Using Constraint Programming to Plan Efficient Data Movement on the Grid

Michal Zerola and Michal Šumbera
Nuclear Physics Institute, ASCR
Czech Republic

Roman Barták
Charles University
Czech Republic

Jérôme Lauret
Brookhaven National Laboratory
USA

Abstract

Efficient data transfers and placements are paramount to optimizing geographically distributed resources and minimizing the time data intensive experiments processing tasks would take. We present a technique for planning data transfers to single destination using a Constraint Programming approach. We study enhancements of the model using symmetry breaking, branch cutting, well studied principles from scheduling field, and several heuristics. Real-life wide area network characteristic is explained and the realization of the computed formal schedule is proposed with an emphasis on bandwidth saturation. Results will include comparison of performance and trade-off between CP techniques and Peer-2-Peer model.

1. Introduction and related works

The needs of large-scale data intensive projects arising out of several fields such as bio-informatics (BIRN, BLAST), astronomy (SDSS) or High Energy and Nuclear Physics (HENP) communities have been brinteasers for computer scientists for years. One of such HENP experiments is the STAR experiment 1. As all running experiments, valuable set of new data are taken every year and then reduced to physics ready data formats. The ever growing samples call for large-scaled storage management and efficient scheme to distribute and move data, while end-users may need to access data sets from any point in time. Coordination is hence needed to avoid random access destroying efficiency due to the sharing of common infrastructure such as storage and network and this applies to efficient use of Data Grids. Decoupling of job scheduling from data movement was studied by Ranganathan and Foster in [6]. Authors discussed combinations of replication strategies and scheduling algorithms, but not considering the performance of the network. The nature of HENP experiments, where data are centrally acquired, implies that replication to geographically spread sites is a must in order to process data distributively. Accessing large-scale data remotely over wide-area network has turned out to be highly ineffective and a cause of often sorely traceable troubles. The authors of [8] proposed and implemented improvements to Condor, a popular batch system. Its data management architecture is based on exploiting workflows and utilizing data dependencies between jobs through study of related DAGs. Since the workflow in HENP data analysis is typically simple and embarrassingly parallel without dependencies between jobs these techniques do not lead to a fundamental optimization in this field. Sato et al. in [7] and authors of [5] tackled the question of replica placement strategies via mathematical constraints modeling an optimization problem in Grid environment. The solving approach in [7] is based on integer linear programming while [5] uses Lagrangian relaxation method. The limitation of both models is a characterization of data transfers which neglects possible transfer paths and fetching data from a site in parallel via multiple links possibly leading to better network utilization.

The purpose of our work is to design and develop an automated system that would efficiently use all network resources in order to bring the aimed dataset to the required destination, without sacrifice of fairness. An initial idea of our presented model originates from Simonis [9] and the proposed constraints for traffic placement problem were expanded primarily on links throughputs and consequently on follow-up transfer allocations in time. The solving approach is based on Constraint Programming techniques. One of the immense advantages of the constrained based approach is a gentle augmentation of the model with additional real-life rules. This work expands the initial study in [11].

2. Problem formalization

In this section we will present a formal description of the problem using mathematical constraints which present restrictions from reality.

1http://www.star.bnl.gov

The network is modeled as a directed weighted graph \( (G = (N, E)) \), where the weight of an edge corresponds to the link bandwidth. The information about demand’s
(d ∈ D) origins (orig(d)) is a mapping of that file to a set of sites where the file is available. The goal of the solver is to produce a transfer path for each file from one origin leading to the destination; and for each file and its selected transfer path, allocate a particular link transfer, such that the resulting plan has minimum makespan.

The solving process is composed of two iterative stages and we will describe each of them separately continuing with the explanation of their interaction and possible reduction of search space exploration.

2.1. Planning and scheduling stage

The aim of planning stage is to generate a valid transfer path for each requested file from one of its origins to the destination node. We have studied link-based and path-based approach. The essential idea for the first one is to use one decision variable (X_{de}) for each demand and link, while in the second one the variable is coupled with demand and path. The detailed description of the constraints for link-based principle can be found in [11], while the constraint model for path-based one is explained in [9].

The goal of the scheduling stage is to evaluate the path configuration in the sense of the required makespan. Essentially, it works as an objective function because the realization of the schedule will not depend on particular transfer times calculated in this phase, as we will show in section 5. We will use the notion of tasks and resources from the area of scheduling. For each link e of the graph that will be used by at least one file demand, we introduce a unique unary resource R_{e3}. Similarly, for each demand and its transfer path we introduce a set of tasks (depicted in Figure 1).

We will assume that all requested files are of the identical unit size. This assumption is not far from reality, because data acquired from detector systems are usually stored in files of a similar size. Instead of defining the link speed as a function of file size over a period of time, we will introduce a term slowdown factor, that quantifies how many units of time are needed for transferring one unit of size. This notion allows us to represent constraints in a simpler way.

2.2. Scale and complexity of the problem

We will show that computational complexity of the problem, in particular of the scheduling stage is strongly NP-hard. The input of the scheduling phase consists of the directed weighted graph (eventual loops are allowed), where the weight of an edge determines its throughput and the set of paths without loops, all of them leading to the same vertex. Each path needs time defined by a throughput of a particular edge for crossing it (for a given edge, this time is identical for all paths). Crossing an edge cannot be interrupted and only one path can cross an edge in a given time. The path must cross its edges in an order defined by the path itself. The goal is to allocate edge crosses for each path in such a way that when the last path reaches destination vertex is minimized. We will use the well-known three field classification \alpha{\beta}{\gamma} for scheduling problems as described in [3]. We will demonstrate a possible polynomial-time reduction from J3|\text{pij}=1|\text{Cmax}, a 3-machine unit-time Job-shop scheduling problem (JSSP). The transformation of the JSSP instance consists of creation a unique link e1, e2, and e3 for every machine M1, M2, and M3 a unique link e1, e2, and e3 is created and a single destination site dest. Every job with ordered actions defines a transfer path with alternating dummy edges, as depicted in Fig. 2. Slowdown for each dummy edge is set to 0, e.g. edge is not causing delays to any transfer. The role of dummy edges is to provide correct paths for each job.

The transformation written above can be realized in a polynomial time, as the size of a transformed instance increases linearly by the factor of 3 caused by dummy edges. Due to the fact that the J3|\text{pij}=1|\text{Cmax} problem belongs to the strongly NP-hard class [10], the direct consequence implies that computational complexity of our problem is at least the same.

3. Constraint model and solving strategy

The principle of the search procedure and iteration of described two stage model is outlined in Algorithm 1. The actual best makespan is used as a bound for scheduling phase (Branch-and-bound strategy). Moreover, the ac-

**Figure 1.** The transfer paths of demands \( d_1 \) and \( d_2 \) share link \( e_3 \), and consequently resource \( R_{e3} \) as well.

**Figure 2.** The job \( J = \langle A_3, A_1, A_2 \rangle \) defines a transfer path to dest using links \( e_1, e_2, e_3 \), with order given by precedences of actions.
some local inconsistencies and thus delete some regions of search space that do not contain any solution. By removing inconsistent values from variable domains the efficiency of the search algorithms is improved. A filtering principle we can apply for the decision $X$ variables in the link-based planning model is following. Let $S_1$ and $S_2$ be two sets of the decision $X$ variables, such that $S_2 \subseteq S_1$. Then, if there exists a constraint stating $\sum_{X \in S_1} X = 1$ and similarly for the second set $\sum_{X \in S_2} X = 1$, we can reduce domains of variables in $S_1 \setminus S_2$ to value 0 (depicted in Fig. 4).

The filtering procedure is simple. During posting of constraints into the model, each set of variables, summation of which must equal to 1, is stored. Then we search for pairs $S_1$ and $S_2$ using a quadratic nested loop over the sets stored.

### 3.2. Search heuristics

We have tested several combinations of variable selection and value iteration heuristics. Initially, in the planning phase well known dom strategy [2] for labeling decision $X$ variables was tested. We suggested also a FastestLink selection for link-based planning approach and similarly FastestPath for path-based one. According to the measurements shown in section 4, the majority of time was spent in the planning phase, hence we proposed an improved variable selection heuristic that exploits better the actual transfer times called MinPath. We have described all mentioned heuristics in [11]. In the scheduling phase two approaches were considered. The first one, called SetTimes, is based on determining Pareto-optimal trade-offs between makespan and resource peak capacity. Detailed description and explanation can be found in [4]. The second one is a texture-based heuristic called SumHeight, using ordering tasks on unary resources. The used implementation originates from [1] and supports Centroid sequencing of the most critical activities.

### 4. Comparative studies

We have implemented and compared performance of both alternatives of the model, namely using link-based and path-based approach. Several combinations of heuristics were tried and in addition comparison with simulated Peer-2-Peer (P2P) method is shown. The P2P approach is based on torrent-like principle, where files are pulled directly from origins. For implementation of the solver we use
Choco \(^2\), a Java based library for constraint programming. The Java based platform allows us an easier integration with already existing tools in the STAR environment.

Regarding the data input part, the realistic-like network graph consists of 5 sites, denoted as BNL, LBNL, MIT, KISTI, and Prague and all requested files are supposed to be transferred to Prague. The distribution of files origins at one particular site, is following: the central repository is at BNL where 100% of files are available. LBNL holds 60%, MIT 1%, and KISTI 20% of all files. All presented experiment were performed on Intel Core2 Duo CPU@1.6GHz with 2GB of RAM, running a Debian GNU Linux.

CPU time limit was used for both phases and more precisely, if the top-level search loop detects that time cumulatively spent in planning and scheduling phase exceeded 30 seconds, the search is terminated. Table 1 shows comparison of six combinations of search heuristics for path-based and path-based model with a Peer-2-Peer one, with an emphasis on a makespan, as quality of the result. We can see that link-based model gives generally better results than path-based, and the most efficient combination of heuristics is FastestLink variable selection in Decreasing value order with the SumHeight heuristic used in a scheduling phase. In this case the solver produced schedules that were better than P2P for all input instances, while the most significant benefits (≈ 50% gain) are for instances up to 50 files. For instances of 40 and more files, all combinations of heuristics reached the time limit of 30 seconds. For P2P all makespans were obtained in less than 1 second. Makespans are in general time units, where 1 unit in reality depends on real link speeds, and we can roughly estimate this 1 unit to the couple of seconds. Hence, the time taken to compute a schedule is paid-off by savings resulting from a better makespan.

In reality the network characteristic is dynamic and fluctuates in time. Hence, trying to create a plan for 1000 or more files that will take several hours to execute is needless, as after the time elapsed the computed plan may no longer be valid. Our intended approach is to work with batches, giving us another benefit of implementing fair-share mechanism in a multi user environment. Particularly, the requests

\(^2\)http://choco.sourceforge.net

<table>
<thead>
<tr>
<th>Files</th>
<th>P2P</th>
<th>Link-based</th>
<th>Path-based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RunTime</td>
<td>% in 1(\text{st})</td>
<td>% in 2(\text{nd})</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 1. Comparison of makespans. Increasing, Decreasing value selection, ST - SumHeight heuristic.

For a better decomposition and estimate of times spent in the phases we studied heuristics combinations such as FastestLink + FastestLink and MinPath (see Table 2). On the basis of detailed measurements, we can see that the majority of time spent in a solving process happens in the planning stage. According to this fact and an average number of scheduling calls (5-th column in Table 2) the implementation of the scheduling stage seems to be fairly efficient. Therefore, we have focused on the improvements for the heuristic MinPath in the planning stage as proposed in section 3 and compared the performance with the FastestLink. Figure 5 shows that convergence of the new MinPath heuristic is faster than the FastestLink and both heuristics achieve better makespan than the P2P approach.

5. Schedule execution

For evaluating the applicability of our model to the real world, let us briefly describe the TCP/IP protocol principle underlying transfers. Transferring a single file consists of splitting the data content into packets of a given size...
(defined by window scaling parameter of the protocol) and sending them one-by-one from the source to the destination. Since the reception of each packet has to be acknowledged by the receiver to achieve delivery guarantee, the time for the acknowledged packet to travel from source to destination and back, so called round-trip time (RTT) plays an important role. In wide area network the RTT is significant. To overcome such delays, data transfer tools use threads to handle several TCP streams in parallel in an attempt to smooth or minimize the intrinsic delays associated with TCP. Another standard approach is the use of multiple sender nodes per link to increase the bandwidth usage. With this last approach, any one instance downtime would be compensated by other active senders transfers. Both approaches are typically combined for best results and it has been experimentally shown that a flat transfer rate could be achieved across long distance.

Our modeling assumes a single file transfer at any time on a link using unary resources. Trying to explicitly model the real network and packets behavior into the solver would over-complicate the problem. Our implementation will consider: (a) every link is supplied by one LinkManager responsible for transferring files over the link if the link is part of their computed transfer path; (b) as soon as a file becomes available, the corresponding LinkManager initiates another instance of the transfer, respecting a maximum allowed simultaneous parallel instances.

The implementation will consider just the plan - the computed transfer paths, because there is no due-time limitation and executes transfers in a greedy manner. However, to allow this, one has to be sure that computed time to complete the schedule would not differ substantially from the real execution of the transfers. Consequently we developed the real-life network simulator along the above facts and comparison of the makespans showed results consistent with each other within a 3% margin, which is negligible. Hence the experiment confirmed that the presented model provides fairly accurate estimate of the real makespan.

6. Conclusions

In this paper we tackle the complex problem of efficient data movements on the network within a distributed environment. The problem itself arises from the real-life needs of the running nuclear physics experiment STAR and its peta-scale requirements for data storage and computational power. We presented the two stage constraint model, coupling path planning and transfer scheduling phase for data transfers to the single destination. The complexity of the problem is introduced and several techniques for pruning the search space, like symmetry breaking, domain filtering, or implied constraints, are shown. We proposed and implemented several search heuristics for both stages and performed experiments for evaluating their applicability. Comparison of the results and trade-off between the schedule of a constraint solver and a Peer-2-Peer simulator indicates that it is promising to continue with the work, thus bring improvements over the current techniques used in the community. In the nearest future we will concentrate on the integration of the solver with real data transfer back-ends, consequently execute tests in the real environment.

Acknowledgment

The investigations have been partially supported by the IRP AVÖZ 10480505, by the Grant Agency of the Czech Republic under Contract No. 202/07/0079 and 205/13/01457, by the grant LC07048 of the Ministry of Education of the Czech Republic and by the U.S. Department Of Energy.

References