BNL-106308-2014-CP

Optimizing Circuit Allocation for Bandwidth Reservations in Dynamic Virtual Circuit Networks

Li Shi

IEEE International Conference on Computing, Networking and Communications(ICNC 2015)
Anaheim, CA
February 16-19, 2015

Feb. 2015

Computational Science Center

Brookhaven National Laboratory

U.S. Department of Energy
OFFICE OF ADVANCED SCIENTIFIC COMPUTING RESEARCH

DISCLAIMER

Optimizing Circuit Allocation for Bandwidth Reservations in Dynamic Virtual Circuit Networks

Li Shi
Department of Electrical and Computer Engineering,
Stony Brook University,
Stony Brook, NY 11794.
li.shi@stonybrook.edu

Sushant Sharma*
Windows Azure,
Microsoft Corp.,
Redmond, WA 98052.
sushant_sharma@outlook.com

Dimitrios Katramatos
Dantong Yu
Computational Science Center,
Brookhaven National Laboratory,
Upton, NY 11973.
{dkat,dtyu}@bnl.gov

***Abstract*—Modern research and education networks, such as ESnet and Internet2, offer the capability to increase the reliability and predictability of big data transfers through the reservation of bandwidth in the form of dynamic (on-demand) virtual circuits interconnecting pairs of geographically distant end sites. Because each end site utilizes a circuit in the same manner as a VLAN, the number of circuits that can be simultaneously active is limited. In this paper, we address the problem of maximizing the number of bandwidth reservations that can be serviced by a set of virtual circuits while minimizing the number of required virtual circuits. We develop an algorithm, called CA, that solves this problem and compare its performance with a simplified version, called S-CA, and a baseline FCFS algorithm, through both offline and online simulations. We demonstrate that CA performs up to 30% better than S-CA and up to three times better than FCFS.**

I. Introduction

In an effort to address the requirements of its science community for reliable and predictable big data transfers over the network, the U.S. Department of Energy (DOE) has funded a number of projects to research and develop services that enable user-driven bandwidth reservations (BRs). The OSCARS project [1] has developed a system for wide-area networks that allows the allocation of bandwidth in the form of dynamic virtual circuits. Two major U.S. research and education networks, ESnet [2] and Internet2 [3], have deployed the OSCARS system [4] into production, providing on-demand virtual circuit connectivity between the end-sites they interconnect, mainly research laboratories and U.S. universities. Numerous data transfers between multiple sites serviced by these networks are initiated through the requests of thousands of researchers to distribute shared datasets and analysis results such as those from the Large Hadron Collider.

The TeraPaths [5] project has developed services to bring the capabilities of this virtual circuit infrastructure to end-site users. Building on the TeraPaths service, the VNOD project [6] establishes virtual network instances comprising multiple end-to-end virtual paths to accommodate data transfers between multiple pairs of source and destination end-sites. VNOD provides predictable data transfers in two steps. First, the system generates an optimized set of bandwidth reservations (BRs) for data transfer requests submitted by users. Within each end site, these BRs are enforced using DiffServ [7].

*The author was with the Computational Science Center at Brookhaven National Laboratory when this work was performed.

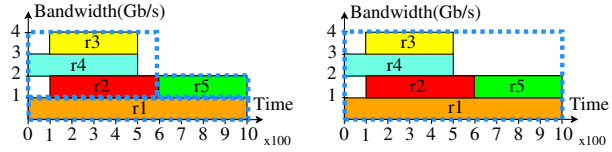


Fig. 1. An example of two different schedules.

The corresponding data flows are regulated and prioritized between source/destination hosts and each site's border router. Next, the system acquires one or more WAN virtual circuits to interconnect the border routers with guaranteed-bandwidth virtual links. Each border router is configured to allocate a specific VLAN for the circuit and to forward the data flows into the circuit using Policy Based Routing (PBR). Therefore, each circuit requires one VLAN identifier (VLAN ID) at each end site.

Several efforts [8], [9], [10], [11] have been pursued to solve the problem of mapping user requests onto BRs. However, none of them considers the problem of allocating virtual WAN circuits for the generated BRs. In this paper, we focus on this critical problem, which is complex because of two main reasons. First, due to hardware and policy restrictions, the number of VLAN IDs that can be used in practice at each end site is limited to much smaller numbers than the 4096 VLAN IDs provided for by the 802.1Q standard. For example, for the TeraPaths project at Brookhaven National Laboratory only 50 VLAN IDs were allowed for such virtual circuit utilization, while in collaborating sites as low as 3. Since each virtual circuit needs one VLAN ID at each end site, such restrictions also greatly limit the number of simultaneously active virtual circuits. Therefore, circuit sharing has to be used to increase the number of requests that can be serviced. With circuit sharing, multiple BRs with common source and destination can be mapped onto the same circuit while each of them maintains its own bandwidth guarantees. However, circuit sharing also causes bandwidth wastage, which can prevent the accommodation of other BRs. The main issue here is to strike a balance between the overall number of circuits and bandwidth utilization, while maximizing the number of accommodated BRs. For example, Fig. 1 shows two different schedules of five BRs. In Fig. 1(a), three circuits (blue dashed line rectangles) are used to service the five reservations, while only one circuit is used in Fig. 1(b) but has larger bandwidth wastage (the white area within the blue dashed line rectangle). Second, the

availability of bandwidth and VLAN IDs varies with time. This variation is caused by previously allocated circuits which reserve a portion of the total bandwidth and use up some VLAN IDs during their lifetime. For the previous example, if there is only one VLAN ID available from times 0 to 1000, then the schedule in Fig. 1(a) is infeasible. On the other hand, if the available bandwidth is only 2 Gb/s from times 600 to 1000, then the schedule in Fig. 1(b) is infeasible.

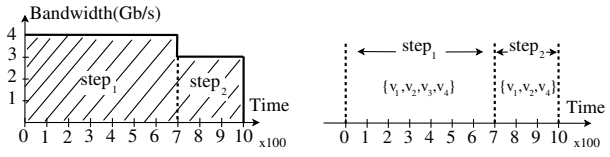
In this paper, we study the problem of allocating circuits for a set of BRs between the same pair of end sites and formulate an objective function to evaluate potential solutions. We develop a Circuit Allocation (CA) algorithm to solve this problem. The algorithm incorporates several problem-specific techniques, such as *Circuit-to-VLAN map*, *Multiple Circuits Consolidation*, and *Single Circuit Allocation*. We study the performance and effectiveness of the CA algorithm through both offline and online simulations.

In section II we describe related work, while in section III we formally define the problem. In section IV we present the CA algorithm and analyze its runtime complexity, then in section V we study the performance of our algorithms through simulations. Finally, in section VI we present our conclusions.

II. Related Work

The problem of scheduling advance and flexible bandwidth reservation requests has been well studied [8], [9], [10], [11]. Sharma *et al.* [8] presents a bandwidth scheduling algorithm, called RRA, to accommodate multiple flexible data transfer requests between a single pair of end sites. The objective of RRA is to generate a set of feasible BRs that maximize the number of accommodated requests. Sharma *et al.* [9] continues the work of [8] by developing an algorithm, called RRPC, to calculate a routing solution as well as accommodate flexible requests. Balman *et al.* [10] develops an algorithm to solve the problem of scheduling a single flexible reservation. Lin *et al.* [11] considers four types of advance bandwidth scheduling problems. However, none of these works considers optimizing the assignment of multiple BRs to virtual circuits, which is necessary for overcoming scalability limitations when implementing BRs in practice.

Problems similar to the one under consideration here can be found in other fields such as operations research. The personnel task scheduling problem (PTSP) has been studied in multiple papers [12], [13], [14]. In this problem, multiple tasks with fixed start and end time need to be assigned to the available personnel. Each task can only be assigned to a specific subset of the personnel. Our problem is similar to the PTSP problem in the view that every BR is a task and every VLAN ID is a worker. In our problem, however, multiple tasks can be merged into one task, and the maximum number of simultaneously active tasks (BRs) is restricted not only by the number of available workers (VLAN IDs) but also by the time-varying availability of bandwidth. These critical differences make our problem more complex than the PTSP problem. The two-dimensional bin packing problem has also been well studied [15]. While this problem seems similar to ours if a circuit reservation is considered a bin and each BR an item, the variation of bandwidth and VLAN ID availability with time makes our problem quite different.



(a) BAG_{in} (shaded area) (b) $VIAG_{in}$

Fig. 2. An example of BAG_{in} and $VIAG_{in}$.

III. Problem Description

We consider two end sites interconnected with a given route through a WAN. The system has received a set of data transfer requests between these end sites, and has generated a set of BRs. Given the limitations on bandwidth and VLAN ID availability, our goal is to map these BRs on a set of virtual circuit reservations (CRs) that maximizes a certain objective function. We can formally describe the input, output, and objective function of the algorithm as follows:

Input: The input to our algorithm has three components. The first component is a set of BRs, represented as set $\mathcal{R}$. The BR r_i in $\mathcal{R}$ is defined as $\{st_i, et_i, bw_i\}$, where st_i is the beginning time; et_i is the end time, and bw_i is the amount of bandwidth to be reserved between st_i and et_i. As an example, the five BRs shown in Fig. 1 can be represented as: {0, 1000, 1 Gb/s}, {100, 600, 1 Gb/s}, {100, 500, 1 Gb/s}, {0, 500, 1 Gb/s}, and {600, 1000, 1 Gb/s}.

The second component is the bandwidth availability graph (BAG) that describes the time-varying availability of bandwidth represented as BAG_{in}. The BAG_{in} is a step function with multiple steps $step_i$ defined as $\{start_i, end_i, maxBW_i\}$, where $start_i$ is the start time of $step_i$, end_i its end time, and $maxBW_i$ the available bandwidth between $start_i$ and end_i. Fig. 2(a) shows an example of BAG_{in} with two steps: {0, 700, 4 Gb/s} and {700, 1000, 3 Gb/s}.

The last component is the VLAN ID availability graph (VIAG) that describes the availability of VLAN IDs denoted as $VIAG_{in}$. The $VIAG_{in}$ also contains a series of steps $step_i$ defined as $\{start_i, end_i, \mathcal{T}_i\}$, where $start_i$ is the start time of $step_i$, end_i its end time, and $\mathcal{T}_i$ the set of available VLAN IDs between $start_i$ and end_i. Fig. 2(b) shows an example of $VIAG_{in}$ with two steps: {0, 700, $\{v_1,v_2,v_3,v_4\}$} and {700, 1000, $\{v_1,v_2,v_4\}$}. Based on the operational requirements of the mechanisms that establish the virtual circuits and forward traffic into them at the end sites, we assume that the VLAN IDs at the two end sites are used in a manner of fixed one-to-one mapping. As a result, we can represent the availability of VLAN IDs for the circuit allocation with the VIAG of one of the end sites.

Objective: Our goal is to generate a set of virtual CRs that maximizes an objective function incorporating three criteria: first, it should help maximize the number of BRs that are successfully assigned to circuits; second, it should help minimize wasted bandwidth caused by mapping multiple BRs onto the same circuit; finally, it should help minimize the total number of required CRs. For a given set of BRs, an objective function that balances all three criteria can be written as follows:

$$\text{Maximize} \quad \alpha\frac{r}{|\mathcal{R}|} - \beta\frac{c}{|\mathcal{R}|} - \gamma w \tag{1}$$

Variable r is the number of BRs that got assigned to a CR, and

variable c is the number of CRs allocated to accommodate the set of given BRs. $|\mathcal{R}|$ represents the total number of input BRs. It is used to normalize r and c. Because the maximum value of r and c equals to $|\mathcal{R}|$, the value of $\frac{r}{|\mathcal{R}|}$ and $\frac{c}{|\mathcal{R}|}$ is in the range of $[0, 1]$. The variable w represents bandwidth wastage and can be calculated as:

$$w = \frac{D_T - D_N}{D_T} \tag{2}$$

in which D_T is the total amount of data that could be transferred with the reserved bandwidth and D_N is the amount of data that actually needs to be transferred, corresponding to the BRs that were assigned to circuits. As an example, the value of w for the circuit shown in Fig. 1(b) (blue dashed line rectangle) is $\frac{3}{10}$. Observe that the value of w is also in the range of $[0, 1]$. The exponents α, β and γ represent the weight of their corresponding variable in the objective function. Since maximizing r is the most important goal in our problem, we usually set α to be larger than β and γ.

Output: The output of the algorithm is a set of CRs. We denote this set by $\mathcal{C}$. Each circuit reservation c_i in $\mathcal{C}$ is defined as $\{t_{a_i}, t_{d_i}, reservedBW_i, vlanId_i, \mathcal{R}_{c_i}\}$, where t_{a_i} is the activation time of c_i; t_{d_i} is the deactivation time; $reservedBW_i$ is the total amount of bandwidth reserved by c_i; $vlanId_i$ is the VLAN ID assigned to this CR; and $\mathcal{R}_{c_i}$ is the set of BRs sharing c_i.

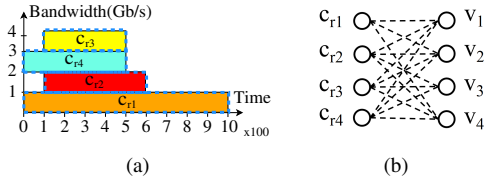
In order to be feasible, the set of CRs should satisfy the following constraints: (i) Any overlapping circuits should not use the same VLAN ID; (ii) At any time t, the VLAN IDs used by all active circuits should belong to the set of available VLAN IDs; (iii) At any time t, the total amount of bandwidth reserved by all active circuits should not exceed the amount of available bandwidth. For example, the three circuits shown in Fig. 1(a) can be presented as: {0, 600, 3 Gb/s, v_1, {r_2, r_3, r_4}}, {0, 1000, 1 Gb/s, v_2, {r_1}}, and {600, 1000, 1 Gb/s, v_1, {r_5}}. Note that a certain VLAN ID can be used in different circuits as long as those circuits are not overlapping in time.

IV. The Circuit Allocation (CA) Algorithm

A. Algorithm Description

We develop an algorithm, called CA, to solve the problem described in the previous section. The CA algorithm begins with the set $\mathcal{R}$, and runs in iterations. During each iteration, the algorithm allocates virtual circuits to a subset of $\mathcal{R}$. Each iteration consists of four phases, as follows:

1) Phase I: Identify Maximum Overlapped Subset: In this phase, the CA algorithm identifies subset $\mathcal{M} \subseteq \mathcal{R}$ that has the largest number of overlapping BRs. For each $r_i \in \mathcal{M}$, a CR, denoted by c_{r_i}, is created. It has the same start time and end time with r_i. Its reserved bandwidth is equal to bw_i of r_i, and only r_i is mapped to it. The VLAN ID for this CR, however, is unknown for now. Set $\mathcal{C}_\mathcal{M}$ contains all CRs created in this manner. For the five BRs shown in Fig. 1(a), Fig. 3(a) shows the maximum overlapped subset $\mathcal{M}$ identified by the CA algorithm in the first iteration. Four CRs (shown as blue dashed line rectangles) are created for set $\mathcal{M}$. Note that the CA algorithm removes all BRs in $\mathcal{M}$ from $\mathcal{R}$ at the end of current iteration. As a result, a new subset $\mathcal{M}$ is identified in the next iteration.



(a) (b)

Fig. 3. An Example of: (a) Maximum overlapped subset, (b) CtoV map.

2) Phase II: Construct Circuit-to-VLAN (CtoV) Map: To be able to actually allocate the circuits corresponding to the CRs in set $\mathcal{C}_\mathcal{M}$, we first need to assign VLAN IDs to the CRs. It is feasible to assign a VLAN ID v_k to a CR c_i if and only if v_k is always available during the active period of c_i. We devised the concept of *Circuit-to-VLAN (CtoV) map* to express the relationship between circuits and VLAN IDs. The CtoV map contains two columns of nodes. If $\mathcal{V}$ is the set containing all available VLAN IDs, each node in the left column represents a circuit corresponding to a CR in $\mathcal{C}_\mathcal{M}$, and each node in the right column represents a VLAN ID in $\mathcal{V}$. Each circuit c_i is connected with dashed lines to the nodes of VLAN IDs that can be assigned to it. Fig. 3(b) shows the CtoV map of the subset in Fig. 3(a). In Fig. 3(b), there are 4 nodes in each side, corresponding to circuits and VLAN IDs. Circuits c_{r_2}, c_{r_3} and c_{r_4} are connected to all VLAN IDs, while c_{r_1} is only connected to v_1, v_2 and v_4.

3) Phase III: Allocate Circuits: In this phase, the goal is to generate a set of implementable CRs (with complete information) to accommodate the BRs in $\mathcal{M}$. In this phase, there are two steps: the *multiple circuits consolidation (MCC)* step and the *single circuit allocation (SCA)* step. Fig. 4 and Fig. 7 shows the pseudocode of the two steps respectively.

Step 1: Multiple Circuits Consolidation (MCC) step. Before introducing the MCC step, we need to clarify two concepts. The first concept is the merging of multiple CRs into a new CR. The time duration of this new CR is the time duration for which at least one of the original CRs is supposed to be active, while its reserved bandwidth is equal to the total bandwidth reserved by all original CRs. The merged CR can accommodate all the requests that are assigned to the original CRs. The second concept is the bandwidth wastage occurring when merging two or more CRs. We express this with the *wastage coefficient*, which is the value of variable w the new CR gets due to the merging of the original CRs. We use Eq. 2 to calculate this value, only taking into account the new circuit and the BRs accommodated by it. For example, the wastage coefficient of c_{r_1} and c_{r_2} is $\frac{5}{20}$.

The MCC step runs in iterations. Each iteration begins by selecting from the current set $\mathcal{C}_\mathcal{M}$ a pair of CRs for which the wastage coefficient is minimum (ties are broken randomly). Assume that the two CRs in such a pair are c_i and c_j. The CA algorithm then finds all the VLAN IDs that can be assigned to both c_i and c_j and adds them to a set, denoted by $\mathcal{V}_{common}$. For each $v_k \in \mathcal{V}_{common}$, we calculate a new objective value corresponding to the merged CR and assign ID v_k to it. c_{i+j} could now be allocated. Because an allocation uses up one of the total available circuits and also increases the number of accommodated BRs, the value of the objective function can change. Assigning a different VLAN ID in $\mathcal{V}_{common}$ to c_{i+j} may cause the objective function to take a different value. This is because some VLAN IDs in $\mathcal{V}_{common}$ may be already

MCC step ($\mathcal{C}_m$, CtoV map, $Object$, BAG_{in}, $VIAG_{in}$)

1. **while** $true$ **do**
2. **for** all $Pair(c_i, c_j)$, $c_i, c_j \in \mathcal{C}_m$ **do**
3. $waste_{ij} \leftarrow$ waste coefficient of $Pair(c_i, c_j)$.
4. Add $Pair(c_i, c_j)$ to List $\mathcal{P}$.
5. **end for**
6. **while** List $\mathcal{P} \neq \{\}$ **do**
7. $Pair(c_i, c_j) \leftarrow argmin_{pair \in \mathcal{P}}\{waste_{ij}\}$
8. Identify $\mathcal{V}_{common}$ for $Pair(c_i, c_j)$.
9. **for** all $v_k \in \mathcal{V}_{common}$ **do**
10. **if** v_k is not assigned to other circuit **then**
11. $O^k_{ij} \leftarrow$ new objective value based on merging c_i and c_j.
12. **else** /*Suppose v_k is assigned to c_o*/
13. $O^k_{ij} \leftarrow$ new objective value based on merging c_i, c_j and c_o.
14. **end for**
15. $v_t \leftarrow argmax_{v_k \in \mathcal{V}_{common}}\{O^k_{ij} | O^k_{ij} > Object\}$
16. **if** $v_t == null$ **then**
17. Remove $Pair(c_i, c_j)$ from List $\mathcal{P}$. **continue**
18. **else**
19. Merge c_o(if exists), c_i and c_j to c_{new}.
20. Assign v_t to c_{new}, and add c_{new} into $\mathcal{C}_m$.
21. Remove c_o (if exists), c_i and c_j from $\mathcal{C}_m$.
22. Update CtoV map and BAG_{in}.
23. $Object \leftarrow O^t_{ij}$. **break**.
24. **end while**
25. **if** List $\mathcal{P} == \{\}$ **then break**.
26. **end while**
27. Add all CRs with VLAN ID to Set $\mathcal{C}$.
28. Remove all CRs with VLAN ID from $\mathcal{C}_m$.

Fig. 4. Pseudocode for the MCC step.

assigned to some other CR c_o in a previous iteration. If we assign an already assigned VLAN ID to CR c_{i+j}, we then need to merge c_i, c_j and c_o together because a single VLAN ID cannot be assigned to two overlapping circuits. In such cases, we accommodate more BRs but don't use up yet another circuit out of the total number available. After calculating the new objective value for each $v_k \in \mathcal{V}_{common}$, we merge the selected CRs and assign to the resulting CR a VLAN ID v_t when the following three feasibility conditions are satisfied:

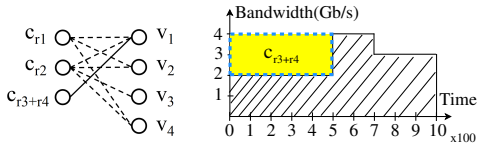
(i) The available bandwidth allows the merge.

(ii) The objective value improves after the merge.

(iii) Compared with assigning other VLAN IDs in $\mathcal{V}_{common}$ to the new CR, assigning v_t improves the objective value the most.

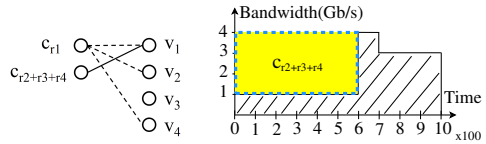
If the two selected CRs cannot be merged, the CA algorithm does not consider this pair anymore and moves to the next pair with the minimum wastage coefficient. If the two CRs c_i and c_j are merged and assigned a VLAN ID v_t, then they are removed from the set $\mathcal{C}_m$, and the new CR c_{i+j} is added to $\mathcal{C}_m$. Subsequently, the CA algorithm updates the CtoV map. It merges nodes c_i and c_j into a new node c_{i+j}, and connects the new node only to node v_t by using a solid line (see, for example Fig. 5(a)). Since c_{i+j} already uses VLAN ID v_t, its node is connected only to the node for v_t. Finally, the CA algorithm updates the BAG_{in} and starts the next iteration.

As an example, consider the CRs in Fig. 3(a). The MCC step begins by selecting the pair (c_{r_3}, c_{r_4}). A new CR $c_{r_3+r_4}$ ({0, 500, 2 Gb/s, {}, {r_3, r_4}}) is generated by merging c_{r_3} and c_{r_4}. By checking the current BAG_{in}, the algorithm finds that $c_{r_3+r_4}$ can be accommodated. Next, it checks whether assigning a VLAN ID to $c_{r_3+r_4}$ improves the objective function value. The set $\mathcal{V}_{common}$ for this pair contains v_1, v_2, v_3, and v_4. For each VLAN ID in $\mathcal{V}_{common}$, a new objective value is calculated[1]. In this example, the new objective value is the same for each assignment of VLAN ID, equal to $10 \times \frac{2}{5} - \frac{1}{5} - \frac{1}{10} = 3.7$. Therefore, a VLAN ID is randomly selected from $\mathcal{V}_{common}$ and assigned to $c_{r_3+r_4}$. Assume that v_1 is selected in this example. We can see that all three feasibility conditions are satisfied, so c_{r_3} and c_{r_4} can indeed be merged and v_1 assigned to the merged CR. The CA algorithm then updates the CtoV map and the BAG_{in}. Fig. 5(a) shows the updated CtoV map. In Fig. 5(b), the shaded area represents the updated BAG_{in} and the yellow rectangle is the allocated CR. In the second iteration, the CA algorithm merges the pair $(c_{r_2}, c_{r_3+r_4})$. and assigns v_1 to the new CR. The objective function value increases to 5.58. The updated CtoV map and BAG_{in} are shown in Fig. 6(a) and Fig. 6(b) respectively.



(a) CtoV map. (b) BAG_{in} (shaded area).

Fig. 5. CtoV Map and BAG_{in} after the 1st iteration of MCC step.



(a) CtoV map. (b) BAG_{in} (shaded area).

Fig. 6. CtoV map and BAG_{in} after the 2nd iteration of MCC step.

If in an iteration the objective value does not improve because it is not feasible to merge any pair of CRs, the MCC step finishes and the CA algorithm moves to the next step, SCA. For the previous example, in the third iteration, the only CR pair left (c_{r_1}, $c_{r_2+r_3+r_4}$) cannot be merged because of insufficient bandwidth availability, which causes MCC to finish.

Step 2: Single Circuit Allocation (SCA) step. In this step, we further improve the objective value by individually allocating the CRs still in $\mathcal{C}_{\mathcal{M}}$. The SCA step begins by removing all the nodes that are connected by a solid line from the CtoV map. Fig. 8(a) shows an example of the updated CtoV map.

The SCA step then runs in iterations. In each iteration, SCA identifies every feasible single CR allocation. An allocation needs to satisfy the following three feasibility conditions: (i) The available bandwidth allows the allocation, (ii) The targeted VLAN ID can be assigned to the CR of interest, (iii) The objective value improves after the allocation. Subsequently, the CA algorithm performs the allocation that causes the largest improvement to the objective function value, breaking ties randomly. After updating the CtoV map and BAG_{in}, SCA then starts a new iteration. If no single CR allocation can be made in some iteration, SCA stops. For the previous example, the SCA step starts with the updated CtoV map shown in Fig. 8(a). In the first iteration, SCA identifies two feasible

[1]In all examples the objective value is calculated using $\alpha = 10$ and $\beta = \gamma = 1$.

SCA Algorithm ($\mathcal{C}_m$, CtoV map, $Object$, BAG_{in}, $VIAG_{in}$)

1. Update CtoV map.
2. **while** $true$ **do**
3. **for** all $c_i \in \mathcal{C}_m$, all v_k that is feasible to c_i **do**
4. $O_i^k \leftarrow$ new objective value after allocating c_i with VLAN ID v_k.
5. **end for**
6. $(c_p, v_t) \leftarrow argmax_{c_i \in \mathcal{C}_m, v_k \in \mathcal{V}}\{O_i^k | O_i^k > Object\}$
7. **if** $(c_p, v_t) == null$ **then break**
8. **else**
9. Assign v_t to c_p, and update $Object$ to O_p^t.
10. Remove c_p from $\mathcal{C}_m$, and add it to $\mathcal{C}$.
11. Update CtoV map and BAG_{in}.
12. **end while**
13. **Return** $\mathcal{C}$.

Fig. 7. Pseudocode for the SCA step.

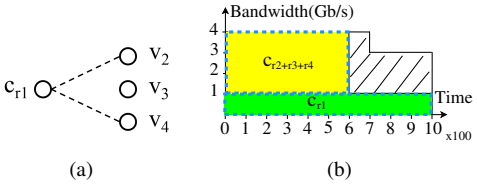


Fig. 8. (a) Updated CtoV map, (b) Circuits allocated after running SCA step.

single CR allocations: (c_{r_1},v_2) and (c_{r_1},v_4), which have the same new objective value of 7.46. SCA randomly selects (c_{r_1},v_2) and then updates the CtoV map and BAG_{in}. In the next iteration, since there are no more circuit nodes in the CtoV map, no allocation can be performed. Therefore, the SCA step stops. Fig. 8(b) shows the allocated CRs after running SCA. The shaded area represents the updated BAG_{in}. After completing the SCA step, the CA algorithm starts its last phase.

4) Phase IV: Update Information: In this phase, the CA algorithm removes all BRs in $\mathcal{M}$ from $\mathcal{R}$ and adds all allocated CRs to set $\mathcal{C}$, as part of the output. The CA algorithm then starts the next iteration. The initial objective value in the next iteration is the current objective value.

5) Termination of the CA Algorithm: In each iteration of CA, a subset of BRs is removed from set $\mathcal{R}$. If set $\mathcal{R}$ becomes empty at the end of an iteration, the CA algorithm terminates. The current set $\mathcal{C}$ contains the final output CRs which will be submitted for activation. For the previous example, the three CRs shown in Fig. 1(a) are the final output.

B. Runtime Complexity

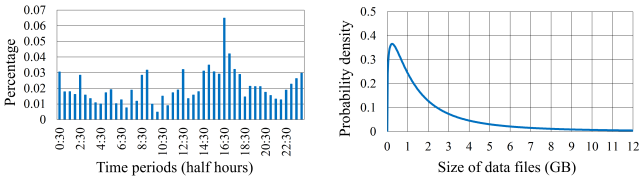
Assume that the number of BRs in $\mathcal{R}$ is N, the number of maximum overlapped subsets is M, and the number of BRs in each subset is n_i. Also assume that the number of steps in BAG_{in} and $VIAG_{in}$ is S and the number of available VLAN IDs is V. The total complexity of the CA algorithm can be written as:

$$O\Big(MNlog(N) + NSV + MVN_{\max}^2 + MN_{\max}^3\Big) \quad (3)$$

where

$$N_{\max} = \max(n_i), i \in [1, M] \quad (4)$$

Note that, in practice, the value of S (number of steps in a BAG) and V (number of available VLAN IDs) is usually much smaller than $N_{\max}$. As a result, $MN_{\max}^3$ usually dominates the complexity of the CA algorithm. We omit the procedure of arriving at Eq. 3 and Eq. 4 due to space limitations.



(a) Histogram of the start time of data transfer requests. (b) PDF of beta prime distribution.

Fig. 9. Distributions used to generate data transfer requests as input.

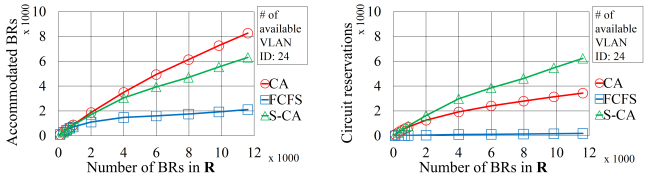
V. Performance Evaluation

A. Simulation Setup

We evaluate the CA algorithm through simulating the complete scheduling procedure, which has two steps. In the first step, we utilized the RRA algorithm [8] introduced in section II to map user requests onto BRs. Next, we run the CA algorithm to map the BRs onto CRs.

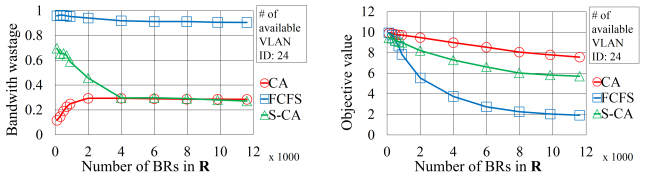
We generate the user requests based on statistics of file transfer requests between the High Performance Storage System (HPSS) of National Energy Research Scientific Computing (NERSC) Center [16] and other locations, like the National Center for Atmospheric Research (NCAR). The time period is from 2010 to 2012. Fig. 9(a) shows the histogram of start time of each request in the scale of 24 hours. In the simulations, we use this statistical data as probability distribution to generate the start time of each request. We also observe that most of the requests have a transferred file size between 100 MB and 500 MB, while the maximum file size is 12 GB. We believe that this observation indicates a significant trend that transfers of smaller files are much more common, at least in this category of scientific data transfers. Based on these observations, we decided to use in our simulations the beta prime distribution with $\alpha = 1.3$ and $\beta = 2$, which is strongly positively skewed, to approximate the real distribution of the file size of each request. Fig. 9(b) shows the probability density function (PDF) of the beta prime distribution. The rate limitation of each request, i.e., the maximum bandwidth can be reserved for this request, is randomly selected from 0 to 400 Mb/s. Furthermore, we also use the normal distribution and the uniform distribution to generate the file size in our simulations. The normal distribution has $\mu = 6$ GB and $\sigma = 2$ GB. The uniform distribution has a uniform density of $\frac{1}{12}$ in the range of [0, 12 GB]. In the simulation results, each data point represents an average value of 50 runs.

We develop two algorithms to compare with the CA algorithm. The first algorithm, called Simple-CA (S-CA), is a simplified version of the CA algorithm. S-CA does not use other strategies developed in the CA algorithm, except the concept of wastage coefficient. The S-CA algorithm processes the BRs one by one in the increasing order of their start time. In each iteration, S-CA selects a BR and tries to merge it with a previously allocated CR that has the minimum wastage coefficient with the selected request. The second algorithm, called FCFS, is a simple heuristic that simulates a mechanism potentially used by a scheduling system without any optimization strategy for circuit allocation. The FCFS algorithm also processes the BRs one by one in increasing order of their start time. In each iteration, however, FCFS simply tries to merge the selected BR into the CR allocated in the previous



(a) Accommodated BRs.

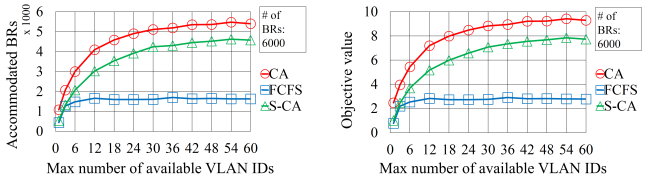
(b) Circuit reservations.



(c) Bandwidth wastage.

(d) Objective value.

Fig. 10. Performance of CA for different number of BRs.



(a) Accommodated BRs.

(b) Objective value.

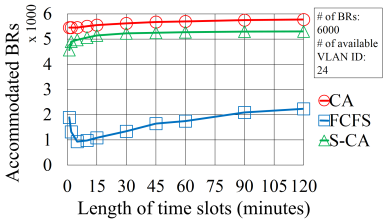
Fig. 11. Performance of CA for different number of available VLAN IDs.

iteration. If such merge is invalid, FCFS then allocates a new CR for the selected BR. By comparing with S-CA and FCFS, we demonstrate the effectiveness of the optimization strategies used by the CA algorithm. Note that we attempted to use the well-known heuristics [13], [15]. However, due to the complexity of our problem, we cannot apply these heuristics without significant modifications. Therefore, comparisons between CA and these heuristics might be considered unconvincing.

B. Simulations based on Statistical Data

Based on the method of generating input requests from statistical data, we perform three sets of simulations. In these simulations, the CA, S-CA, and FCFS algorithms run after running the RRA algorithm first.

In the first simulation, we study how the CA algorithm performs as the number of requests increases from 100 to 12,000. A multi-step BAG and a VIAG are randomly generated. For every step in the BAG, the value of available bandwidth is randomly selected between 0 and 10 Gb/s. The maximum number of available VLAN IDs in each step of VIAG is 24. Fig. 10(a) shows the number of accommodated BRs. The CA algorithm performs nearly four times better than FCFS and 35% better than S-CA. Fig. 10(b) shows the number of circuit reservations created by each algorithm. The CA algorithm creates nearly 45% less circuit reservations than that of S-CA, but more than that of FCFS. Fig. 10(c) shows the bandwidth wastage of each algorithm. The FCFS algorithm has the largest bandwidth wastage, followed by the S-CA and CA algorithms. From Fig. 10(b) and Fig. 10(c), we observe that the CA algorithm achieves good balance between c and w and accommodates the largest number of BRs. Fig. 10(d) shows the objective value of each algorithm. The CA algorithm has the largest objective value, followed by the S-CA and FCFS



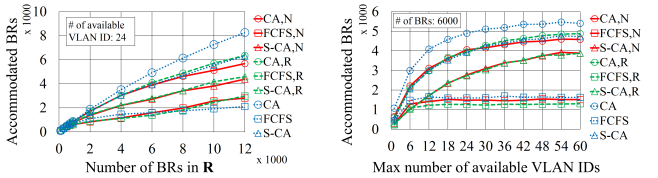
(a) Accommodated BRs.

Fig. 12. Performance of CA in online simulations.

algorithms. As far as the running time is concerned, when the number of BRs in $\mathcal{R}$ is 12000, all three algorithms finish a single run in less than 40 seconds. All the simulations were performed on a 2.7 GHz processor.

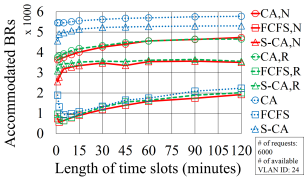
In the second simulation, we demonstrate the performance of the CA algorithm when the maximum number of available VLAN IDs increases from 1 to 60. Such an increase also increases the number of circuits that can be simultaneously activated. We present the results obtained for 6000 BRs. A BAG and a VIAG are randomly generated as in the first simulation. Fig. 11(a) shows the number of accommodated requests. The CA algorithm accommodates 20% more BRs than S-CA and nearly 230% more than FCFS. Simulation results (which we have to omit due to space limitations) also show that S-CA allocates the largest number of circuit reservations and FCFS has the largest bandwidth wastage. This is similar to the results of the first simulation. Finally, Fig. 11(b) shows that the objective value of CA is nearly 20% superior to that of S-CA and nearly 230% superior to that of FCFS. For the running time, when the maximum number of available VLAN IDs is 60, all three algorithms finish a single run in less than 30 seconds. All the simulations were, again, performed on a 2.7 GHz processor.

Finally, we study the performance of the CA algorithm through a set of online simulations, in which the time axis is divided into multiple time slots with length Δt. Essentially, we simulate a real-world scenario where the scheduling system periodically runs the scheduling algorithms to address the data transfer requests received in the latest period. A submission time is randomly generated for each request. All requests whose submission time is in the same time slot are addressed together at the end of that time slot, except for those requests whose start time is within that time slot. The system addresses the latter type of requests immediately and individually. In the simulations, a set of 6000 BRs is generated as input. In each simulation, at any time t, the initial available bandwidth is 10 Gb/s and the initial number of available VLAN IDs is 24. This simulates the initial state in which no circuit reservation has been allocated. Fig. 12(a) shows the number of accommodated BRs by each algorithm at the end of an online simulation. In online simulations, we treat the number of accommodated BRs as the most important metric. We can see that the CA algorithm accommodates nearly 20% more requests than S-CA and nearly 200% more than FCFS in the beginning. As Δt increases, the number of requests accommodated by CA increases slightly and then stabilizes at approximately 5800 requests. Even though the difference between the performance of CA and S-CA decreases as Δt increases, we notice that the acceptance rate of CA reaches 97% after stabilizing.



(a) Simulations with increasing number of BRs.

(b) Simulations with increasing number of available VLAN IDs.



(c) Online simulations with increasing value of Δt.

Fig. 13. Performance of CA when using the normal and uniform distributions.

C. Simulations using other probability distributions.

In this section, we demonstrate the performance of the CA algorithm when the normal and uniform distributions introduced in section V-A are used to generate the file size of each request. Due to the space limitations, we only present the number of requests accommodated by each algorithm, because this is the most important goal in our objectives. Fig. 13 shows the results of the simulations. In the graph legends, the letter "N" after the algorithm's name indicates that the normal distribution is used, and the results are plotted with red solid lines; the letter "R" indicates that the uniform distribution is used, and the results are plotted with green dashed lines. Finally, the simulation results obtained in previous sections are shown with blue dotted lines.

In Fig. 13, we can see that each algorithm generally accommodates less BRs, when the normal or random distribution is used to generate input. This is because more data transfer requests with larger file size are generated when using the normal or random distribution. Fig. 13(a) shows the results of simulation with increasing number of BRs. We can observe that the CA algorithm accommodates nearly 40% more than S-CA and nearly 100% more than FCFS, when either the normal distribution or the uniform distribution is used. Fig. 13(b) shows the results of simulation with increasing number of available VLAN IDs. The performance of CA is around 25% superior than that of S-CA and around 200% superior than that of FCFS. Fig. 13(c) shows the results of online simulations. The CA algorithm accommodates 20%-40% more requests than S-CA and 250%-400% more requests than FCFS.

VI. Conclusion

In this paper, we studied the problem of optimizing the number of bandwidth reservations that can be serviced by a set of virtual circuits. We formulated an objective function, and proposed an algorithm called CA to solve the problem by maximizing the objective function. We studied the performance of the CA algorithm with both offline and online simulations. In the simulations, we generated the input based on the statistic data of the NERSC HPSS and developed two algorithms to compare with CA. One of these algorithms, called S-CA, is a simplified version of CA; the other, called FCFS, is a simple heuristic. Simulation results shows that the CA algorithm outperforms S-CA by a factor of 0.1-0.3 and FCFS by a factor of 2-4. Furthermore, when using the normal and random distributions to generate input, the CA algorithm performs up to 40% better than S-CA and up to 4 times better than FCFS.

Acknowledgements

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-98CH10886 to BNL. The authors would like to thank Alex Sim and Arie Shoshani of Lawrence Berkeley National Laboratory for providing statistical data transfer data from the NERSC HPSS Storage Resource Manager. The work from LBNL was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02- 05CH11231.

References

[1] C. Guok, D. Robertson, E. Chaniotakis, M. Thompson, W. Johnston, and B. Tierney, "A User Driven Dynamic Circuit Network Implementation" in *Proc. IEEE Distributed Autonomous Network Management Systems (DANMS),* November 2008.

[2] "Energy sciences network," http://www.es.net/

[3] "Internet 2," http://www.internet2.edu/

[4] "OSCARS: On-Demand Secure Circuits and Advance Reservation System," http://code.google.co-m/p/oscars-idc/

[5] D. Katramatos, D. Yu, K. Shroff, S. McKee, and T. Robertazzi, "TeraPaths: End-to-end network resource scheduling in high-impact network domains," *International Journal On Advances in Internet Technology,* vol 3, no. 1–2, pp. 104–117, 2010.

[6] D. Katramatos, S. Sharma, and D. Yu, "Virtual Network On Demand: Dedicating Network Resources to Distributed Scientific Workflows," in *Proc. ACM HPDC, Workshop on Data Intensive Distributed Computing (DIDC)*, Delft, Netherlands, June 18, 2012.

[7] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474, December, 1998.

[8] S. Sharma, D. Katramatos, and D. Yu, "End-to-end network QoS via scheduling of flexible resource reservation requests," In *Proc. ACM/IEEE Supercomputing Conference (SC),* Seattle, WA, November, 12–18, 2011.

[9] S. Sharma, D. Katramatos, D. Yu, and L. Shi "Design and implementation of an intelligent end-to-end network QoS system," In *Proc. ACM/IEEE Supercomputing Conference (SC),* Salt Lake City, UT, November 10–16, 2012.

[10] M. Balman, E. Chaniotakis, A. Shoshani, and A. Sim, "A flexible reservation algorithm for advance network provisioning," In *Proc. ACM/IEEE Supercomputing Conference (SC),* Seattle, WA, November 12–18, 2011.

[11] Y. Lin, and Q. Wu, "Complexity analysis and algorithm design for advance bandwidth scheduling in dedicated networks," *IEEE/ACM Transcations on Networking,* vol. 21, no. 1, pp. 14–27, 2013.

[12] E. Arkin, and E. Silverberg, "Scheduling jobs with fixed start and end times," *Discrete Applied Mathematics,* vol. 18, no. 1, pp. 1–8, 1987.

[13] M. Krishnamoorthy, A. Ernst, and D. Baatar, "Algorithms for large scale shift minimisation personnel task scheduling problems," *European Journal of Operational Research,* vol. 219, no. 1, pp 34–48, 2012.

[14] P. Brucker, R. Qu, and E. Burke, "Personnel scheduling: Models and complexity," *European Journal of Operational Research,* vol. 210, no. 3, pp. 467–473, 2011.

[15] A. Lodi, S. Martello, and M. Monaci, "Two-dimensional packing problems: A survey," *European Journal of Operational Research,*, vol. 141, no. 2, pp. 241–252, 2002.

[16] A. Shoshani, A. Sim, and J. Gu. "Storage resource managers: Middleware components for grid storage," in *Proc. IEEE Symposium on Mass Storage Systems,* College Park, MA, 2002.