PROCEEDINGS OF SCIENCE

Building Efficient Data Planner for Peta-scale Science

Michal ZEROLA*
†
Nuclear Physics Institute, ASCR
E-mail: michal.zerola@ujf.cas.cz

Jérôme Lauret
Brookhaven National Laboratory
E-mail: jlauret@bnl.gov

Roman Barták
Faculty of Mathematics and Physics, Charles University
E-mail: bartak@kti.mff.cuni.cz

Michal Šumbera
Nuclear Physics Institute, ASCR
E-mail: sumbera@ujf.cas.cz

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

PROCEEDINGS OF SCIENCE

Unprecedented data challenges both in terms of Peta-scale volume and concurrent distributed computing have seen birth with the rise of statistically driven experiments such as the ones represented by the high-energy and nuclear physics community. Distributed computing strategies, heavily relying on the presence of data at the proper place and time, have further raised demands for coordination of data movement on the road towards achieving high performance. Massive data processing will be hardly "fair" to users or unlikely be using network bandwidth efficiently whenever diverse usage patterns and priorities will be involved unless we address and deal with planning and reasoning of data movement and placement. Although there exist several sophisticated and efficient point-to-point data transfer tools, the lack of global planners and decision makers, answering questions such as "How to bring the required dataset to the user?" or "From which sources to grab the replicated data", is for most part lacking.

We present our work and status of the development of an automated data planning and scheduling system, ensuring fairness and efficiency of data movement by focusing on the minimal time to realize data movement (delegating the data transfer itself to existing transfer tools). Its principal keystones are self-adaptation to the network/service alteration, optimal selection of transfer channels, bottlenecks avoidance and user fair-share preservation. The planning mechanism relies on Constraint Programming and Mixed Integer Programming techniques, allowing to reflect the restrictions from reality by mathematical constraints. In this paper, we will concentrate on clarifying the overall system from a software engineering point of view and present the general architecture and interconnection between centralized and distributed components of the system. While the framework is evolving toward implementing more constraints (such as CPU availability versus storage for a better planning of massive analysis and data production), the current state of our implementation in use for STAR is limited to a multi-user, multi-site and multi-source environment for data transfers and we will present the implications and benefit of our approach as well as a use case in practice based on requests made with multiple choice for sources.

13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research
February 22-27, 2010
Jaipur, India

---

*Speaker.

†The work has been supported by the grants LC07048 and LA09013 of the Ministry of Education of the Czech Republic, the project Czech Science Foundation P202/10/1188 and by the Office of NP within the U.S. DOE Office of Science.

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

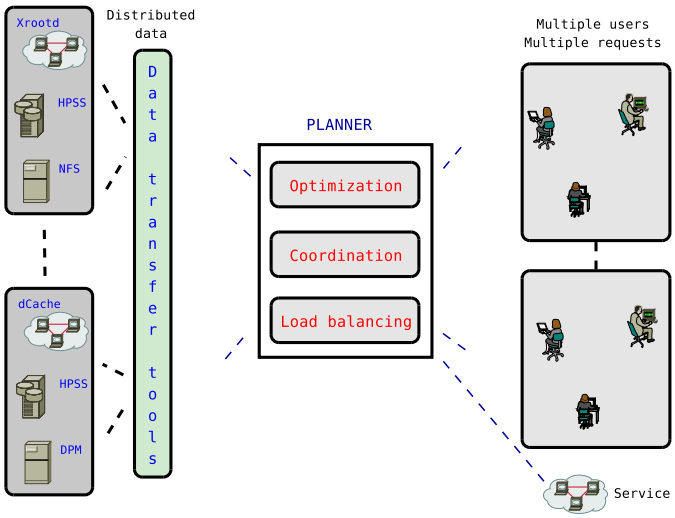

Figure 1: General view of the automated planning system. The goal is to achieve controlled and efficient utilization of the network and data services with a proper use of existing point-to-point transfer tools. At the highest level of abstraction, the planner should appear as a "box" between the user's requests and the resources.

1. Introduction

As it is widely known, distributed computing offers large harvesting potential for computing power and brings other benefits as far as it is properly exploited. On the other hand it introduces several pitfalls including concurrent access, synchronization, communications scalability as well as specific challenges such as answering key questions like "how to parallelize a task?" knowing where my data and CPU power are located. In data intensive experiments, like the one from HENP community and the STAR [1] [1] experiment, the problem is even more significant since the task usually involves processing and/or manipulation of large datasets.

This massive data processing will be hardly "fair" to users and hardly using network bandwidth efficiently unless we address and deal with planning and reasoning related to data movement and placement. In this paper we present and focus on the implementation and software engineering part of our ongoing work, while we refer to our previously published papers explaining in more depth the underlying model and theoretical background.

The purpose of our research and work is to design and develop an automated planning system acting in a multi-user and multi-service environment as shown in Fig. 1. The system acts as a "centralized" decision making component with the emphasis on **optimization**, **coordination** and **load-balancing**. The optimization guarantees the resources are not wasted and could be shared and re-used across users and sources. Coordination ensures multiple resources do not act independently so starvation or clogging do not occur, while load-balancing avoids creating bottle-necks on the resources. The intent is not to create another point-to-point data transfer point-to-point tool, but to use available and practical ones in the efficient manner.

[1] Solenoidal Tracker at Relativistic Heavy Ion Collider is an experiment located at the Brookhaven National Laboratory (USA). See http://www.star.bnl.gov for more information.

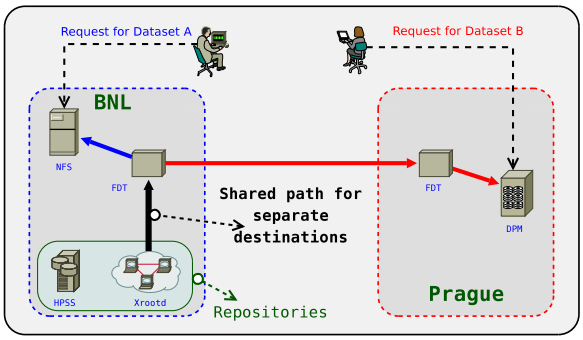

Figure 2: Optimization of the transfer paths with regards to the network structure and link bandwidth. Some network path may be re-used to satisfy multiple requests for the same data.

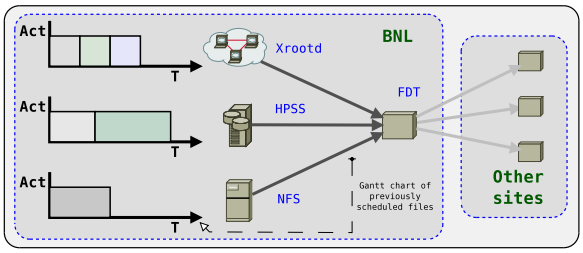

Figure 3: Optimization of the transfer paths with regards to the different data service performance/latency. Multiple sources for the same data may be naturally combined alternatively to avoid overload and service clogging.

We describe the most important optimization characteristic with the help of figures Fig. 2 and 3. Let us suppose there are requests for the same (or overlapping) dataset from two users, while each of them needs the dataset to be processed at his/her specific location. The system has to reason about the possible repositories for the dataset, select the proper ones for every file (the granularity is specified by the files in our case) and produce the transfer paths for each file. The output plan should be optimal with an objective to the overall completion time of all transfers. Thus, this optimization characteristic is focusing on the network structure and respective link bandwidth. As illustrated in Figure 2, it is conceivable in our example that optimization will cause data movement to occur once on some network links while datasets will be moved to two different destinations. Moreover, the files are usually served by several data services (such as Xrootd [6], Posix file systems, Tape systems [8], ...) with different performance and latencies. Therefore, the optimization and reasoning on where to take the files available from multiple sources choice will allow making the proper selection for a file repository, respecting their intrinsic characteristic (communication and transfer speed) and scalability (Fig. 3). In other words, as soon as multiple services and sources are available, load balancing would immediately be taken into account by our planner.

2. Architecture

In this section, we will describe the architecture of the system, explaining briefly each com-

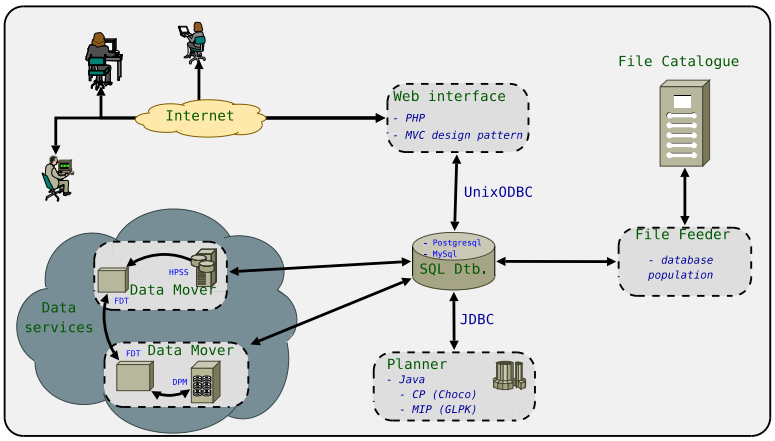

Figure 4: Architecture of the system.

ponent following the work-flow (see Fig. 4 for illustration). End users (or stand-alone services) generate requests using the web interface, written in *PHP* following the *MVC* design pattern. A request is an encapsulation of the meta-data query (as understood by STAR's File and Replica Catalogue) and the destination. The request is stored in a *SQL* database (system supports *MySQL* and *PostgreSQL*) in a Catalog agnostic manner (any Catalog should work as far as they have a LFN/PFN concept our approach relies on) with the additional information like user name, group or date of the request. Later, the component called *File Feeder* contacts the *File and Replica Catalogue* and makes the query for the requested meta-data. The output information is stored back to the database, including all possible locations for every file in a request.

The brain of the system, a component called the *Planner*, takes a subset of all requests for files to be transferred according to the preferred fair-share function. It creates the plan (transfer paths) for the selected requests and stores the plan back to the database. The individual file transfers are handled by the separate distributed component called *Data Mover*. The role of these workers is to perform a point-to-point data transfer on a particular link following the computed plan. The results and intermediate status is continuously recorded in the database and user can check the progress at any time.

We can see that the whole mechanism is a combination of **deliberative** (assuring optimality) and **reactive** planning (assuring adaptability to the changing environment). Since this is crucial to the argument, in the next section we will describe the respective two components (*Planner* and *Data Mover*) serving up as a "reasoner" and a "worker".

2.1 Planner

The *Planner* (Fig. 5-left), the brain of the system, is built on the constraint-based mathematical model. The theoretical background and our continuous progress were published in several papers ([11], [10], [12]). Therefore, we will not go into details in this paper, but only sketch out the main principles. Constraint based approach ([7]) brings a fundamental advantage in a straight forward mapping of the reality restrictions into the mathematical model. The solver uses methods from Constraint Programming and Mixed Integer Programming and the logic tries to minimize the

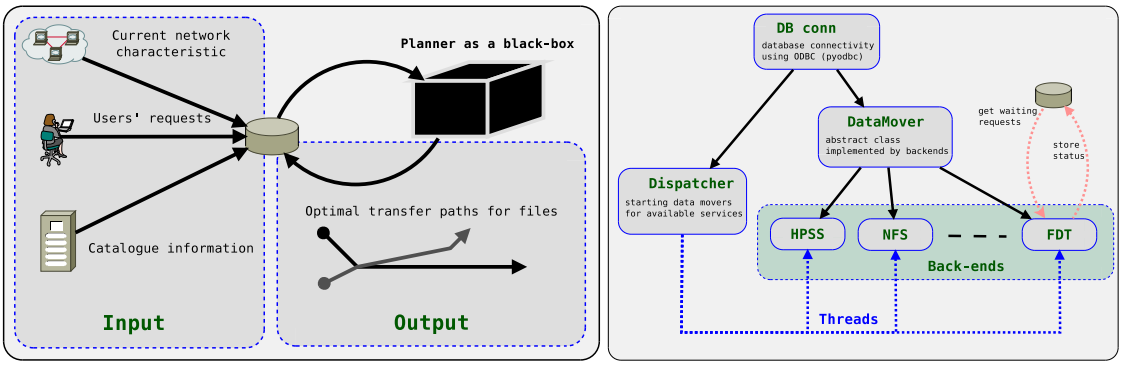

Figure 5: Left: Planner as a black box. **Right:** Data Mover component.

makespan considering all possible combinations. The tree of possibilities may very well contain solutions where transferring data once on a given link lead to a minima or balancing between services lead to the fastest transfers. In all cases, the optimal solution will only be determined by the input parameters. The input consists of three parts: current characteristic of the link or network, requests to be planned (size, logical files) and information from a *File (replica) Catalogue* about possible repositories. Having all these information the solver starts a computation and stores the results directly into the database. The result is a computed transfer path (repository and oriented path to the destination) for each request. Note that multiple requests for the same files would be treated and accounted for in the plan. Our planning is also incremental - we have previously demonstrated ([9]) that a full plan or incremental planning would not make a large difference on the make span overall - the gain of an incremental approach is the ability to self-adapt based on the *Mover's* feedback.

For implementation of the solver we use **Choco** ([2]), a Java based library for constraint programming and **GLPK** ([4], [5]), a library for Mixed Integer Programming. The Java based platform allows us an easier integration with already existing tools in the STAR environment.

2.2 Data Mover

The *Data Mover* is the distributed component responsible for performing data transfers in a reactive way. Each instance is controlling data services within a given computing site and also the wide-area network connections from/to the site. It relies on the underlying data transfer tools and uses them for data movement. In our implementation, we did not address interoperability of data transfer tools (which is not the object of this work) but settled in using by the **F**ast **D**ata **T**ransfer tool (FDT [3]). The way data movers operate is reactive that is, as soon as a file appears at the source node (either at a data service or in a cache space before WAN transfer) it is marked as "ready for transfer" and moved by the proper underlying tool. As soon as the transfer is finished another instance realizes the file is available and initiates the next move (along the computed path from the solver). Our approach is also adaptive: from the initial transfer and consequent monitoring, the real speed can be inferred and re-injected as a parameter for the next incremental plan, helping the system to converge toward realistic transfer rates rather than relying on theoretical optimum alone.

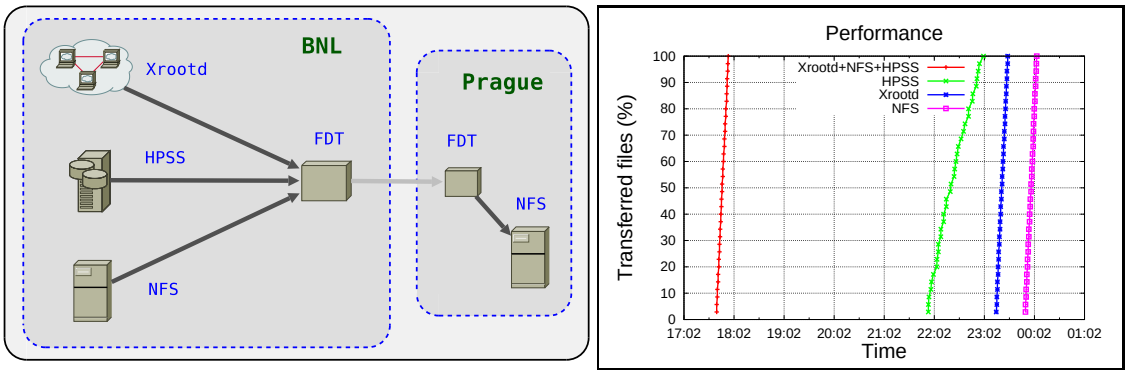

Figure 6: Left: The network and service configuration for the tests. **Right:** The performance of the system using 4 different configurations. On the X axis, we represent the time of transfers while Y is the percentage completion. The x-range of each curve is hence representative of the makespan.

The *Data Mover* is written in *Python* language and concurrent link/service control is achieved by separate threads (Fig. 5-right).

3. Show case

To prove the validity of our planning strategy, a use case was designed and implemented. The purpose of the test was to affirm the software components work and communicate in the expected way and the quality of the computed plan is confident. The environment was for simplicity formed by two computing sites, the central **BNL** and remote **Prague**. The available data services at *BNL* were: *Xrootd*, *NFS* and *HPSS*, while in *Prague* only *NFS* was available. The wide area network (WAN) transfer was controlled by FDT. The configuration is shown in Fig. 6-left. The test hence challenges the planner in making proper decisions when multiple sources are available at the same site.

The request consisted of files available at all data services at *BNL* at the same time and the task was to bring them to the *Prague* NFS service. The test was composed of four different configurations. The planner consecutively considered:

- only Xrootd repository
- only NFS repository
- only HPSS repositories
- a combination of Xrootd, NFS and HPSS repository concurrently

The results of each configuration are shown in Fig. 6-right. As expected, while all files are located on mass storage in STAR, transfers from *HPSS* (in green) are the longest to accomplish and hence, lead to the longest delays in delivery. In our setup, the green and blue curves are near equivalent (*NFS* direct transfers are slightly faster) but it is to be noted that not all files are held on *NFS* (central storage) in STAR and pulling all files from *Xrootd* may cause significant load on a system in use primarily for batch based user analysis (hence, an additional load is not

desirable). When we combined all storage sources, the makespan was equivalent to the one from *Xrootd* while the relative ratio of files transfers from the diverse sources was 19%, 38% and 43% for *HPSS*, *NFS* and *Xrootd* respectively with no load caused on any of the services. At the end, the overall bottleneck was only the WAN transfer speed - we infer our test proved the planner works as expected, since the full reasoning considering all possible repositories led to the optimum makespan. Additionally, the utilization of all services brings the advantage in the form of load-balancing and automatic use of replicas.

4. Conclusions

When multiple sources for files or datasets are available along with many CPU resources in a distributed computing environment, planning is needed to ensure load balancing, efficient and fair data movement and best use of the resources. Random access to files and datasets by users could easily destroy efficiency or render sites inoperative and within this in mind, we have tackled the challenge of coordination of data transfers.

In this work, we specifically presented the architecture components and implementation of a framework, in test mode in the STAR experiment, which goal is to address the planning challenges of transfers over widely distributed resources. Based on constraint and mixed integer programming techniques, the tool was designed to incorporate elements to achieve optimization, coordination and load-balancing. Its simple yet robust architecture allows users to express their requests for files via a Web interface while a back-end planner and set of data movers take care of the movement on the user's behalf. Within our test example of moving files to a single destination considering a dataset available from multiple-sources, we have showed that our approach lead to an optimal plan that is, producing the shortest possible makespan while causing no load on any of the storage systems by automatically load-balancing. With our model (showed to work in simulated mode [10]) and this proof of principles, we are equipped with a corner stone functional architecture and we will pursue as next steps multi-users and multi-sites transfers.

References

[1] STAR Collaboration: J. Adams. Experimental and theoretical challenges in the search for the quark gluon plasma: The STAR collaboration's critical assessment of the evidence from RHIC collisions. *Nuclear Physics A*, 757:102, 2005.

[2] Choco. http://www.emn.fr/.

[3] FDT. http://monalisa.cern.ch/FDT.

[4] GLPK. http://www.gnu.org/software/glpk/.

[5] GLPK-java. http://glpk-java.sourceforge.net/.

[6] A. Hanushevsky, A. Dorigo, and F. Furano. The Next Generation Root File Server. In *Proceedings of the Computing in High Energy and Nuclear Physics (CHEP) conference*, pages 680–683, 2005.

[7] Helmut Simonis. Constraint applications in networks. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 25, pages 875–903. Elsevier, 2006.

[8] Danny Teaff, Dick Watson, and Bob Coyne. The Architecture of the High Performance Storage System (HPSS). In *Proceedings of the Goddard Conference on Mass Storage and Technologies*, pages 28–30, 1995.

[9] Michal Zerola, Roman Barták, Jérôme Lauret, and Michal Šumbera. Using constraint programing to resolve the multi-source / multi-site data movement paradigm on the grid. In *Advanced Computing and Analysis Techniques in Physics Research* `PoS(ACAT08)039`, 2008.

[10] Michal Zerola, Roman Barták, Jérôme Lauret, and Michal Šumbera. Efficient Multi-site Data Movement in Distributed Environment. In *Proceedings of the* 10th *IEEE/ACM International Conference on Grid Computing (GRID)*, pages 171–172. IEEE, 2009.

[11] Michal Zerola, Roman Barták, Jérôme Lauret, and Michal Šumbera. Planning Heuristics for Efficient Data Movement on the Grid. In *Proceedings of the* 4th *Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*, pages 768–771, 2009.

[12] Michal Zerola, Roman Barták, Jérôme Lauret, and Michal Šumbera. Using Constraint Programming to Plan Efficient Data Movement on the Grid. In *Proceedings of the* 21st *International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 729–733. IEEE, 2009.