GreenDataFlow: Minimizing the Energy Footprint of Global Data Movement

MD S Q Zulkar Nine[1], Luigi Di Tacchio[1], Asif Imran[1], Tevfik Kosar[1], M. Fatih Bulut[2], Jinho Hwang[2]
[1]Department of Computer Science and Engineering, University at Buffalo, Buffalo, New York
[2]IBM TJ Watson Research Center, Yorktown Heights, New York
Email: {mdsqzulk, luigidit, asifimra, tkosar}@buffalo.edu, {mfbulut, jinho}@us.ibm.com

***Abstract*—The global data movement over Internet has an estimated energy footprint of 100 terawatt hours per year, costing the world economy billions of dollars. The networking infrastructure together with source and destination nodes involved in the data transfer contribute to overall energy consumption. Although considerable amount of research has rendered power management techniques for the networking infrastructure, there has not been much prior work focusing on energy-aware data transfer solutions for minimizing the power consumed at the end-systems. In this paper, we introduce a novel application-layer solution based on historical analysis and real-time tuning called GreenDataFlow, which aims to achieve high data transfer throughput while keeping the energy consumption at the minimal levels. GreenDataFlow supports service level agreements (SLAs) which give the service providers and the consumers the ability to fine tune their goals and priorities in this optimization process. Our experimental results show that GreenDataFlow outperforms the closest competing state-of-the art solution in this area 50% for energy saving and $2.5\times$ for the achieved end-to-end performance.**

I. Introduction

The era of artificial intelligence (AI) has made data the most important resource, in turn the efficient data handling is the key to use compute, network, and storage resources more effectively. Not like compute and storage resources, the network resource needs more sophisticated control as it involves the end-to-end efficiency. The annual data transfer rate over global IP networks has already exceeded zettabyte scale [46]. The energy footprint of this global data movement is estimated at more than 100 terawatt hours per year at the current rate, costing more than 20 billion US dollars annually to the world economy in addition to the environmental side effects [19], [24], [36], [39], [46]. This fact has resulted in considerable amount of work focusing on power management and energy efficiency in hardware and software systems [9], [13], [14], [25], [27], [32], [41], [42], [44], [47], [52] as well as on power-aware networking [6], [18], [20], [21], [28], [36].

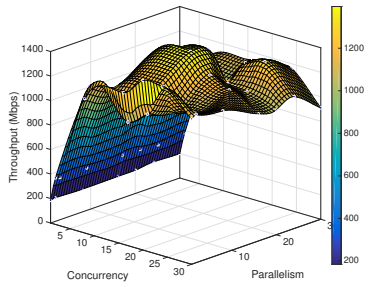
Majority of the existing work on power-aware networking focuses on reducing the power consumption on networking devices (e.g., routers, switches, and hubs). Gupta et al. [24] were amongst the earliest researchers to advocate conserving energy in the networking infrastructure. They suggested different techniques such as putting idle sub-components (e.g., line cards) to sleep [23], which were later extended by other researchers. Nedevshi et al. proposed adapting the rate at which switches forward packets depending on the traffic [38]. IEEE Energy Efficient Ethernet Task Force proposed the 802.3az standards [1] for making Ethernet cards more energy efficient. They defined a new power state called Low-Power Idle (LPI) that puts the Ethernet card to low power mode when there is no network traffic. Other related research in power-aware networking has focused on architectures with programmable switches [22], switching layers that can incorporate different policies [30], and power-aware network protocols for energy efficiency in network routing [10].

The existing approaches suffer from the following drawbacks: (1) the solution is too costly (e.g., replacing all switches with energy efficient ones); (2) the solution is unpractical in the short term (e.g., replacing TCP with a more energy-efficient version); (3) the solution penalizes performance while increasing energy efficiency (e.g., sleeping some components while not in use). In this paper, we propose an application-layer solution called GreenDataFlow which is low cost, very easy and practical to deploy, and does not penalize the performance while increasing energy efficiency. With the added benefits and simplicity to adopt, service providers can directly benefit from GreenDataFlow as they can offer it as-a-service in their cloud platforms, while making sure that the SLA requirements of customers are satisfied using our SLA-based algorithms.

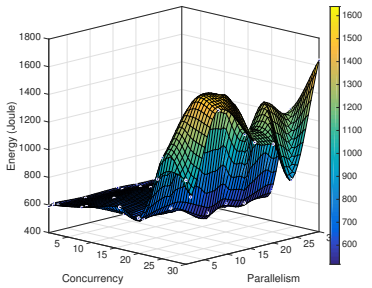
GreenDataFlow provides novel two-phase dynamic optimization models to minimize energy and increase throughput at the same time. It is based on mathematical modeling with offline knowledge discovery and adaptive online decision making. During the offline analysis phase, we analyze historical transfer logs to perform knowledge discovery about the characteristics of the past transfers with similar requirements. During the online phase, we use the discovered knowledge from the offline analysis along with real-time investigation of the network condition to optimize the protocol parameters for both minimal energy consumption and maximum transfer throughput. Our models use historical knowledge about the network and data to reduce the real-time investigation overhead while ensuring near optimal results for each transfer. Specifically our contributions in this paper are as follows:

1) GreenDataFlow minimizes the energy footprint of big data transfers by operating in the application-layer, without any need to change the existing infrastructure nor the low-level networking stack, which makes its integration to existing applications easier.
2) GreenDataFlow integrates knowledge-based offline analysis with real-time tuning to achieve close-to-optimal

978-1-5386-5035-6/18/$31.00 ©2018 IEEE



(a) Achieved Throughput for different cc and p



(b) Energy Consumption for different cc and p

Figure 1: Achieved throughput and energy consumption of a single transfer under different parameter combination. Surface interpolation is perform using piece-wise cubic spline.

data transfer throughput while reducing energy consumption.

3) GreenDataFlow applies adaptive tuning in real-time and uses pre-computed mathematical optimization based decisions to provide faster convergence toward maximally achievable throughput.
4) Our experimental results show that GreenDataFlow outperforms the closest competing solution in this area up to 50% for energy saving and up to $2.5\times$ for the achieved end-to-end performance. When we compare it with baseline cases (without any optimization), the energy savings go up to 80% and performance improvement reaches $10\times$.

The rest of the paper is organized as follows: Section II provides background information and gives a formal definition of the problem; Sections III and IV present our novel two-phase dynamic optimization models; Section V evaluates our models; Section VI describes the related work in this field; and Section VII concludes the paper.

II. Background and Problem Definition

Large-scale data transfers can have suboptimal performance and high energy footprint in a long RTT WAN network due to the protocol inefficiencies. Changing the protocol stack requires low-level updates, and its adaptation by large-scale needs considerable time and effort. Therefore, application level solutions are more lucrative and easy to deploy in the user space. Application level data transfer protocol parameters (i.e., concurrency, parallelism, pipelining) can be tuned to increase the data transfer throughput and decrease the energy footprint significantly. Figure 1 shows the throughput and energy consumption of a single transfer under different parameters. A short description of these parameters is given below.

Concurrency (cc) controls the number of server processes where each process can transfer an individual file. It can accelerate the transfer throughput when a large number of files need to be transferred. Each server process can transfer a different portion of a file in parallel. We define the number of parallel streams for each process as *Parallelism* (p). Increasing number of parallel data streams can increase the achievable throughput for large files. However, excessive use of streams may lead to packet loss and force TCP to initiate slow-start phase that may lead to severe throughput loss. Control channel idleness is a significant bottleneck in transferring a large number of files. After each file transfer, the server process sends an acknowledgment to initiate the next file transfer. This acknowledgment can take at least one Round-Trip-Time (RTT) between each file transfer. It may hurt the overall throughput of a dataset containing a large number of small files in a long RTT network. This issue can be solved by queuing multiple file transfer requests without waiting for the acknowledgments. We define the size of the outstanding file transfer request queue as *Pipelining* (pp).

Energy consumption is a major concern in data centers and end-systems that perform large-scale data transfers. Minimizing the energy consumption can reduce the data center operating cost while utilizing the network bandwidth efficiently. Energy-efficiency can be achieved through optimal hardware resource (e.g.,CPU, memory, disk and NIC) scheduling to the data transfer tasks. Over-provisioning of the compute resources can increase the energy cost. On the other hand, the under-provisioning can reduce the data transfer rate and takes more time to finish transfer job. As the resources are occupied for a longer period of time, the energy consumption could be higher than optimal solution while provides a low data transfer rate.

There is a trade-off between performance and energy constraints. A data center administrator has an immediate incentive to optimize the energy consumption as it is directly related to the operating cost. However, end users might want the freedom to choose from a different range of services. A service provider can advertise energy constraint solutions with a lower price tag. In this work, we introduce easy to describe energy-aware Service Layer Agreement (SLA) categories that a user or an administrator can initiate. SLA is a contract between the user and the service provider on service quality and specific rights of both parties. It may include the description of services agreed to be provided, monitoring and reporting of quality of service (QoS) matrices. For energy efficient transfers, SLA space can be sub-divided into two main categories:

(1) TYPE - T: A user may need a throughput guarantee (e.g., throughput sensitive transfers, deadline-aware transfers). In this case, the service provider can choose an energy efficient solution without violating the throughput SLA. Therefore, the optimization problem can be expressed as :

$$\begin{array}{ll} \underset{\{parameters,resources\}}{\text{minimize}} & (E) \\ \text{subject to.} & T_{act} \geq T_{sla} \end{array} \quad (1)$$

Where T_{act} is the achieved throughput and T_{sla} is the throughput guarantee that the user wants.

(2) TYPE - E: A user might want to limit energy consumption to reduce the data transfer cost and ask for the best possible transfer rate. Therefore, actual energy consumption (E_{act}) can be constrained by energy consumption limit specified in SLA, (E_{sla}). The optimization problem is to maximize throughput, T under a specified energy constraint and can be expressed as:

$$\begin{array}{ll} \underset{\{parameters,resources\}}{\text{maximize}} & \int_{\tau_s}^{\tau_f} T \\ \text{subject to.} & E_{act} \leq E_{sla}. \end{array} \quad (2)$$

In this paper, we apply our optimizations to GridFTP [33] which is based on TCP. GridFTP is widely used in the scientific community, and it supports easy tuning of parameters such as parallelism, concurrency, and pipelining.

III. Solving Optimization Problem

The optimization problems defined in Section II require accurate modeling of throughput and energy consumption. Therefore, we need to know the impact of application level parameters and compute resource allocations on both energy consumption and throughput. We can express both throughput and energy consumption as a function of application level parameters and allocated compute resources.

$$\begin{aligned} T &= f_1(data, net, cc, p, pp, \mu_{cpu}, \mu_{mem}, \mu_{disk}, \mu_{nic}) \\ E &= f_2(data, net, cc, p, pp, \mu_{cpu}, \mu_{mem}, \mu_{disk}, \mu_{nic}) \end{aligned} \quad (3)$$

Where $data$ is dataset, net is network, μ_{cpu}, μ_{mem},μ_{disk}, and μ_{nic} are CPU, memory, disk and NIC allocations.

To understand these relationships in Equation 3, we can record throughput and energy consumption of a data transfer that is performed with different parameters, resource allocations and then use these data to fit a regression or interpolation model. However, this simplistic approach has four significant drawbacks - (1) The collected synthetic data might not be representative of the real-world scenario. (2) Other contending traffic in the same network can impact the data transfer. Therefore, we need to model their impact as well. (3) Alan et al. [3] show that the parameters can have a different effect on throughput and energy when data transfer requests are different. For example, small file transfers require high concurrency and pipelining to achieve high throughput. On the other hand, large files require high parallelism to achieve high throughput. Therefore, clustering is necessary to group similar file transfers first. (4) We need to select a proper regression or interpolation model based on the behavior of the data.

We decided to use historical data transfer logs to address the drawback (1). Then we perform clustering on these logs to group the similar transfers. After that, we use a proper interpolation model on each log cluster. Then we solve the optimization problem. Steps and the design choices are explained below.

Step 1 – Storing Historical Logs: Historical data transfer logs are collected periodically and stored in a log server. These logs collect information about the network characteristics (e.g., round trip time, buffer size, queuing delay, packet loss rate), application level parameters (cc, p, pp), end-system resource allocation information (e.g., CPU, memory, NIC), dataset information (e.g., size, number of files/objects), energy-related information (e.g., CPU utilization, memory utilization, NIC card utilization, disk I/O utilization). These logs provide insight to optimize transfers energy-efficiently under different circumstances.

Step 2 – External Load Modeling: In a shared network environment, the data transfer task has to compete with other contending transfers. Contending transfers can be a mixture of known incoming/outgoing streams in both source/destination and completely unknown transfers. We can define the throughput of the known transfers as T_{ext_kn} and the throughput of the unknown transfers as T_{ext_unk}. Some bandwidth might be wasted by TCP congestion (δ_{congst}). However, the optimal number of streams can offset congestion loss significantly. We can say that the entire bandwidth of the link is the sum of the achieved throughput of our data transfer task T_{act}, all other known, unknown transfers, and the bandwidth wasted due to congestion and slow start. Therefore, we can model bandwidth.

$$BW = T_{act} + \sum T_{ext\_kn} + \sum T_{ext\_unk} + \sum \delta_{congst} \quad (4)$$

We can estimate $\delta_{congestion}$ from the round trip time and queuing delay of the network. We can also determine the combined throughput of known transfers. Therefore, from Equation (4), we can get a rough estimate of the combined throughput of the unknown external traffic.

Step 3 – Clustering historical logs: As we are using the historical logs, categorizing logs into groups based on their similarity would provide us a more structured view of the log information. After analyzing the logs, we can see that the log metadata (e.g., dataset information, network characteristics, external load condition) are responsible for performance and energy consumption variability in different transfers using the same parameters and resources. We also notice that some log metadata have direct precedence over others. For example, source and destination information and network characteristics have more impact on performance and energy consumption variability compared to the dataset information. Therefore, we can see an explicit hierarchy in clustering. The high-level clusters (Tier-1) are based on source, destination and connecting network. Each of these clusters can be sub-divided into more clusters based on dataset information (Tier-2). Each Tier-2 cluster can be sub-divided using external load information, and so on. To meet this requirement, we use a modified version of *Hierarchical Agglomerative Clustering* [37] which can cluster logs in a bottom-up manner. We set the log metadata hierarchy

as explained earlier. The clustering algorithm initially treats each log entry as a singleton cluster and starts merging them based on last tier log metadata. Then the formed clusters are merged again based on second to the last tier log metadata, and it continues until it reaches the Tier-1.

Step 4 – Interpolation and optimization: We are interested to find the relationship expressed in Equation 3 for each cluster from Step-3. Historical logs might not have log information for all the parameters and resource allocations. We need an interpolation technique to model the relationships so that we can predict the performance, energy consumption of parameters and resource allocations those are not present in the historical logs. After analyzing the historical logs, we can see that both relations are strictly non-linear and follow a continuous cubic pattern. Therefore, we modeled both throughput and energy using piece-wise cubic spline interpolation (Figure 1). This technique stitches multiple cubic functions with smoothness guarantee up to the second derivative. All the continuity constraints and the smoothness constraints are linear. Therefore, the coefficients can be computed by solving the system of linear equations. This interpolation has some explicit benefits. If we choose the optimal parameters from the historical logs without any interpolation, there is a good chance that the historical logs do not have the parameters optimal for the transfer. Interpolation can predict the parameters those are not present in the historical logs. We have used 70% logs to perform the interpolation and the rest of the 30% logs are test the prediction of the interpolation method. We used standard Root Mean Squared Error (RMSE). This interpolation can achieve 93% accuracy. Now we can model throughput and energy using the parameters and resource allocation (Equation 3). We used a Matlab solver to compute optimal parameters and resource allocations.

Algorithm 1: Dynamic Tuning

```
// Application level parameter: params; Resource allocation: res; Data transfer request: req; Network status: net_status; Network delay, δ
Periodically check:
if SLA_type == 'Energy Constraint' then
    if instantaneous power consumption is higher than SLA then
        if High queuing delay: fairness_control()
    else
        opportunistic_decrease(SLA)
    params, res ← get_precomputed_results(req, net_status, δ, SLA)
    update_resource_groups(res)
    Perform rest of the transfer with new parameters
else if SLA_type == 'Throughput Guarantee' then
    if Throughput is less than SLA then
        if High queuing delay: fairness_control()
    else
        opportunistic_increase(SLA)
    params, res ← get_precomputed_results(req, net_status, δ, SLA)
    update_resource_groups(res)
    Perform rest of the transfer with new parameters
check_stream_perf()
redistribute_pipelining()
```

Algorithm 2: Stream Fairness Control Algorithm

```
// Queuing delay: Q_rtt
procedure back_off_control()
    Q_rtt ← measure_queuing_delay()
    pkt_loss_rate ← get_packet_loss_rate()
    if Q_rtt ≪ Q_rtt.expected) : pp ← pp − 1
    else: pp ← pp + 1
    if pkt_loss_rate ≪ pkt_loss_rate.threshold : cc ← cc − 1
    else: cc ← cc + 1
```

IV. Model Design

In our proposed models, we considered two crucial design challenges: (1) Optimization from Section III can introduce overhead when we solve it during the transfer; (2) Network condition might change during the transfer, and initial parameter choices and resource allocation might become sub-optimal. To address these issues, we decided to perform the optimization offline. So that the user can access the precomputed parameters and resource allocation to perform the transfer. We also propose a dynamic tuning of the transfer adapt to the real-time network conditions. These two design choices are explained below.

A. Offline Optimization

Precomputing the constrained optimization (detailed in Section III) during offline phase can have two major benefits: (1) it eliminates any real-time latency for optimal parameters, and (2) these precomputed results can be reused for many subsequent transfers, which can effectively amortize the initial cost of analysis. As the user can put constraints over throughput or energy consumption, during the offline phase, we would have to solve the optimization problem for all possible SLA values of throughput or energy or power, which is not feasible. We observed that, when two SLA values are close, they produce similar solutions. Therefore, instead of solving the optimization problem for all possible SLA values we partition the SLA feasible region based on historical log data and solve one optimization problem for each partition. For example, a transfer takes minimum 600 Joules to perform the task, and the maximum throughput can be achieved with 1000 Joules. Based on historical logs we can partition this into three regions, (1) 600 - 750 (Joules), (2) 751 - 850 (Joules), and 850-1000 (Joules). Service provider can advertise these SLA partitions with associated price tag. This approach can give us a constant number of SLA levels for both achievable throughput and energy consumption. The pre-computed optimization result for each cluster is stored in a hash table. The key is the cluster name and values are a vector of all parameter values and resource allocations. We can access these results through a

function called `get_precomputed_results(key)` from the hash table.

B. Dynamic Tuning

Dynamic tuning is the real-time monitoring of the health of the data transfers, simultaneously it controls the aggressiveness of the protocol (maintains fairness), while ensuring strict SLA requirement. As we have two different categories of SLAs, we need two different strategies as well. However, the core control is mostly similar. An overview of the tuning module for different SLAs is introduced in Algorithm 1.

SLA – Energy Constraint *(Lines 2-9)* **:** Energy consumption of a transfer increases for the following reasons: (1) extra work due to excessive retransmission when packet loss or congestion increases, (2) over-provisioning of compute resources. When instantaneous power consumption goes higher, dynamic tuning checks RTT and queuing delay to detect congestion. If delay is detected, then it immediately initiates a sub-routine called `fairness_control()` (Explained in Algorithm 2). This sub-routine reduces the parameter values to ensure fairness among the users during congestion. It measures queuing delay and packet loss rate. When queuing delay is higher than expected, it reduces parallelism value by one as a congestion avoidance measure. However, packet loss has more severe consequences. Therefore, when it detects packet loss, it decreases the concurrency value by one. On the other hand, When queuing delay is reasonable, the high energy consumption might be due to resource over-provisioning. It checks the remaining data to be transferred, network condition, and SLA. Then it uses `get_precomputed_results()` function to get new allocation from offline optimization and update resource allocation (Lines 7-8). In a congested link, SLA violation might occur during the data transfer. To compensate for the violation, we can decrease the SLA energy constraint to a lower value whenever it is possible. We introduce `opportunistic_decrese()` of SLA (Line 6). It monitors the network conditions, external loads and asks the offline optimization module for parameters and allocations that can guarantee the decreased SLA. This decreased SLA ensures enough buffer efficiency for any previous or upcoming SLA violation due to congestion.

SLA – Throughput Guarantee *(Lines 10-17)* **:** The major reason for throughput SLA violation is the congestion and under provisioning of resources. When it detects throughput is less than the SLA, it immediately checks for congestion. If congestion is detected, it initiates `fairness_control()` to dynamically tune parameters (same as Energy constraint SLA) (Line 12). Otherwise, it asks `offline_optimization` for new resource allocation to resolve under-provisioning using function `get_precomputed_results()` (Line 15). In an uncongested link, it can provide continuous throughput guarantee. However, a congested link may force the transfer to reduce the parameters (ensure fairness) that directly impact the throughput. `opportunistic_increase()` compensates this inevitable throughput drop with a new throughput goal that is higher than SLA (Line 14). During uncongested time interval, the dynamic tuning module will ask offline analysis module to provide parameters that can achieve new SLA goal. We also introduce a buffer capacity of achievable throughput higher than SLA. This buffer size is based on historical data analysis. Opportunistic increase function tries to fill this buffer, whenever it senses available throughput. This buffer pro-actively offsets any future throughput degradation.

Specifications	IBM IDCN	XSEDE
Bandwidth (GBps)	1	10
RTT (ms)	65	40
Buffer size (MB)	8	32
File system		Lustre
Cores	2	16
Memory (GB)	2	32

Table I: System and network specification of test sites

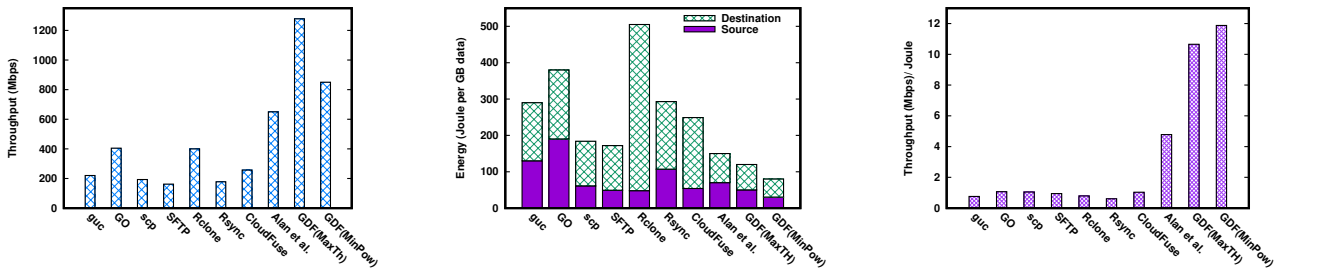
V. Evaluation

We performed experiments on WAN links between IBM datacenters located in Washington, D.C. and San Jose, CA. We also used XSEDE, a production level high-speed computing infrastructure for scientific computations. An overview of systems and network information is provided in Table I. We collected historical log data over two weeks of time from transfers performed in IBM datacenters. To measured energy consumption we used Intel Running Average Power Limit (RAPL) model in all our experiments. It is highly reliable and uses hardware counters to measure energy consumption.

We compared our model with many existing data transfer solutions. However, there has been done very little work on energy efficient data transfer optimization. We evaluate our model against the model proposed by Alan et. al. [3], `globus-url-copy` (guc) [16], Globus Online (GO) [11], `scp`, `SFTP`, Rsync, Rclone [43], and CloudFuse [12]. Alan et al. provide a High Throughput Energy-Efficient Algorithm (HTEE) that uses a heuristics based approach. It starts with one channel and periodically increases it by two until it reaches to a user-defined limit. Then it computes energy efficiency for each level and picks the best one. This periodic additive increase is slow. `Globus-url-copy` and Globus Online are GridFTP based data transfer tools to achieve high performance during transfer. However, they are not energy optimized tools. Scp and SFTP are widely used secure file transfer tools. Rsync and Rclone are high-performance data synchronization applications. CloudFuse provides cloud-based Managed File Transfer (MFT) service and offers migration, sync and other file management capabilities to the end users.

A. Comparison with Other Solutions

Figure 2 shows an elaborate experimentation and performance analysis of different state-of-the-art solutions and our proposed approach. Most of the models do not support SLA. Therefore, to make comparison fair we set the SLA of our model in two extreme cases - (1) Maximum achievable



(a) Achieved throughput (medium files) (b) Energy consumption (medium files) (c) Throughput efficiency (medium files)

Figure 2: Achievable throughput and corresponding energy consumption of different optimization objectives.

throughput (MaxTh) and (2) Minimum possible energy consumption (MinPow). To test the efficiency of different types of file transfers, we tested all data transfer solutions for small (1 - 5MB), medium (100 - 500MB), and large (1 - 4GB) files.

Figure 2 (d-f) contain performance comparison for medium files. We compared achieved throughput, energy consumption, and the throughput efficiency (achieved throughput per unit energy). As we can see off-the-shelf tools like scp and SFTP perform poorly due to single data channel allocation and control channel inefficiency. Their energy consumption is also high because low throughput transfer needs more time to finish and longer time of execution increases the static power component (power consumption when the resource is idle and waiting). Similarly, `globus-url-copy (guc)`, a GridFTP based tool, also performs poorly with base-line parameter settings ($cc = 1$ & $p = 1$). Globus Online is a statically tuned cloud service that uses GridFTP protocol. Due to the use of multiple streams, we observe that it can reach up to $2\times$ performance improvement compare to `scp`, `SFTP`, and `guc`. However, it consumes $2\times$ more energy. The reason is the sub-optimal parameter choice. Moreover, there is no way to limit the resource utilization as well.

`Rclone` and `Rsync` are file sync tools. By default, `Rclone` uses 4 parallel data connections to transfer a single file. We observe that it can achieve similar performance as `GO`. However, there is no way to make any dynamic adjustment to parallel connections. We see an unusually high energy consumption at the destination due to sync operation.

CloudFuse achieves slightly better performance than `guc`. It consumes slightly less energy compared to `Rsync` and `guc`. As we see in the Figure 2, Alan et al. model performs much better than the solutions discussed above since it initiates an online parameter search. However, there is no real-time control over parameters. Therefore, when external traffic changes, those parameters may become sub-optimal. And additive parameter search may take a toll on the achieved throughput. However, due to the search for energy-efficient parameters, it can manage to keep energy consumption less than the other approaches mentioned above.

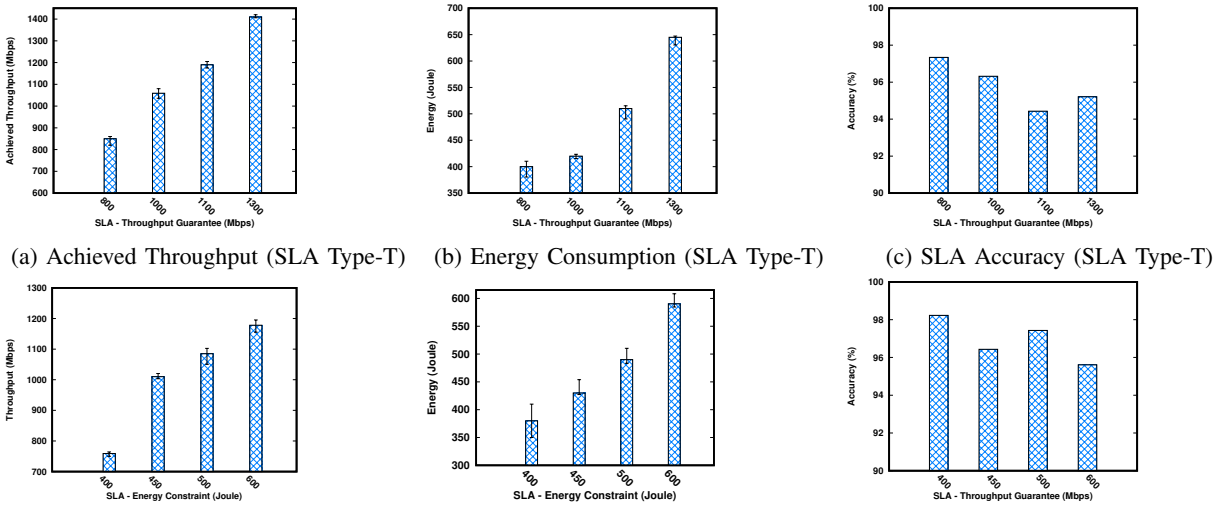
Both of GreenDataFlow algorithms (MaxTh and MinPow) outperform all the listed solutions. MaxTh provides $6\times$ throughput performance improvement over the baseline performance of `globus-url-copy` and almost $2\times$ improvement over the closest competitor Alan et al. model due to the historical analysis and real-time tuning of the parameters. As it achieves high throughput, the execution time reduces as well which reduces the static power consumption along with constrained resource scheduling in end-systems. MinPow is aimed to decrease the total energy consumption. It consumes $8\times$ less energy than the energy-hungry `rclone` and almost 36% less energy than the closest competitor Alan et al. due to optimal resource scheduling.

B. SLA-based Performance Analysis

Figure 3 shows the performance for different SLA levels. It can be seen that SLA violations are rare unless there exist over-subscription of throughput or severe capacity reduction for a long period. Most of the cases in Type-T SLA (Figure 3 (a-c)), our model can achieve performance higher than the SLA, due to the `opportunistic_increase()` strategy. It also keeps the resource utilization manageable by putting a dynamic restriction on usage. SLA violation error is ranged from 3% to 6%. For energy constrained (Type-E) SLA, we observed the SLA violation occurs due to heavy congestion which forces retransmission and initiates slow start phase. Moreover, excessive concurrent processes can consume extra power while congesting the network. As our model cautiously monitors and budgets the required future energy usage, it can achieve high accuracy in SLA commitment.

VI. Related Work

The work on network throughput optimization focuses on tuning transfer parameters such as parallelism, pipelining, concurrency and buffer size. The first attempts to improve the data transfer throughput at the application layer were made through buffer size tuning. Various dynamic and static methods were proposed to optimize the buffer size [29], [40], [45]. However, Lu et al. [35] showed that parallel streams could achieve a better throughput than buffer size tuning and then several others [5], [26], [49] proposed throughput optimization solutions by means of tuning parallel streams. Another transfer parameter used for throughput optimization was pipelining, which helped in improving the performance of transferring a large number of small files [8], [15], [17]. Liu et al. [34] optimized network throughput by concurrently opening multiple transfer sessions and transferring multiple



(a) Achieved Throughput (SLA Type-T) (b) Energy Consumption (SLA Type-T) (c) SLA Accuracy (SLA Type-T)

(d) Achieved Throughput (SLA Type-E) (e) Energy Consumption (SLA Type-E) (f) SLA Accuracy (SLA Type-E)

Figure 3: Achieved throughput, energy consumption and SLA commitment accuracy for different SLA types. (a-c) SLA with throughput guarantee, (d-f) SLA with energy constraint, and (g-i) SLA with power constraint.

files concurrently. They proposed increasing the number of concurrent data transfer channels until the network performance degrades. Globus Online [4] offers fire-and-forget file transfers through thin clients over the Internet. It partitions files based on file size and transfers each partition using partition-specific protocol parameters. However, the protocol tuning Globus Online performs is non-adaptive; it does not change depending on network conditions and transfer performance.

The work on power-aware networking focuses on saving energy in the networking devices. Gupta et al. [24] were among the earliest researchers to advocate conserving energy in networks. They suggested techniques such as putting idle sub-components (i.e., line cards, etc.) to sleep [23], which were later extended by other researchers. S. Nedevshi et al. [38] proposed adapting the rate at which switches forward packets depending on the traffic load. IEEE Energy Efficient Ethernet task force proposed the 802.3az standards [1] to make ethernet cards more energy efficient. They defined a new power state called low power idle (LPI) that puts the ethernet card to low power mode when there is no network traffic. Other related research in power-aware networking has focused on architectures with programmable switches [22] and switching layers that can incorporate different policies [30]. Barford et al. proposed power-aware network protocols for energy-efficiency in network design and routing [10]. Bertozzi et al. [7] investigated the energy trade-off in networking as a function of the TCP receive buffer size and show that the TCP buffering mechanisms can be exploited to significantly increase the energy efficiency of the transport layer with minimum performance overheads.

Several highly-accurate predictive models [31], [50], [51] were developed which require as few as three sampling points to provide accurate predictions for the parallel streams giving the highest transfer throughput for the wired networks. Yildirim et al. analyzed the combined effect of parallelism and concurrency on data transfer throughput [48]. Alan et al. explored the impact of parallelism and concurrency on end-to-end data transfer throughput versus energy consumption in WAN networks using precalculated values for these parameters and proposed a heuristic approach to improve them [2], [3].

VII. Conclusion

In this paper, we introduced a novel set of data transfer algorithms (collectively called GreenDataFlow) based on historical analysis and real-time tuning, which can achieve high data transfer throughput while keeping the energy consumption during the transfers at the minimal levels. GreenDataFlow supports service level agreements (SLAs) which give the service providers and the consumers the ability to fine tune their goals in this optimization process. Our experimental results show that GreenDataFlow outperforms existing solutions in this area both in terms of energy saving and the achieved end-to-end performance. Considering the massive energy footprint of global data movement, our presented GreenDataFlow techniques have a great potential to decrease this footprint and contribute to the efforts of achieving greener Internet.

Acknowledgements

This project is in part sponsored by the National Science Foundation (NSF) under award numbers OAC-1724898 and OAC-1842054, and by IBM Research under award number OCR-W1771224. This work also used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF grant number ACI-1548562.

References

[1] IEEE energy efficient ethernet standards. 10.1109/IEEESTD.2010.5621025, Oct. 2010.

[2] I. Alan, E. Arslan, and T. Kosar. Energy-performance trade-offs in data transfer tuning at the end-systems. *Sustainable Computing: Informatics and Systems*, 4(4):318–329, 2014.

[3] I. Alan, E. Arslan, and T. Kosar. Power-aware data scheduling algorithms. In *Proceedings of IEEE/ACM Supercomputing Conference (SC15)*, November 2015.

[4] B. Allen, J. Bresnahan, L. Childers, I. Foster, G. Kandaswamy, R. Kettimuthu, J. Kordas, M. Link, S. Martin, K. Pickett, and S. Tuecke. Software as a service for data scientists. *Communications of the ACM*, 55:2:81–88, 2012.

[5] E. Altman and D. Barman. Parallel tcp sockets: Simple model, throughput and validation. In *Proceedings of IEEE INFOCOM*, 2006.

[6] G. Ananthanarayanan and R. Katz. Greening the switch. In *In Proceedings of HotPower, December 2008*.

[7] D. Bertozzi, A. Raghunathan, L. Benini, and S. Ravi. Transport protocol optimization for energy efficient wireless embedded systems. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*, page 10706. IEEE Computer Society, 2003.

[8] J. Bresnahan, M. Link, R. Kettimuthu, D. Fraser, and I. Foster. Gridftp pipelining. In *Proceedings of TeraGrid*, 2007.

[9] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th annual international symposium on Computer architecture*, ISCA '00, pages 83–94, New York, NY, USA, 2000. ACM.

[10] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *In Proceedings of IEEE INFOCOM, April, 2008*.

[11] K. Chard, I. Foster, and S. Tuecke. Globus: Research data management as service and platform. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, page 26. ACM, 2017.

[12] Why cloudfuze: Quickly connect with powerful providers like google drive, dropbox, or box from a single screen and login. *https://www.cloudfuze.com/why-cloudfuze/*, 2016.

[13] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan. Full-system power analysis and modeling for server environments. In *Proc. of Workshop on Modeling, Benchmarking, and Simulation*, 2006.

[14] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. *ACM SIGARCH Computer Architecture News*, 35(2):13–23, 2007.

[15] K. Farkas, P. Huang, B. Krishnamurthy, Y. Zhang, and J. Padhye. Impact of tcp variants on http performance. *Proceedings of High Speed Networking*, 2, 2002.

[16] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997.

[17] N. Freed. SMTP service extension for command pipelining. http://tools.ietf.org/html/rfc2920.

[18] W. Fu and T. Song. A frequency adjustment architecture for energy efficient router. *ACM SIGCOMM Computer Communication Review*, 42(4):107–108, 2012.

[19] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav. It's not easy being green. *ACM SIGCOMM Computer Communication Review*, 42(4):211–222, 2012.

[20] E. Goma, M. C. A. L. Toledo, N. Laoutaris, D. Kosti, P. Rodriguez, R. Stanojev, and P. Y. Valent?n. Insomnia in the access or how to curb access network related energy consumption. In *In Proceedings of ACM SIGCOMM 2011*.

[21] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel. The cost of a cloud: Research problems in data center networks. In *In ACM SIGCOMM CCR, January 2009*.

[22] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Towards a next generation data center architecture: Scalability and commoditization. In *In ACM PRESTO, pages 5762, 2008*.

[23] M. Gupta and S. Singh. Energy conservation with low power modes in ethernet lan environments. In *IEEE INFOCOM (MiniSymposium) 2007*.

[24] M. Gupta and S. Singh. Greening of the internet. In *ACM SIGCOMM, pages 1926, 2003*.

[25] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, and M. Kandemir. Using complete machine simulation for software power estimation: The softwatt approach. In *Prpc. of 8th High-Performance Computer Architecture Symp.*, pages 141–150, 2002.

[26] T. J. Hacker, B. D. Noble, and B. D. Atley. Adaptive data block scheduling for parallel streams. In *Proceedings of HPDC '05*, pages 265–275. ACM/IEEE, July 2005.

[27] K. Hasebe, T. Niwa, A. Sugiki, and K. Kato. Power-saving in large-scale storage systems with data migration. In *IEEE CloudCom 2010*.

[28] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In *Proceedings of NSDI 2010*.

[29] M. Jain, R. S. Prasad, and C. Dovrolis. The tcp bandwidth-delay product revisited: network buffering, cross traffic, and socket buffer auto-sizing. 2003.

[30] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *SIGCOMM CCR 38(4):5162, 2008*.

[31] J. Kim, E. Yildirim, and T. Kosar. A highly-accurate and low-overhead prediction model for transfer throughput optimization. In *Proc. of DISCS Workshop*, November 2012.

[32] R. Koller, A. Verma, and A. Neogi. Wattapp: an application aware power meter for shared data centers. In *Proceedings of the 7th international conference on Autonomic computing*, pages 31–40. ACM, 2010.

[33] W. Liu, B. Tieman, R. Kettimuthu, and I. Foster. A data transfer framework for large-scale science experiments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 717–724. ACM, 2010.

[34] W. Liu, B. Tieman, R. Kettimuthu, and I. Foster. A data transfer framework for large-scale science experiments. In *Proceedings of DIDC Workshop*, 2010.

[35] D. Lu, Y. Qiao, P. A. Dinda, and F. E. Bustamante. Modeling and taming parallel tcp on the wide area network. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 68b–68b. IEEE, 2005.

[36] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. A power benchmarking framework for network devices. In *In Proceedings of IFIP Networking, May 2009*.

[37] F. Murtagh and P. Legendre. Wards hierarchical agglomerative clustering method: which algorithms implement wards criterion? *Journal of classification*, 31(3):274–295, 2014.

[38] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wether-all. Reducing network energy consumption via rate-adaptation and sleeping. In *Proceedings Of NSDI, April 2008*.

[39] U. of Minnesota. Minnesota internet traffic studies (mints), 2012.

[40] R. S. Prasad, M. Jain, and C. Dovrolis. Socket buffer auto-sizing for high-performance data transfers. *Journal of GRID computing*, 1(4):361–376, 2003.

[41] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for internet-scale systems. In *ACM SIGCOMM computer communication review*, volume 39, pages 123–134. ACM, 2009.

[42] F. Rawson and I. Austin. Mempower: A simple memory power analysis tool set. *IBM Austin Research Laboratory*, 2004.

[43] Rclone-rsync for cloud storage. *https://rclone.org/commands/rclone_sync/*, 2017.

[44] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. *HotPower*, 8:3–3, 2008.

[45] J. Semke, J. Mahdavi, and M. Mathis. Automatic tcp buffer tuning. *ACM SIGCOMM Computer Communication Review*, 28(4):315–323, 1998.

[46] C. Systems. Visual networking index: Forecast and methodology, 2015–2020, June 2016.

[47] S. V. Vrbsky, M. Galloway, R. Carr, R. Nori, and D. Grubic. Decreasing power consumption with energy efficient data aware strategies. *FGCS*, 29(5):1152–1163, 2013.

[48] E. Yildirim, E. Arslan, J. Kim, and T. Kosar. Application-level optimization of big data transfers through pipelining, parallelism and concurrency. *IEEE Transactions on Cloud Computing (TCC)*, 4(1):63–75, 2016.

[49] E. Yildirim, D. Yin, and T. Kosar. Balancing tcp buffer vs parallel streams in application level throughput optimization. In *Proceedings of DADC Workshop*, 2009.

[50] E. Yildirim, D. Yin, and T. Kosar. Prediction of optimal parallelism level in wide area data transfers. *IEEE Transactions on Parallel and Distributed Systems*, 22(12), 2011.

[51] D. Yin, E. Yildirim, and T. Kosar. A data throughput prediction and optimization service for widely distributed many-task computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6), 2011.

[52] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Y. Wang. Modeling hard-disk power consumption. In *FAST 2003*.