Real-time GDA

ABSTRACT

ACM Reference format:
. 2016. Real-time GDA. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 3 pages.
DOI: 10.475/123_4

1 CLARINET

1.1 Given claims

The paper's introduction gives the following claims which were interrogated:

(1) Formulating an optimal solution for a multi-query, network-aware, joint query planning/placement/scheduling is computationally intractable.
✓ Cites two sources that even for a single query with the given scheduling assumptions is NP-hard.

(2) CLARINET does joint multi-query planning. It picks the best WAN-aware QEP/task placement/take scheduling per query and provides "hints" (location and start times of each task) to the query execution layer. It optimizes for minimal average query completion times. Task placement and task scheduling within a query are not done jointly.
✓ Joint multi-query planning although not optimal because of NP-hardness.

(3) They show how to (heuristically) compute the WAN-optimal QEP for a single query which includes task placement and task scheduling. The solution relies on reserving WAN links. This is an effective heuristic for the best single-query QEP, that decouples placement and scheduling.
✓ Task placement and scheduling are decoupled.
× But not a jointly optimal decomposition which is NP-hard.
× Task placement is minimizes time usage of WAN with a linear program, not completion time which would become a piecewise linear program.
✓ Given a task placement for a single query, the scheduling problem gives the optimal (non-interruptible) schedule to minimize that query's completion time.

(4) They allow for cross-query (heuristic) optimization of n queries. They order the queries by optimal QEP expected completion time. Then they choose the ith query's QEP considering the WAN impact of the $i-1$ preceding queries.
✓ This mimics the heuristic rule Shortest Job First (SJF).

(5) They heuristically compact the schedules tightly in time by considering the above order and groups of $k \leq n$ queries to combat resource fragmentation.
✓ This is done at a cost to average completion time.

(6) They extend the above heuristic to accommodate (i) fair treatment of queries, (ii) minimize WAN bandwidth costs, and (iii) online query arrivals.
✓ This is done, but only heuristically.

DOI: 10.475/123_4

1.2 Model

There are n queries and each query j has a set QEP-Set QS_j from which one QEP must be chosen. Also chosen are the task locations and task start times.

DAG assumptions:

(1) Each node in the DAG represents a stage which is a group of tasks. (Josh says: We were thinking of DAG nodes representing all tasks, not just a group of tasks.)
(2) Each stage in the DAG has a known amount of output data D_j at site j.
(3) Each stage is assumed to be either a set of Map tasks or a set of Reduce tasks. (Josh says: This seems pretty restrictive.)
(4) Each Map stage is placed at the site holding its input data.
(5) Each Reduce stage can be distributed to all sites and is represented by a continuous fraction r_j, i.e. the fraction of reduce tasks for the stage at site j. (Josh says: I think this simplification is the reason why they choose to model only Map and Reduce tasks.)

Scheduling assumptions:

(1) Network transfers do not overlap. In fact, they have a theorem that proves that any optimal schedule has an equivalent non-overlapped schedule.
(2) Obtain non-interruptible transfer schedules.
(3) Bandwidth B_{ij} between sites i and j is known and constant.
(4) The compute phase of a task can only start after all its inputs are available to it at its site.

1.3 Single Query Problem

The goal is to minimize the query execution time. First, the tasks are placed across the sites. Then, the data transfers are scheduled.

Given a DAG: Within each stage (single node of the DAG), the decision of (Reduce) task placement is *decided* independently from other stages but *have* a dependency on each other through a variable τ_{ij}. This value is the length of time already reserved on the link between sites i and j for stages that do not have a DAG ordering with this stage.

1.3.1 *Task placement within a (Reduce) stage.*

The following linear program is given to decide the distribution of reduce tasks $\{r_j\}$ in a particular stage that respects the current reservation status of each link:

$$\min_{r} \quad \sum_{i,j} \left(\frac{D_i r_j}{B_{ij}} + \tau_{ij} \right) \tag{1a}$$

$$\text{s.t.} \quad \sum_{j} r_j = 1 \tag{1b}$$

$$r_j \geq 0 \quad \forall j \tag{1c}$$

After $\{r_j\}$ is decided, each τ_{ij} is incremented by $\dfrac{D_i r_j}{B_{ij}}$.

(Josh says: This formulation is a bit strange. First, given the above formulation τ_{ij} is a constant, and so has no effect on the linear programming problem. Second, the summation is minimizing total time usage of the WAN and *not* stage execution time. To minimize stage execution time, it should be the max operator and not the summation operator. But then the order in which these problems are solved will matter. However, there is no discussion in order in which these problems are solved since it doesn't matter with the summation operator being used.)

After deciding the task placement for each stage, task scheduling for the QEP is decided.

1.3.2 Scheduling tasks in a QEP. The DAG is augmented in two ways. First, each data transfer is made into a node that is place between two the associated stages. Second, tasks from the same stage at the same site are coalesced into a sub-stage. From here on, only the start times s of the sub-stages will be scheduled.

Given two (data transfer) sub-stages u and v in a DAG, if u must finish before v then we have the following constraint:

$$s(v) \geq s(u) + d(u) \tag{2}$$

where $s(u)$ is the start time and $d(u)$ is the duration of stage u.

To decide the order between sub-stages that don't have an ordering but need to have a non-overlapping schedule we add the following constraints:

$$s(v) \geq s(u) + d(u) - N(1 - z_{uv}) \tag{3a}$$
$$s(u) \geq s(v) + d(v) - N(z_{uv}) \tag{3b}$$

where z_{uv} is a binary decision variable that indicates if v is executed after u, and N is a large constant.

With the above scheduling constraints the i-th QEP for the query is:

$$\Phi^i := \max_{u \in \text{QEP}^i} s(u) + d(u) \tag{4}$$

which is a binary integer linear program.

1.4 Multiple Query Problem

1.4.1 Handling previously scheduled queries. Suppose low(b) and high(b) represents the start and end times of a reservation on a link for a previously scheduled query. Then we add the following two constraints to the scheduling problem:

$$s(u) \geq \text{high}(b) - N(1 - x_{ub}) \tag{5a}$$
$$\text{low}(b) \geq s(u) + d(u) - N(x_{ub}) \tag{5b}$$

where x_{ub} is a binary indicator denoting that u is scheduled after interval b.

(Josh says: Below, only the high-level ideas are given because they are all explicitly heuristics.)

1.4.2 Handling resource fragmentation. As queries get scheduled, resources become fragmented from reservations happening without accounting for other queries. They present a heuristic that optimally packs subsets of queries of size $k \leq n$.

1.4.3 Fairness. The multi-query heuristics above favor shorter queries over longer queries. To combat this, they target that each query should finish within a specific multiple of its completion time if it was the only query running. Then they only pick the k queries from a set of queries that are far from this target.

1.4.4 WAN utilization. To limit WAN usage, they only select QEPs that have a WAN usage below a threshold β.

1.4.5 Online Arrivals. When a new query arrives, update and recompute for all queries that have yet to have any tasks start to be executed.

(Josh says: In thinking about our own problem to solve, I think our main focus could be on (i) individual task placement, and (ii) for a general DAG of tasks since these are the weakest parts of this paper. Other sub-areas we could focus on are the problems that they only attack with heuristics such as: (i) WAN usage, (ii) online multiple queries, (iii) fairness of multiple queries)

2 SINGLE MAPREDUCE QUERY

2.1 Model

Let $\mathcal{L}$ be the set of sites that holds data and runs tasks. For each inter-site WAN link, let B_{ij} be the bandwidth from site $i \in \mathcal{L}$ to site $j \in \mathcal{L}$. We assume that the bandwidths are stable within the time frame of doing real-time data analytics.

We perform the map tasks at the sites that contain the associated data and denote D_i as the output data from all of the map tasks at site i. The fraction of reduce tasks assigned to site j is denoted as r_j which is also the fraction of all other sites' data that must be transfered through the WAN to j. This means that the total WAN usage is for a given task distribution r is:

$$\sum_i \sum_{j \neq i} D_i r_j \tag{6}$$

which can be rearranged to:

$$\sum_i \sum_{j \neq i} D_i r_j = \sum_i D_i \sum_j r_j - \sum_j D_j r_j = \sum_i D_i - \sum_j D_j r_j \tag{7}$$
$$= \sum_j D_j (1 - r_j) \tag{8}$$

If we are given a start time s after the map steps are all completed and a data shuffle deadline t for the reduce tasks, then the completion time is bounded as such:

$$s + \max_{(i,j):j \neq i} \left\{ \frac{D_i r_j}{B_{ij}} \right\} \leq t \tag{9}$$

which is caused by the heterogeneous WAN bandwidth and has a bottlenecking link(s). We assume that $t > s$.

2.2 Problem

Minimize WAN usage. When minimizing WAN usage for a MapReduce data shuffle we have the following optimization problem:

$$\min_r \quad \sum_j D_j (1 - r_j) \tag{10a}$$
$$\text{s.t.} \quad \sum_j r_j = 1 \tag{10b}$$
$$r_j \geq 0 \quad \forall j \tag{10c}$$

Feasibility to meet deadline. We want to find a feasible r so that the data shuffle finishes at or before t:

$$\text{find } r \text{ s.t.} \quad \frac{D_i r_j}{B_{ij}} \leq t - s \quad \forall (i,j) : j \neq i \tag{11a}$$
$$\sum_j r_j = 1 \tag{11b}$$
$$r_j \geq 0 \quad \forall j \tag{11c}$$

Minimize WAN usage given a shuffle deadline. When minimizing WAN usage for a MapReduce data shuffle for a given deadline t we have the following optimization problem:

$$\min_r \quad \sum_j D_j (1 - r_j) \tag{12a}$$
$$\text{s.t.} \quad \frac{D_i r_j}{B_{ij}} \leq t - s \quad \forall (i,j) : i \neq j \tag{12b}$$
$$\sum_j r_j = 1 \tag{12c}$$
$$r_j \geq 0 \quad \forall j \tag{12d}$$

2.3 Optima

Since Problem (12) is a convex optimization problem (linear program), we look at the Karush-Kuhn-Tucker (KKT) conditions for optimality using the following dual variables $(\mu_{ij}, \theta, \lambda_j) : \forall(i,j), i \neq j$:

$$-D_j + \sum_{i \neq j} \frac{D_i}{B_{ij}} \mu_{ij} + \theta - \lambda_j = 0 \quad \forall j \tag{13a}$$

$$\mu_{ij} \left(\frac{D_i}{B_{ij}} r_j - (t-s) \right) = 0 \quad \forall(i,j) : i \neq j \tag{13b}$$

$$r_j \lambda_j = 0 \quad \forall j \tag{13c}$$

$$\mu_{ij} \geq 0 \quad \forall(i,j) : i \neq j \tag{13d}$$

$$\lambda_j \geq 0 \quad \forall j \tag{13e}$$

$$\frac{D_i}{B_{ij}} r_j - (t-s) \leq 0 \quad \forall(i,j) : i \neq j \tag{13f}$$

$$\sum_j r_j - 1 = 0 \tag{13g}$$

$$r_j \geq 0 \quad \forall j \tag{13h}$$

where (13a) are the first stationary conditions, (13b) (13c) are the complimentary slackness conditions, (13d) (13e) are the dual feasibility conditions, and (13f) (13g) (13h) are the primal feasibility conditions.

Notice that for every site j, (13f) implies that there could be bottlenecking link if $\max_{i \neq j} \{D_i / B_{ij}\} > t - s$. There would not be a bottlenecking link only if making $r_j = 1$ meets the deadline. Let us denote

$$U_j := \max_{i \neq j} \{D_i / B_{ij}\} \tag{14}$$

as the maximum upload time for site j if $r_j = 1$, and denote

$$b_j := \arg\max_{i \neq j} \{D_i / B_{ij}\} \tag{15}$$

as the set of bottlenecking data sources. If for any $i \notin b_j$, then (13f) is satisfied if it satisfied for b_j and also $D_i / B_{ij} < t - s$. This fact and (13b) implies that $\mu_{ij} = 0$ if $i \notin b_j$. Now the KKT conditions (13) have redundant conditions that can be eliminated to:

$$-D_j + U_j \sum_{i \in b_j} \mu_{ij} + \theta - \lambda_j = 0 \quad \forall j \tag{16a}$$

$$\mu_{ij} \left(U_j r_j - (t-s) \right) = 0 \quad \forall(i,j) : i \in b_j \tag{16b}$$

$$r_j \lambda_j = 0 \quad \forall j \tag{16c}$$

$$\mu_{ij} \geq 0 \quad \forall(i,j) : i \in b_j \tag{16d}$$

$$\lambda_j \geq 0 \quad \forall j \tag{16e}$$

$$U_j r_j - (t-s) \leq 0 \quad \forall(i,j) : i \in b_j \tag{16f}$$

$$\sum_j r_j - 1 = 0 \tag{16g}$$

$$r_j \geq 0 \quad \forall j \tag{16h}$$

We have three cases for each r_j:

(1) $r_j = (t-s)/U_j$. Since $t - s > 0$ and $U_j > 0$, then $r_j > 0$. Then (16c) results in $\lambda_j = 0$. In this case (16a) turns into:

$$\theta = D_j - U_j \sum_{i \in b_j} \mu_{ij} \tag{17}$$

Since $\sum_{i \in b_j} \mu_{ij} \geq 0$ from (16d), then this means that $\theta \leq D_j$.

(2) $0 < r_j < (t-s)/U_j$. Then (16c) results in $\lambda_j = 0$ and (16b) results in $\mu_{ij} = 0 : \forall i \in b_j$. In this case (16a) turns into:

$$\theta = D_j \tag{18}$$

(3) $r_j = 0$. Then (16b) results in $\mu_{ij} = 0 : \forall i \in b_j$. In this case (16a) turns into:

$$\theta = D_j + \lambda_j \tag{19}$$

Since (16e), then $\theta \geq D_j$.

Since θ can only be a single value, the above cases give a natural ordering. For a particular θ: if $D_j > \theta$, then Case 1 applies; if $D_j < \theta$ then Case 3 applies; if $D_j = \theta$ then Cases 1,2, or 3 could apply. This also means that we can restrict θ to the set $\{D_j : \forall j\}$ without restricting the solution space of the task placements $r_j : \forall j$.

Also, if all sites are in Case 1 and we observe that from (16g) $\sum_j \frac{1}{U_j} < \frac{1}{t-s}$, then the deadline t is too soon and the problem is infeasible. Let us denote the lower bound on t as:

$$\underline{t} := s + \frac{1}{\sum_j \frac{1}{U_j}} \tag{20}$$

On the other hand, let $l := \arg\max_j \{D_j\}$ and if $U_l < t - s$, then $r_l = 1$ and $r_j = 0 : \forall j \neq l$. Let us denote the upper bound on t as:

$$\bar{t} := s + U_l \tag{21}$$

If $t \geq \bar{t}$, then $r_l = 1$, $r_j = 0 : \forall j \neq l$, and the WAN usage is $\sum_i D_i - D_l$.

If $t = \underline{t}$, then $r_j = (\underline{t} - s)/U_j : \forall j$ and the WAN usage is $\sum_i D_i - (\underline{t} - s) \sum_j (D_j / U_j)$.

Note that if either the maximum deadline range

$$\bar{t} - \underline{t} = U_l - \frac{1}{\sum_j \frac{1}{U_j}} \tag{22}$$

(23)

or the maximum WAN savings

$$D_l - \frac{1}{\sum_j \frac{1}{U_j}} \sum_j \frac{D_j}{U_j} \tag{24}$$

has significant cost, then developing an optimization algorithm is worthwhile.

2.4 Algorithm

We have the following algorithm:

(1) **Initialize:**
- Order D_j in descending order and relabel the indexes for this ordering.
- Set $y := 1$, $k := 1$, and $r_j := 0 : \forall j$
- For each site j, set:
  $U_j := \max_{i \neq j} \{D_i / B_{ij}\}$
  $b_j := \arg\max_{i \neq j} \{D_i / B_{ij}\}$.

(2) **Test for feasibility:**
- If $t \leq s + \frac{1}{\sum_j \frac{1}{U_j}}$, then stop because the problem is infeasible.

(3) **Process:**
- Replace $r_k := \min\{y, (t-s)/U_k\}$.
- Update $y := y - r_k$ and $k := k + 1$.
- If $y \leq 0$ or $k > |\mathcal{L}|$, then stop and output $r_j : \forall j$. Otherwise repeat Step (3).

APPENDIX