Elastic Data Transfer Infrastructure (DTI) on the Chameleon Cloud

Joaquin Chung, Zhengchun Liu, Rajkumar Kettimuthu, Ian Foster
Argonne National Laboratory. USA
Email: jchung@mcs.anl.gov, zhengchun.liu@anl.gov, kettimut@mcs.anl.gov, foster@anl.gov

***Abstract*—Many science workflows are distributed in nature and rely on wide area networks (WANs) to move data between geographically distributed resources for analysis, sharing, and storage. In spite of continued enhancements in campus cyberinfrastructure, data transfer nodes (DTNs) are grossly underutilized. Our previous analysis of logs from 1,800 DTNs shows that they were completely idle for 94.3% of the time in 2017. Motivated by the opportunity to optimize DTN usage, here we present an elastic data transfer infrastructure (DTI) architecture in which the pool of nodes allocated to DTN activities expands and shrinks over time, based on demand. Our results show that this elastic DTI can save up to ∼95% of resources compared with a typical static DTN deployment.**

I. Motivation

Data transfers over wide area networks (WANs) [1] may be negatively impacted by security configurations on campus networks. The Science DMZ [2] model allows research institutions to support high-speed wide area data transfers by moving data transfer nodes (DTNs) close to the campus network perimeter. DTNs at the Science DMZ [2] are optimized for high-speed scientific data transfer. However, we previously [3] analyzed GridFTP server usage logs from about 1,800 DTNs. We showed that DTNs are completely idle (i.e., no transfers) 94.3% of the time, and 80% of them are active less than 6% of the time.

One limitation of the current Science DMZ model is its statically provisioned DTNs, that are configured to sustain peak traffic demands. However, many science workflows are bursty, and thus their computing and network demands fluctuate significantly over time [4–7]. To handle these fluctuating demands efficiently, university campuses often consolidate compute resources into an elastic private cloud. Furthermore, elastic management of virtual network functions (VNFs) has been demonstrated on the Global Environment for Network Innovation (GENI) using software-defined networking (SDN) and control theory [8]. We argue that such an elastic infrastructure is required for data transfers as well, because statically provisioned DTNs result in underutilization of resources.

II. Elastic Data Transfer Infrastructure (DTI)

In this section we present the architecture, design, and implementation of an elastic data transfer infrastructure (DTI) that expands and shrinks dynamically to conserve resources.

Architecture: The elastic DTI architecture, shown in Fig. 1, is composed of a thin dedicated DTN, on-demand DTNs, an orchestrator, and agents. The thin dedicated DTN is a minimal resource replica of a regular DTN that is always online, while on-demand DTNs are dynamically allocated according to the load in the elastic DTI. The orchestrator runs on the thin dedicated DTN, and it makes decisions about allocating resources based on DTI usage measurements. Agents run on the thin dedicated DTN as well as on each on-demand DTNs, and they collect resource utilization statistics.

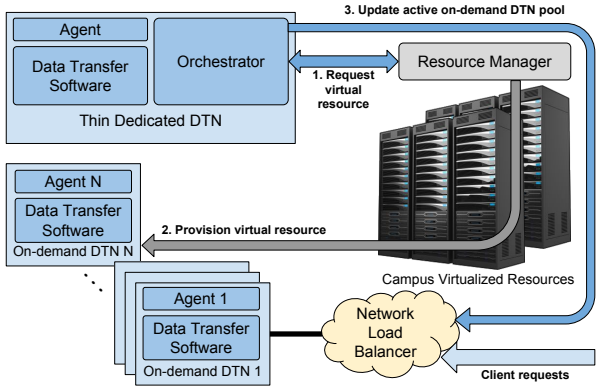

Fig. 1: Elastic DTI architecture

Design: To realize an elastic DTI, we need to answer the following question: What type of resources are more suitable for realizing an elastic DTI, and how do we (de)provision them? When do we (de)provision these resources? Where do we (de)provision resources?

The minimal resource unit that can be provisioned dynamically is a CPU core and portion of memory. Virtual machines and containers [9] can be used to (de)provision resources on demand. The question of when to provision or remove resource can be answered by applying different schemes on the decision engine ranging from simple thresholds to machine learning techniques. To decide where to (de)provision resources, we can delegate this task to existing high-performance computing resource managers and container orchestration systems (e.g., Kubernetes).

Implementation: We realized an instantiation of our proposed elastic DTI in the National Science Foundation's Chameleon testbed [10]. The implementation of our orchestrator and agents collect statistics on CPU usage, network throughput, and active transfers. Our elastic DTI is written in Python and uses gRPC for communication between orchestrator and agents. We chose Globus GridFTP as our

978-1-7281-2700-2/19/$31.00 2019

data transfer software because the extensive logs we had already collected [3] helped us create traffic traces for the evaluation. We used containers as the type of resources we can (de)provision and Docker as the campus resource manager. For this proof of concept, we implemented the communication between the orchestrator and agents using a polling model.

III. Evaluation on Chameleon

Experimental Setup: We evaluated our architecture in the Chameleon cloud [10] testbed, which provides bare metal nodes with 48 cores of an Intel Xeon CPU E5-2670 v3 with 2.30 GHz and 128 GB of RAM. Four of these bare metal nodes served as the underlay of our elastic DTI experiments. Our minimal resource unit is a Docker container with one core and 2.667 GB of RAM. All containers in the same bare metal node share the same 10 Gigabit Ethernet card. Our evaluations use a trace with transfers whose characteristics follow that of real datasets (both file size and dataset size) [3] and whose arrival times follow a Poisson distribution with $\lambda = 3$ seconds.

Results: We first ran microbenchmarks for (de)provisioning time of a node in the DTI, which on average is in the order of tens of milliseconds including communication overhead between Orchestrator and agents. For the evaluation of our elastic DTI, we focused on when and where to (de)provision resources. For when to (de)provision, we used both static thresholds (defined as percentages of average CPU utilization per container) and provisioning upon arrival (U.A.) of a new transfer. For where to (de)provision, we used the core (container) count and network throughput usage of the bare metal nodes as our metrics. These parameters generate four possible elastic DTI schemes as shown in Table I.

TABLE I: Elastic DTI Schemes

Scheme	When	Where
CPU+Count	CPU Usage	Core count
CPU+Net	CPU Usage	Network throughput
U.A.+Count	Upon arrival of new transfer	Core count
U.A.+Net	Upon arrival of new transfer	Network throughput

For our experiments, we measured the cores saving with respect to a baseline that matches the specifications of a typical static DTN [11]. For static threshold schemes, we derived four *(de)provisioning schemes* defined by a combination of high and low thresholds, and we used the *"HIGH/LOW"* notation for naming those schemes. We kept the low CPU usage threshold at 10% for the first three schemes and varied the high usage threshold as 30%, 50%, and 70%; for the fourth scheme the low threshold was 20% and the high threshold was 70%. Initial results show that at most ∼95% of the CPU core resources can be saved when compared with a typical DTN deployment (see Fig. 2). When using the U.A.+Count scheme, however, CPU savings reach only 20%.

IV. Conclusion

The Science DMZ as a network design pattern has had a notable impact on the science community by improving the performance of wide area file transfers significantly. Yet it falls short in efficient utilization of data transfer nodes. We presented the design, implementation, and initial evaluation of an elastic DTI that grows and shrinks based on demand. We realized an instantiation of this elastic DTI in the Chameleon testbed and showed that up to ∼95% of the CPU resources can be saved when compared with a typical DTN deployment. For future work, we will further investigate other metrics for when and where to (de)provision resources. We propose to evaluate adaptive thresholds for the elastic DTI schemes.

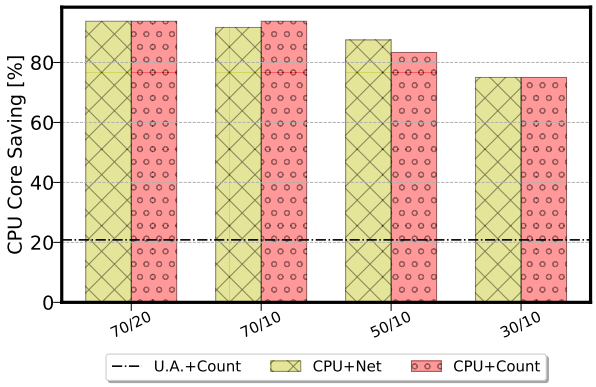

Fig. 2: CPU resources saved vs. typical DTN deployment

Acknowledgments

This work was supported by the U.S. Department of Energy, Office of Science contract DE-AC02-06CH11357. The authors will like to thank Paul Ruth and Kate Keahey for their help setting up the testbed on Chameleon.

References

[1] R. Kettimuthu, Z. Liu, D. Wheeler, I. Foster, K. Heitmann, and F. Cappello, "Transferring a petabyte in a day," *4th International Workshop on Innovating the Network for Data Intensive Science*, pp. 1–11, 2017.

[2] J. Crichigno, E. Bou-Harb, and N. Ghani, "A comprehensive tutorial on Science DMZ," *IEEE Communications Surveys & Tutorials*, 2018.

[3] Z. Liu, R. Kettimuthu, I. Foster, and N. S. Rao, "Cross-geography scientific data transfer trends and user behavior patterns," in *27th ACM Symposium on High-Performance Parallel and Distributed Computing*. ACM, 2018.

[4] R. F. da Silva, S. Callaghan, and E. Deelman, "On the use of burst buffers for accelerating data-intensive scientific workflows," in *12th Workshop on Workflows in Support of Large-Scale Science*. ACM, 2017, pp. 2:1–2:9.

[5] Z. Liu, R. Kettimuthu, S. Leyffer, P. Palkar, and I. Foster, "A mathematical programming- and simulation-based framework to evaluate cyber-infrastructure design choices," in *IEEE 13th International Conference on e-Science*, 2017, pp. 148–157.

[6] "Nuclear physics network requirements workshop, 2008," http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Np_net_req_workshop.pdf.

[7] Y. Sun, S. Marru, and B. Plale, "Experience with bursty workflow-driven workloads in LEAD science gateway," in *TeraGrid Conference*, 2008.

[8] "NFV Tutorial: Managing a Virtual Network Function using SDN and Control Theory," https://groups.geni.net/geni/wiki/GENIExperimenter/Tutorials/NFV/Ryu, accessed: 2019-08-19.

[9] linuxcontainers.org, "Linux containers," https://linuxcontainers.org/.

[10] "Chameleon Cloud," https://www.chameleoncloud.org/.

[11] "Data Transfer Node Reference Implementation," https://fasterdata.es.net/science-dmz/DTN/reference-implementation/, accessed: 2018-12-14.