Bandwidth Allocation for Iterative Data-dependent e-Science Applications

Eun-Sung Jung, Sanjay Ranka and Sartaj Sahni
Computer and Information Science and Engineering Department
University of Florida, Gainesville, FL 32611
Email: {ejung,ranka,sahni}@cise.ufl.edu

***Abstract*—We develop a novel framework for supporting e-Science applications that require streaming of information between sites. Using a Synchronous Dataflow (SDF) model, our framework incorporates the communication times inherent in large scale distributed applications, and can be used to formulate the bandwidth allocation problem with throughput constraints as a multi-commodity linear programming problem. Our algorithms determine how much bandwidth is allocated to each edge while satisfying temporal constraints on collaborative tasks. Simulation results show that the bandwidth allocation by the formulated linear programming outperforms the bandwidth allocation by simple heuristics.**

***Keywords*-Bandwidth allocation; Throughput; Iterative data-dependent; E-Science**

I. Introduction

Advances in optical communication and networking technologies, together with computing and storage technologies, are dramatically changing the ways scientific research is conducted. A new term, *e-Science*, has emerged to describe the "large-scale science carried out through distributed global collaborations enabled by networks, requiring access to very large-scale data collections, computing resources, and high-performance visualization" [31]. Well-quoted e-Science (and the related grid computing [11]) examples include high-energy nuclear physics [3], radio astronomy [19], geoscience [8], and climate studies [7]. To support e-Science activities, a new generation of high-speed research and education networks has been developed. These include Internet2 [17], the Department of Energy's ESnet [9], National Lambda Rail [25], CA*net4 [4] in Canada, and the pan-Europe GÉANT2 [12].

E-Science activities often requires the transport of large volumes of data at very high rates among a large number of collaborating sites [26], [3], severely stressing network resources. For instance, the high-energy physics (HEP) data are expected to grow from the current petabytes (10^{15}) to exabytes (10^{18}) by 2015 [3]. Beyond the obvious need for large amounts of data to be transferred, e-Science requirements for network use are significantly different from the traditional network applications [20], [10], [18]. The underlying applications require schedulable, high-bandwidth, low-latency connectivity with known future characteristics or performance guarantees [6] for real-time remote visualization, interactions with instruments, distributed simulation or data analysis, and so on. In a distributed workflow system that involves many entities, such as distant parties, scientific instruments, computation devices, as well as complex feedback in various stages of the workflow, unintended delays due to a lack of planning for future communication paths can ripple through the entire workflow environment, slowing down other participating systems as they wait for intermediate results, thus reducing the overall effectiveness [18].

The focus of this paper is on supporting e-Science applications that require streaming of information between sites. We present a framework for bandwidth scheduling of streaming e-Science applications. These applications include interactive visualization of simulations, large data streaming coordinated with job execution for producer consumer applications, and networked supercomputing [10]. The main contributions of the paper are as follows:

1) We have adapted the Synchronous Dataflow (SDF) model to model and analyze iterative data-dependent applications in e-Science. Synchronous dataflow was proposed in late 1980s as a modeling method for digital signal processing (DSP) applications, but it ignores the communication times. Our model incorporates the communication times that are inherent in large-scale distributed applications.
2) We have formulated the bandwidth allocation problem of iterative data-dependent e-Science applications with temporal constraints as a multi-commodity linear programming problem. It incorporates optimal rates and buffer minimization for streaming applications that can be represented by a SDFG. Our algorithms determine how much bandwidth is allocated to each edge while satisfying temporal constraints on collaborative tasks. Using the solution of the bandwidth allocation problem, buffer requirements for the schedule is achieved using procedures similar to the ones presented in [14].

To the best of our knowledge, this represents the first attempt to analyze the temporal behavior of collaboratively iterative tasks and to determine the bandwidth allocations among distributed nodes.

Ongoing research projects for supporting e-Science applications (e.g., HOPI [16], Ultralight [32], Teragrid [30], CHEETAH [5], DRAGON [23], ESnet [9], and OS-

CARS/BRUW [27]) have mainly focused on setting up a fast data plane with a large footprint and sufficient connectivity and setting up a basic but functional control plane, such as developing signaling and control middleware for handling user requests for elementary network services, ensuring security and improving reliability. However, the control plane mechanisms lack sophisticated network service support or efficient service reservation algorithms. They normally only support fixed bandwidth guarantee by reserving circuits or lightpaths. Using such a restricted set of services or simplistic resource management algorithms to support diverse e-Science activities can lead to inefficient utilization of the network resources (especially for the new-generation, high-speed, coarse-granular networks, such as the wavelength-based systems) and/or not provide the level of performance required by those activities in desired but varied performance metrics.

The rest of the paper is organized as follows. We provide a detailed description of SDF and its operational semantics and examine its applicability to e-Science applications in Section II. We present an overall process of problem-solving, including a mathematical formulation as a linear programming and a discussion of the detailed deployment of the obtained solution for the linear programming in real systems in Section III. We show that our approach outperforms a naive heuristic, also given by us, in Section IV. Lastly, we conclude with a summary and discussion of the practicality of our proposal in Section V.

II. Synchronous Dataflow for e-Science Applications

The SDF model of computation was first proposed by Lee in [22]. The SDF model has been found to be very useful for expressing DSP applications that have the following features: infinitely looping execution, discretized communication expressed by tokens, and parallelism to be exploited for maximizing throughput. Most of the existing research for these problems is limited to deriving maximal rates and buffer minimization.

SDFG is a directed graph defined by $G = (V, E, I, O, \tau, \Phi)$, where V and E represent a set of nodes and a set of edges, respectively. Each node in SDFG is called an *actor* and the edge in SDFG is called a *communication channel* or *channel*. The notation is based on its earlier use in DSP applications comprising function blocks and the communication channels interconnecting them. An *actor* repeats its task infinitely, and the execution of its task is called *firing*. In this paper, we use the terms *node* for *actor*, and *edge* for *channel* interchangeably. An *actor* can produce and consume data per *channel* at different rates, which are specified by the number of *tokens*.

The number of tokens is a positive integer. If multiple inputs and outputs are associated with an actor, it is assumed that the actor waits until all input buffers have their tokens to be consumed ready for use and all output buffers are available. *Homogeneous* SDFG, where at most one token can be produced or consumed on each edge, is a special case of SDFG.

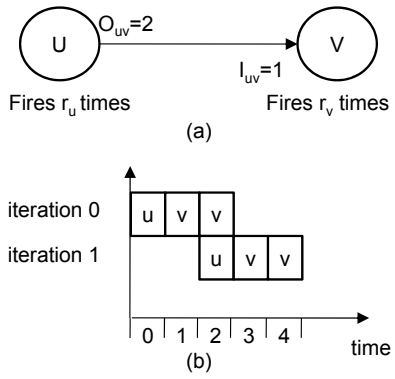

Figure 1. An example of SDFG

The number of tokens that actors produce and consume is specified by sets, I and O. I is a set of numbers of tokens consumed by destination actors of edges, and O is a set of numbers of tokens produced by source actors of edges. Thus, each edge (u, v) is associated with two integer values, I_{uv} and O_{uv}. Consider the sample SDFG shown in Figure 1 (a). The edge (u, v) has two associated integer values, I_{uv} and O_{uv}, which are 1 and 2, respectively. This represents the fact that actor u produces 2 tokens at each firing and actor v consumes 1 token at each firing. In addition, τ is a set of execution times of actors, and the execution time of each actor's firing is denoted by τ_i. Finally, a set Φ represents the initial numbers of tokens on edges, which are necessary for the start of iterative operations of a SDFG.

Using the known properties of a *homogeneous* SDFG allows us to derive the maximal computation rates as well as buffer requirements. Also, it can be shown that any arbitrary SDFG can be converted into a *homogeneous* SDFG, although this conversion may increase the size of the network exponentially.

To adapt the SDFG model for e-Science applications, it is important to understand the key differences between e-Science and DSP applications. A summary of differences between DSP and e-Science applications is provided in Table I. Unlike DSP applications, e-Science applications can be represented by acyclic graphs, have fixed start and end time, and have communication times to be considered. The time unit of DSP applications is on the order of a few milliseconds, compared to the time unit of e-Science applications that may be from a few hours to several days. Throughput is the most important objective in both DSP and e-Science applications. However, for DSP applications, tradeoffs are between throughput and buffer size, while for e-Science, the tradeoff is generally between throughput and network resource requirements. The focus of our work is on optimizing these resources.

Category	DSP application	e-Science application
Inter-task dependency	Cycles are allowed.	Usually acyclic.
Execution period	Infinite.	Finite.
Time unit	Small (a few milliseconds).	Ranges from small to large (a few minutes).
Compute resource	Unlimited.	Unlimited or limited if compute resource should be co-allocated.
Communication time	Assumed to be 0.	Needs to be considered.
Temporal constraints	Objective is maximizing computation rate (throughput).	Throughput.
Schedule	Static or dynamic.	Static.

Table I
COMPARISON BETWEEN DSP AND E-SCIENCE APPLICATIONS

Lee [21] divided scheduling of parallel computation defined by SDFG into four classes: fully dynamic, static assignment, self-timed, and fully static. Fully dynamic scheduling schedules actors at run-time only. In static assignment, assignment of actors to processors is done off-line and a local run-time scheduler of each processor invokes actors assigned to the processor. In self-timed scheduling, the assignment and ordering of actors on each processor is determined off-line and exact firing time is scheduled at run-time. The actor starts executing when all the input tokens are available. Token(s) are produced after the execution. Finally, fully static scheduling determines all information off-line.

Based on the above classification, the target e-Science applications can be considered to be self-timed. A node of SDFGs for e-Science applications represents one site, such as a data server or a computing node. This implies that every actor is assigned to a unique processor that only manages that task.

As described earlier, a SDFG is represented by $G = (V, E, I, O, \tau, \Phi)$. Since actors can produce or consume tokens at different rates, a feasible schedule should guarantee that tokens are not infinitely accumulated. In Figure 1 (a), actor u produces 2 tokens at each firing, while actor v consumes 1 token. To prevent infinite buffer overflow, actor u should be fired once for every two firings of actor v. Formally, this can be stated by the equation, $r_u \times 2 = r_v \times 1$, where r_u and r_v denote firing rates of actor u and v, respectively. These kinds of equations are called *balance equations* or *state equations*. To solve balance equations formally, we need to define a topology matrix, where e_i denotes the ith edge and O_{e_i} and I_{e_i} denote the number of produced tokens and consumed tokens, respectively, on an edge e_i.

Definition 1 (Topology matrix):

$$\text{topology matrix } \Gamma \text{ is a } |E| \times |V| \text{ matrix.}$$
$$\Gamma_{ij} = \begin{cases} O_{e_i} & \text{if an edge } e_i = (v_j, v_k), \\ -I_{e_i} & \text{if an edge } e_i = (v_k, v_j), \\ O_{e_i} - I_{e_i} & \text{if an edge } e_i = (v_j, v_j), \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The topology matrix for Figure 1 (a) is: $\Gamma = \begin{pmatrix} 2 & -1 \end{pmatrix}$. The existence of a solution, as well as a method to solve the balance equations, can be shown using the following theorem.

Theorem 1 ([22]): A connected SDF graph with actors has a periodic schedule if and only if its topology matrix Γ has rank $n-1$. Further, if its topology matrix has rank $n-1$, then there exists a unique smallest integer solution to the balance equations $\Gamma q = 0$. It can be shown that the entries in the vector q are coprime.

Given rates of actors obtained by Theorem 1, $\{r_1, r_2, \cdots, r_n\}$, one *iteration* is defined as a schedule containing r_i firings of actor i. Figure 1 (b) shows the optimal schedule for a SDFG in Figure 1 (a) when both actor u and v have self-dependency loops and the execution times of actors are all 1.

Theorem 2 ([28]): For a homogeneous SDFG represented by $G = (V, E, I, O, \tau, \Phi)$, the maximal computation rate of every node in the graph is given by

$$\min_{\forall C} \frac{\sum_{(i,j)\in C} \Phi_{ij}}{\sum_{i \in C} \tau_i}. \quad (2)$$

where C is any cycle in the graph.

Regardless of the SDFG type, i.e., homogeneous or multi-rate, the *computation rate* of a SDFG is defined as the number of iterations per unit time. The maximal computation rate of a homogeneous SDFG can be derived by examining all cycles in the graph. Theorem 2 says the maximal computation rate of a homogeneous SDFG is bounded by the minimum initial token-to-time ratio cycle in the graph. As for a homogeneous SDFG, the maximal computation rate of an iteration equals to the maximal computation rate of a node since the number of firings of a node in one iteration is 1. For a multi-rate SDFG, we can compute the maximal computation rate of a node in two steps. We can compute the maximal computation rate of an iteration after converting the multi-rate SDFG into a homogeneous SDFG. Figure 2 shows The corresponding homogeneous SDFG that is derived from the multi-rate SDFG in Figure 1. A self-dependency loop is added on each node. A node u with a rate r_u in a multi-rate SDFG will be expanded to r_u number of nodes in the homogeneous SDFG [13]. Hence, the maximal computation rate of an iteration with regard to Figure 2 is $\frac{1}{2}$ through the

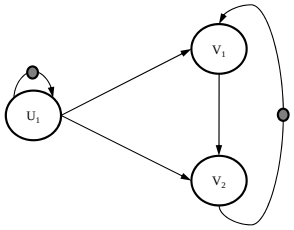

Figure 2. A homogeneous SDFG converted from Figure 1 (a)

equation

$$\min\{\frac{1}{1 \times r_u}, \frac{1}{1 \times r_v}\} = \min\{1, \frac{1}{2}\}.$$

Next, we can compute the maximal rate of each node by multiplying the maximal rate of an iteration by the rate of the node. In this example, the maximal rate of node u and v are $\frac{1}{2}(= \frac{1}{2} \times r_u)$ and $1(= \frac{1}{2} \times r_v)$, respectively. For the rest of this paper, we call the number of firings of a node per unit time, the *throughput* of the node.

III. Problem Formulation

In this section, we propose an algorithm for determining efficient bandwidth allocations required by an e-Science application whose data dependency is given by a SDFG. In addition, with these bandwidth allocations, we can minimize buffer size requirements and find the corresponding schedules.

The overall process of finding the full-fledged solution for an e-Science application is summarized as below.

1) Discretization step: In this step, both time and data size are discretized. Discretization is an important requirement for using the SDFG model. For target applications, a base unit for execution and communication times can be chosen and appropriate rounding can be performed. A base time unit should be fine-grained enough to differentiate each actor's execution time and temporal constraints. The base unit can be a few seconds to several hours, depending on the application.
2) Firing rates of actors: Using Theorem 1, firing rates of actors guaranteeing well-behaved SDFG, i.e., free of deadlock and infinite buffer accumulation, can be calculated. In MATLAB, the firing rates of actors can be obtained through a simple operation, $null(\Gamma,' r')$, where Γ is a topology matrix for the SDFG and $null$ is a MATLAB function returning a solution, Z, for $\Gamma \times Z = \mathbf{0}$.
3) Path bandwidth selection: The e-Science applications are distributed, and connection-oriented communication paths among distributed nodes are set up on demand or in advance. The bandwidth of paths is guaranteed by network technologies, such as multi-protocol label switching (MPLS) and general multi-protocol label switching (GMPLS). The communication time of a path to which bandwidth is allocated on request within available bandwidth is inversely proportional to the allocated bandwidth. Hence, path bandwidth allocation should also be taken into account since it can affect throughput as well as the total network resource consumption. A formal problem formulation will be presented in Section III-B.
4) The amount of buffer space requirements is the total number of tokens queued on every edge. Clearly, different schedules can lead to different buffer space requirements. The following buffer minimization problem is shown to be NP-complete [24]: Given a homogeneous SDFG, is there a valid schedule for the SDFG. It is easier, however, to find the minimum buffer space when the computation rate is fixed, even though the problem is still NP-complete. Using an approach similar to Govindaraja [14] but adapted to e-Science applications, we use a two-phase approach for first finding the solution for the bandwidth allocation problem, then use this solution to minimize the buffer requirements. Using this two-phase approach, as mentioned above, we obtain the solution for the bandwidth allocation problem, then minimize buffer requirements based on the obtained previous solution. After the solution to the BAFS problem (described in the next section) is obtained, we have to find exact schedules and minimize buffer space requirements. For buffer assignment, the previously developed algorithms for buffer space requirement minimization can be directly applied. The buffer space requirement minimization problem has been solved in the context of DSP applications [14] [33] [29]. The actual buffer assignment is beyond the scope of this paper.
5) Adjust for deployment in a real system: The implementation of the derived solution requires a few other considerations. The generated solution consists of discretized values in terms of the properly chosen base time and data unit. As long as we can ensure that the discretized problem has stricter constraints than the original problem, such as higher production rates and lower consumption rates, the resulting solution should be feasible. Additionally, in the absence of a global clock, synchronization issues need to be considered to force firings of tasks to follow the computed schedule. Self-timed scheduling may not achieve the maximal rate without the global clock if buffer space is limited and not properly synchronized with actors' schedules. However, for reasonable buffer sizes, the deterioration of the maximal rate should be small.

A. Illustrative Example

We pick the visualization application in [15] as a representative example of e-Science applications that can be modeled by an extended synchronous dataflow graph (ESDFG). The visualization application shown in Figure 3 (a)

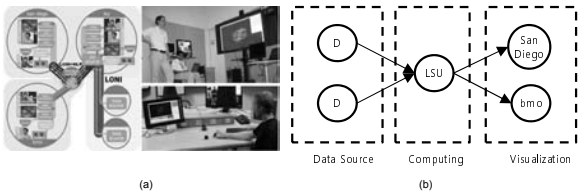

Figure 3. A real example of e-Science applications [15]

Item	Continuous value	Discretized value
Data centers		
Production	*2560 Mbyte*	128
Execution time	*1 second*	1
Computing site at LSU		
Consumption	256 Mbyte	256
Production	1 frame *(1 Mbyte)*	1
Execution time	100 ms	2
Visualizing site at San Diego		
Consumption	1 frame *(1 Mbyte)*	1
Execution time	100 ms	2
Throughput	At least 5 frames/sec	0.25
Visualizing site at Brno		
Consumption	1 frame *(1 Mbyte)*	1
Execution time	50 ms	1
Throughput	At least 5 frames/sec	0.25

Base time unit: 50 ms, Base data unit: 1 Mbyte

Table II
SUMMARY OF SYSTEM PARAMETERS OF THE VISUALIZATION APPLICATION

has a use-case scenario as follows. For the demonstration in San Diego, CCT/LSU (Louisiana), CESNET/MU (Czech Republic) and iGrid/Calit2 (California) participated in a distributed collaborative session. The visualization front-end is located at LSU running Amira for the 3D texture-based volume rendering for distributed visualization. The visualization back-end (data server) also was executed at LSU. The actual data set for the demonstration had a size of 120 Gbytes and contained 4003 data points at each timestep (4 bytes data/point for a 256 Mbyte/timestep).

We assume a more general model, similar to the use-case in [10], extended from this application such that data servers reside at different sites from computing sites. This general model can be abstracted as the diagram in Figure 3 (b).

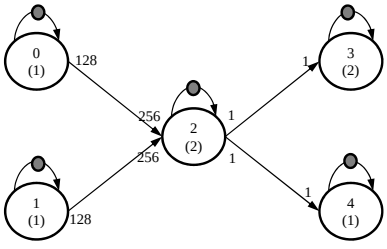

Figure 4. An ESDFG model for Figure 3

The system parameters of the visualization application is summarized in Table II. If not explicitly mentioned, all the parameters are per one firing. The figures marked by *italic* type are parameters that are not explicitly given in [15]. These are chosen by us based on a reasonable range of the associated hardware's performance. The discretized values for the parameters are computed with appropriately chosen base time and data unit. For example, for the data production speed of data centers, the value 2560 Mbyte/s, is discretized into 128 unit packets per unit time. This assumes that the unit packet size is equal to 128 Mbytes, and the unit time period is 50 ms. The resultant ESDFG for the application is shown in Figure 4.

Second, the firing rates of nodes can be calculated using the topology matrix of the ESDFG based on the description in Section II. The solution for rates of nodes is given by $[2, 2, 1, 1, 1]$. Each element of the solution vector corresponds to r_1 through r_5, respectively.

B. Optimal Bandwidth Allocation with a Feasible Schedule

To include temporal constraints such as throughput, we define extended SDFG (ESDFG) as follows.

Definition 2 (Extended SDFG (ESDFG)): An ESDFG is represented by

$G = (V, E, I, O, \tau, D, st, et, T)$,
$where\ V, E, I, O, \tau, D$ are same as SDFG,
st, et are start and end time of execution period of a SDFG, and
$T\ is\ \{(v, T_v) | v \in V, T_v \in \mathbb{R}\}$.

The set, T, has elements of throughput constraints defined as two-tuple (v, T_v), where v is the node whose throughput should be equal to or greater than T_v. st and et are used for in-advance bandwidth reservations. Suppose that we manage data structures for in-advance bandwidth reservations such as time-bandwidth lists representing how much available bandwidth varies over time on each edge, we can easily obtain a subgraph whose available bandwidth on each edge is set to maximum available bandwidth during the period $[st, et)$. For example, if an edge e_{ij} has available bandwidth 1 and 2 over time period $[0, 1)$ and $[1, 2)$, and st and et are given as 0 and 2, the e_{ij} of subgraph has a value of 1 as an available bandwidth. This formulation should also work for in-advance scheduling. Informally we can define the bandwidth allocation with a feasible schedule (BAFS) problem as follows: Given a network topology represented by $G = (V, E)$ and iterative data-dependent tasks represented by an ESDFG, $G_t = (V_t, E_t, I_t, O_t, \tau_t, st, et, T)$, what is the efficient bandwidth allocation with a feasible schedule that minimizes network resource consumption and meets temporal requirements?

The formal problem formulation will be presented below after discussion of how to model communication times in established paths for e-Science applications.

1) Modeling Communication Times: A communication delay is composed of four factors: processing delay, transmission delay, queueing delay, and propagation delay. Processing delay is associated with operations such as packetizing, thus is proportional to the data size as is transmission delay. Queueing delay is stochastic, and propagation delay is constant for a certain link. In this paper, we assume that e-Science applications run on dedicated networks, i.e., MPLS or GMPLS networks, where the paths are established using label switched paths (LSPs). For such scenarios, queueing delay and propagation delay can be ignored. We make a simplifying assumption that transmission delay dominates total delay. The processing delay is incorporated into transmission delay because both of them are proportional to the size of the data sent. Effectively, by optimizing for transmission delay, we account for the predominant cost of communication for data intensive applications.

We now investigate how to incorporate communication times into an optimal computation rate problem given a SDFG. Figure 5 (a) shows a SDFG consisting of two actors, u and v. Actor u produces 2 tokens per firing, actor v consumes 1 token per firing. We assume that it takes 2 units of time for actor u to send 2 tokens to actor v. The value in parenthesis inside a node indicates the execution time of the node.

There are two ways to integrate the communication time within the SDF model.

1) The communication time can be included in the execution time of producing actor u (Figure 5 (b))
2) The communication time can be included by having a dummy actor c whose execution time is set to the communication time (Figure 5 (c))

The first option (Figure 5 (b)) implies that communication can occur right after tokens are produced in the producer's buffer and the producer cannot be fired again until transfer of produced tokens is done. This is the most conservative way of modeling a communication time since the relation between the execution and communication is assumed to be synchronous. We call this model **the conservative model** in this paper. If we are not sure how the program is implemented internally, we can take this *conservative model* to guarantee the final solution meets the throughput requirements. The second option (Figure 5 (c)) implies that communication can run independently of the producer. This, in general, can lead to higher buffer space requirements, but may result in a higher computation rate. As can be seen in Figure 5 (c), the optimal schedule shows more throughput compared to the optimal schedule in Figure 5 (b). We call this model **the optimistic model** as opposed to **the conservative model** in this paper. Either of these two models can be chosen arbitrarily per each node, and the details on how this issue can be dealt with in the problem formulation are presented in Section III.

In some cases, there may be multiple outgoing

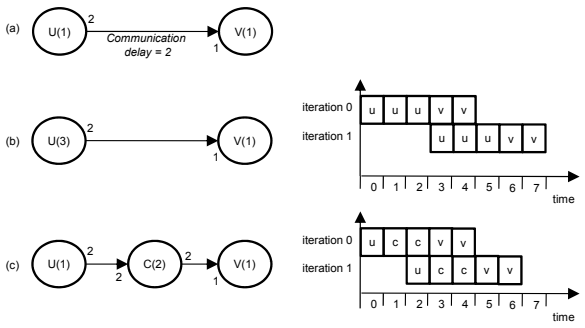

Figure 5. Modeling communication time in a SDFG

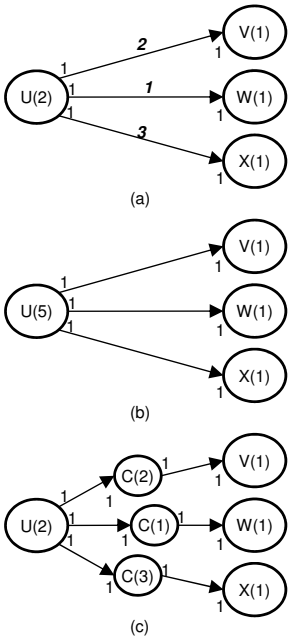

Figure 6. Modeling communication time in the case of multiple communication channels

communication channels (Figure 6 (a)). Again, one can make a choice between two options: a conservative approach and an optimistic approach. The conservative approach adds $\max\{communication\ times\ of\ outgoing\ communication\ channels\}$ to the execution time of a producer actor. Figure 6 (b) shows such a case where the execution time of actor u increases by 3, $\max\{2,1,3\}$. One of drawbacks of this model is that it deters early executable actors from starting their schedules. For example, actor w in Figure 6 (b) cannot be fired 1 unit time after u finishes its execution. Instead it should wait 2 unit times more.

For the optimistic case, the approach of single channel as described in Figure 6 (c), can be extended to multiple channels i.e., for each channel, a logical actor accounting for a corresponding communication time is inserted between the original producer/consumer actors.

A more elaborate analysis of a certain actor's execution pattern may lead to a more exact modeling, and Figure 7 shows in what cases and how we can improve our models.

Category	Notation	Description
Function	$v_t(v)$	$v_t : \mathbf{Z} \rightarrow \mathbf{Z}$, maps a vertex, v, in V into a vertex in V_t.
	$Com(a)$	$Com : V \rightarrow boolean$, returns true, if an actor a is a dummy node to model communication time.
Constant or Set	G	(V, E), original network topology.
	G_t	$(V_t, E_t, I_t, O_t, \tau_t, st, et, T)$, an ESDFG specifying iterative data-dependent tasks.
	J_c	$\{(s_i, d_i) \mid s_i \in V, d_i \in V, (v_t(s_i), v_t(d_i)) \in E_t\}$, A set of communication jobs modeled by the conservative approach, defined by two tuples of source and destination nodes.
	J_o	$\{(s_i, d_i) \mid s_i \in V, d_i \in V, c \in V_t, (v_t(s_i), c) \in E_t, (c, v_t(d_i)) \in E_t\}$, A set of communication jobs modeled by the optimistic approach, also defined by two tuples of source and destination nodes.
	J	J_c or J_o depending on the approach.
	s_j	$s_j \in V, j \in J_c \bigvee j \in J_o$, source node of job j.
	d_j	$d_j \in V, j \in J_c \bigvee j \in J_o$, destination node of job j.
	τ_i	Execution time of node (actor) $i \in V_t$.
	r_i	Rate of node (actor) $i \in V_t$.
	I_j	$j \in J$, amount of data (number of tokens) consumed by actor $v_t(d_j)$.
	O_j	$j \in J$, amount of data (number of tokens) produced by actor $v_t(s_j)$.
	C_{lk}	Available bandwidth on edge $(l, k) \in E$ during the period $[st, et)$.
	V_{tf}	A set of front-end nodes whose throughputs are concerned, $V_{tf} \subset V_t$
	T_d	Throughput requirement of node (actor) $d \in V_{tf}$, specified by users.
Variable	R_{max}	The maximal computation rate of an iteration.
	t_d	Throughput of node (actor) $d \in V_{tf}$.
	f_{lk}^j	Flow of job j on an edge $(l, k) \in E$.
	D_j	Allocated bandwidth for job j.

Table III
NOTATION FOR PROBLEM FORMULATION

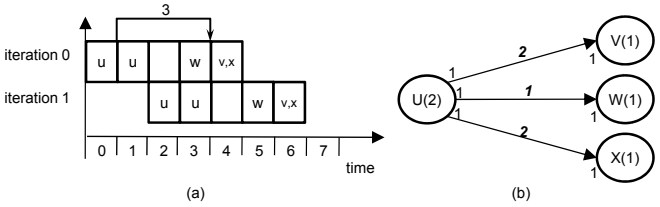

Figure 7. Deriving additional parallelism in case of multiple communication channels

The semantics of SDF enforces that the output of an actor is generated at the end of the execution of the actor. Hence, in case of Figure 6 (a), actor v, w and x can start their own execution at least 2, 1 and 3 unit time after actor u's firing is done, which means actor x can start at time 5 if actor u is fired at time 0. However, suppose that the output data for actor x is generated at time 1, the communication time on the channel between actor u and x can be adjusted by 2 as in Figure 7. Effectively, we are transforming the model to an equivalent model, with lower communication times and then using the transformation in Figure 6 to convert it into the graphs in Figure 6 (b) and (c).

The next procedures for incorporating communication time into SDF model will take either of Figure 6 (b) and (c).

2) Problem Formulation: The notation for the BAFS problem is summarized in Table III. The BAFS problem can be formulated as linear programming problems shown in Figure 8 and 9, for the conservative and the optimistic models, respectively.

The problem formulation allows the use of the multi-commodity flow problem, for which a variety of efficient solutions exists in the literature [2]. The major differences between a typical use of multi-commodity flow problem and this problem formulation are as follows:

1) The demand of each job is not a constant, but a decision variable. This determines how much bandwidth is allocated to a job, i.e., communication between producer and consumer actors.
2) The decision variables are constrained by temporal constraints pertaining to throughputs of actors.

The objective of the linear programming (given in Equation 1 and 11) is to minimize network resource consumption, which is the total amount of bandwidth allocated to all edges in the original network topology. If the demands were constant values, the objective can be regarded as minimizing average hops of all jobs (communication channels) if we approximate the average hop number as $\frac{total\ network\ traffic}{total\ demand}$. However, since we have demands as decision variables, this objective can be thought of as minimizing allocated network

Objective

$$\text{minimize} \sum_{j \in J,(l,k) \in E} f_{lk}^{j} \tag{1}$$

Multi-commodity flow constraints

$$\sum_{k:(l,k) \in E} f_{lk}^{j} - \sum_{k:(k,l) \in E} f_{kl}^{j} = 0,\ l \neq s_j, l \neq d_j, \forall j \in J \tag{2}$$

$$\sum_{j \in J} f_{lk}^{j} \leq C_{lk},\ \forall (l,k) \in E \tag{3}$$

$$\sum_{k:(l,k) \in E} f_{lk}^{j} - \sum_{k:(k,l) \in E} f_{kl}^{j} = \begin{cases} D_j, & \text{if } l = s_j \\ -D_j, & \text{if } l = d_j \end{cases}, \forall l \in V, j \in J \tag{4}$$

$$0 \leq f_{lk}^{j}, \forall j \in J, (l,k) \in E \tag{5}$$

$$0 \leq D_j \tag{6}$$

Temporal constraints

$$R_{max} \leq \frac{1}{r_i(\tau_i + \frac{O_j}{D_j})},\ i \in V_t, j \in J_c, (v_t(s_j), v_t(d_j)) \in E_t \tag{7}$$

$$t_d = R_{max} \cdot r_d,\ d \in V_{tf} \tag{8}$$

$$T_d \leq t_d \leq \frac{r_d}{r_i(\tau_i + \frac{O_j}{D_j})},$$
$$d \in V_{tf}, i \in V_t, j \in J_c, (v_t(s_j), v_t(d_j)) \in E_t \tag{9}$$

$$T_d \cdot (\tau_i D_j + O_j) \leq \frac{r_d}{r_i} D_j,$$
$$d \in V_{tf}, i \in V_t, j \in J_c, (v_t(s_j), v_t(d_j)) \in E_t \tag{10}$$

Figure 8. BAFS problem formulation in case of the conservative model

Objective

$$\text{minimize} \sum_{j \in J,(l,k) \in E} f_{lk}^{j} \tag{11}$$

Multi-commodity flow constraints

$$\sum_{k:(l,k) \in E} f_{lk}^{j} - \sum_{k:(k,l) \in E} f_{kl}^{j} = 0,\ l \neq s_j, l \neq d_j, \forall j \in J \tag{12}$$

$$\sum_{j \in J} f_{lk}^{j} \leq C_{lk},\ \forall (l,k) \in E \tag{13}$$

$$\sum_{k:(l,k) \in E} f_{lk}^{j} - \sum_{k:(k,l) \in E} f_{kl}^{j} = \begin{cases} D_j, & \text{if } l = s_j \\ -D_j, & \text{if } l = d_j \end{cases}, \forall l \in V, j \in J \tag{14}$$

$$0 \leq f_{lk}^{j}, \forall j \in J, (l,k) \in E \tag{15}$$

$$0 \leq D_j \tag{16}$$

Temporal constraints

$$R_{max} \leq \frac{1}{r_i \cdot \frac{O_j}{D_j}},\ \text{if } Com(i) = true \text{ and } j \in J_o, (v_t(s_j), i) \in E_t \tag{17}$$

$$t_d = R_{max} \cdot r_d,\ d \in V_{tf} \tag{18}$$

$$T_d \leq t_d \leq \frac{r_d}{r_i \cdot \frac{O_j}{D_j}},$$
$$\text{if } Com(i) = true \text{ and } d \in V_{tf}, j \in J_o, (v_t(s_j), i) \in E_t \tag{19}$$

$$T_d \cdot O_j \leq \frac{r_d}{r_i} D_j,$$
$$\text{if } Com(i) = true \text{ and } d \in V_{tf}, j \in J_o, (v_t(s_j), i) \in E_t \tag{20}$$

Figure 9. BAFS problem formulation in case of the optimistic model

resources regardless of average hop number of all jobs.

The constraints can be divided into two parts. The first part is typical of multi-commodity flow constraints. Equation 2 and 12, the flow conservation constraint, mandate that for all jobs, the net flow to a node is zero, i.e., the incoming and outgoing flows to a node are balanced unless the node is a source or a destination. Equation 3 and 13, the capacity constraint, mandate that the flow along any edge cannot exceed the capacity of the edge. Equation 4 and 14 ensures that the source and the destination of any job should produce and consume, respectively, the flow of the job, D_j.

The second part concerns temporal constraints, guaranteeing the throughputs of front-end actors. As discussed earlier, Theorem 2 states that the maximum computation rate is limited by the cycle whose cost-to-time ratio is minimum. Accordingly, Equation 7 and 17 account for communication times on outgoing edges of a certain node. Since the target application is acyclic e-Science application, the cycles we should consider for the maximum computation rate are limited to self-dependency loops, where the number of tokens is 1 and the total execution time are $r_i(\tau_i + \frac{O_j}{D_j})$ and $r_i \cdot \frac{O_j}{D_j}$ for the conservative and optimistic models, respectively. The term $\frac{O_j}{D_j}$ accounts for communication times of outgoing edges of actor i. In addition, since Theorem 2 is for homogeneous SDFs, considering the conversion from the given ESDFG to the homogeneous SDF [24], the execution time, $(\tau_i + \frac{O_j}{D_j})$ or $\frac{O_j}{D_j}$, should be multiplied by the firing rate r_i as in Equation 7 and 17. Since R_{max} is the number of iterations per unit time and the firing rate is the number of firings per iteration, the throughput of a certain node d equals $R_{max} \cdot r_d$, as in Equation 8 and 18. Equation 8 and 18 can be transformed into Equation 9 and 19, respectively, since the required throughput can be guaranteed if any $R_{max} \cdot r_d$ is greater than or equal to T_d specified by users. With a few transformations, the throughput constraints result in Equation 10 and 20, which are linear inequalities.

The solution for the linear programming determines the efficient bandwidth allocation on edges, and exact schedules and associated buffer space allocation can then be computed in [14].

IV. Experimental Evaluation

There is no other research work on BAFS problem in the context of grid computing. We compare our LP-based algorithm for BAFS problem with a heuristic (Algorithm 1) that uses the definition of throughput between two actors in the assignment process. The heuristic (Algorithm 1) works as follows. For each pair of producer actor and consumer actor in the SDFG graph, it finds all possible logical paths that connect them. For each such logical path, it determines the maximum communication time on any given edge that

is required for sustaining the throughput assuming that the communication time on each edge is the same. This process is repeated for all pairs of producer actors and consumer actors and the communication time of a edge for the SDFG graph is the minimum value over all such communication times. This communication time is used for a multicommodity formulation given in Equation 2 through 5 in Figure 8 to derive the physical bandwidth required for each edge of the SDFG graph. Clearly, the solution of multicommodity flow may be infeasible in some cases.

Algorithm 1 A heuristic for BAFS problem

Input: An SDFG

1: Initialize the communication time on edge i, e_i^d, as ∞.
2: **for** For all the possible paths from data producers to data consumers **do**
3: Compute maximum d required to satisfy temporal requirements for the path assuming that the communication time, d, on all the edges of the path is the same.
4: **if** $e_i^d > d$ **then**
5: $e_i^d = d$
6: **end if**
7: **end for**
8: Compute the bandwidth on each edge based on e_i^d.

The heuristic does not consider possible parallelism of tasks, or possible balanced bandwidth allocations on edges since it computes communication times by assuming all communication times on a path to be the same.

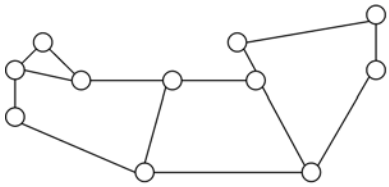

Figure 10. The Abilene network

We compare two algorithms in terms of rejection ratio of requests. The bandwidths of edges are randomly selected from a uniform distribution between 10 to 1024 unit data per base unit time. We varied the number of requests from 1 to 16 on the Abilene network [1] (see Figure 10), and each request is a specified task graph (Figure 4). The nodes of a request were constrained to have a matching node in the original network topology graph, and the matching node is randomly assigned using uniform distribution.

The experimental results are shown in Figure 11. Both of our LP-based approaches have better rejection ratios and are lower than the heuristic by 5 to 30%. The drawback of the heuristic is two-fold. First, it does not consider the fact that schedules of iterations can be overlapped. Second, it cannot allocate bandwidth to links interconnecting nodes according to the current network status, i.e., the current available bandwidth on each link. The optimistic approach, which assumes that data transfers can also occur in parallel with

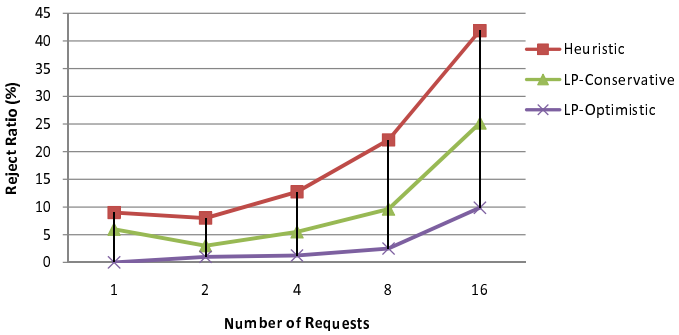

Figure 11. Rejection ratio vs. number of requests

computational executions, uses the least amount of network resources to achieve throughput requirements given by users. As our results show, the optimistic approach leads to a lower rejection ratio because it uses the network resources for one request most effectively. This leave more bandwidth for future requests.

V. Conclusion and Future Work

The dedicated network on which e-Science applications operate guarantees that a certain path can have a reserved bandwidth over a given period, which means the communication times vary depending on allocated bandwidths. We develop a SDF-based model for iterative data-dependent e-Science applications that incorporates variable communication times and temporal constraints, such as throughput. We formulate the problem as a variation of multi-commodity linear programming with an objective of minimizing network resource consumption while meeting temporal constraints. The resulting solution can then be used to derive buffer space requirements by previously developed algorithms in the context of DSP applications. Finally, an illustrative example of an e-Science application shows that the framework and algorithm we propose is valid to model and analyze iterative data-dependent e-Science applications. The simulation results show that the bandwidth allocation by using our linear programming formulation outperforms the bandwidth allocation by a simple heuristic in terms of rejection ratio of requests.

In the current paper, we did not focus on scheduling the assignment of processors to computation. We assumed that this was provided by the application. In many cases, there is flexibility in where the computations are executed (or where the data is visualized). We will investigate approaches that can uses this flexibility to improve overall performance.

Acknowledgment

This work was supported, in part, by the National Science Foundation under grant 0312038 and 0622423. Any findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

REFERENCES

[1] Abilene. http://abilene.internet2.edu/.

[2] Ravindra Ahuja, T. Magnanti, and J. Orin. *Network flows : theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs N.J., 1993.

[3] J. Bunn and H. Newman. Data-intensive grids for high-energy physics. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, Inc, 2003.

[4] CA*net4. http://www.canarie.ca/canet4/index.html.

[5] CHEETAH: Circuit-switched High-speed End-to-End Transport Architecture. http://www.ece.virginia.edu/cheetah/.

[6] T. DeFanti, C. d. Laat, J. Mambretti, K. Neggers, and B. Arnaud. TransLight: A global-scale LambdaGrid for e-science. *Communications of the ACM*, 46(11):34–41, Nov. 2003.

[7] The Earth System Grid (ESG). http://www.earthsystemgrid.org/.

[8] EarthScope. http://www.earthscope.org/usarray/data_flow/archiving.php.

[9] Energy Science Network (ESnet). http://www.es.net.

[10] Tiziana Ferrari. *Grid Network Services Use Cases from the e-Science Community*. The Open Grid Forum, Dec. 2007. http://www.ogf.org/, site last visited on Feb. 18, 2008.

[11] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[12] GÉANT2. http://www.geant2.net/.

[13] R. Govindarajan and Guang Gao. Rate-optimal schedule for multi-rate DSP computations. *The Journal of VLSI Signal Processing*, 9(3):211–232, April 1995.

[14] R. Govindarajan, Guang R. Gao, and Palash Desai. Minimizing buffer requirements under Rate-Optimal schedule in regular dataflow networks. *The Journal of VLSI Signal Processing*, 31(3):207–229, July 2002.

[15] Andrei Hutanu, Gabrielle Allen, Stephen D. Beck, Petr Holub, Hartmut Kaiser, Archit Kulshrestha, Milo Lika, Jon MacLaren, Ludk Matyska, Ravi Paruchuri, Steffen Prohaska, Ed Seidel, Brygg Ullmer, and Shalini Venkataraman. Distributed and collaborative visualization of large data sets using high-speed networks. *Future Gener. Comput. Syst.*, 22(8):1004–1010, 2006.

[16] Hybrid Optical and Packet Infrastructure. http://networks.internet2.edu/hopi.

[17] Internet2. http://www.internet2.edu.

[18] W. E. Johnston, J. Metzger, M. OConnor, M. Collins, J. Burrescia, E. Dart, J. Gagliardi, C. Guok, and K. Oberman. Network communication as a service-oriented capability. In L. Grandinetti, editor, *High Performance Computing and Grids in Action, Vol. 16*. IOS Press, 2008. http://www.es.net/pub/esnet-doc/ESnet-SciDAC-Review-Summer-2007.pdf.

[19] Joint Institute for Very Long Baseline Interferometry (JIVE). http://www.jive.nl/.

[20] Lawrence Berkeley National Laboratory. *FES Network Requirements Workshop Final Report*, March 2008. http://www.es.net/pub/esnet-doc/FES-Net-Req-Workshop-2008-Final-Report.pdf.

[21] E.A. Lee and S. Ha. Scheduling strategies for multiprocessor real-time DSP. In *Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyond. GLOBECOM '89., IEEE*, pages 1279–1283 vol.2, 1989.

[22] Edward Ashford Lee. *A coupled hardware and software architecture for programmable digital signal processors (synchronous data flow)*. PhD thesis, University of California, Berkeley, 1986.

[23] T. Lehman, J. Sobieski, and B. Jabbari. DRAGON: A framework for service provisioning in heterogeneous grid networks. *IEEE Communications Magazine*, March 2006.

[24] Praveen Murthy. *Scheduling techniques for synchronous and multidimensional synchronous dataflow.* Electronics Research Laboratory College of Engineering University of California, Berkeley, 1996.

[25] National Lambda Rail. http://www.nlr.net.

[26] H. B. Newman, M. H. Ellisman, and J. A. Orcutt. Data-intensive e-science frontier research. *Communications of the ACM*, 46(11):68–77, Nov. 2003.

[27] OSCARS: On-demand Secure Circuits and Advance Reservation System. http://www.es.net/oscars.

[28] Raymond Reiter. Scheduling parallel computations. *J. ACM*, 15(4):590–599, 1968.

[29] S. Stuijk, M. Geilen, and T. Basten. Throughput-Buffering Trade-Off exploration for Cyclo-Static and synchronous dataflow graphs. *Computers, IEEE Transactions on*, 57(10):1331–1345, 2008.

[30] TeraGrid. http://www.teragrid.org/.

[31] The U.K. Research Councils. http://www.research-councils.ac.uk/escience/.

[32] Ultralight: An Ultrascale Information System for Data Intensive Research. http://www.ultralight.org.

[33] M. Wiggers, M. Bekooij, P. Jansen, and G. Smit. Efficient computation of buffer capacities for multi-rate real-time systems with back-pressure. In *Hardware/software codesign and system synthesis, 2006. CODES+ISSS '06. Proceedings of the 4th international conference*, pages 10–15, 2006.