Needle in A Haystack: Cost-Effective Data Analytics for Real-Time Cloud Sharing

Yu Hua
Wuhan National Lab for Optoelectronics, School of Computer
Huazhong University of Science and Technology
Wuhan, China
csyhua@hust.edu.cn

Dan Feng
WNLO, School of Computer
Huazhong University of Science and Technology
Wuhan, China
dfeng@hust.edu.cn

***Abstract*—Real-time file sharing is important to improve the quality of cloud services. Current cloud systems however fail to efficiently offer cost-effective data analytics due to slow response and energy inefficiency. In order to support real-time processing and improve energy efficiency in the cloud, this paper proposes a novel cost-effective data analytics scheme, called Needle. The idea behind Needle is to use flat-addressing and content-aware naming approaches to deliver high performance of network transmission. Needle leverages a suitable division-of-labor model between core and edge network nodes to efficiently support content-aware queries in the cloud data center networks. Experimental results demonstrate that Needle is able to support cloud sharing service in a real-time and efficient manner.**

I. Introduction

Cloud services generally use the network transmission in large-scale data centers that leverage the design principles evolved from enterprise and Internet networking to build communication infrastructure. The Internet principles and architecture were established in the 1960s and 1970s [1]. The design goal was to implement the resource sharing and facilitate the ease of remote use. In order to construct the conversation for data transmission between exactly two hosts, IP packets are used to contain the source and destination addresses, thus becoming the *waist* of entire Internet protocol stacks due to its simplicity and ease of configuration. Large-scale data centers essentially require solid support, including communication services and energy efficiency, from the network to improve the system performance and provide cloud service guarantee [2].

The rapid growth of cloud data centers deployed by Internet service operators can support cloud sharing services. A large-scale data center typically contains tens of thousands of servers and consumes Mega-Watts of power for running and cooling the equipments. With the increasing demands for cloud services, energy consumed by data centers has been significantly increased. For example, for a 3-year server and 15-year infrastructure amortization, the energy-related costs are estimated to 41.6% of operation costs of large-scale data centers [3]. Moreover, although current cloud data centers have obtained significant increase in communication and computation capabilities, they unfortunately can not efficiently meet the needs of the increasing demands of energy consumption. The costs on powering up (and cooling) these data centers also significantly increase. For example, in the United States, data center facilities consume approximately 2% of all electricity, with an estimated growth rate of 12% per year. In the worldwide, data centers now consume about 1.3% of the worldwide electricity and this percentage will grow to 8% by 2020 [4]. Data centers have to afford skyrocketing power consumption and electricity bills. For instance, Google has over 1,120GWh and 67M, and Microsoft has over 600GWh and 36M [5].

In the cloud sharing service, high energy consumption not only results in large electricity costs, but also incurs high carbon emission. In the United States, generating 1 kWh of electricity emits about 500g of CO2 on average [4]. Each 100MW power station will cost $60-100 million dollars to build and emit 50 million tons of CO2 during its operation [6]. Hence, IT carbon footprints currently occupy 2% of global greenhouse gas emissions [7]. Moreover, there exist potential power outages. For instance, Amazon experienced an outage in October 2012 in its US-East-1 region. This outage leads to a series of failures in the power infrastructure [8]. Therefore, efficient and cost-effective energy management of data centers is becoming ever more important, especially for data center operators, e.g., Amazon, Facebook, Google and Microsoft.

To alleviate the energy consumption and support sharing service, efficient energy management in large-scale cloud data centers have received many attentions from both academia and industry. The key issues include flat-addressing and content-aware naming schemes to obtain energy savings and execute real-time data analytics. Performing their design and implementation is non-trivial since there exist two main challenges to be well addressed.

Challenge 1: Inefficient Combination of Location and Identity. The wide use of IP addresses potentially becomes the sources of many inefficient design schemes due to combining the network locations and host identities, such as failing to handle mobile hosts for moving users or the migrated virtual machines. This design has resulted in performance decrements in content-aware applications, such as optimizing the content addressing and content-based dynamic routing. A *location* is used to allow a request to identify an object in the network, while an *identity* can uniquely and consistently specify that object. In order to separate these two representation forms, existing solutions often rely on the resolution in endpoint namespace to transform a name to the corresponding location (e.g., an address or forwarding directive [9]). By using the transformation, cloud data center network uses location information to deliver the packet to the destination. In practice, the location-dependence design essentially increase the costs and complexity of the configuration and implementation complexity of network services due to the transformation mapping from content to host locations [10]. Therefore, it is well-recognized that the combination of identity and location into a single network address fails to meet the needs of current cloud data center network to support cloud sharing service in an efficient manner.

Challenge 2: Non-Scalability of Hierarchical Addressing. Conventional IP-based design leverages a hierarchical addressing, in which the addresses are grouped to indicate the relationships, which allows the addressed entities to manipulate the packets routing. The hierarchical addressing is non-scalable due to the expensive maintenance for massive states and control message overheads, which grows with the size of the network scale. Specifically, the identification of a target content needs to carry out the static operations of prefix matching multiple times in the intermediate nodes. These nodes serve for source-destination paths, while overlooking the differentiated requirements for data transmission. Performing the navigation through the entire addressed hierarchy potentially leads to the poor scalability. For a given network topology, the higher-level nodes become the performance bottleneck with higher probability, while requiring more redundant nodes as backups to avoid the potential single-point-of-failure. Furthermore, by using the hierarchical addressing, the mobility is also difficult to handle because IP addresses are hierarchical and tightly coupled with the scalability of the routing protocols. Therefore, although the hierarchical addressing works and supports various network applications for over 40 years, it has already limited the network scalability for the cloud data centers.

In order to improve hierarchical design performance and allow flat-addressing network to be efficient to real-world cloud applications, we need to bridge the gap between the accurate representation of content-aware namespace and the simplicity of hashing. In this paper, we propose Needle, a content-aware and cost-effective scheme in end-host routing protocol stacks, to efficiently support packet addressing and network services for cloud data centers. In the context of Needle design, a *service* is interpreted as a group of processes offering the same functionality, such as content distribution, energy efficiency and retrieval [11]. We make the following contributions.

First, in order to handle the inefficient combination, Needle provides a location-independent and content-aware naming scheme that facilitates the name-based routing and support content retrieval. Needle achieves this through a suitable "division of labor" [12] between core and edge network nodes. Each name is associated with a group of service instances that have similar functionalities, such as the contents of web servers and files in storage systems. In the edge nodes, we obtain the content-aware names for data packets by using Robin fingerprint [13] to partition entire data file into self-contained segments. Each segment becomes one dimension of that file. Needle further builds each name through Locality Sensitive Hashing [14] with constant-scale complexity. On the other hand, in the core nodes, they only need to execute approximate matching and forwarding operations upon arriving packets.

Second, in order to alleviate the performance bottleneck from hierarchical addressing, Needle makes use of the flat name for name-based routing and addressing in the cloud data center networks. Specifically, the packets from the same file contain the same name that is fixed-size and content-aware. At each network node, Needle carries out bit-aware operations to identify which ports are most similar to the arriving packets for fast forwarding. This design is light-weight without the need of maintaining and updating a routing table. The name-based addressing is independent of the physical addresses of network nodes and can adapt to the changes of network topology.

Third, in order to examine the performance of Needle, we implement a real prototype that consists of data servers in the cloud. Experiments use real-world datasets to evaluate the system efficiency and energy consumption. Moreover, the Needle design is compatible with existing applications and protocols running at end hosts. Extensive experimental results demonstrate the efficacy and efficiency of our proposed schemes.

The rest of the paper is organized as follows. Section II presents the backgrounds. Section III presents the Needle design and implementation details. Section IV shows the namespace generation and representation. Section V shows the routing in network elements and resolution in end-hosts. Section VI shows the performance evaluation. Section VII presents the related work. We conclude our paper in Section VIII.

II. Backgrounds and Motivations

Cloud data center networks support typical applications, such as queries in cloud storage and virtual machine migration. In general, the content queries are more important than the location of these contents. Due to the hierarchical address-based design in IP packets, the access to the contents needs to support the mapping from the *what* users actually care about to *where* the locations are, thus causing the significant increase of complexity and costs [15]. This case is exacerbated by the centralized hierarchical addressing that is inefficient to offer network system scalability. One promising solution is to provide alternatives or improvements to the IP waist.

The advantages of IP's success for over 40 years include the simplicity and weak demands for underlying layers, such as stateless and best-effort delivery. These salient features unfortunately cause the inefficiency of the IP-based network architecture for cloud services. Specifically, the best-effort service model is difficult to meet the needs of the QoS in many real-time applications, such as IPTV and video-conference. The data center network is also vulnerable due to equipment failures, software bugs and human-based configuration mistakes. Furthermore, a large-scale data center consumes expensive equipment costs and energy. These issues essentially come from the lack of an efficient addressing to provide scalable and content-aware namespace and support various emerging cloud applications.

IP can efficiently support the ubiquitous inter-connectivity and was designed for conversations between communications endpoints, thus supporting the content distribution. Current content distribution networks (CDN) are essentially a massive overlay infrastructure by using a large number of machines to cache and serve data. Different CDNs are isolated from each other. The CDN's performance is limited by its own server capabilities. Needle alleviates the CDN's limitations and offer content distribution to support cloud sharing service.

Various research schemes have been recently used and widely discussed. Content-Centric Networking (CCN) [15] moves the universal component of the network stack from IP to "*chunk of content*" and supports the hierarchical name aggregation with longest-match lookup. SecondNet [16], as a virtual data center, achieves scalability by distributing all the virtual-to-physical mapping, routing, and bandwidth reservation state in server hypervisors. With the aid of hierarchical addressing, DIFANE [12] handles wildcard rules in intermediate flow-based switches. SCAFFOLD [17] supports flow-based anycast with service instances. In essence, these approaches mainly use the hierarchical addressing scheme. The hierarchy design leads to the inefficiency in routes, and the use of location-dependent addresses complicates the mobility and management.

Unlike the hierarchical addressing, flat addressing [18] is attractive and promising to efficiently support the ease-of-use configuration of data centers. Flat names have recently been recognized to be useful since flat addressing simplifies the handling of topology changes and end-host mobility, without the needs of reassigning addresses. ROFL (routing on flat labels) [10] evaluates the possibility of routing directly on semantic-free flat labels. Disco [19] routes on flat, location-independent identifiers with guaranteed scalability and low stretch. SEATTLE [20] provides plug-and-play functionality via flat addressing, while ensuring scalability and efficiency through shortest-path routing and hash-based resolution of host information. These approaches determine routing paths based on a hash of the destination's identifier, thus incurring a stretch penalty.

In the cloud sharing service, the goals of performance improvements and energy efficiency can be achieved via a suitable design. In order to optimize energy usage, conventional approaches often mitigate individual performance metric, while not decreasing the workload's possible completion time or throughput. For example, Brown et al. considered energy efficiency as an optimization problem [21]. To minimize the total energy, an energy efficient system needs to adjust the system's hardware resources dynamically. Moreover, Rivoire et al. used two complementary ways to address the energy-efficiency problem. This scheme allows the functionality of energy efficiency to be integrated into the initial design of computer systems, and adaptively manages the power consumption changing with the workload [22].

III. The Needle Design

In large-scale data center networks, a suitable "division of labor" between the core and edge network nodes helps support the content distribution and retrieval in a scalable and efficient way. We first elaborate the design principle of Needle, i.e., "Division of Labor", and then illustrate the proposed architecture in data center networks.

A. *"Division of Labor"*

Large-scale data center networks exhibit imbalanced workloads between core and edge network nodes [23], which handle heterogeneous tasks in terms of processing data packets. Specifically, the core nodes with heavier workloads are responsible for fast forwarding the arriving data to facilitate end-to-end routing. On the other hand, the edge nodes need to (de)encapsulate the packets and identify the data semantics to support higher-level services. The architecture design of data center networks hence needs to simplify the operation complexity in core network nodes and some customized operations can be completed in the edge nodes. The benefits include the improved throughputs and the maximized utilization of network resources.

By following the above design principle, we propose a content-aware architecture for cloud data center networks, called Needle. The idea behind Needle is to employ *flat addressing*, to simplify the operations on topology changes and host mobility, without the needs of address re-assignment. In order to ensure scalability and efficiency, Needle executes locality-sensitive hashing (LSH) [24] operations with $O(1)$ complexity in the core nodes, while allocates relatively complex operations of namespace management in the edge nodes.

In order to support the "division of labor" in the entire end-to-end network, compared with conventional IP-based protocol stack, the data center networks require a simpler but more efficient waist to obtain significant performance improvements. Needle remains compatible with existing applications and protocols running at the end hosts and the intermediate routers. Needle makes use of *policy* and *manipulation* as the higher and lower levels around Needle. The policy refers to the requirements for applications, such as QoS and reliability. The manipulation executes practical operations to facilitate the finer-grained and dynamic optimization under changing conditions.

B. *Content-aware Design*

We propose the content-aware scheme, called Needle, to support accurate and efficient addressing and retrieval. Figure 1 shows the architecture of our proposed scheme. This design follows the above "division of labor" principle between core and edge network nodes. Specifically, the tasks in edge nodes include *data partition*, *content identification*, and *name representation*. After being processed in the edge nodes, the data packets then route in the data center networks according to their name, not the IP addresses. On the other hand, the core nodes only need to carry out simple operations, including *similarity matching* and *data forwarding*, which incur very low complexity. Needle can implement the above functionalities with the aid of rabin fingerprint [13] for data partition, locality-sensitive hashing (LSH) [24] for data identification and namespace representation. In practice, this combination, unfortunately, does not work well due to low accuracy of Rabin fingerprint and so-called *curse of dimensionality* [24], [25]. We hence need to handle this issue in implementing the system prototype.

C. *The Function Components*

Needle is a narrow waist to support namespace representation and facilitate data retrieval with dynamically-changing service instances. As an architecture for network communication with services, rather than the devices, Needle includes function components in the transport layer, along with traditional network-layer functions. The Needle contains three main components:

Packet Names: A packet name includes its content ID and network address of communicating end-hosts. The data center network uses the content ID to direct a data flow to an instance of the named service by executing ID matching multiple times.

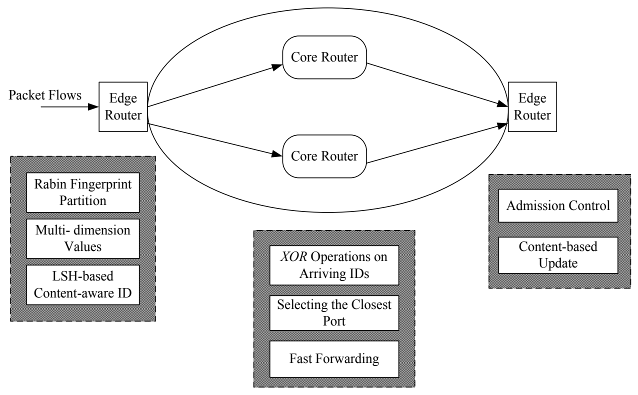


Fig. 1. The Needle architecture.

The network addresses can ensure the continuous communication with that service instance.

Network Elements: The network elements consist of the intermediate routers that can direct a packet to a service instance based on content ID. Different from existing schemes, the routers in Needle contain the ID-based forwarding table that makes the forwarding independent from routing. It is promising to balance the states between the packet headers and the network nodes, thus allowing both stateless and stateful operations.

End-hosts: In Needle, end-hosts need to carry out the data retrieval in the local nodes and provide the queried data contents to satisfy the specific service instances. Applications connect with content IDs and the addresses, thus allowing the changes with the moving end-hosts.

IV. Namespace Generation and Representation

This Section shows the namespace generation and representation in the Needle architecture for large-scale cloud data center networks.

A. Packet Name

The names of packets are generated at the source node, where Needle carries out content-based analysis and representation. This is achieved by using two-step process, i.e., data partition and content identification as shown in Figure 2.

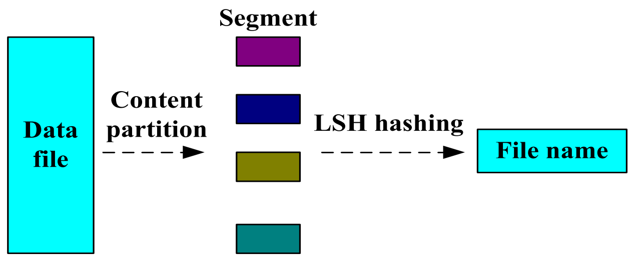


Fig. 2. The generation of packet names.

A Needle ID is a fixed-length, location-independent and content-aware name for representing particular data and supporting service naming. The packet header includes the 1-bit flag and 32-bytes generated name, as shown in Figure 3, as well as the host addresses and the socket ID. The previous two parts are fixed-length for fast processing in hardware. The 1-bit flag indicates the packet's types, such as query requests and data transmission. Moreover, the 32-bytes name comes from the LSH computation results to describe the similarity among networked data. We will show the details of namespace generation and representation in the following sections.

In addition, for the host addresses, if the packet contains query requests, the address indicates which host issues the request. On the other hand, if it contains the actual data, the address shows where the data come. Hence, the actual data contents only exist in the data transmission. For the query request, it does not contain the actual data. In addition, in order to facilitate the re-compose packets into original files at the end-hosts, the transmitted data also contains the names of its former and latter packets. If a packet is the first (last) one, its former (latter) names become vacant.

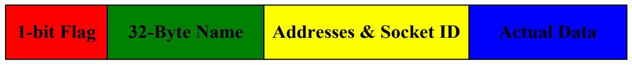


Fig. 3. The Needle packet components.

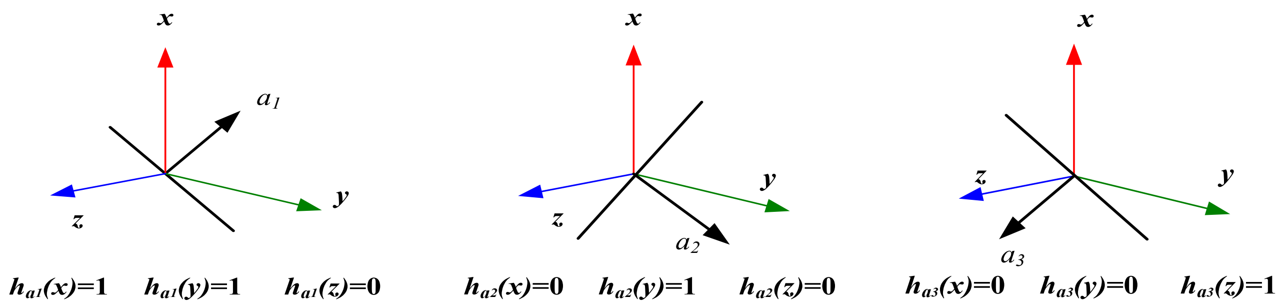
B. Content Partition

In order to improve the accuracy and efficiency of addressing and routing, Needle uses Rabin fingerprint [13] to allow the name to be content-aware. In general, we use short tags to represent and label larger objects. Rabin fingerprint is a method for implementing public key fingerprints using polynomials over a finite field. Needle uses it to study the contents of data flows and carry out the content-based partitions from the "0/1" sequences.

One advantage of Rabin fingerprint is only to execute *Shift* and *XOR* operations for bit object string. For example, if we consider the string matching problem to find the substring $x_1x_2\dots x_n$ from original string $y_1y_2\dots y_m$ $(n \leq m)$, the time complexity of using Rabin fingerprint is $O(n+m)$, which is much better than the straightforward one-by-one checking solution with complexity $O(nm)$.

By using the Rabin fingerprint, Needle can efficiently partition entire data file into multiple content-aware segments. Each segment is then considered as one dimension of that file. A file thus has multiple dimensions, even thousands-scale for large and varying files. The high dimensions lead to the significant increments of computation complexity. In order to alleviate the problems of high dimensions and efficiently exploit the semantics behind these dimensions, we use locality-sensitive hashing that achieves this goal with constant-scale complexity at edge network nodes. In addition, it is unnecessary for the flat-naming design architectures to specify how clients learn the Needle IDs. Instead, users can leverage the hashing-based approaches, say LSH, to identify human-readable names to Needle IDs.

A Needle name consists of a series of 0/1 bits that contain potential semantic correlation. In general, more correlated files exhibit more identical bits in their names. For example, as shown in Figure 4, there are three file vectors, i.e., x, y and z, under the measures of three random vectors a_1, a_2 and a_3. The vectors hence obtain their LSH-based names, i.e., $name(x) = 100$, $name(y) = 110$ and $name(z) = 001$.


x
a_1
y
z
$h_{a_1}(x)=1$ $h_{a_1}(y)=1$ $h_{a_1}(z)=0$
x
y
z
a_2
$h_{a_2}(x)=0$ $h_{a_2}(y)=1$ $h_{a_2}(z)=0$
x
z
y
a_3
$h_{a_3}(x)=0$ $h_{a_3}(y)=0$ $h_{a_3}(z)=1$

(a) Random vector a_1. (b) Random vector a_2. (c) Random vector a_3.

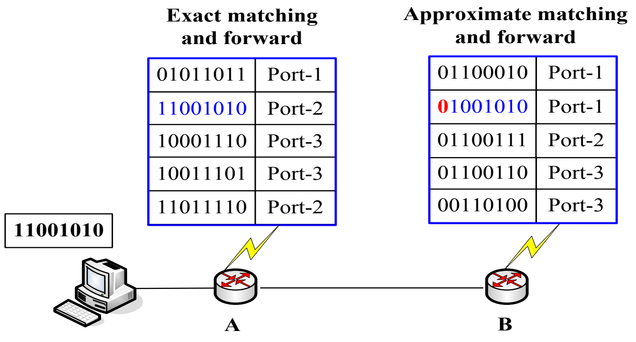
Fig. 4. The name generation by using LSH-based projection under multiple random vectors.

V. Routing in Network Elements and Resolution in End-hosts

This Section first describes the routing scheme in the Needle network architecture, especially in the named routing, fault tolerance and automatic update. We then present the resolution of end-hosts through the revised sockets.

A. *Name-based Routing*

Needle implements source routing via the operations of matching and forwarding in the intermediate network elements. Figure 5 shows an example of using 1-byte name. For simplicity, we use 1-byte as an example and the system implementation still uses 32-byte names. The network elements can support both exact and approximate matching to facilitate fast forwarding.



Exact matching and forward

01011011	Port-1
11001010	Port-2
10001110	Port-3
10011101	Port-3
11011110	Port-2

Approximate matching and forward

01100010	Port-1
01001010	Port-1
01100111	Port-2
01100110	Port-3
00110100	Port-3

Fig. 5. Network elements to support both exact and approximate matching for packet forwarding.

Needle carries out the local operations to identify the most correlated ports for forwarding. When a packet arrives, an intermediate network node needs to identify the port that has the largest number of the same bits as the transmitted packet. For instance, as shown in Figure 5, for the first hop, the node *A* contains a port that has the same name with the arriving packet. This packet is then forwarded from port-2. Moreover, for the second hop, if the same name does not exist, the intermediate node identifies that port-1 is the most approximate port that has the largest number of the same bits than others. The packet is then forwarded from port-1.

Fault Tolerance. The support for approximate matching can improve fault tolerance of networked systems and in the meanwhile facilitate approximate representation from user's query requests. Specifically, if an intermediate network node does not work, Needle can still support routing and identify the queried data through approximate matching with neighboring network nodes. Since the content-aware names attempt to identify more correlated nodes, the packets can always identify an alternative end-to-end path with incurring small extra costs. On the other hand, due to the infrequent appearance of vague query requests from users, the query requests need to obtain approximate, rather than exact, queries results. In this case, the approximate matching in the Needle routing can well meet the needs of the quality of query service.

Automatic Update. When an end-host or a service instance fails or moves to a new location, Needle automatically updates the states in the intermediate network nodes to accurately direct new data flows. The end-host with explicit representation for available data updates the connection with a new address and socket ID. Moreover, the network changes can be classified to be predicted, such as periodical maintenance and virtual machine migration, and unpredicted, such as server failure and client mobility. In order to ensure a fast response to both changes, Needle makes use of tight integration between end-hosts and the network.

In order to support network services in Needle, typical cloud sharing applications need to initiate the communication with service names. Needle constructs a protocol family for representing content names, rather than the network addresses of IP protocol family. The sockets API's flexibility allows such protocol families to be defined and implemented.

The network addresses of the communicating endpoints do not need to be exhibited to applications. The addresses management for applications is important since these addresses may change over time if a session or process moves. Other high-level socket interfaces are related with cloud applications.

VI. Performance Evaluation

In this section, we present the experimental results of a Needle prototype implementation. We examine the performance via real-world datasets in terms of energy efficiency and response latency.

A. *Experiments Configuration*

We implement the components and real-time scheduling schemes of Needle in a real data center. Each node is equipped with Intel 2.0GHz dualcore CPU, 2GB DRAM, 250GB disk and 1000PT quad-port Ethernet network interface card. The prototype is developed in the Linux kernel 2.4.21 environment.

Our evaluation leverages real-world electricity datasets of Google Data Centers in the U.S., which contain the following locations: Mountain View, CA, Houston, TX, and Atlanta, GA. The datasets reported here were obtained from publicly available government agencies [26], [27]. In general, the electricity system for data centers in the US is organized in cross-state regions such as New England, PJM (primarily covering Pennsylvania, New Jersey, and Maryland), and ERCOT (Texas).

The data centers have the properties that the regulated utility and deregulated wholesale structures exist in different regions. Specifically, Mountain View and Houston are in the deregulated electricity regions, and Atlanta is located in the regulated utility region. Moreover, for two deregulated regions, California has an hour-ahead forward market, whereas Texas has a 15-min ahead forward market. Based on these data, we obtain the average daily and hourly electricity prices for all these locations in May 2009.

We use three datasets from Google data center locations to emulate a dynamic multi-electricity-market environment. These locations form a subset of the whole Google data center locations in the United States. Without loss of generality, we select two groups of data on May 2 and May 16, 2009 at GMT-2:00 Time. To emulate Google power consumption, we assume that the total workload from five front-end Web portal servers to three back-end data centers is the requests per second. Furthermore, the delay constraint is 1ms. Each server operates at 100W. The processing speed of each server is 2.25 requests per second, 1.5 requests per second, and 2 requests per second, respectively in three data centers. Moreover, we compare our energy-aware scheduling scheme with their original management approach that is considered to be standard.

B. Evaluation Results and Analysis

We present the evaluation results in terms of energy consumption, entire running costs and request completion latency.

1) Energy Consumption: Figure 6 shows the hourly energy consumption in three data centers on May 2 set. Compared with the standard approach, the average energy savings in Needle are respectively 28.5% in Houston, TX, 18.7% in Mountain View, CA, and 36.8% in Atlanta, GA. The reason of improving energy efficiency in Needle is the efficient slack stealing scheme for scheduling real-time and non-real-time tasks. Needle hence can significantly improve the utilization and decrease the completion time, which is also verified by the results in Section VI-B3. We also observe that the May 16 set also exhibits the similar results. The energy savings are respectively 26.3% in Houston, TX, 20.5% in Mountain View, CA, and 37.3% in Atlanta, GA as shown in Figure 7. We hence argue that Needle is helpful to decrease the energy consumption.

2) Entire Running Costs: Figure 8 shows the entire running costs on May 2 set. Since Needle can efficiently decrease the energy consumption, it in the meantime reduces the entire running costs. The average decrease is 35.7% in May 2 set. Moreover, we also observe that the costs in Needle change in the smoother manner than the standard approach. Needle alleviates the usage peak that is important to a reliable data center. The observations and conclusions are also supported by the May 16 set. As shown in Figure 9, the average decrease of costs is 32.9%, compared with the standard approach.

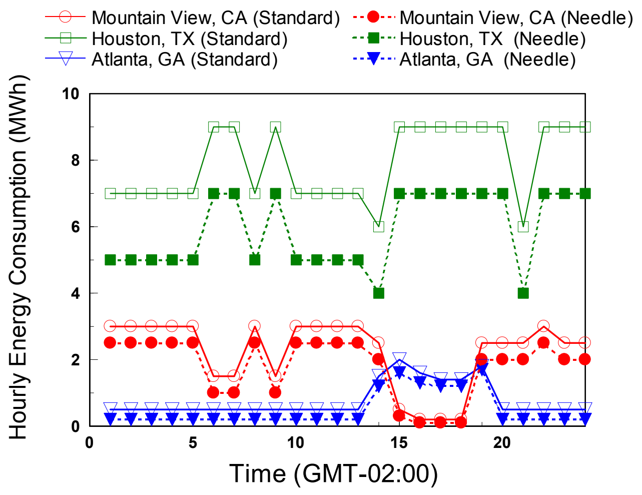


Fig. 6. Energy consumption in the May-2 set.

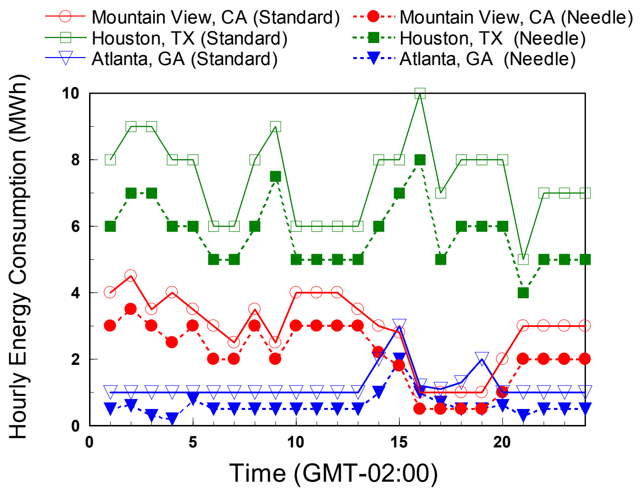


Fig. 7. Energy consumption in the May-16 set.

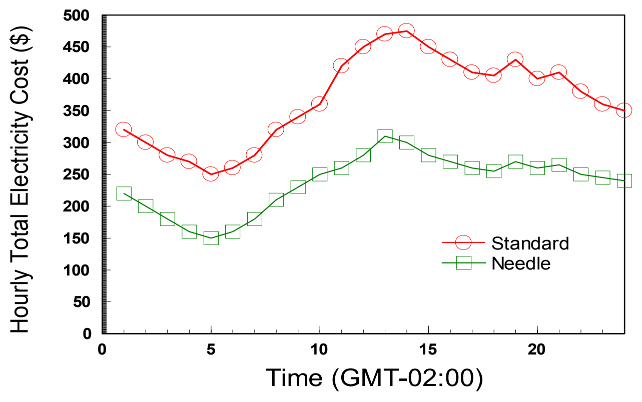


Fig. 8. Hourly energy consumption for three locations on May 2, 2009.

3) Request Completion Latency: We examine the real performance of data centers in terms of request completion latency. Figures 10 and 11 show the latencies in May 2 and 16 datasets. Specifically, based on the request inputs, the response latencies are respectively 26.5ms in Mountain

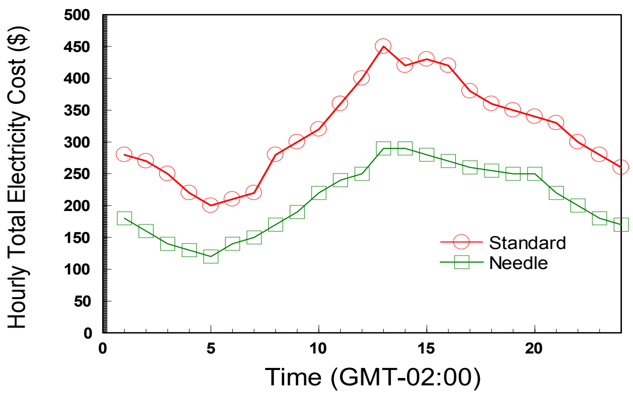


Fig. 9. Hourly energy consumption for three locations on May 16, 2009.

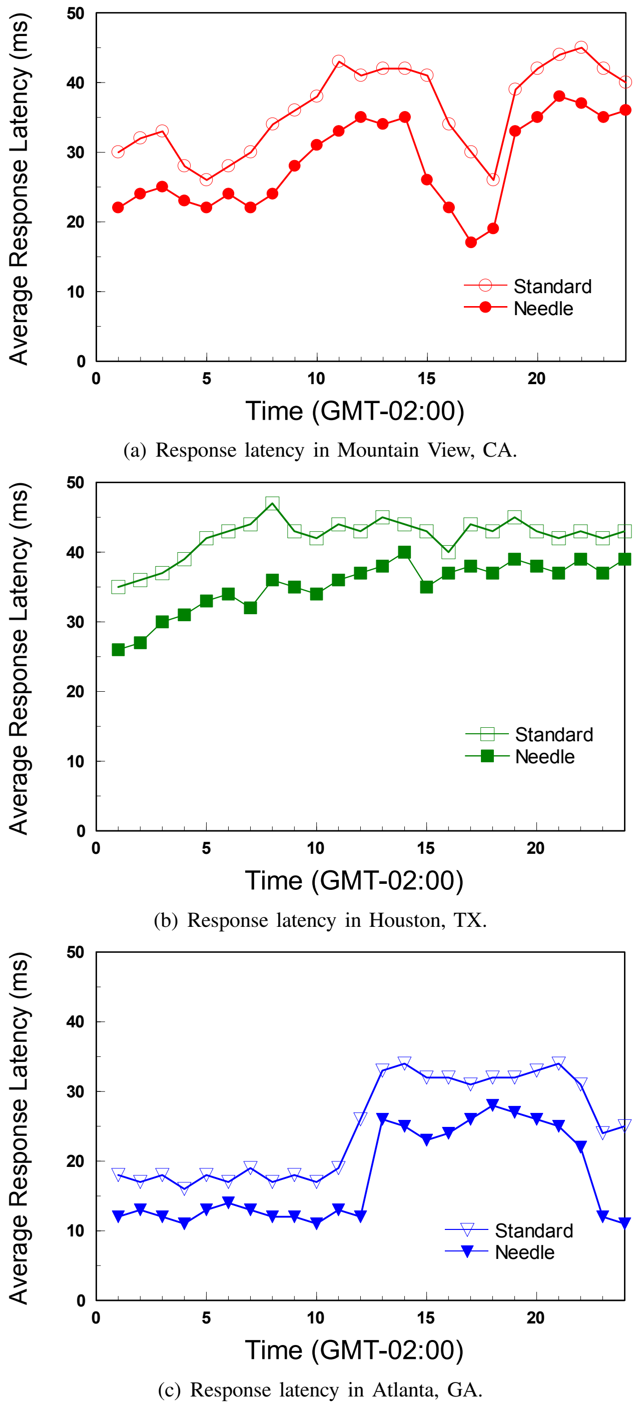
View, CA, 32.6ms in Houston, TX, and 15.2ms in Atlanta, GA, by using Needle, which are much smaller than 35.1ms in Mountain View, CA, 47.8ms in Houston, TX, and 24.9ms in Atlanta, GA, by using the standard approach.

In the May 16 set, based on the requests, the latencies in Needle are respectively 17.8ms in Mountain View, CA, 21.7ms in Houston, TX, and 8.2ms in Atlanta, GA. Those in the standard approach are 29.3ms in Mountain View, CA, 27.6ms in Houston, TX, and 17.5ms in Atlanta, as shown in Figure 11. The advantages of Needle in terms of latency savings are derived from the efficient use of flat addressing that can significantly reduce the entire completion latency.

VII. Related Work

With the rapid growth of cloud sharing services, the electricity consumption by data centers has significantly increased, as well as the performance requirements in real-time scheduling. The energy savings and efficient scheduling are becoming more and more important to large-scale data centers. Both industry and academia have proposed various technologies to reduce the electricity consumption and improve the execution performance in data centers [28].

Real-time Task Scheduling. Heo et al. [29] have developed an adaptation graph analysis scheme to address the conflicts between inter-acting adaptive components, e.g., On/Off and dynamic voltage scaling policies in server farms. This scheme can minimize the energy consumption. Two load-scheduling systems GreenSlot [30] and GreenHadoop [31] were well proposed. They assume that the data center is powered by solar energy and grid without energy storage. GreenSlot [30] is used to execute batch jobs, while GreenHadoop [31] is used to support the scheduling on Hadoop jobs. Aksanli et al. [32] designed an adaptive scheduler for mixed batch and Web service jobs. This scheduler leverages two separate job arrival queues, respectively for the Web services and the batch jobs. Each server has the limited capacity of the amount of workloads (including the Web requests and batch jobs). In order to obtain suitable tradeoff between the Web services and batch jobs, they are carried out in different cases. For example, Web services are executed when there exist available computing resources. In the meantime, the batch jobs are executed by exploiting the availability of the renewable energy. Moreover, an energy-aware cluster was proposed [33]



(a) Response latency in Mountain View, CA.

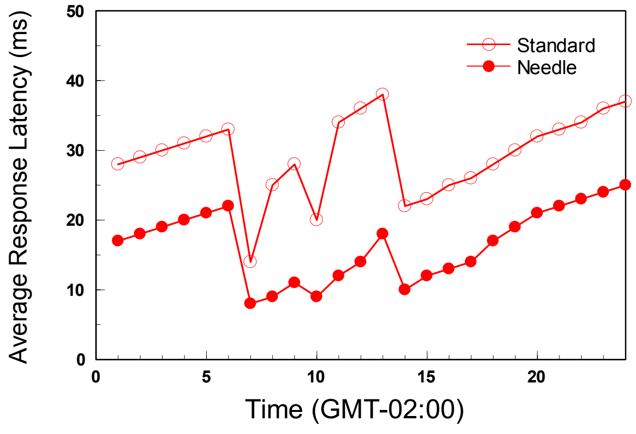
(b) Response latency in Houston, TX.

(c) Response latency in Atlanta, GA.

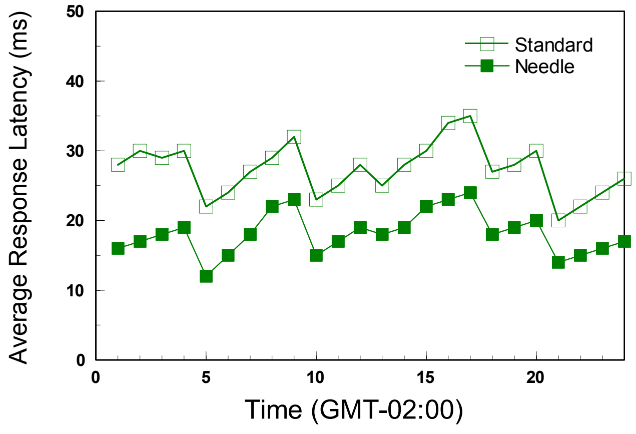
Fig. 10. The average response latency for requests in the May-2 set.

to slightly defer long running batch jobs. This scheme can improve the performance of the use wind energy.

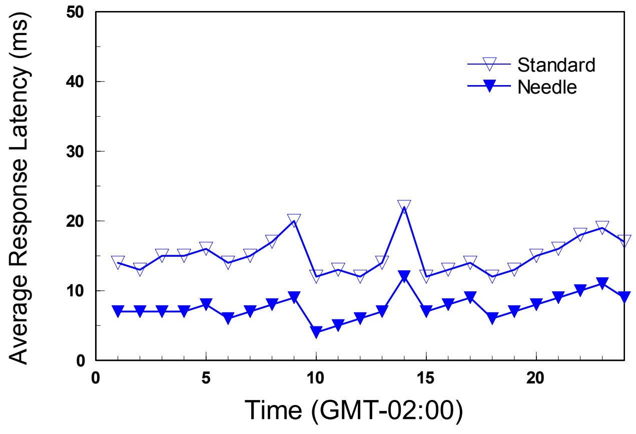
Flat names: Previous content-aware networking schemes mainly leverage unstructured and self-certifying content based labels. ROFL (routing on flat labels) [10] constructs a circular namespace to efficiently support accurate routing and allows extra pointers to decrease the routes. SEATTLE [20] utilizes flat addressing with a one-hop DHT to offer a directory service with reactive address resolution and service discovery. These systems need to show explicit contents to notify the DHT in



(a) Response latency in Mountain View, CA.



(b) Response latency in Houston, TX.



(c) Response latency in Atlanta, GA.

Fig. 11. The average response latency for requests in the May-16 set.

terms of the location before being queried. This design often obtains a cached copy of data along with their routed path, and fails to guarantee the closest copy. Unlike them, Needle leverages content-aware naming scheme to support the fast routing and similarity-based retrieval.

Content-Centric Networking: Content-Centric Networking (CCN) [15] is a networking architecture built on IP's engineering principles. The CCN scheme makes use of the named content rather than host IDs. It is unnecessary for the CCN routing to consider the host addresses. Furthermore, in order to support on-line services and deal with the service replication and dynamism, SCAFFOLD [17] names a stateful service and includes the possibly changing host addresses in each packet. This approach in essence uses the hierarchical addressing. DONA(Data-Oriented Network Architecture) [34] replaces DNS names with flat, self-certifying names. Each resolution handler needs to maintain a large-size forwarding table that can offer the next-hop information. The benefits of large-scale packet caching [35] include the decrease of redundant content transmission. Routers can identify previously forwarded contents and replace the content portion with a representative fingerprint. Unlike them, Needle studies the actual contents to build the names of transmitted packets to facilitate further fast forwarding in network nodes.

VIII. Conclusion

In order to support cloud sharing service, this paper proposes an cost-effective data analytics scheme, called Needle. Needle has the salient features of real-time scheduling performance and energy efficiency. In order to achieve these design goals, Needle leverages content-aware naming and efficient routing schemes. We further examine the performance by using real-world datasets in terms of energy consumption and response latency. Extensive experimental results demonstrate the efficiency of the proposed schemes.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (NSFC) under Grant 61173043, National Basic Research 973 Program of China under Grant 2011CB302301 and NSFC under Grant 61025008.

References

[1] J. Rexford and C. Dovrolis, "Future Internet architecture: clean-slate versus evolutionary research," *Communications of the ACM*, vol. 53, no. 9, pp. 36–40, 2010.

[2] N. Zhu, X. Liu, J. Liu, and Y. Hua, "Towards a cost-efficient mapreduce: Mitigating power peaks for hadoop clusters," *Tsinghua Science and Technology*, vol. 19, no. 1, pp. 24–32, 2014.

[3] J. Hamilton, "Cooperative expendable micro-slice servers (CEMS): low cost, low power servers for internet-scale services," *Proc. Conference on Innovative Data Systems Research (CIDR)*, 2009.

[4] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 211–222, 2012.

[5] A. Qureshi, *Power-demand routing in massive geo-distributed systems*. PhD thesis, Massachusetts Institute of Technology, 2010.

[6] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam, "Carbon-aware energy capacity planning for datacenters," *Proc. IEEE 20th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 391–400, 2012.

[7] G. Cook, "How clean is your cloud," *Catalysing an energy revolution*, 2012.

[8] F. . O. Amazon Cloud Outage KOs Reddit, "[Online]. Available: http://www.datacenterknowledge.com/archives/2012/10/22/amazon-cloud-outage-affecting-many-sites/," Oct. 2012.

[9] D. Clark, R. Braden, A. Falk, and V. Pingali, "FARA: Reorganizing the addressing architecture," *Proc. ACM SIGCOMM workshop on Future Directions in Network Architecture*, 2003.

[10] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: routing on flat labels," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 363–374, 2006.

[11] Y. Hua, B. Xiao, B. Veeravalli, and D. Feng, "Locality-sensitive bloom filter for approximate membership query," *IEEE Transactions on Computers*, vol. 61, no. 6, pp. 817–830, 2012.

[12] M. Yu, J. Rexford, M. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *Proc. ACM SIGCOMM*, pp. 351–362, 2010.

[13] M. Rabin, "Fingerprinting by random polynomials," *Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University*, 1981.

[14] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," *Communications of the ACM*, no. 1, pp. 117–122, 2008.

[15] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," *Proc. CoNEXT*, 2009.

[16] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees," *Proc. ACM CoNEXT*, 2010.

[17] M. Freedman, M. Arye, P. Gopalan, S. Ko, E. Nordstrom, J. Rexford, and D. Shue, "Service-Centric Networking with SCAFFOLD," *Technical Report TR-885-10, Department of Computer Science, Princeton University*, September 2010.

[18] A. Dhamdhere and C. Dovrolis, "The Internet is Flat: Modeling the Transition from a Transit Hierarchy to a Peering Mesh," *Proc. ACM CoNEXT*, 2010.

[19] A. Singla, P. B. Godfrey, K. Fall, G. Iannaccone, and S. Ratnasamy, "Scalable Routing on Flat Names," *Prov. ACM CoNEXT*, 2010.

[20] C. Kim, M. Caesar, and J. Rexford, "Floodless in seattle: a scalable ethernet architecture for large enterprises," *Proc. ACM SIGCOMM*, 2008.

[21] D. J. Brown and C. Reams, "Toward energy-efficient computing," *Communications of the ACM*, vol. 53, no. 3, pp. 50–58, 2010.

[22] S. Rivoire, M. A. Shah, P. Ranganatban, C. Kozyrakis, and J. Meza, "Models and metrics to enable energy-efficiency optimizations," *IEEE Computer*, vol. 40, no. 12, pp. 39–48, 2007.

[23] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 75–86, 2010.

[24] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," *Proc. STOC*, pp. 604–613, 1998.

[25] S. Berchtold, C. Böhm, and H. Kriegal, "The pyramid-technique: towards breaking the curse of dimensionality," *ACM SIGMOD Record*, vol. 27, no. 2, pp. 142–153, 1998.

[26] F. E. R. Commission, "[Online]. Available: http://www.ferc.gov/,"

[27] I. S. U.S. Energy Information Administration (EIA) and Analysis, "[Online]. Available: http://www.eia.doe.gov/,"

[28] Y. Hua, H. Jiang, Y. Zhu, D. Feng, and L. Tian, "Semantic-aware metadata organization paradigm in next-generation file systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 2, pp. 337–344, 2012.

[29] J. Heo, D. Henriksson, X. Liu, and T. Abdelzaher, "Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm case-study," *Proc. IEEE International Real-Time Systems Symposium (RTSS)*, pp. 227–238, 2007.

[30] Í. Goiri, R. Beauchea, K. Le, T. D. Nguyen, M. E. Haque, J. Guitart, J. Torres, and R. Bianchini, "Greenslot: scheduling energy consumption in green datacenters," *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011.

[31] Í. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "Greenhadoop: leveraging green energy in data-processing frameworks," *Proc. ACM European conference on Computer Systems*, pp. 57–70, 2012.

[32] B. Aksanli, J. Venkatesh, L. Zhang, and T. Rosing, "Utilizing green energy prediction to schedule mixed batch and service jobs in data centers," *ACM SIGOPS Operating Systems Review*, vol. 45, no. 3, pp. 53–57, 2012.

[33] A. Krioukov, S. Alspaugh, P. Mohan, S. Dawson-Haggerty, D. E. Culler, and R. H. Katz, "Design and evaluation of an energy agile computing cluster," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2012-13*, 2012.

[34] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *Proc. ACM SIGCOMM*, 2007.

[35] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," *Proc. ACM SIGCOMM*, 2008.