for a variety of error states to manifest so that the framework’s efficiency at detecting and recovering these errors could be assessed.

| Number Of Jobs | Percent (%) | Job Submissions |
|----------------|-------------|-----------------|
| 102,043        | 96.6023     | 1               |
| 2,895          | 2.7406      | 2               |
| 493            | 0.4667      | 3               |
| 103            | 0.0975      | 4               |
| 83             | 0.0785      | 5               |
| 6              | 0.0056      | 6               |
| 3              | 0.0028      | 7               |
| 6              | 0.0056      | Not Runnable    |

A job is considered successful if for a given input file on tape both the output files and reconstruction log also exist on tape and are not truncated in any way, over 99% of jobs achieved success. The compounded slot utilization efficiency, taking into account the resubmissions was calculated at 97(±2)%. As this represents a slightly higher level of efficiency than local production, the outcome is very good indeed.

7. Conclusion
STAR has vast computing resource requirements and a history of offloading work to other sites whenever possible. We presented a finite state workflow for processing production jobs leveraging STAR’s KISTI Tier 1 site resources. This production framework leveraged past knowledge and features finite state checking and recovery using an automated yet simple set of components. This approach achieved an unprecedented 97(±2)% level of utilization efficiency. The effort shows that job efficiency running on the grid can be as good as local production.

Acknowledgments
We thank the KISTI Center in Korea, the Open Science Grid consortium, ESnet, and KREONET for providing resources and support. This work was supported in part by the Office of Nuclear Physics within the U.S. DOE Office of Science.
References

[1] Hjort E, Lauret J and Hajdu L 2006 Data and Computational Grid decoupling in STAR – An Analysis Scenario using SRM Technology. International Conference on Computing in High Energy and Nuclear Physics (Tata Institute of Fundamental Research, Mumbai, India)

[2] Newman H 2004 Networking for High Energy and Nuclear Physics as Global E-Science (International Conference on Computing in High Energy and Nuclear Physics, Interlaken, Switzerland)

[3] Allcock W 2003 GridFTP: Protocol Extensions to FTP for the grid (Global Grid Forum GFD) page 20

[4] Porter J 2001 STAR Database Implementations with MySQL (International Conference on Computing in High Energy and Nuclear Physics, CERN)

[5] Lauret J and Yu D. 2010 ERADAT and Data Carousel systems at BNL: A tool and UI for efficient access to data on tape with fair share policies capabilities (ACAT 2010 proceedings, PoS ACAT 023)

[6] Yu D and Lauret J 2011 Tape storage optimization at BNL (CHEP 2010 proceedings) J. Phys. Conf. Ser. 331 042045 doi:10.1088/1742-6596/331/4/042045

[7] Lauret J and Hajdu L 2006 Meta-configuration for dynamic resource brokering: the SUMS approach (International Conference on Computing in High Energy and Nuclear Physics, Tata Institute of Fundamental Research, Mumbai, India)

[8] Frey J, Tannenbaum T, Foster I, Livny M and Tuecke S 2001 Condor-G: A computation management agent for multi-institutional grids Proc. of the Tenth IEEE Symposium High Performance Distributed Computing (HPDC) (San Francisco) pp 7–9

[9] Foster I 2006 Globus Toolkit Version 4: Software for Service-Oriented Systems (IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779) pp 2-13

[10] Tuecke S, Welch V, Engert D, Pearlman L and Thompson M 2004 RFC3820 Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile