Future Generation Computer Systems 89 (2018) 10–18

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Toward a smart data transfer node

Zhengchun Liu [a,*], Rajkumar Kettimuthu [a], Ian Foster [a,b], Peter H. Beckman [a]

[a] Mathematics & Computer Science Division, Argonne National Laboratory, Lemont, IL 60439, USA
[b] Department of Computer Science, University of Chicago, Chicago, IL 60637, USA

HIGHLIGHTS

- A DTN powered by reinforcement learning to achieve self-optimization.
- A knowledge engine that can identify optimal parameter values for a DTN.
- A knowledge engine guides a DTN to achieve stabler and better performance.

ARTICLE INFO

Article history:
Received 31 January 2018
Received in revised form 23 May 2018
Accepted 18 June 2018
Available online 21 June 2018

Keywords:
Data transfer node
Smart computing
File transfer
Reinforcement learning

ABSTRACT

Scientific computing systems are becoming significantly more complex, with distributed teams and complex workflows spanning resources from telescopes and light sources to fast networks and Internet of Things sensor systems. In such settings, no single, centralized administrative team and software stack can coordinate and manage all resources used by a single application. Indeed, we have reached a critical limit in manageability using current human-in-the-loop techniques. We therefore argue that resources must begin to respond automatically, adapting and tuning their behavior in response to observed properties of scientific workflows. Over time, machine learning methods can be used to identify effective strategies for autonomic, goal-driven management behaviors that can be applied end-to-end across the scientific computing landscape. Using the data transfer nodes that are widely deployed in modern research networks as an example, we explore the architecture, methods, and algorithms needed for a smart data transfer node to support future scientific computing systems that self-tune and self-manage.

Published by Elsevier B.V.

1. Introduction

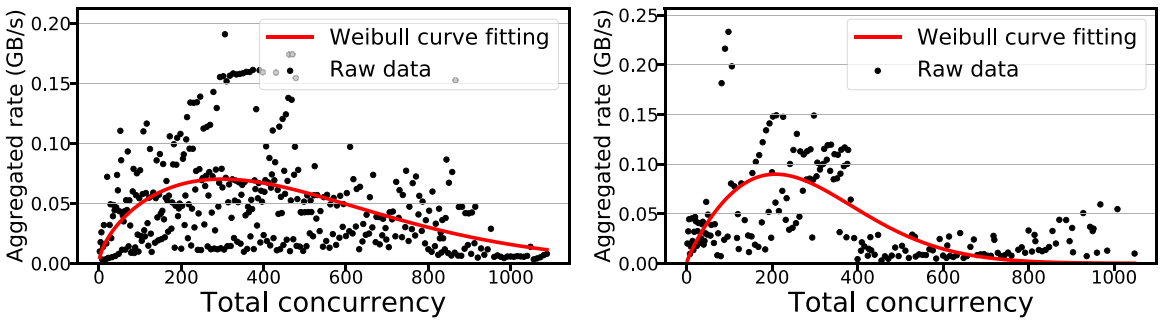
Scientific computing systems are becoming significantly more complex and have reached a critical limit in manageability using current human-in-the-loop techniques. To address this situation, we need to devise autonomic, goal-driven management actions, based on machine learning, applied end-to-end across the scientific computing landscape. The high-performance computing center was previously the nexus of the scientific computing universe, both administratively and computationally. Users brought their codes and their data to computing facilities, and the operational teams managing the systems carefully configured and monitored systems to achieve the required uptimes and queue wait times. As science workflows get complex, spanning distributed resources and involving a distributed team of researchers, no single, centralized administrative team and software stack can coordinate and manage all the resources. Thus, smart systems that achieve self-configuration, self-optimization, self-healing, and self-protection have garnered the attention of researchers in both academia and industry [1–5].

Data transfer nodes (DTNs) [6] are compute systems dedicated for data transfers in distributed science environments. In previous work [7,8], we determined via the analysis of millions of Globus [9] data transfers involving thousands of DTNs that DTN performance has a nonlinear relationship with load. Aggregate DTN throughput first increases with transfer load but, after a threshold, decreases because of overload (see Fig. 1). A DTN thus has an optimal operating point. As instantaneous DTN load is determined by the characteristics of the transfers that are currently running, which are in turn determined by the parameters of those transfers, the optimal operating point cannot easily be determined analytically.

An ideal scheduling algorithm will ensure that a given DTN always works in its optimal operating point. since resources used to transfer data over wide area network – for example, networks and storage systems at the source and destination endpoints are shared with other applications – a static policy is powerless to deal with these dynamics. Analytical methods are inadequate in capturing the collective impact of application parameters because these

* Correspondence to: Bldg. 240, Argonne National Lab., 9700 S. Cass Avenue, Lemont, IL 60439, USA.
E-mail addresses: zhengchun.liu@anl.gov (Z. Liu), kettimut@anl.gov (R. Kettimuthu), foster@anl.gov (I. Foster), beckman@mcs.anl.gov (P.H. Beckman).

https://doi.org/10.1016/j.future.2018.06.033
0167-739X/Published by Elsevier B.V.



(a) DTN at NERSC (b) DTN at JLab

Fig. 1. Aggregate incoming transfer rate vs. total concurrency (i.e., instantaneous number of GridFTP server instances) at two heavily used endpoints, with Weibull curve [10] fitted.
Source: [7].

parameters are often categorical and their impact on performance and power usage is opaque to such methods [11]. We report here on a preliminary study in which we apply a deep reinforcement machine learning-based knowledge engine to power a DTN with smartness in order to achieve self-awareness, self-configuration, and self-optimization. Our goal is to make the DTN always run at its optimal operating point (or at least avoid overloading) if there are sufficient transfer tasks. The key difference between this paper and other studies [12,13] on optimizing wide-area data transfer performance is that we try to maximize the aggregated throughput of a DTN, whereas others try to optimize the throughput of a given transfer.

The rest of the paper is organized as follows. In Section 2 we present the architecture of a smart cyberinfrastructure in which each subsystem has the ability to act autonomously. In Section 3 we detail the design and implementation of a smart DTN. Our experiment results are discussed in Section 4, where we present two experiments to show the effectiveness of the knowledge engine. In Section 5 we review related work, and in Section 6 we summarize our conclusions and briefly discuss future work.

2. Motivation

Extraordinary advances in computing, communication networks, and information technologies have produced an explosive growth of highly interconnected systems, which are increasingly becoming complex, dynamic, heterogeneous, labor intensive, and challenging to operate and manage with existing approaches [14]. For large organizations such as DOE's science community, with thousands of geographically interconnected systems, traditional distributed systems operation and management based on static behaviors, interactions, and configuration are proving to be inadequate. We define a system architecture in which, as shown in Fig. 2, each edge resource has its own knowledge engine (KE). This component acts as the "brain" of an edge resource, generating control commands based on the current system state as determined from monitoring data and on learned knowledge of the relationship between actions and cost/benefits. Thus, the KE has three key components: (1) input features that reflect the current state of the physical system, (2) output control commands that steer the physical system to operate optimally for current tasks, and (3) machine learning models that capture and optimize the relationship between input and output.

Fig. 2 shows an architecture of science autonomous infrastructure, which consists of five layers: fabric, perception, processing, collective, and application. The **fabric layer** consists of the resources or things in the smart science ecosystem: for example, computational resources, storage systems, network resources, and instruments. The **perception layer** gathers static and dynamic information about the resources to discover the state, structure and capabilities as well as provides mechanism to steer (or perform actions on) the resources. The *processing layer* stores, analyzes, and processes data that comes from the perception layer. It can use learning algorithms to detect patterns, find anomalies, predict performance and make proactive decisions. The **collective layer** defines the necessary communication and authentication protocols and mechanisms for the exchange of data among the KEs in the processing layer of different resources in the science ecosystem. The **application layer** comprises of the user applications within the smart science ecosystem.

As one can see, each component in Fig. 2 is powered by a KE capable of sensing information from the environment and/or other components and optimizing itself by learning from its history. In the current era of distributed and data-intensive science, data movement is a critical aspect. Thus, in this work we focus on designing a smart data transfer node, a first step toward designing a smart distributed science ecosystem.

Here, a DTN is typically a PC-based Linux server, built with high-quality components and configured specifically for wide-area data transfer. It is a key component in distributed science environments [15], and its performance has direct influence on the productivity of the whole ecosystem. A DTN typically mounts the parallel file system that serves the compute cluster and is connected to a high-speed wide area network. For a given transfer, a DTN either pulls data from the storage and sends the data over its network interface card or receives data from its network card and writes the data to its storage system. Heavily used DTNs, such as those at national supercomputer centers, serve as both source and destination for multiple concurrent transfers [16]. Based on our extensive study of millions of Globus transfer logs [7,17], we find a big space for improvement in data transfer performance by optimizing DTN behavior. Self-configuration and self-optimization are two of the key features we achieve in this paper. Basically, these features mean that the DTN is self-aware and knows how to steer itself. The following are a few examples of ways in which a smart DTN can adapt its behavior to optimize desired outcomes.

- **Network packet pacing.** Packet pacing can improve performance [18], but its value depends on the characteristics of the destination endpoint. Typically, it is desirable only when transferring data from a higher bandwidth endpoint to a lower bandwidth endpoint. Since a DTN may transfer data to endpoints of different types (e.g., other DTNs, desktops, laptops), a smart DTN should apply pacing differentially to different edges (source to destination endpoint pair) based on destination capabilities.
- **File transfer order.** Overall performance can be improved by rearranging the order in which files are transferred based