Optics Communications 381 (2016) 437–442

Contents lists available at ScienceDirect

Optics Communications

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

User-access-frequency statistics based hotspot adjustment in all-optically interconnected metro-embedded datacenters

Gang Chen [a], Hongxiang Guo [a,*], Dongxu Zhang [a], Jian Wu [a], Xiaoyuan Cao [b], Noboru Yoshikane [b], Takehiro Tsuritani [b], Itsuro Morita [b]

[a] Beijing University of Posts and Telecommunications, No.10 Xitucheng Road, Haidian District, Beijing 100876, China
[b] KDDI R& D Laboratories Inc, 2-1-15 Ohara, Fujimino, Saitama 356-8502, Japan

ARTICLE INFO

Article history:
Received 8 April 2016
Received in revised form
30 June 2016
Accepted 7 July 2016

Keywords:
Metro-embedded datacenter
Optical burst switching
Location aware
Hotspots adjustment

ABSTRACT

Traditional datacenters are typically deployed with highly clustered resources and built-in remote areas far away from users, requiring huge floor space and power consumption. The proposed metro-embedded datacenter architecture with distributed resources spreading across the metro area close to users could be a promising solution, which exhibits lower access latency and more flexibility for service orchestration. Based on metro-embedded datacenter, we propose the conception of "follow the user" with user access location aware dynamic hotspots adjustments. By enabling service hotspots adjustment according to user access frequency statistics, the quality of service can be significantly improved in terms of reduced latency. Finally, we experimentally demonstrate low-latency service provisioning based on the proposed principle.

1. Introduction

The typical warehouse datacenters (DCs) are built with highly clustered resources [1–3], which take up large floor space and consume huge amount of energy, so that most of them must be built in remote areas far away from users. The recently proposed metro-embedded DC architecture breaks traditionally centralized DCs down to small pieces which are called "micro datacenters" (mDCs). mDCs spread across metro regions upon optical switching networks [4], which can provide higher agility and reduced access latencies for users.

In the case that a user requiring cloud service moves back and forth between different places (e.g. the house and workplace or somewhere else), the metro-embedded DC system may dynamically maintain or migrate user's data with user location awareness. Wherever the user goes, the distributed cloud infrastructure guarantees the optimal user experience by making service contents follow the user.

Specifically, the control plane will collect user access frequency statistics and strategically set up "service *hotspots*" in different regions which ensure users' data can be accessed as easily as possible. When users' access preference has changed, the hotspots can also be dynamically adjusted. Meanwhile, contents synchronization needs to be carried out among users' hotspots in order to provide correct service.

To realize the content-floatable DC architecture described above, considerations should be taken from both data plane and control plane. To support highly dynamic data transmission and high scalability inside/among the mDCs, the interconnection network requires elastic bandwidth, low latency, reduced energy consumption, and easy extension. Time–frequency two-dimensional based optical sub-wavelength switching networks have inherent advantages in these aspects [5–7], and herein we employ fine-grained optical burst switching (OBS) [8,9] and optical burst switching over wavelength switched optical network (OBS-over-WSON) [10,11] paradigm for intra- and inter- mDCs respectively. Meanwhile, to achieve unified scheduling of dispersed resource of compute, storage and networking, SDN [12] technologies (including OpenStack [13] and OpenFlow [14]) are used for intelligent resources orchestration.

In this paper we first introduce the metro-embedded DC architecture and the principle of "follow the user" low latency service provisioning. Then we discuss the corresponding control plane design for the floatable resource orchestration, including the signaling workflow and some performance simulations based on location aware hotspots assignment. At last, we present an experimental prototype of the proposed metro-embedded DC architecture, with which we emulate the scenario of user location change and demonstrated the user location aware hotspots setup and adjustment under the coordination of SDN control plane. We

* Corresponding author.

http://dx.doi.org/10.1016/j.optcom.2016.07.018
0030-4018/

measured user access latencies to validate the feasibility of the proposed resource scheduling schemes.

2. The metro-embedded datacenter architecture

The architecture of metro-embedded DC [4] is shown in the left part of Fig. 1. User-level hotspots setup and content synchronization need the network to support fine-grained and dynamic bandwidth allocation. Therefore, we employ sub-wavelength optical switching to construct the network data plane. Specifically, inside each mDC, optical burst switching (OBS) ring is used to all-optically interconnect server racks, and the inter-mDC network adopts OBS over WSON technology to realize high bandwidth and low energy consumed transparent data communication.

To achieve more agile service mobility and stronger robustness, the control plane takes advantages of SDN control to holistically organize the distributed IT and network resources, which include a metro network controller and several mDCN controllers, as well as an OpenStack platform on the top working as a unified orchestrator.

Note that every mDC have an Access Node (AC), which all-together serve as distributed network gateways of the holistic metro-embedded DC. The entire metropolitan region can be divided into several small areas that each contains an mDC, and users' service requests launched from a specific area will be responded by the AC which is exclusively responsible for that area. In other words, during a service session, the data streams between user and servers are always delivered via the user's closest AC, such that the users can be guaranteed with the minimum access latency. However, it is neither efficient nor necessary to have a copy of users' data at every mDC to take advantage of resource distribution and decrease latency. In fact, considering system feasibility, we can define some service *hotspots* for all users based on their frequently accessed locations, and then strategically adjust these hotspots as users' access frequency statistics change.

For example, as shown in Fig. 1, when users access their data from AC1 and AC2 more frequently than from AC3, the control plane would allocate resources in mDC1 and mDC2 to establish the users' optimal hotspots respectively. At the same time, the network resources are also scheduled to accommodate data streams caused by users' hotspots synchronization. But as users' behavior changes, they may access the cloud services via AC3 more than AC1, thus the control plane would perceive the changes of these users' access location preference, and then migrate users' hotspots accordingly.

We call the feature described above as “follow the user”, which can provide services with low latency. It involves precise coordination between the unified orchestrator (e.g. OpenStack) and network control plane to achieve dynamic IT resource reallocation and optical network reconfiguration. In the next section, we further discuss the detailed operation schemes from the perspective of control plane.

3. Location aware service hotspots assignment

Take the scenario depicted in Fig. 1 for instance. Assuming that a user has registered his (or her) cloud services and the orchestrator allocated two initial hotspots (Hotspot1 and Hotspot2), the working mechanism for the user accessing his (or her) data is shown in Fig. 2. If the user is at location1, the service request would be forwarded to the nearest AC1, and then AC1 would look up the service database and connect the user to a nearest hotspot (Hotspot1). During an active service session, the data streams between user and hotspot1 would go through gateway AC1. If the user moves to other locations, he (or she) will be served by other ACs which in turn select nearest hotspots to fetch user's data. And the hotspots would always keep data synchronization with each other.

Nevertheless, as users' frequently accessed AC points change over time, the control plane may constantly update the users' locality statistics and strategically trigger hotspots readjustment, as shown in Fig. 3. The orchestrator would monitor access amount of different ACs in real time, and trigger hotspots readjustment when necessary. With the knowledge of global topology and IT resource distributions, and according to hotspot adjustment strategy, orchestrator would setup or remove hotspots then configure metro and intra-mDC optical bandwidth by the metro-net controller and mDCN controller respectively to migrate hotspot data. After that, the orchestrator may adjust subwavelength interconnections within and among mDCs, because adjustment of hotspots could change the bandwidth configurations for content synchronizations. Also, the orchestrator would update the service database and forwarding rules in related ACs and network nodes.

It is up to the unified scheduling strategy for deciding how many and in which locations the hotspots should be established for all users. Empirically, the mDCs that are most frequently accessed by users should be assigned as hotspots. Higher amount of hotspots would be beneficial for lowering access latency, but also increase cost and resource wastes; fewer hotspots save costs but may damage user experience. In this regard, different hotspot assignment strategies might be designed to strike for the balance between costs and performance. Here, we introduce a simple solution as shown in Fig. 4. The basic principles are that: (1) the number of hotspots should be large enough so that users' total

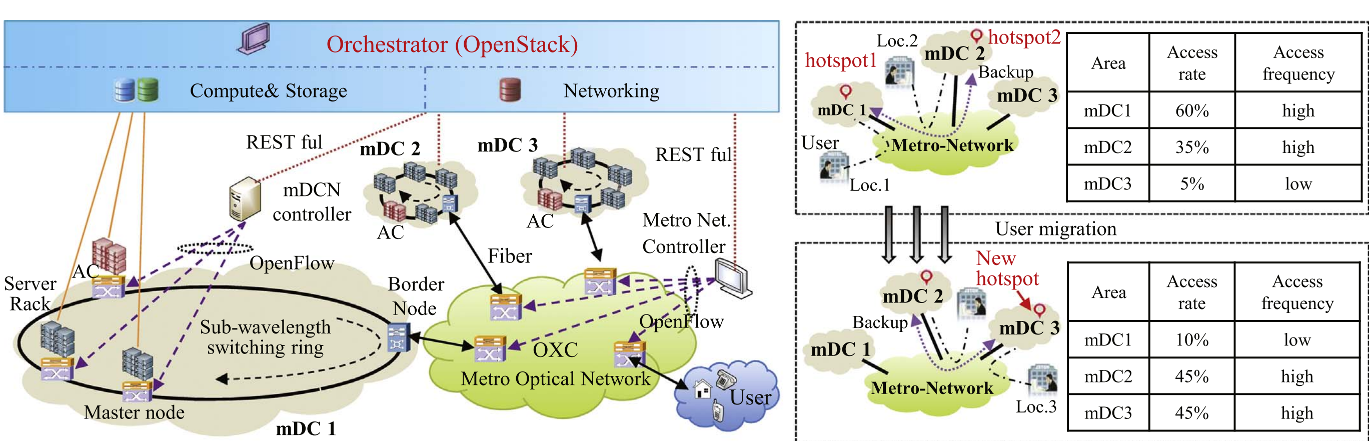

Fig. 1. The architecture of metro-datacenter and hotspots synchronization and adjustment along with user migration.