Programming SDN-Native Big Data Applications: Research Gap Analysis

Khaled Alwasel and Yinhao Li
Newcastle University, UK

Prem Prakash Jayaraman
Swinburne University of Technology, Australia

Saurabh Garg
University of Tasmania, Australia

Rodrigo N. Calheiros
Western Sydney University, Australia

Rajiv Ranjan
Newcastle University, UK

Software-Defined Networking has involved as a preferred abstraction for sharing network resources within a cloud datacenter in response to simultaneous data retrieval and computation demands from around the world. However, several research challenges need to be investigated before SDN powered-cloud datacenters are able to efficiently process big data as defined by its "4V" characteristics. Big data enabled-systems have to be able to respond to concurrent requests and allocate computing (e.g., virtual machine instances), storage (e.g., disk space) and networking (e.g., bandwidth) resources efficiently and effectively.

Traditionally, most cloud datacenters are based on monolithic networking systems where network intelligence is distributed among forwarding devices (switches and routers), that require very complex protocols to operate and low-level configuration management (e.g., via command-line interface).[3] As a result, it is widely accepted in industry and academia that traditional monolithic networking systems impose significant limitations on cloud datacenters hosting big data applications. These include: (1) promoting monopoly and silos by introducing proprietary, dedicated network infrastructure hardware; (2) raising the cost for obtaining and maintaining several hundred to thousands of heterogonous and complex network components; (3) lacking capabilities to meet application quality of service (QoS) and service level agreement (SLA) demands at run-time; and (4) requiring specialized expertise and specific training to integrate and maintain networks from different vendors. Software-defined networking (SDN) has emerged as a promising solution to overcome such drawbacks of the traditional datacenter network approach.

Software-Defined Networking—Overview

SDN is a networking approach originally derived from the work of Martin Casado in 2005. It became popular with the invention of the OpenFlow (OF) SDN protocol in 2007.[4–6] OF is the new protocol to control the *flows of streams of packets* in the network. SDN was initially designed to serve a specific purpose, which was to simplify the process of network management and configuration for network

 2325-6095/17/$33.00

administrators. The four main pillars[1,3] of SDN include:

- Separation of control and data planes
- The network devices can be programmed through software applications and/or application programming interfaces (APIs)
- Logical centralization of (possibly physically distributed) network control (e.g., routing logic, bandwidth assignment) logic to an SDN controller which has global view of data flow across network
- Forwarding decisions are flow-based (stream of packets) rather than packet-based

SDN breaks the ideology of self-driven device decisions (distributed intelligence) and integrates the network intelligence into a logically centralized SDN controller.[3] Decoupling network control mechanisms from devices (switches and routers) may allow datacenter operators to achieve a global network view, superior network QoS optimization, fine-grained control, improved network consistency, partial/total reconfiguration of networks, and fast failure detection/recovery. It may also enable optimal performance, either solely through network-based load balancing in anycast service environments, or through selection of an anycast service node using a combination of network and server conditions such as congestion.[7]

Figure 1 shows an example of the SDN architecture, which consists of three layers: data, control, and management.[3] First, the data layer contains forwarding devices that implements the OF protocol. OF allows the remote management and access of heterogonous forwarding devices without exposing their low-level, internal designs and functionalities. Every forwarding device maintains an OF routing table, allowing an SDN controller to dynamically configure the state of devices such as add, retrieve, remove, and/or update routing entries (rule, action, and stats) on behalf of respective management layer network manipulation functions (e.g., routing, monitoring, traffic load-balancing, etc.).

Second, the control plane contains an SDN controller, which exposes southbound and northbound APIs to the data and management layers, respectively. The SDN controller leverages OF table (as shown in Figure 1) of OF protocol for sending data flow rules (rule, action, stats) to network devices at the data plane layer. The OF talbe can be dynamically updated across network devices, which is what they (router and switches) use for routing frame and packets across the network.

Third, the management layer plane consists of various network applications to configure (e.g., firmware updates), deploy, control, and orchestrate the entire network, without having direct communications with forwarding devices in the data plane.

Though the SDN paradigm provides a well-defined communication standard (OF) and programming interface (northbound and southbound APIs) to the datacenter network, the *availability of software frameworks and algorithmic techniques* to dynamically configure and optimize the SDN-based datacenter network for improving big data application performance and availability is still nascent.

In the SDN landscape, network configuration is accomplished using one or multiple SDN controllers by deploying network policies, protocols, and algorithms. In a broader sense, SDN controllers have default network configuration mechanisms for resource discovery and maintenance of network devices. From such default configurations, SDN controllers operate and orchestrate traffic flows once their intended networks are up and running. As discussed in Figure 1, the management layer is the place where configuration parameters are specified by network management applications (routing, load balancing, etc.) and injected into SDN controllers, resulting in network behavior changes. In other words, SDN can enable scheduling network resources in terms of network paths based on packets and flows of respective server/host/virtual machine (VM) IP addresses and implemented network algorithms (equal-cost multi-path, priorities, etc.). SDN technology, in a broader sense, should be leveraged to improve the run-time QoS of all types of cloud-hosted applications (e.g., content delivery networks, n-tier web applications, and scientific applications). However, each of these different application types has diverse data and control flow complexities as well as heterogeneous QoS demands. This article, therefore, will point to the challenges in programming, configuring, deploying and managing SDN-native big data applications. It is well

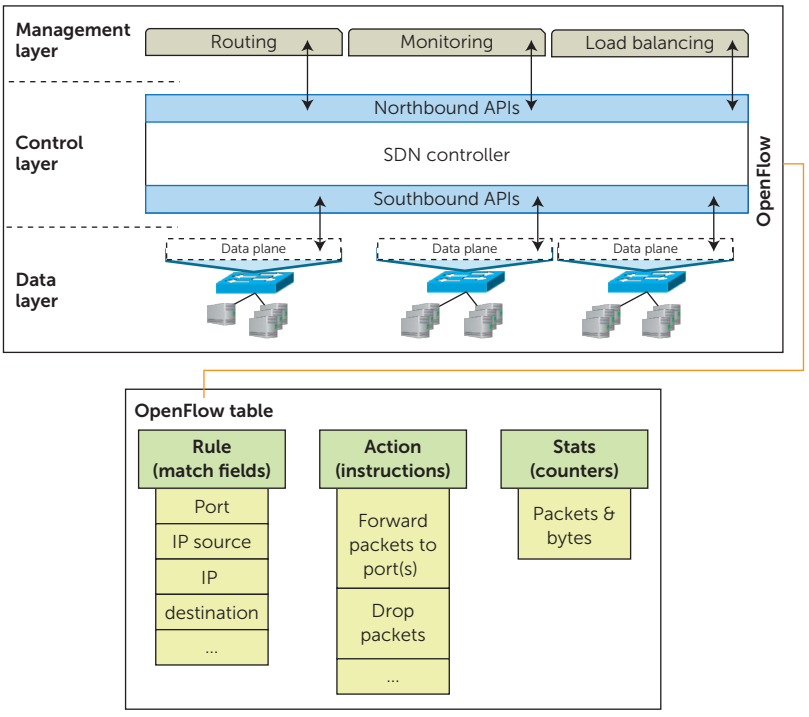

FIGURE 1. Software-defined networking and OpenFlow.

understood and validated by industry and academic experts that big data application workflows[8] incur excessive resource demands on datacenter network, storage, and compute resources.

Programming of Big Data Application Workflows—Overview

Big data frameworks (such as Apache Hadoop, Apache Spark, Apache Storm, and Apache Kafka) are increasingly recognized as powerful solutions to develop and deploy big data applications in cloud datacenters. Due to their ability to handle the 4Vs of big data, they are considered as suitable alternatives to traditional data processing frameworks (e.g., Microsoft Excel, MySQL, Oracle, etc.). Each big data framework deals with different aspects of big data requirements. For example, a batch-processing framework (such as Apache Hadoop; http://hadoop.apache.org) analyses data in simultaneous and stateless fashions, while a stream processing framework (such as Apache Storm; http://storm.apache.org) handles continuous big data streams in a consecutive and stateful manner.[1] Such big data frameworks have found application in several domains such as public healthcare, business sectors and financial trading for solving problems, such as curing chronic diseases, understanding consumer behaviors, and detecting frauds, to name a few.

Big data management systems (BDMS), also known as resource negotiators or data operating systems, provide a generic programming layer for managing lifecycle operations related to big data application workflows (e.g., composition, mapping, QoS monitoring, and dynamic reconfiguration).[8] These BDMSs such as Apache Hadoop YARN (https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html) and MESOS (http://mesos.apache.org/) implement intelligent algorithmic techniques to obtain superior performance among diverse big data applications hosted on shared datacenter resources (see Figure 2). Programming BDMSs and the underlying big data frameworks in a SDN-native fashion would lead to several advantages, such as the amount of network resources allocated to each big data framework hosted on a cloud dataceenter

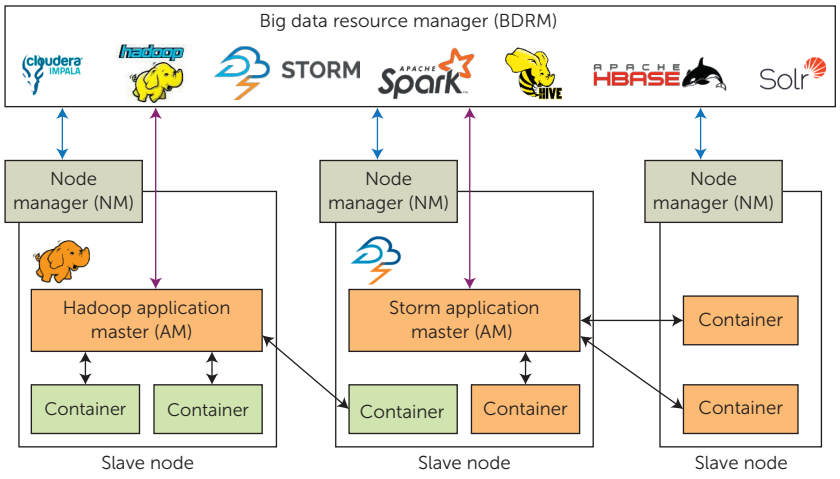

FIGURE 2. Big data cluster (resource management).

can be controlled and dynamically reconfigured to meet changes in data flows (4Vs). It will enable explicit allocation of bandwidth to one or more big data frameworks in periods of high data flow, and release bandwidth in favour of one or more other big data frameworks in periods of low data flow.

Consider a disaster management big data application workflow such as flood monitoring, detection and response as described in Center for Earth Systems Research[9] and Ranjan et al.[8] managed by a BDMS such as YARN. Such an application workflow is dependent on data from multiple sources including weather stations, traffic flow, water depth, and water flow sensors. Other information sources include population of the area, emergency evacuation and alerting plans, location of emergency response teams, etc. It is clear that heterogeneous[8] big data frameworks will process each of these data sources.

During the normal mode of operations, the sensors and other data sources transmit data at moderate volume and velocity to big data frameworks hosted in cloud datacenters via a network of routers and switches. In case of the occurrence of a flooding event, the sensors and other data sources stream data into the cloud at much higher volume and velocity in order to parametrize, calibrate, and validate the flood prediction models in real-time. However, this leads to new run-time QoS requirements for network providers. These include (1) the ability to obtain data from the location of flooding at a higher velocity and volume; (2) reconfiguring the inter-networking routes to provide priority paths to the data originating from data sources near to the flooding area; (3) reconfiguring the sensor and data sources to produce data at a higher frequency; and (4) additional resources to be allocated to big data frameworks in the cloud for managing and processing the big data stemming from multiple data sources.

To address the aforementioned challenges and the QoS demands imposed by big data application workflows such as flood modeling programmed using multiple big data frameworks on cloud datacenters, the BDMSs and the underlying network need to work collaboratively in a coordinated fashion. However, achieving this is a non-trivial activity and requires seamless exchange of data, QoS requirements, application demands, data sources capabilities across each layer of the cloud, and the inter-networks (in case of an SDN, this will include the data, control, and management plane as described earlier).

Why Program SDN-Native Big Data Applications?

SDN is a promising technology and when tightly integrated with the resource management capabilities offered by datacenters' server hypervisors BDMSs (e.g., Apache YARN, Apache MESOS) can provide significant benefits as regards to improving application and network QoS. In a broader

sense, making cloud applications (including big data applications) SDN-native can provide remarkable benefits including: (1) avoiding traditional network issues (route congestion and network bottlenecks); (2) enhanced scheduling decisions via cross-layer (server-to-network and vice versa) QoS optimization (availability, load, throughput, available budget, etc.) at run-time; (3) creating dynamic routes for application-specific data flows; (4) higher degree of run-time control of quality of services (throughput, delay and packet loss), software quality assurance (e.g., fault-tolerance, reliability, etc.), and SLAs; (5) availability of APIs for run-time network traffic engineering (isolation, aggregation, characterization, and route reservation); and (6) dynamic network reconfiguration in response to physical network link failures and improper network behaviors.

Big Data Management System (BDMS)—Overview

Let's focus on YARN in order to illustrate a typical BDMS that can be used to program large-scale application workflows. YARN decouples the resource management responsibilities and big data programming frameworks into two separate components. It consists of four elements: a central resource manager, node managers, application masters (e.g., Apache Hadoop's master node, Apache Storm's master node, etc.), and worker containers, as shown in Figure 2. The big data resource manager (BDRM) is a central authority that schedules and reconciles resource contention among different application masters. A node manager (NM) is responsible for managing node resources (memory, CPU, etc.) and reporting node failures/availability to BDRM. Every newly deployed big data framework has an application master (AM), which is in charge of the application lifecycle and resource negotiations with BDRM (e.g., number of containers, priorities, and preferred locations). Last, a container is a lightweight, virtualized resource deployed on a slave node to run an application master-specific data processing task.

From the user point of view scheduling and resource allocation, each big data framework application-master explicitly specifies the amount of memory, CPUs, and queues to be allocated for its containers and application master to the YARN's BDRM. Following that, the BDRM tries to reserve such resource requirements using resource availability information obtained from the NM. The resource allocation decision can be affected by many factors such as SLA (e.g., data processing deadlines, available budget) requirements. YARN has a built-in configuration management component in order to automatically configure and deploy resources and applications. In order to configure YARN, numerous XML files need to be configured based on users' requirements, such as *yarn-site.xml* (https://hadoop.apache.org/docs/r2.7.0/hadoop-yarn/hadoop-yarn-common/yarn-default.xml), *hdfs-site.xml*, *mapred-site.xml*, etc. Every file contains hundreds of parameters that need to be specified explicitly, such as the amount of memory and CPU for every container. Once the configuration files are in place, YARN starts the deployment automatically (e.g., RM, NM, containers, etc.). Furthermore, for configuring YARN in cloud environments (e.g., EC2), users can leverage the power of deployment tools such as Fabric and Puppet, which leads to greater flexibility and automation.

Integration of BDMS with SDN—Current State of the Art

As big data applications can be compute-intensive, data-intensive, or both, it is of importance to tune network and server component configurations constantly. To achieve this, BDMSs such as YARN and MESOS should be able to schedule and allocate both compute and network resources accordingly. From the system point of view, scheduling and resource allocation spans many levels—from the allocation of storage (e.g., HDFS), to allocation of virtualized compute (server) resources among different types of application containers (e.g., Hadoop, Storm, Spark, etc.), to allocation of network bandwidth and routes to application masters and containers depending on the data flow behaviors. However, from the network point of view, YARN and other BDMSs have no authority to configure, schedule, and dynamically allocate network resources; instead, it assumes the networks' best-effort delivery for the data flow across the application containers. Moreover, YARN and other BDMSs have no built-in intelligence for managing network traffic; hence, it delegates such tasks to underlying network controllers. To summarize, existing BDMSs have no capabilities that can allow

them to leverage the power of SDN in order to optimize the datacenter network scheduling and allocation based on data flow across big data frameworks.

Nevertheless, current SDNs are network-centric, having no embedded capabilities to support requirements posed by higher-level big data application workflows and their respective BDMSs. Moreover, existing SDNs do not directly provide support for any network scheduling capabilities to BDMSs. Therefore, further work is required to develop holistic integration between SDN-BDMSs that considers different scheduling requirements of big data application workflows, users QoS/SLA, and run-time data flows.

For the last few years, some research has been conducted in the context of integrating a particular type of big data framework (e.g., Apache Hadoop) with SDN for achieving faster and efficient data processing. However, there have been very few efforts that really propose any framework or system to achieve this. Qin et al.[10] proposed a scheduling system called BASS (Bandwidth Aware Scheduling with SDN) which integrates the scheduling engine of Hadoop with SDN Controller. It utilizes an SDN controller to obtain the bandwidth availability and allocates tasks locally or remotely based on network bandwidth. This work shows promising results. Han et al.[11] proposed an architecture of an SDN-enabled content delivery network system. They utilized a Social TV analytics application on Hadoop as the use case and utilized the flow forwarding feature of SDN. However, none of the existing algorithmic techniques (for scheduling and resource allocation) available in the literature are able to exploit the full potential of SDN in context of complex big data application workflows (that combines multiple big data frameworks and heterogeneous data flows from multiple sources) managed by systems such as YARN and MESOS. In the next section, we propose a novel architecture for programming an SDN-native BDMS that combines the resource management capabilities of BDMSs with the network control and management capabilities of SDN.

An Architecture for Programming SDN-Native Big Data Application Workflows

To the best of our knowledge, there has been relatively little research on the development of a holistic programming architecture for making BDMSs SDN-native. To this end, we propose a system-level description of SDN-native BDMS architecture (Figure 3), which focuses on three main components: configuration management, scheduling, and traffic management.

As shown in Figure 3, the key layers of an SDN-native BDMS architecture are resource, control and management, and application. The application layer is made of diverse big data application workflows that leverage instances of one or more big data frameworks. The resource layer contains compute, storage, and networking resources. The compute and storage resources will be managed by traditional BDMS APIs and network resources are managed by an SDN controller using southbound APIs. The requirements of these different application workflows and current resource availability will be monitored by the middle layer (control and management layer). This layer will interact with an SDN controller to manage network resources. This management layer is comprised of a new, intelligent BDRM (to be realized within Apache YARN or Apache MESOS), containing three components (scheduling, traffic, and configuration managers).

Scheduling manager: The role of scheduling manager is to make decisions about which application master and containers (an instance of big data processing framework) should utilize which storage, server, and network resources and at what time. It will also need to dynamically provision (on some instances reprovision) network resources based on data flow and analytics workload seen across the big data processing frameworks. In order to dynamically allocate network resources, the scheduling management will need to constantly monitor the data flow across big data processing frameworks using a new class of APIs that connects to both BDMSs as well as the SDN controller. The scheduling manager will need to counter various scenarios including contention and diverse QoS/SLA requirement across multiple application workflows (priorities, fairness, deadline, budget, urgency, etc.). In general, such scheduling problems are NP-hard, thereby requiring computationally complex heuristics, which must act in near-real time, increasing the overall challenges of building such systems.

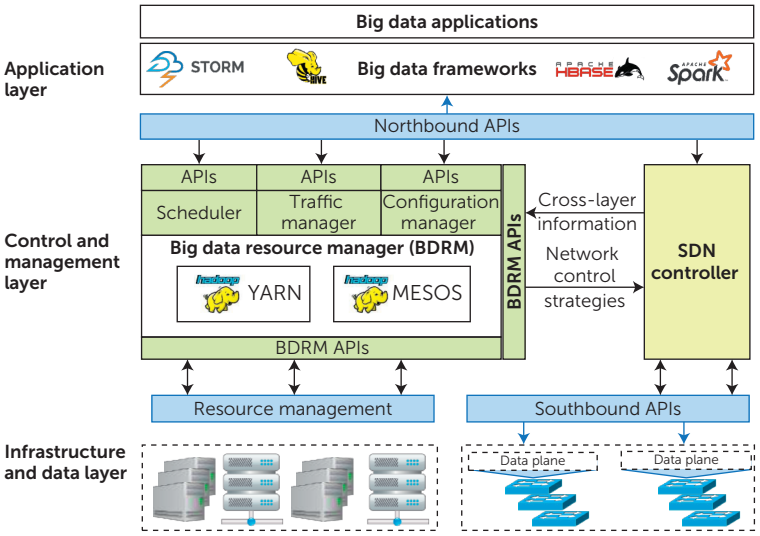

FIGURE 3. SDN-native big data application programming architecture.

Traffic manager: The traffic manager monitors the network availability by interacting with the SDN controller using BDMS APIs. Given the network requirements may vary over time, the traffic manager will need to update the network information regularly. The more up-to-date the information is the better the network resource allocation decision the scheduling manager will be able to take. To this end, the traffic manager will need to monitor network link congestion, failure, and bandwidth utilization.

Configuration manager: A configuration manager is one of the most important components of the system architecture, which performs various system operations and configurations on behalf of users. In a general sense, it is responsible for configuring the entire resource stack (e.g., big data processing frameworks, server, storage, and network). It should provide a simpler and automated application deployment with higher accuracy, flexibility, and fault tolerance compared with manual scripting-based methods. The configuration manager will obtain the application deployment graph (topology connecting application masters and containers) from the scheduling manager to configure compute, storage, and network resources in a very flexible and efficient manner. As configurations should be performed programmatically, there is less chance of errors and inconsistencies.

Future Research Directions for Programming SDN-Native Big Data Applications

To implement the proposed SDN-aware BDMS and conduct scheduling, traffic, and configuration management in holistic manner, there are several research challenges that need be tackled:

- *Integrating network requirements of diverse application workflows:* It is important to first understand data flows in the life cycle of a big data application, then based on this, to define network requirements for each data processing component (batch processing vs. stream processing vs. transactional). For example, BDMS-MapReduce batch processing applications often require good network performance during the mapping phase (copying of initial data to map functions) and shuffling phase (copying of intermediate data to reduce functions). In comparison, stream applications often demand continuous, excellent network performance as they process unbounded real time data.
- *Simultaneous consideration of volume and velocity in SDN-BDMS:* With more and more application workflows concentrating on real-time data ingestion and processing, there is a requirement to investigate techniques that not only enable scheduling decisions considering volume aspects

as done in previous studies but also velocity. The new class of scheduling technique will need to reconcile high-speed (including burstiness) arrival of data to the BDMS that needs to be processed in a timely manner along with large volumes historical data. The former aspect, typically addressed by streaming frameworks (such as Apache Spark and Heron), demands fine-grained resource management and scheduling: As the arriving flow of data varies in intensity, different amount of storage, server, and network resources need to be allocated to process this real-time data. These frameworks usually manage it at operator level, where an operator is a small computation that is applied to each received tuple, and the stream application is composed of a workflow of such operators. The SDN controller needs to constantly optimize network resources to handle different operator workflows and variation in the velocity of data arrival, while maintaining enough resources for batch processing frameworks (e.g., Apache Hadoop) that need to co-exist in the BDMS. Research is needed towards a new class of scheduling algorithms to enable such advanced, cross-layer, cross-framework resource management. Computing optimal configuration across network, storage, server, and big data processing frameworks in response to changing data volume and velocity is NP-complete individually. In other words scheduling problems in the context of SDN-native big data applications can be classified as strongly NP-complete ones,[12] as can even the simple problem of compute or storage resource allocation in distributed computing architectures.[13]

- *Conflicting resource requirements of multiple applications:* In a large distributed environment such as a cloud datacener, multiple big data application workflows are simultaneously competing for the resources. Many of the applications' resource requirements change over time and depend on the time and space. For example, during non-emergency periods, it is not essential to store or process all the weather data, as discussed earlier. However, during emergency situations, the collected data needs to be fine grained to enable better disaster management. Thus there is a need of deriving new, dynamic scheduling policies to deal with resource management based on various factors (priorities, fairness, etc.). This is made even harder due to the conflicting objectives of minimizing aggregate resource requirements while attempting to provide "near-infinite" elasticity to each individual workload.
- *Real-time network monitoring and configuration:* In general, it is almost impossible to permanently reserve network resources in a large cloud datacenter where many application workflows will be sharing resources based on dynamic resource allocation and mapping/binding. As a massive number of application workflows randomly and continuously run on cloud networks, it is impossible to assure network configurations are stable. Network links will be congested, failed, over-utilized, and under-utilized at some point due to random workload distribution of application workflows. Thus, new classes of monitoring techniques should be investigated for dynamically monitoring routes and bandwidths based on a number of options (e.g., user-level, application-level, flow-level, etc.). The data provided by such monitoring techniques can be used to formulate failure recovery techniques and mechanisms, as well as developing models for predicting performance anomalies.
- *New application-level and SDN level API:* In order to allow every layer to communicate and coordinate with its adjacent layers, new dedicated APIs must be developed. For example, southbound APIs (which drive the interaction between the controller and network devices) need to be extended to allow management and monitoring of both network, and compute and storage resources. The new communication and coordination protocols between the scheduling manager and SDN controller is also needed for obtaining real-time resource utilization/availability and enforcing policies.
- *Accurate application demand and network traffic prediction:* Scheduling is not a trivial task when several diverse factors are involved which vary with time and space. Thus, the scheduling decision should not be changed very frequently. This requires not only the accurate estimation of current demand but also prediction of future

demand and resource availability. Some work has been done to date regarding partitioning and transitioning microflows to smoothly evolve the configuration of subnetworks under changing conditions.[14]

- *Scalable control and management layer:* Depending on how rapid and frequent updates are required within a system, decision making delays should be minimal. Now, in reality this layer needs to monitor and configure several thousands of applications and resources with numerous configurations. Thus, this layer needs to make decisions efficiently which requires decentralized and scalable implementation, especially in the case of hyperscale cloud datacenters which may have hundreds of thousands of interconnected physical servers.

With the advancement of software-defined networking technology, datacenter networks are emerging as the premier resource and/or infrastructure which make it possible to make the application stack SDN-native by programming the network stack all the way from physical topology to flow level traffic control. In this Blue Skies piece, we proposed tight integration of application stack (i.e., big data programming models) and network stack for jointly optimizing big data application performance and SDN performance at run-time. How to fully leverage and integrate SDN capabilities with existing BDMSs (e.g., YARN, MESOS) and how program big data applications in an SDN-native manner requires a number of important advancements. To this end, we attempted to briefly explore the research challenges and open issues related to integration of SDN and BDMSs, and discuss future research directions that need to be further investigated.

References

1. L. Cui, F. Yu, and Q. Yan, "When Big Data Meets Software-Defined Networking: SDN for Big Data and Big Data for SDN," *IEEE Network*, vol. 30, no. 1, 2016, pp. 58–65.
2. G. Wang, T. Ng, and S. Anees, "Programming Your Network at Run-Time for Big Data Applications," *Proceedings of the First Workshop on Hot Topics in Software Defined Networks,* 2012, pp. 103–108.
3. D. Kreutz et al., "Software-Defined Networking: A Comprehensive Survey," *Proc. IEEE*, vol. 103, no. 1, 2015, pp. 14–76.
4. M. Casado and N. McKeown,"The Virtual Network System," *ACM SIGCSE Bull.,* vol. 37, no. 1, 2005, pp. 76–80.
5. N. McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Comm. Rev.*, vol. 38, no. 2, 2008. pp. 69–74.
6. M. Casado et al., "Ethane: Taking Control of the Enterprise," *ACM SIGCOMM Computer Comm.*, vol. 37, no. 4, 2007, pp. 1–12.
7. J. Weinman, "Better Together: Quantifying the Benefits of the Smart Network," 3 March 2013; http://www.joeweinman.com/Resources/SmartNetwork.pdf.
8. R. Ranjan et al., "Orchestrating BigData Analysis Workflows," *IEEE Cloud Computing*, vol. 4, no. 3, 2017, pp. 20–28.
9. Centre for Earth Systems Engineering Research at Newcastle University, October 2017; http://www.ncl.ac.uk/ceser/research/integrated-systems/cities/citycat.
10. P. Qin et al., "Bandwidth-Aware Scheduling with SDN in Hadoop: A New Trend for Big Data," *IEEE Systems J.*, vol. PP, no. 99, 2015, pp. 1–8.
11. H. Hu et al, "Toward an SDN-Enabled Big Data Platform for Social TV Analytics," *IEEE Network*, vol. 29, no. 5, 2015, pp. 43–49.
12. M. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," *WH Free*, 1978, pp. 90–91.
13. J. Weinman, "Cloud Computing is NP-Complete," 21 February 2011; http://www.JoeWeinman.com/Resources/Joe_Weinman_Cloud_Computing_Is_NP-Complete.pdf.
14. R. Wang, D. Butnariu, and J. Rexford, "OpenFlow-Based Server Load Balancing Gone Wild;" http://static.usenix.org/event/hotice11/tech/full_papers/Wang_Richard.pdf.

KHALED ALWASEL *has a BS and MS in information technology from Indiana University–Purdue University Indianapolis (2014) and from Florida International University (2015), USA. He is currently working toward a Ph.D. in the School of Computing Science at Newcastle University, UK. Khaled's interests*

lie in the areas of software-defined networking (SDN), Big Data, and Cloud Computing. Contact him at K.S.A.Alwasel2@newcastle.ac.uk

YINHAO LI *is a PhD student in the School of Computing at Newcastle University, UK. His research interests include Cloud Computing, Edge Computing and Internet of Things. He previously received his MSc in Computer Science from the China University of Geoscience. Contact him at y.li119@ncl.ac.uk*

PREM PRAKASH JAYARAMAN *is a research fellow at the Swinburne University of Technology. His research interests include Internet of Things, cloud computing, big data analytics, and mobile computing. Jayaraman has a PhD in computer science from Monash University. Contact him at prem.jayaraman@gmail.com or http://www.premjayaraman.com.*

SAURABH GARG *is currently working as a lecturer in the Department of Computing and Information Systems at the University of Tasmania, Hobart, Tasmania. He was one of the few Ph.D. students who completed in less than three years from the University of Melbourne in 2010. He has published more than 30 papers in highly cited journals and conferences with Hindex 20. His doctoral thesis focused on devising novel and innovative market-oriented metascheduling mechanisms for distributed systems under conditions of concurrent and conflicting resource demand. He has gained about three years of experience in the Industrial Research while working at IBM Research Australia and India. Contact him at Saurabh.Garg@utas.edu.au.*

RODRIGO N. CALHEIROS *is a Lecturer in the School of Computing, Engineering and Mathematics, Western Sydney University, Australia. He works in the field of Cloud computing and related areas since 2008, and since then he has been carrying out R&D supporting research in the area. His research interests also include Big Data, Internet of Things, Fog Computing, and applications of these technologies. Contact him at R.Calheiros@westernsydney.edu.au.*

RAJIV RANJAN *is an associate professor (reader) in the School of Computing Science at Newcastle University, UK, and a visiting scientist at Data61, Australia. His research interests include cloud computing, content delivery networks, and big data analytics for Internet of Things (IoT) and multimedia applications. Ranjan has a PhD in computer science and software engineering from the University of Melbourne (2009). He has published more than 200 scientific papers. Contact him at raj.ranjan@ncl.ac.uk or http://rajivranjan.net.*