Provenance-aware optimization of workload for distributed data production

Dzmitry Makatun\textsuperscript{1,3}, Jérôme Lauret\textsuperscript{2}, Hana Rudová\textsuperscript{4} and Michal Šumbera\textsuperscript{3}

\textsuperscript{1} Faculty of Nuclear Physics and Physical Engineering, Czech Technical University, Czech Republic
\textsuperscript{2} RHIC/STAR Experiment, Physics Department, Brookhaven National Laboratory, USA
\textsuperscript{3} Nuclear Physics Institute, Czech Academy of Sciences, Czech Republic
\textsuperscript{4} Faculty of Informatics, Masaryk University, Czech Republic

E-mail: makatun@rcf.rhic.bnl.gov

Abstract. Distributed data processing in High Energy and Nuclear Physics (HENP) is a prominent example of big data analysis. Having petabytes of data being processed at tens of computational sites with thousands of CPUs, standard job scheduling approaches either do not address well the problem complexity or are dedicated to one specific aspect of the problem only (CPU, network or storage). Previously we have developed a new job scheduling approach dedicated to distributed data production – an essential part of data processing in HENP (pre-processing in big data terminology). In this contribution, we discuss the load balancing with multiple data sources and data replication, present recent improvements made to our planner and provide results of simulations which demonstrate the advantage against standard scheduling policies for the new use case. Multi-source or provenance is common in computing models of many applications whereas the data may be copied to several destinations. The initial input data set would hence be already partially replicated to multiple locations and the task of the scheduler is to maximize overall computational throughput considering possible data movements and CPU allocation. The studies have shown that our approach can provide a significant gain in overall computational performance in a wide scope of simulations considering realistic size of computational Grid and various input data distribution.

1. Introduction

Data production is an important part of computations in High Energy and Nuclear Physics (HENP) where a set of raw (input) files has to be processed before it can be used for further analysis. During its lifetime, raw data can undergo several passes of production. The additional passes are typically performed when the code or calibration data are improved. Therefore, the time span between repeated processing of the same data is large (compared to time of the processing itself). A single non-stop pass over a fixed dataset is called a data production campaign. Similar computational patterns can be found in other big data applications, where this workflow is also referred as event reconstruction or preprocessing. In modern HENP experiments data production involves Petabytes of data being processed on hundreds of thousands CPUs at facilities distributed all over the world. Commonly used load balancing (and job scheduling) approaches either do not scale to the problem size or consider only a single aspect of optimization (CPU usage, data access or network usage). For this reason, many experiments use their custom setups for a given infrastructure \[2, 10, 6, 7, 4\]. However,
such solution is not universal, and requires an increased effort to be adjusted to changes in the infrastructure or parameters of the workload.

In order to fill the gap between general load balancing methods, job scheduling, distributed data management systems and current specifics of computations in HENP we have previously developed a new job scheduling approach [14, 16, 13]. Our main goal was to maximize the overall computational throughput of a distributed system for a data intensive application. The developed approach combines load balancing for CPUs, storage and network across a distributed system. The key idea is to distribute the data between computing facilities during the computation in such a way, that I/O waiting time is minimized and network/storage capacity are not exceeded at any given time. Our scheduling is based on the network flow maximization problem [1], which has a polynomial complexity. Such formulation significantly improves the scalability compared to general scheduling problem with exponential complexity. It is also self-adjusting to changing environment including performance fluctuations, background load and resource addition/withdrawal/reconfiguration. These properties are in high demand for modern data-intensive applications with growing usage of opportunistic resources, cloud computing. Moreover, the computational power aggregated at smaller computational facilities (Tier-2 and Tier-3) is increasing during the last years [3]. However, due to lower storage space and limited connectivity to the main infrastructure the utilization of those sites is often restricted to the tasks with lower I/O demands (e.g. simulations). Our scheduling approach provides an efficient way to utilize a large number of loosely connected small sites for data production. For this reason, the simulations in this work are extended to model data production in large-scale heterogeneous computational networks.

In data replication, the raw (input) data recorded from the detector is stored at the central site (Tier-0) and a second copy is distributed to one or more secondary sites (Tier-1). The number of those sites, the degree of file replication and data shares at sites may vary. Data replication allows to improve data safety against loss, to increase access speed and to distribute I/O load. While the low level parity mechanisms are beyond the scope of our research, the duplication of files at multiple sites is an important factor to consider in our scheduling approach as it introduces additional complexities in our modeling. First, the scheduler should consider multiple input data locations and select an optimal one to access the data from each particular computing facility. Second, excessive data transfers and duplication of computational jobs should be avoided. The first aspect was addressed in our previous work [15] where we have introduced reasoning on multiple data sources into our scheduling approach. In this work we focus on the second aspect and discuss the recent implementation of handling of the replicated files for data production.

2. Model summary
In order to achieve a reasonable complexity of the scheduling problem we used specific properties of data production. First, in a single campaign each portion of data has to be processed exactly once. Second, all the computational jobs are independent and interchangeable which means that they can be executed in arbitrary order. Moreover, since a single data production campaign consists of a large set of similar files undergoing the same type of processing, the parameters of upcoming jobs can be predicted using the statistics of previously finished ones. Despite particular jobs can vary in their parameters, for a large enough dataset and long enough time interval we can rely on the average values. Following these assumptions, the main idea of our approach is to plan resource load in advance for a limited time step (planning time interval) and then distribute data and computational jobs accordingly, rather than producing a complete schedule. Planning for limited time intervals (e.g. 12 hours) provides adaptability necessary for dynamic systems. Planning repeatedly for shorter time steps allows to deal with such uncertainties by correcting predictions and adjusting to the current state and performance of the resources.

Fig. 1 illustrates the input for our planner: at the beginning of each planning time interval
3. Handling of replicated files

The data flow of a HENP experiment begins at a central site where the detector is located. Typically, the site also provides a persistent storage for all the data related to the experiment and is called a Tier-0. In addition to the main storage, overlapping subsets of data are copied across multiple Tier-1 sites [3]. The degree of replication for particular files can vary and change in time depending on the computation history and collaboration plans. Each file can be assigned a logical filename (lfn) uniquely defining it. A single lfn can point to multiple physical files identified by physical filenames (pfm).

We assume that the data is replicated before the data production starts. The planner never deletes those initial (persistent) file replicas. It considers initial data locations and creates temporal file replicas when it allows to speedup data access. Therefore the planner distinguishes two types of a physical file instance: persistent and temporal. Persistent instance is the one that was created before the data production started and has to be kept. Temporal instance is created when the file is transferred to a new site as a result of the plan execution. Only one temporal instance per lfn can exist at a time. This instance is deleted after the file is processed.

Forwarding a temporal replica from one site to another implies creation of a new temporal replica at the destination and deletion of an old one from the source. Temporal replica can be understood as a “traveling” instance which hops between sites according to the plan until it gets processed.
There is a central service (called file catalog) which keeps track of all the \textit{lfn}'s of the input files in data production. The handlers at individual sites can communicate through this service in order to ensure that all \textit{lfn}s are processed exactly once and data are never transferred to a site where another its copy is already present. File catalog provides information on the number of physical instances for each \textit{lfn} and their location. It also stores the status information related to each \textit{lfn}. Two statuses are used in the current implementation: "queued" and "used". \textit{Queued} status means that the \textit{lfn} waits to be processed, no temporal copy was created, neither was the job started. \textit{Used} status means the file is being processed at one of the sites which has its persistent copies, or a temporal copy was created to be processed at another site.

Initially all the file statuses are set to "queued". When a file is selected to be processed at one of its initial locations the corresponding \textit{lfn} status is changed to "used". When a file is transferred from its initial location the corresponding \textit{lfn} status is changed to "used" and a temporal replica is created at the destination. From this moment, only this temporal replica is allowed to be processed. Through the change of the status the other sites with a persistent replica of the file are prevented from processing or sending it out again. Since a max-flow problem solution contains no flow cycles the temporal instance travels a limited path in the Grid until it is processed, for this reason no excessive transfers take place.

At the beginning of data production each handler at a source site (a) registers persistent replicas at its site to the file catalog and sets their status to "queued"; (b) organizes input files at the site into a queue where files with less replicas are prioritized. The former allows to process files with less replicas first and leave more options for later planning cycles.

During the data production each handler periodically scans its local queue and removes files with status "used" from it. Incoming input files forwarded from other sites (the temporal instances) are placed at the beginning of the queue and kept. Whenever there is a capacity to process a file, or to send it to another site, the handler takes the next file from the queue.

Each new plan depends on the current state of the system but not on previously issued plans. Therefore, failures during execution of a plan do not affect future planning cycles. Recovery can be done using standard approaches. Loss of replicas at a given site would be detected by handlers and removed from the possible replicated candidates. File corruptions can be detected using checksums stored in the file catalog. The next cycle of planning would consider a new file distribution landscape. Timed-out and failed actions can be re-queued within one planning cycle. A handler can perform self-recovery at the start of a planning cycle (i.e. when a new plan is issued). In such case a handler verifies the content of the local disc and running jobs and then it can proceed with the current plan. In case of a planner failure, it restarts and requests current status from all the sites and then continues to issue plans as normal. More statuses of the file replicas can be introduced in order to ensure safe transactions. However, failures of the file catalog service are beyond the scope of this paper.

4. Simulations
During the development of our scheduler we conducted numerous simulations to validate various potential usecases and compared against commonly used scheduling methods. In our previous work we have demonstrated the gain we may obtain in a simple network setup with several sites [13]. In that work we have presented the dependence of makespan improvement on network bandwidth and amount of CPUs at sites. Later, in work [16], we showed that the usage of our planner is advantageous in shared networks with significant background traffic. The scalability of our method to the infrastructure of one of the largest HENP experiment was also demonstrated in that work. In addition, in reference [15] we presented results of simulations with multiple input sources and more general usecases: when the computational infrastructure consists of many tens of randomly connected (following the power law) smaller sites (hundreds of CPUs). Those simulations have shown, that in such case the planner can provide a dramatic decrease in
makespan compared to a widely used PULL method. In this work our simulations are focused on a recently added and extremely important features – data replication and an heterogeneous infrastructure composed of Tier-0,1,2 sites.

The simulations are implemented using GridSim [5] and based on the data collected in real production environment. The parameters of computational sites and network are taken from online monitoring tools such as [18, 17, 12]. The parameters of computational jobs and files were extracted from log files of data production of the STAR experiment [11]. The simulated infrastructure (see Fig. 2) is described in the following text.

Ten Tier-1 computational sites (B0 – B9, colored magenta) and the primary network infrastructure of one of the largest HENP experiments are used as a core. The number of CPUs, storage size and network bandwidth are downscaled by the factor of two. This allows to run the simulations within reasonable time. The total number of simulated CPUs at those ten sites is 19,536. We assume that each Tier-1 site initially stores an equal portion of input data (replicated from Tier-0).

Another Tier-1 site (B10, colored blue) with a significant portion of CPUs (6,000), slow connection to Tier-0 and no locally stored input data is added to the system in order to challenge compared scheduling approaches. The site represents a large computing facility which is outside of the primary network infrastructure. Such case can illustrate the usage of opportunistic resources or resources of an external volunteering organization possibly from a distant part of the globe.

A single Tier-0 site (colored red) is considered as a storage only, i.e. it is another source of input files and the only destination for output files. We assume that the Tier-0 site persistently stores the entire input dataset, and all the output files have to be copied there. In many of the experiments the Tier-0 has a significant computational power and is often used for data production. However, processing data at the Tier-0 site does not require remote data access and, therefore, is removed out of scope of our simulations. This simplification decreases the number of simulated CPUs and, therefore, reduces the complexity of the simulations. Nevertheless, our approach can include data processing at Tier-0 site with just a trivial change in configuration.

The number of CPUs and storage size at Tier-2 sites were assigned randomly with respect to values observed in real infrastructure (~100 CPUs, ~3 TB). The sites were connected to the rest of the infrastructure using an algorithm for a scale-free network generation in order to represent properties of real networks [9]. The bandwidth of the corresponding links was set within 0.1 – 1 Gbps. The simulated infrastructure contains 39 Tier-2 sites (C0 – C38, colored blue) with 11,019 CPUs in total.

The resulting computational network used in our simulations consists of 51 sites with 36,555 CPUs in total and 82 network links. Five different datasets were created using random selection from the original set of log records of data production of the STAR experiment. Each dataset consists of 600,000 input files with total size of 2.7 PB on average. The corresponding average total size of output files is 1.8 PB. The total time for sequential processing of such dataset is 314 CPU years on average.

In recent simulations our job scheduling approach (further referred as PLANNER) was compared against the commonly used PULL method. Under this method whenever a site has a free CPU slot to run a computational job it transfers an unprocessed input file from the closest (by ping) available source, performs the computations and transfers the output file to the destination site. When all the input files at the closest source are processed, the site switches to the next closest one until all the data are processed.

5. Results of the simulations
The simulations with the described infrastructure have shown that comparing to PULL the PLANNER allows to process each given dataset with a makespan shorter by 7% on average.
with deviation 0.8%. A typical dependence of the total CPU usage over time during the data production for both PULL and PLANNER is compared at Fig. 3. As one can see on the plot, the PULL model fails to utilize all the available CPUs efficiently in the simulated case: a significant fraction of CPUs is waiting for input/output transfers and, for this reason, the actual computations are delayed. In such situation, when the network performance becomes a bottleneck, uncoordinated concurrent data transfer by multiple processes becomes inefficient as it leads to congestion and increased latency. In contrast, the PLANNER transfers balanced portions of input data in advance before their processing, output transfers do not delay CPUs and the network usage is planned in order to avoid congestion. This allows to reach the maximum CPU usage which otherwise would be hindered by the limited network performance.

For better illustration we provide CPU usage per site for PULL and PLANNER at Fig. 5 and 6 respectively. Each plot shows the performance of seven sites with the lowest CPU utilization under the PULL approach. Sites, such as C3, C11 and C5 (see Fig. 2), share network links for data access. As a result, simultaneous I/O access saturates the network capacity and the CPU usage at those sites is degraded. C21 and C24 switch to distant input sources when the closest ones are depleted which increases the latency. Sites, such as B10 and C14 do not have enough connection bandwidth to keep all their CPUs utilized. Despite alternative input sources and transfer routes are available for those sites, the PULL model does not distribute (balance) network load across them. Using the PLANNER, the transfers are distributed in time, between possible sources and over alternative routes in order to match the network capacity. This allows to transfer the data to/from the bottleneck sites efficiently.

In order to study the advantage of data replication we have performed an additional simulation where all the input data are initially stored at the Tier-0 site only. The results are presented at Fig. 4. Comparing it to Fig. 3 one can conclude that for the studied usecase the PLANNER provides a close computational efficiency with and without data replication. This result can be understood considering that our PLANNER creates temporal copies asynchronously to the data processing that is, moves data closer to the resources as the CPUs are busy for later consumption. The time for moving files being less than the processing time (and the storage capacity allowing the data copies), the difference between replication and no replication is not seen, illustrating the importance (for this use case) of networking bandwidth as a resource. In contrast, the data replication is required to increase the performance of the PULL for two reasons. First, it allows to increase peak performance due to a better distribution of the network load. Second, without data replication the PULL had a noticeable makespan increase. This delay is due to a small fraction of slow jobs, which are executed at large network distances from the data placement and, therefore, suffer an I/O overhead. Those jobs are observed as a "tail" at the end of the plot at Fig. 4. Such effect is reduced with data replication where jobs have a choice of input sources and use the one with the fastest connection. Nevertheless, the PLANNER has shown advantage both with/without data replication. The planner considers how much data can be processed during the planning time interval and distributes data and jobs accordingly: if the data can not be processed at a remote site faster than at its current location it will not be sent there. Therefore, the overall makespan is not compromised by a small fraction of delayed jobs. In this simulation the PLANNER provides 19% of makespan improvement compared to the PULL. Similar observations were made in the simulations from our previous work [15].

Another important observation made in the presented simulations is that it takes 30 ms on average with deviation 14 ms (500 planner runs analyzed) for the PLANNER to generate a plan for 12 hours of data production. As the scale of the simulated computational network compares to real systems in HENP, such solving time meets the requirements for planning in real infrastructures.
Figure 3. Results of simulation: total CPU utilization during distributed data production. The planner has achieved a better CPU usage and shorter makespan due to more efficient utilization of network bandwidth.

Figure 4. Results of simulation without data replication. Under the PULL approach the overall makespan is prolonged due to a small fraction of jobs with increased I/O latency. The PLANNER allows to avoid such problem.

Figure 5. Results of simulation with PULL approach: 7 sites with the lowest CPU usage. The performance is degraded due to network congestion as a result of uncoordinated concurrent usage of shared network links.

Figure 6. Results of simulation with PLANNER approach: 7 sites with the lowest CPU usage. All the CPUs are occupied with computations due to intelligent planning of network and storage load.

6. Conclusion
In previous work [14, 16, 13] we have developed a new job scheduling for data production in HENP which is focused on optimization of resource usage {CPU, disk, network}. With the help of simulations we have shown that our approach adapts to changing environment, background network traffic and can balance network load between self-discovered alternative transfer paths when necessary. It was also demonstrated that it systematically provides better makespan than other common methods such as PULL and PUSH, and can scale for online planning for the infrastructure of the currently largest HENP experiments due to reasonable computational complexity of the underlying algorithm.

In this work we have extended our approach to reason on multiple input sources and data replication across distributed system, which is of great importance for real life data intensive applications. We have performed series of simulations based on log data from real systems. These simulations were featuring a large-scale realistic network of heterogeneous computational resources. Several "bottleneck" (non-optimized) sites were considered in simulations to challenge compared scheduling approaches. Our planner has consistently shown a significant (7% on average) makespan improvement compared to the standard one (PULL). Moreover, in contrast to PULL, in the studied infrastructure our planner can provide the same improved efficiency without relying on multiple data sources (replication). Planning of data production is performed in time steps. The data for the next planning step are distributed across the system while the
CPUs are processing the data from the previous step. This allows to transfer the data from the Tier-0 to the rest of the system without delaying the computation. The planner considers the current state and previous performance of the resources at each planning step. It allows to establish an optimal data distribution with respect to computing resources, storage and network. The overall computational throughput is increased due to more efficient usage of sites with limited connectivity. Our approach meets the demand for efficient load balancing with an increasing usage of opportunistic and cloud resources outside the primary infrastructure. It becomes even more important in the era when ever-growing amount of data emphasizes the need for its efficient storage, transfer and processing. In future we plan to continue improvement of the planner and perform detailed simulations of data production of more HENP experiments. The final goal is to integrate our approach into data production system of the STAR experiment.

Acknowledgments
This work has been supported by the Czech Science Foundation 13-20841S, the MEYS grant LG15001 and the Office of Nuclear Physics within the U.S. Department of Energy. Furthermore, the authors would like to acknowledge the Institute of Physics of the Czech Academy of Sciences for providing access to their computing facility for running simulations.

References
[1] R. K. Ahuja, T. L. Magnati, and J. B. Orlin. Network flows: theory, algorithms, and applications. Prentice Hall, 1993.
[2] S. Bagnasco et al. AliEn: ALICE environment on the grid. In Journal of Physics: Conference Series, volume 119, page 062012, 2008.
[3] I. Bird et al. Update of the Computing Models of the WLCG and the LHC Experiments. Technical Report CERN-LHCC-2014-014. LCG-TDR-002, CERN, Geneva, Apr 2014.
[4] B. Bockelman et al. Commissioning the htcondor-ce for the open science grid. In Journal of Physics: Conference Series, volume 664, page 062003, 2015.
[5] R. Buyya and M. Murshed. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. The Journal of Concurrency and Computation: Practice and Experience (CCPE), 14, 2002.
[6] K. De et al. The future of PanDA in ATLAS distributed computing. In Journal of Physics: Conference Series, volume 664, page 062035, 2015.
[7] E. Deelman et al. GriPhyN and LIGO, building a virtual data grid for gravitational wave scientists. In 11th IEEE International Symposium on High Performance Distributed Computing, HPDC-11, pages 225–234. IEEE, 2002.
[8] M. Drozdowski. Scheduling for Parallel Processing. Springer-Verlag London, 1st edition, 2009.
[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In ACM SIGCOMM computer communication review, volume 29, pages 251–262. ACM, 1999.
[10] M. U. Garcia et al. Integration of cloud resources in the LHCb distributed computing. In Journal of Physics: Conference Series, volume 513, page 513, 2014.
[11] L. Hajdu et al. STAR experience with automated high efficiency Grid based data production framework at KISTI/Korea. In HEPiX Spring 2015 Workshop. Oxford University, UK, 2015.
[12] LHC Optical Private Network. http://lhcpp.web.cern.ch/lhcpp/, 2015.
[13] D. Makatun, J. Lauret, and H. Rudová. Distributed data production planning for High Energy and Nuclear Physics. (Under submission process), 2017.
[14] D. Makatun, J. Lauret, H. Rudová, and M. Šumbera. Model for planning of distributed data production. In Proceedings of the 7th Multidisciplinary International Scheduling Conference (MISTA), pages 699–703, 2015.
[15] D. Makatun, J. Lauret, H. Rudová, and M. Šumbera. Network flows for data distribution and computation. Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling and Network Design, 2016.
[16] D. Makatun, J. Lauret, H. Rudová, and M. Šumbera. Simulations and study of a new scheduling approach for distributed data production. Journal of Physics: Conference Series, 762(1):012023, 2016.
[17] MonAlisa: Grid online monitoring data of the ALICE experiment. http://alimonitor.cern.ch/, 2015.
[18] WLCG REsource, Balance & USage. https://rebus.cern.ch/, 2015.