Accepted Manuscript

Lark: An effective approach for software-defined networking in high throughput computing clusters

Zhe Zhang, Brian Bockelman, Dale W. Carder, Todd Tannenbaum

PII: S0167-739X(16)30054-1
DOI: http://dx.doi.org/10.1016/j.future.2016.03.010
Reference: FUTURE 2984

To appear in: *Future Generation Computer Systems*

Received date: 14 November 2015
Revised date: 17 February 2016
Accepted date: 14 March 2016

Please cite this article as: Z. Zhang, B. Bockelman, D.W. Carder, T. Tannenbaum, Lark: An effective approach for software-defined networking in high throughput computing clusters, *Future Generation Computer Systems* (2016), http://dx.doi.org/10.1016/j.future.2016.03.010

Lark: An Effective Approach for Software-Defined Networking in High Throughput Computing Clusters

Zhe Zhang[a], Brian Bockelman[a], Dale W. Carder[b], Todd Tannenbaum[b]

[a]*Department of Computer Science and Engineering, University of Nebraska-Lincoln, USA*
[b]*Department of Computer Science, University of Wisconsin-Madison, USA*

Abstract

High throughput computing (HTC) systems are widely adopted in scientific discovery and engineering research. They are responsible for scheduling submitted batch jobs to utilize the cluster resources. Current systems mostly focus on managing computing resources like CPU and memory; however, they lack flexible and fine-grained management mechanisms for network resources. This has increasingly been an urgent need as current batch systems may be distributed among dozens of sites around the globe like Open Science Grid. The Lark project was motivated by this need to re-examine how the HTC layer interacts with the network layer.

In this paper, we present the system architecture of Lark and its implementation as a plugin of HTCondor which is a popular HTC software project. Lark achieves lightweight network virtualization at per-job granularity for HTCondor by utilizing Linux container and virtual Ethernet devices; this provides each batch job with a unique network address in a private network namespace. We extended HTCondor's description language, ClassAds, so users can specify networking requirements in the job submission script. HTCondor can perform matchmaking to make sure user-specified network requirements and resource-specific policies are fulfilled. We also extended the job agent, *condor_starter*, so that it can manage and configure the job's network environment. Given this important building block as the core, we implement bandwidth management functionality at both the host and network levels utilizing software-defined networking (SDN). In addition to HTCondor, Wide area network bandwidth management for GridFTP traffic is designed and implemented. Our experiments and evaluations show that Lark can effectively manage network resources simultaneously for both applications inside the cluster environment. By not resorting to heavyweight VMs, we keep startup overheads minimal compared to "regular" batch jobs. This mechanism provides the users with better predictability of their job execution and the administrators more policy flexibility in allocation of network resources.

Keywords: high throughput computing, HTCondor, bandwidth management, software-defined networking, network-aware scheduling

1. Introduction

Cluster computing has become the workhorse powering scientific discovery and engineering research. At the heart of many compute clusters is a general-purpose workload management batch system such as PBS [1] or Grid Engine [2]. The goal of these batch systems is to ensure submitted jobs ultimately run to completion on cluster resources according to a supplied policy. Achieving this goal is facilitated by resource management mechanisms including scheduling, monitoring, accounting, and binding to specific jobs [3]. However, batch systems rarely treat the network as a first-class resource that can be directly managed and bound to jobs. SLURM [4] can perform topology-aware job placement to reduce communication overhead; however, it cannot interact with the network layer directly, thus leading to limited functionality. For example, policies that allocate or prioritize network access to different sets of jobs cannot be effectively monitored or enforced. Network-based tools on their own are of little help to enact job-specific policies because the network has no ability to distinguish between different jobs running on the same host.

The need for batch systems to manage the network is increasing as the size of data sets being processed grows, and as high-throughput computing clusters have become federated into wide-area international computing grids that connect hundreds of clusters across dozens of countries [5]. In these sorts of environments, as well as single cluster installations, HTCondor [6] is a widely deployed workload management system in both commercial and academic settings. Traditionally, the computing resources fully managed by HTCondor have been CPU, memory, and disk. Accordingly, we started the Lark project that aims to make networking a first-class managed resource of HTCondor. In this paper, we explain how the Lark software leverages recent functionality in the Linux kernel to implement a mechanism that provides a unique network identity for each compute job by binding jobs to unique network address. Given a one-to-one mapping of network addresses and jobs, HTCondor can now interact with and alter the network layer based on its internal policies. We also explain how Lark defines new job and machine attributes in a policy language, allowing users to describe networking requirements at job submission time and resource administrators to enforce network policies. The existing HTCondor matchmaking techniques can then match jobs to appropriate hosts.