# STAR'S APPROACH TO HIGHLY EFFICIENT END-TO-END GRID PRODUCTION

**G. Agakishiev[1], N. Balashov[1], W. Betts[2], L. Didenko[2], L. Hajdu[2,a], V. Korenkov[1], E. Kuznetsov[1], J. Lauret[2], V. Mitsyn[1], Y. Panebratsev[1]**

[1] *Joint Institute for Nuclear Research, 6 Joliot-Curie, Dubna, Moscow region, 141980, Russia*

[2] *The STAR experiment, Brookhaven National Laboratory, Upton, New York 11973, USA*

E-mail: [a] LBHajdu@BNL.GOV

STAR's RHIC computing facility provides over 15K dedicated slots for data reconstruction. This number of slots is not always sufficient to satisfy an ambitious and data-challenging physics program, and harvesting resources from outside facilities is paramount to scientific success. However, constraints of remote sites (e.g. CPU time limits) do not always provide the flexibility of a dedicated farm. STAR has a breadth of smaller datasets (both in runtime and size) that can be easily offloaded to remote facilities with many such limits. Scavenged resources can be run with efficiency comparable to that of the local production and contributes additional computing time to an experiment that runs every year and therefore needs fast turnaround. We will discuss STAR's software stack of our grid production framework including features dealing with multi-site submission, automated re-submission, job tracking, as well as new challenges and possible improvements.

Keywords: grid, STAR, RHIC, BNL, JINR, production system, workflow, scheduler, Globus, HTCondor, HTCondorCE, data carousel, GridCE, file transfer

## 1. Introduction to the STAR experiment

The STAR (Solenoidal Tracker At RHIC) detector is located in one of the interaction regions of the RHIC (Relativistic Heavy Ion Collider), currently the second-highest-energy heavy-ion collider in the world, with a circumference of 2.4 miles (3.9 km). RHIC is extremely versatile with an energy ranging from 7.7 GeV to 200 GeV, 510GeV for protons and a wide particle species range from protons to uranium.

STAR's first physics run was in year 2000. The detector takes one data run, lasting about six months, every year-currently we are on our 17th physics run. This poses a huge data challenge as the dataset from one run has to be calibrated and reconstructed before the next dataset is captured. STAR's local computing facility (the RHIC Computing Facility or RCF) provides ~15,000 computing slots for this data processing. This allows between 1.2 to 1.4 passes of reconstruction whereas the typical high energy physics experiment takes in excess of five passes. In order to speed up the rate of scientific discoveries, STAR started utilizing grid resources in 2001 for simulation requests, where small efficiency losses are tolerable, and over-submission can cover for losses. Today the bulk of computing requests are reconstruction of detector data which requires higher efficiency as there is only a precious finite amount of input. Detector upgrades have provided exponential increases in detector data while the computing resources remain mostly flat. To date STAR has collected about 50 Petabytes of data. So there is a desire and need to supplement local computing resources with grid computing resources especially for detector data reconstruction.

## 2. STAR's current and former production sites

STAR has received support over the years from a shifting array of cloud and grid production sites. Currently STAR is using the JINR (Joint Institute for Nuclear Research) tier-2 site with an agreement to opportunistically utilize 500 slots and later increase to 1,000 slots if good utilization is demonstrated, along with STAR's own Online cluster which is a mix of Xeon Phi systems and conventional systems under STAR's grid production resource pool. As all sites are heterogeneous the reconstruction software is pre-compiled to insure that there will be a compatible payload for target sites.

Grid sites utilized by STAR but not using the grid production system software include NERSC's (National Energy Research Scientific Computing Center) PDSF cluster which is used for complex simulation along with user analysis and NERSC's Cori which is an HPC cluster used for detector data reconstruction. The envisioned payload for such a cluster is one which requires a massive number of processes with low latency inter-communication between all of the threads. This has traditionally not been a requirement of STAR data reconstruction, which can run on inexpensive commodity hardware which can even be geographically separated at different sites.

Another typical attribute of current HPC clusters is a high ratio of CPU cores to memory. In order to efficiently utilize such HPC clusters, event level parallelization is required because the "pleasantly parallel" computing model requires too much memory per process (or core) and the implied input size is greatly reduced. With event level parallelization the base memory footprint of the payload can be shared amongst many event level reconstruction threads. Event level parallelization requires one input file to be split into event ranges, each of which is assigned to one of the reconstruction threads. Each thread produces its own output file which must be merged back into one contiguous file. This is done on the host site and requires a common buffer space. With this workflow, host sites without a common buffer space cannot utilize event level workload splitting, instead whole files are processed in a "pleasantly parallel" model. However care must be taken in this case to insure all events can be processed within the host site's queue time limits otherwise, the process will be evicted before it has time to finish its reconstruction and copy back its output. Another benefit of a local host site providing common buffer space is that the output payload can be copied to the buffer and the processes can terminate allowing the output to be copied back outside of the job's runtime. We will discuss data transfer modes in more detail in the upcoming section.

## 3. Data transfer modes

When running detector reconstruction on a remote site it is necessary to transfer the raw detector files to the site which will be used as the reconstruction job's input and to transfer back the reconstructed output files. Simulation tasks are easier because there are no input event files to transfer to the remote site (only trivially small configuration files need to be sent), but simulation is a relatively small percentage of STAR's computing workload. Transfer modes for workflows with both input and output can be divided into two groups, those using a shared buffer space on the host site and those without such a space. It is relatively common to have sites share computing slots but provide no local disk space as storage is a more valuable resource requiring a file retention time far in excess of an individual job's own runtime to be viable.

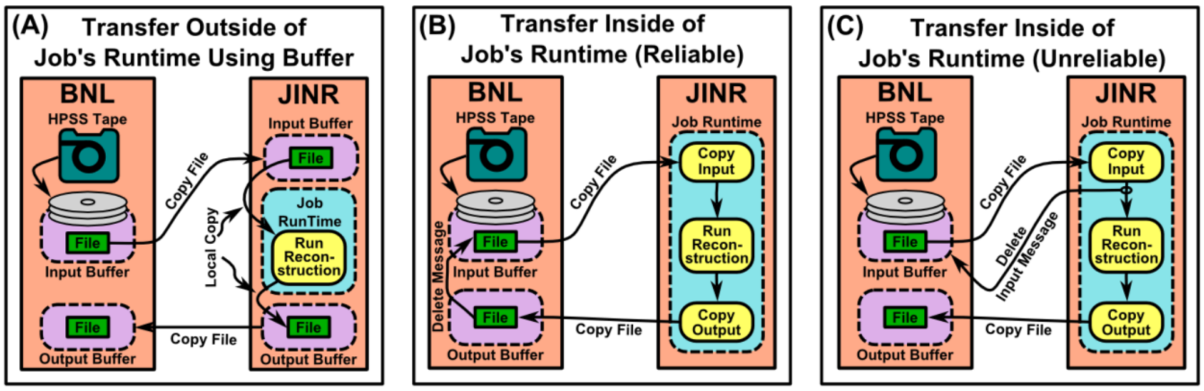

Figure 1. Transfer modes from submitter (BNL) to host site (JINR)

In all modes the input file is restored from a tape located in the drive silo (HPSS) to a local buffer at BNL. Figure 1 shows three transfer modes labeled (A), (B), and (C). Mode (A) depicts a transfer utilizing a shared buffer space on the remote site (JINR). Then the input (raw event file) is copied to a local buffer on the remote site. As it is a shared space, once the job starts up it can do a local copy into the job's local temporary directory where the job has low latency access to the input. As the job runs and processes the input it produces the output file which it also writes into the job's local temporary directory. Just before the job finishes it does a local copy to the local output buffer, which may be the same as the input buffer. The benefit of mode (A) is that site-to-site transfers occur outside of job runtime. Interruptions or slow transfer of the input and output files will not consume job runtime as wasted CPU cycles and will not accumulate towards queue runtime limits, potentially losing the output files when jobs are killed by the batch system. Network connectivity interruptions between the two sites can be very large without affecting recovery of the outputs, given a sufficiently large output buffer. The negative aspect of this mode is that more logic is needed to manage the remote site buffers, keep them clean and insure they do not overfill.

Modes (B) and (C) are variations on transfers without remote shared buffer space. As a consequence all transfers are transacted inside of job runtime. The STAR grid production system supports both modes (B) and (C). We have extracted statistics for a period of six months from the production system's database to determine the overhead of performing the transfers within job runtime between BNL and JINR. The average reconstruction time was 1,784.3 minutes and the average input file size was 2.05 GB with an average input file copy time of 12.9 minutes which constitutes 0.7% of the total runtime. The average output file size is 1.31 GB and the average output file copy time is 5.36 minutes constituting 0.3% of the job's total runtime. The average job spends 99.0% of its time in actual reconstruction. From these statistics it can be concluded that the transfer overhead is negligible. An identical conclusion was reached looking at a large production STAR ran at KISTI (Korea Institute of Science and Technology) in 2015 [1].

In mode (B) the input file is restored from HPSS to a local buffer, and once the job starts it pulls in its input file directly to the job's local temp space. After reconstruction is completed the job copies (pushes) back its output file and log files from the worker node's temp space directly back to the submitting site's output buffer. Once the output files are verified to be intact and correct, the production system removes the input file from the submitting site's input buffer. If the output files are not intact or are missing, the input file in the local buffer remains, and a new job is submitted to attempt to process the input again. One disadvantage of this method is that the local buffer must be large enough to hold a copy of the input file for every running job. With input files of several GB each multiplied by the number of available slots the buffer may need to be very large indeed. If the input buffer is insufficient it will limit the number of job slots that can be filled.

Mode (C) overcomes the above problem of requiring large local input file buffers. Once the input file is restored and the job pulls it in, the job sends back a message indicating that the framework may remove the input files from the local site. As a side note this signal is always sent, but as it plays no role in the other modes it has not been drawn in the other illustrations. Once the input file is removed the space can be reused for the next job. In this way a small local input file buffer can support a large number of jobs. However if the job fails after sending back the "input copy completed" message then the input file must be restaged again from tape. For this reason this method has been christened "unreliable" just as the IP network protocol is unreliable. However, the method is used within the context of the production system, and the system always verifies completion, so even if the job fails multiple times due to transient problems (up to the configured retry limit), the job is likely to be ultimately successful, albeit with more overhead in the case of a re-submission. As the latency of file restoration from HPSS is very, large especially when the files requested are noncontiguous, this method is typically only used when the first pass efficiency is very high and the number of slots available exceeds the input buffer limit. First pass efficiency is defined as the percentage of jobs that successfully complete without any resubmission attempts.

In all three models above (A,B,C), gsiftp is used as the transfer protocol as implemented by the globus-url-copy [2] program for both pulls and pushes of files. Plans are underway to deprecate the Globus software stack including the globus-url-copy program as the developers no longer wish to support it. Globus-url-copy performs the copy, user mapping, and authentication. The OSG [3] (Open Science Grid) has made a commitment to support the Globus software stack for at least three years. However this component will eventually have to be replaced with a substitute.

## 4. JINR lessons learned

As with any site added to the STAR grid production system, extensive testing and tuning is first done to insure the best possible efficiency, such as job first pass efficiency and bandwidth testing of the network connection in both directions. The initial configuration consisted of submission to a CREAM CE [4] from an HTCondor [5] client (HTCondor is distributed with a built-in CREAM submission plug-in). For authentication a grid proxy with VOMS [6] extension was used. It was initially believed this configuration could provide a robust method of submission between the OSG and WLCG environments. And it did work to first order allowing the submission of jobs. However the first pass efficiency was poor. Most of the inefficiency was attributed to the VOMS extension. For non-technology (administrative) related reasons the lifetime of the proxy was limited to three days, and renewal of the proxy on the submission side did not always propagate. Automated password-less renewal of the proxy returned successfully however the internal VOMS proxy certificate chain was mangled and was unable to actually authenticate hence renewal had to be done by hand. With these and other problems it was decided to switch to a native HTCondorCE on the JINR (host site) side from the OSG distribution and use a long-lived grid proxy to authenticate. This achieved our goal of over 90% first pass efficiency. Some bugs still exist - most notable HTCondor will lose track of some jobs, reporting them as held in the local HTCondor queue when they are in fact still running. This was realized by the fact that the held jobs later returned output. This inconsistency in state tracking is observed in the presence of network transient outages between the CE and HTCondor client.

A resource constraint we had not encountered before was a tight limit on queued jobs (jobs waiting for a free slot to run). If jobs exceed the limit they go into the held state. To ensure this limit is not exceeded we simply adjusted a parameter in the HTCondor client to limit the number of jobs pushed over to the JINR host site at any one time. This way the local queue (submitting side) can hold as many jobs as required up to the resource limits of the queue and only push over the preconfigured quantity into the host sites local queue. Once the remote host site's batch system digests some of these jobs and the number of jobs in the remote queue drop, HTCondor will push over more jobs up to its limit again and repeat this feeding cycle as long as jobs exist.

## 5. STAR grid production system components and data flow

The STAR production system uses mostly existing software components maintained by STAR itself or the grid community, mainly OSG. Hence development time is minimized and the quality of these components is high as most of the debugging was carried out in advance and maintenance over time is assured. The STAR local components have an additional advantage in that the collaboration decides the end-of-life date of all components instead of external groups. Some of the STAR components include the STAR Unified Meta Scheduler (SUMS) [7], the STAR Data Carousel [8,9] and STAR File Catalog.

SUMS, first deployed by STAR in 2002, provides a uniform interface for users submitting jobs both locally and via the grid across all STAR sites. It also allows sites to transparently switch between batch systems without disturbance to the user's job submission and expert shaping of resource requirements. SUMS takes a JDL (Job Description Language) request which describes a dataset, operation, and optional resource requirements such as memory, time (events or files per hour) and produces jobs based on a tailored configuration which it submits to a batch system or grid or cloud interface. This fits very well into the production system because different JDLs can be produced for the processing of different datasets. It is the component that transforms the actual request into jobs.

The STAR Data Carousel is a tool which restores datasets from HPSS to disk for reconstruction. It has a sophisticated feature, holding a queue of file restoration requests from multiple users re-orders the queue by the tape on which the requested files reside. This will minimize the mount-dismount cycles, as the time required for the robotic arms to fetch and mount a tape is relatively large.

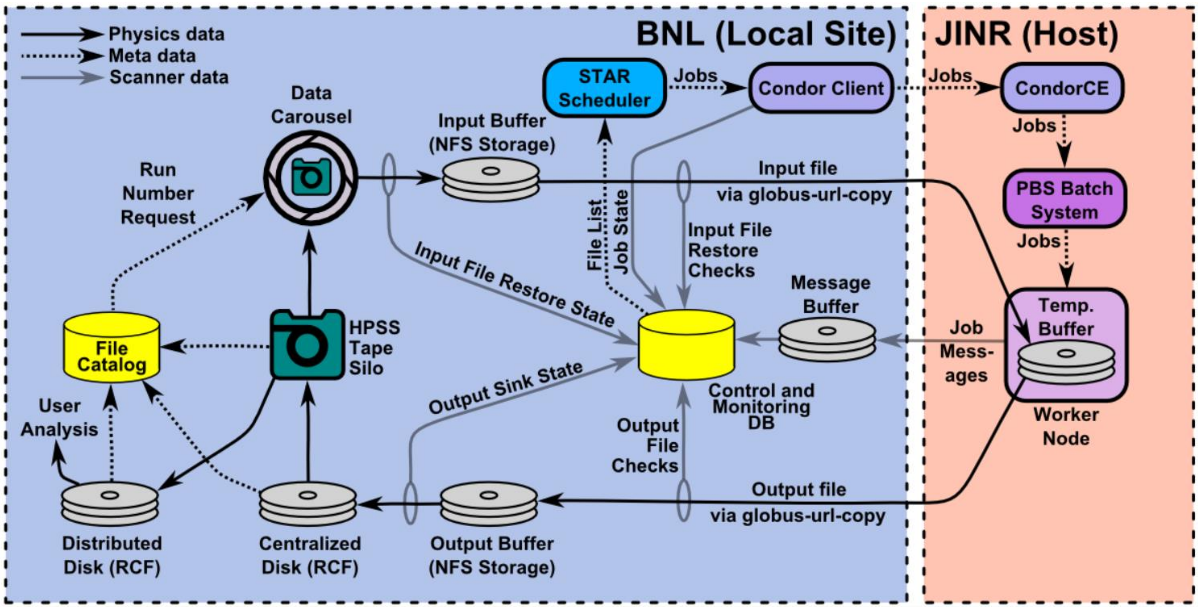

Figure 2. STAR Grid Production System Dataflow

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 input file buffer on NFS disk at any one time. Typically 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

HTCondor grid client which passes the jobs to the remote site's batch system where they eventually execute on the worker node and pull in their input files. While a job is running it passes back messages. Once a job is done it copies its output file back to the local site. 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.

## 6. STAR's grid production system's finite state system

Using JINR's resources, STAR has a high (93.2%) first-pass efficiency. However the requirement is that the nth-pass efficiency is 100% of all files that can be reconstructed. A finite state system model is used to track the progress of each file to be reconstructed. This includes loops to retry actions such as copying files in or out and, if needed resubmitting the job which would count as a first-pass efficiency failure. Retries are limited to a finite number of attempts. Figure 3 illustrates the finite state automaton's states and transition conditions which are kept track of in the production system's control and monitoring database illustrated in Figure 2.

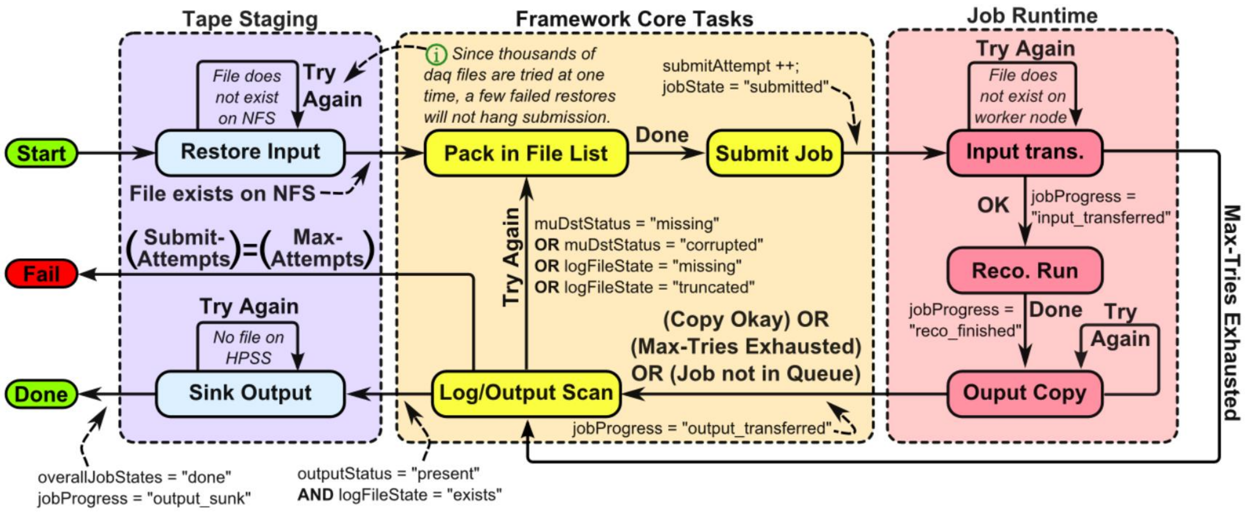
Tape Staging
Framework Core Tasks
Job Runtime
Start
Restore Input
File does not exist on NFS
Try Again
Since thousands of daq files are tried at one time, a few failed restores will not hang submission.
File exists on NFS
(Submit-Attempts)=(Max-Attempts)
Fail
Try Again
No file on HPSS
Sink Output
Done
Pack in File List
Done
Submit Job
submitAttempt ++; jobState = "submitted"
Try Again
muDstStatus = "missing" OR muDstStatus = "corrupted" OR logFileState = "missing" OR logFileState = "truncated"
(Copy Okay) OR (Max-Tries Exhausted) OR (Job not in Queue)
Log/Output Scan
jobProgress = "output_transferred"
Try Again
File does not exist on worker node
Input trans.
OK
jobProgress = "input_transferred"
Reco. Run
jobProgress = "reco_finished"
Done
Try Again
Ouput Copy
Max-Tries Exhausted
overallJobStates = "done" jobProgress = "output_sunk"
outputStatus = "present" AND logFileState = "exists"

Figure 3. Finite state machine diagram for grid production system

## 7. Resubmission of failed jobs

Key features of any software claiming to call itself a production system versus by-hand-submission is the ability to feed, monitor, and resubmit failed jobs. The software must accurately determine if a job has finished and was successful. Our system does this by polling the batch system every hour to check if the job is still in the queue. Once the job is out of the queue, the system checks the expected location of output files for their existence, size, and the log file for known error patterns. In an older version the software checked checksums taken on the worker node verses the checksum of the same file on the output buffer, however this was later replaced by a simpler and faster size check, which does not require reading back all of the output files. It is critical to check returned output quickly so it can be flagged for sinking to tape as it could overflow the output buffer if too many returned output files linger in the buffer for too long.

## 8. Job feeding

So far we have discussed first-pass efficiency exclusively, however there exist other efficiency metrics such as utilization efficiency, which is the percent of available slots versus the number of jobs actually running. When there exists a non-zero number of idle jobs on all sites, utilization efficiency will be one hundred percent of the resource, even in an opportunistic environment where the number of available slots changes with time. The key to achieving full utilization is to ensure a sufficient number of restored files. We have already discussed the method used to achieve restoration even with a small output buffer in section 3 of our paper. There exists a propagation delay, checking for input files, wrapping them into a job, submitting the jobs to the grid, and waiting for the batch system to do its matchmaking. But if we ensure there are always idle jobs for the batch system to grab the propagation time is irrelevant as propagation happens long before the slot is actually available. There are some inherent limits which lead to job feeding, for example, the whole dataset size cannot fit into the input buffer at one time, but even if the buffer could be sized to restore the entire dataset at once it is not ideal to submit them simultaneously as they would divide between sites with different capacities where one site finished well in advance of another. Attempting to calculate the capacity of a site providing opportunistic resources in order to pre-allocate workload (jobs) is impossible as it would require accurate knowledge of other users resource requests before they actually make them. Instead, our production system polls the HTCondor client queue once per hour in order to keep a "pad" of idle jobs on each site as represented in Figure 4. When the idle job pad starts to deteriorate, and the jobs go into the running state, the feeder replenishes it by submitting

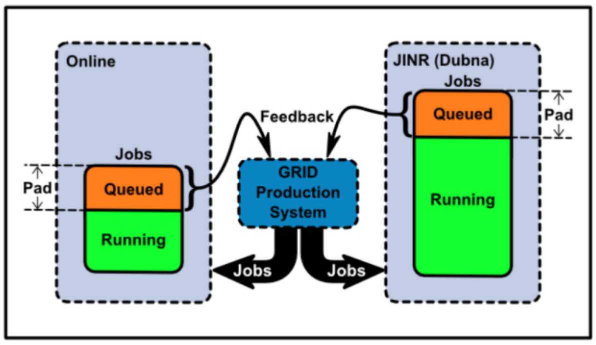

Figure 4. Jobs Feeding

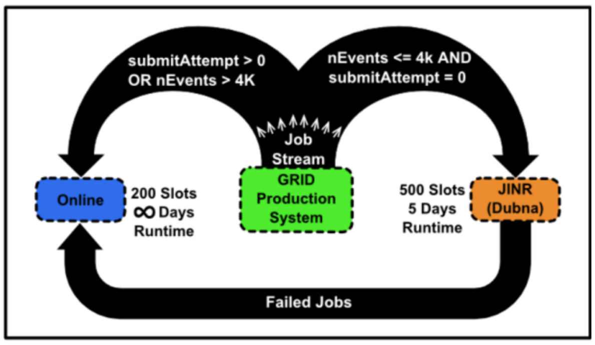

Figure 5. Multi-site feeding algorithm example

more jobs up to the preconfigured limit. This forms a negative feedback loop holding the pad size essentially constant and insuring each sites workload is scaled correctly so that the dataset finishes on all site's at approximately the same time.

## 9. Site selection

To further boost efficiency the production system has some logic to do matchmaking at the site level. For example for a particular dataset which had a high number of tracks per event it was noted that some jobs were approaching the queue time limit of five days at JINR. A quick plot was produced of events versus runtime from jobs that had already been processed, and it was determined that four thousand events represent a point where the job is unlikely to exceed the queue limit. A rule was added for the dataset to only submit jobs containing four thousand events or less to JINR and pass the rest to the Online farm which has no queue time limit. And for good measure a second rule was added that if a job fails at JINR it should not be resubmitted there but instead to the Online farm. This will improve the second pass efficiency as jobs that exceed the time limit of the queue will not be retried on the same site where they are likely to fail again. This is a powerful feature but care needs to be taken to ensure no inadvertent unbalance is caused between sites.

## 10. Efficiency

In a recent ten-month period, STAR successfully reconstructed 25,627 files containing 151 million events using JINR resources and our own Online farm. A breakdown of production statistics between the two sites can be seen in Figure 6. It should be noted that the Online cluster has considerably fewer slots, takes longer jobs, and is only available after STAR finishes data-taking

| Site | Files | Events | Runtime (Hours) | Dataset Size GB |
|---|---|---|---|---|
| **Online:** | 4,847 | 27M | 332,633 | 14,293 |
| **JINR:** | 20,780 | 138M | 534,324 | 23,878 |
| **Total:** | 25,627 | 165M | 866,957 | 38,171 |

Figure 6. Multi-site feeding algorithm example

each year and so it has produced fewer files. Our first pass efficiency was 93.2% of all jobs returning valid input and log files in the first attempt with the remainder being resubmitted. Causes of job failures include jobs running past the queue runtime limits, AFS errors, gsiftp errors, and an Online farm power outage.

## 11. Acknowledgement

STAR thanks JINR for making their site available for reconstruction jobs and assisting to solve technical problems along the way.

## 12. Conclusion

Scavenging additional resources allows for the reconstruction of a few additional small datasets per year which allows additional analysis work in turn and is therefore of value to STAR. Our framework has an excellent first pass efficiency of 93.2%. The efficiency is comparable to, but slightly lower than, local efficiency, which is 98%, due to the additional complexity of the added software interfaces. STAR has put a lot of thought and iterative refinement into the design of this production system. Extensive preproduction testing and a well-structured finite state system applied to each job, with retries in the event of interruption, applied to each job contributes to this high efficiency. What makes this even more remarkable is that it is running on heterogeneous nodes.

## References

[1] Hajdu L. et al. Automated Finite State Workflow for Distributed Data Production // Journal of Physics: Conference Series, 762 012006 doi:10.1088/1742-6596/762/1/012006
[2] Allcock W. 2003 GridFTP: Protocol Extensions to FTP for the grid (Global Grid Forum GFD) page 20
[3] Pordes R. et al. The Open Science Grid // Journal of Physics: Conference Series, 2007, Vol. 78, Boston, Massachusetts, USA
[4] Andreetto P. et al. Status and Developments of the CREAM Computing Element Service // Journal of Physics: Conference Series, 2011, doi:10.1088/1742-6596/331/6/062024

[5] Bockelman B. et al. Commissioning the HTCondor-CE for the Open Science Grid // Journal of Physics: Conference Series, 2015, doi:10.1088/1742-6596/664/6/062003
[6] Ceccanti A. et al. VOMS/VOMRS utilization patterns and convergence plan // Journal of Physics: Conference Series, 2010, doi:10.1088/1742-6596/219/6/062006
[7] Hajdu L. et al. Meta-configuration for dynamic resource brokering: the SUMS approach (2006 International Conference on Computing in High Energy and Nuclear Physics, Tata Institute of Fundamental Research, Mumbai, India)
[8] Lauret J. et al. ERADAT and Data Carousel systems at BNL: A tool and UI for efficient access to data on tape with fair share policies capabilities (2010 ACAT 2010 proceedings, PoS ACAT 023)
[9] Lauret J. et al. Tape storage optimization at BNL (CHEP 2010 proceedings) // Journal of Physics: Conference Series, 331 042045 doi:10.1088/1742-6596/331/4/042045