MISTA 2009

Planning Heuristics for Efficient Data Movement on the Grid

Michal Zerola · Roman Barták · Jérôme Lauret · Michal Šumbera

1 Problem area

Computationally challenging experiments such as the STAR at RHIC (Relativistic Heavy Ion Collider [1]) located at the Brookhaven National Laboratory (USA), have developed a distributed computing approach (Grid) to face their massive computational and storage requirements. Since the peta-bytes of data produced by STAR are geographically spread, it is necessary to face the question of efficient data transfers and placements in order to bring requested dataset to a particular site for further analyses. Our aim is to create a plan how to transfer data from data-warehouses to the requested site in the shortest time. The presented approach is based on Constraint Programming with initial modeling idea from [2] further extended in [3]. This paper brings a new search heuristic used during the selection of routes.

2 Formal model

The input of the problem consists of two parts. The first part represents the Grid network and file origins. The network, formally a directed weighted graph, consists of a set of nodes **N** (sites) and a set of directed edges **E** (links). The weight of an edge describes the number of time units needed to transfer one size unit. Information about files' origins is a mapping of each file to a set of sites where the file is available. The second part of the input is a user request, namely the set of files that need to be transferred to a common destination site. The solving process is composed of two stages: a transfer path for each file, i.e., one origin and a valid path from the origin to the destination, is selected (*planning*); second, for each file and its selected transfer path, the particular transfers via links are scheduled in time such that the resulting plan has minimal makespan (*scheduling*). Both stages iterate until the plan of transfers with the minimal makespan is found (see Alg. 1). In [3] we identified that about 90% of overall time is spent in the planning stage hence we put our effort to improve this stage. The following formalism is used to define a constraint model describing the planning

Michal Zerola (corresponding author)
Nuclear Physics Institute, Academy of Sciences, Czech Republic
E-mail: michal.zerola@ujf.cas.cz

Algorithm 1 Pseudocode for a search procedure.

```
makespan ← sup
plan ← Planner.getFirstPlan()
while plan != null do
  schedule ← Scheduler.getSchedule(plan, makespan) {B-a-B on makespan}
  if schedule.getMakespan() < makespan then
    makespan ← schedule.getMakespan() {better schedule found}
  end if
  Planner.getNextPlan(makespan) {next feasible plan with cut constraint}
end while
```

sub-problem. The set $\mathbf{OUT}(n)$ consists of all edges leaving node n, the set $\mathbf{IN}(n)$ of all edges leading to node n. Input received from a user is a set of demands $\mathbf{D}$ needed at the destination site *dest*. For every demand $d \in D$ we have a set of sources $\mathbf{orig}(d)$ - sites where the demanded file d is already available. We will present the *link-based* approach for modeling planning constraints, another approach called *path-based* can be found in [3].

The essential idea of the link-based approach is using one decision $\{0, 1\}$ variable X_{de} for each demand and link of the network, denoting whether demand d is routed over edge e or not. Constraints (1-3), ensure that if all decision variables have assigned values then the resulting configuration contains transfer paths. These constraints alone allow isolated loops along with the valid paths and therefore *precedence constraints* (4) are used to eliminate such loops.

$$\forall d \in \mathbf{D}: \sum_{e \in \cup \mathbf{OUT}(n|n \in \mathbf{orig}(d))} X_{de} = 1, \sum_{e \in \cup \mathbf{IN}(n|n \in \mathbf{orig}(d))} X_{de} = 0 \tag{1}$$

$$\forall d \in \mathbf{D}: \sum_{e \in \mathbf{OUT}(dest(d))} X_{de} = 0, \sum_{e \in \mathbf{IN}(dest(d))} X_{de} = 1 \tag{2}$$

$$\forall d \in \mathbf{D}, \forall n \notin orig(d) \cup \{dest(d)\}:$$
$$\sum_{e \in \mathbf{OUT}(n)} X_{de} \leq 1, \sum_{e \in \mathbf{IN}(n)} X_{de} \leq 1, \sum_{e \in \mathbf{OUT}(n)} X_{de} = \sum_{e \in \mathbf{IN}(n)} X_{de} \tag{3}$$

Precedence constraints (4) use non-decision positive integer variables P_{de} representing possible start times of transfer for demand d over edge e. Let dur_{de} be the constant duration of transfer of d over edge e. Then constraint

$$\forall d \in \mathbf{D}\ \forall n \in \mathbf{N}: \sum_{e \in \mathbf{IN}(n)} X_{de} \cdot (P_{de} + dur_{de}) \leq \sum_{e \in \mathbf{OUT}(n)} X_{de} \cdot P_{de} \tag{4}$$

ensures a correct order between transfers for every demand, thus restricting loops. Unfortunately, constraints (4) do not restrict the domains of P_{de} until the values X_{de} are known and therefore we suggest using a redundant constraint (5) to estimate better the lower bound for each P_{de}. Let *start* be the start vertex of e not containing demand d ($start \notin orig(d)$):

$$\min_{f \in \mathbf{IN}(start)} (P_{df} + dur_{df}) \leq P_{de} \tag{5}$$

Variables P_{de} can be used not only to break cycles but also to estimate makespan of the plan. The idea is that according to the number of currently assigned demands per

	Solution time		Makespan		
Files	*FastestLink*	*MinPath*	*FastestLink*	*MinPath*	*P2P*
25	3.862	1.431	14	14	24
50	26.508	27.556	36	32	40
100	8.627	3.176	73	73	80
150	16.52	14.618	111	110	120
200	26.167	14.031	146	146	160

Table 1 Comparison of heuristics with emphasis on time when the best solution was found and the makespan.

some link and their possible starting times, we can determine the lower bound of the makespan for schedule that will be computed later in the scheduling stage. Hence if we have some upper bound for the makespan (typically obtained as the best solution from the previous iteration of planning and scheduling) we can restrict plans in next iterations by the following constraint:

$$\forall e \in \mathbf{E} : \min_{d \in \mathbf{D}}(P_{de}) + \sum_{d \in \mathbf{D}} X_{de} \cdot dur_{de} + SP_e < makespan, \tag{6}$$

where SP_e stands for the value of the shortest path from the ending site of e to $dest$.

3 Search heuristics

The constraint model needs to be accompanied by a clever branching strategy to achieve good runtimes. In [3] we proposed *FastestLink* variable selection heuristic that suggests for each demand d to instantiate first the variable X_{de_j} corresponding to the fastest link ($j = \arg\min_{i=1,\ldots,m} dur_{de_i}$). In this paper we propose a new variable selection heuristic that exploits better the actual transfer times by using information from variables P_{de}. In particular, the heuristic, called *MinPath*, suggests to instantiate first variable X_{de} such that the following value is minimal:

$$\inf P_{de} + dur_{de} + SP_e, \tag{7}$$

where $\inf P_{de}$ means the smallest value in the current domain of P_{de}.

4 Experiments

We compared the performance of the *FastestLink* and *MinPath* heuristics and a Peer-2-Peer model [4] that is currently the most frequently used approach to solve the problem. The constraint model was implemented in the Java-based constraint programming library **Choco** 2.0.0.3 and all experiments were performed on Intel Core2 Duo (@1.6GHz) with 2GB of RAM, running Debian GNU Linux operating system. We used a realistic-like network graph and file distribution as described in [3]. Figure 1 shows that convergence of the new *MinPath* heuristic is faster than the *FastestLink* and both heuristics achieve better makespan than the P2P approach. Table 1 shows similar comparison of heuristics and the P2P model including the time when the best solution was found for several input instances.

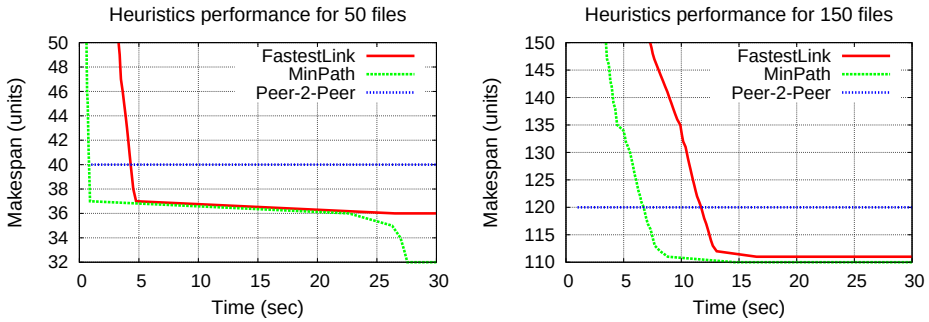

Fig. 1 Convergence of makespan during the search process for *FastestLink* and *MinPath*.

5 Conclusions

In this paper we tackle the complex problem of efficient data movements on the network within a Grid 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 as well. We have focused on improvements of the search heuristic for the planning stage using domains of start time variables that were added to the constraint model with precedence constraints. Experimental results confirmed that the new heuristic converges faster toward the optimal solution.

Acknowledgements The investigations have been partially supported by the IRP AVOZ 10480505, by the Grant Agency of the Czech Republic under Contract No. 202/07/0079 and 201/07/0205, by the grant LC07048 of the Ministry of Education of the Czech Republic and by the U.S. Department Of Energy.

References

1. J. Adams, e.a.: 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**, 102–183 (2005)
2. Simonis, H.: Constraint applications in networks. In: F. Rossi, P. van Beek, T. Walsh (eds.) Handbook of Constraint Programming, chap. 25, pp. 875–903. Elsevier (2006)
3. Zerola, M., Barták, R., Šumbera, M.: Using constraint programming to plan efficient data movement on the grid. (Submitted to CPAIOR 2009) (2009)
4. Kaashoek, M.F., Stoica, I. (eds.): Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers, *Lecture Notes in Computer Science*, vol. 2735. Springer (2003)