Experimental Demonstration of SDN-enabled Reconfigurable Disaggregated Data Center Infrastructure

Xiaotao Guo(1), Fernando Agraz(2), Xuwei Xue(1), Bitao Pan(1), Albert Pagès(2), Shaojuan Zhang(1), Georgios Exarchakos(1), Salvatore Spadaro(2), Nicola Calabretta(1)

(1) Institute of Photonic Integration, Eindhoven University of Technology, Netherlands, x.guo@tue.nl
(2) Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract *A 4-node prototype of SDN-controlled disaggregated data-center network is experimentally demonstrated based on the nanoseconds optical switch, enabling flexible hardware resource provisioning and dynamic resource reallocation. Experimental results show that, based on monitoring statistics, real-time reconfiguration reduces the memory node access latency by 21%.*

Introduction

Emerging cloud applications (such as scientific computation and web serving) require diverse hardware resources (including CPU, GPU, and memory) from data center (DC) infrastructures[1]. However, the fixed hardware of the server-centric (CPU, memory, etc.) results in hardware resource underutilization despite the high power consumption and costs. To solve these issues, the disaggregated DC concept was recently proposed, which decouples all the hardware from the server-unit and replaces the on-board bus by network interconnection[2,3]. Hence, the data center network (DCN) scope is extended to enclose the fabric interconnecting the hardware devices that will compose the compute nodes. Exploiting separated hardware resource pools, the disaggregated DC can achieve flexible resource provisioning for different applications and high power-efficiency during operation. However, there are still several challenges for the implementation of disaggregated DCs. At first, the extra access latency is introduced for the communication among hardware nodes. Thus, a low latency and high bandwidth network is critical for the disaggregated architecture. Secondly, there are much more hardware nodes to be interconnected, which requires a scalable interconnect network. Finally, massive hardware nodes make resource allocation difficult to find the optimal mapping, and dynamic requirements of applications demand a real-time orchestration system for resource reallocation.

Several studies have been carried out to implement the disaggregated DC. An emulated remote memory system was implemented in [4], but the setup is still within a server-centric, which lacks the connection network. In [5], the authors proposed a hierarchical disaggregated memory orchestrator based on RDMA protocol. However, current multi-layer electrical networks may degrade the application performance. A 2-node disaggregated prototype was implemented based on optical networks[6], but this prototype only shows a fix compute-to-memory traffic and no orchestration and control plane were employed. In our previous work, a rack-scale disaggregated architecture based on nanoseconds optical switches (NOS) was only numerically investigated[7].

In this work we experimentally demonstrate the full operation of the NOS based disaggregated DC architecture empowered by a software-defined network (SDN) controller and orchestrator for reconfigurable disaggregated compute nodes composition. Exploiting

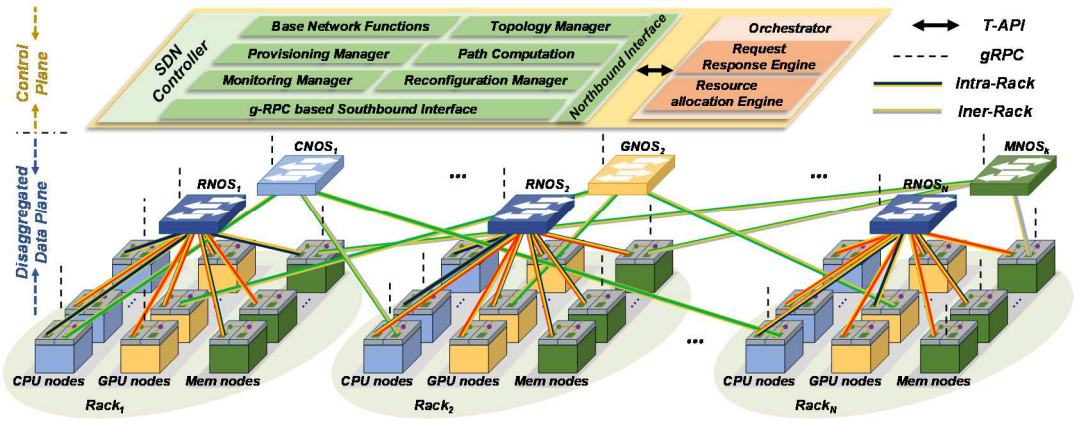

Fig. 1: Architecture of the SDN-enabled reconfigurable disaggregated DC.

978-0-7381-4679-9/21/$31.00 ©2021 IEEE

nanoseconds SOA-based switches, the data plane sets up the low latency and high bandwidth connection among 4 disaggregated hardware nodes with monitoring capability. Due to the dynamic resource requirement, applications allocated to the same hardware node may occur in resource contention; hence degrading the application performance. Based on monitored statistics, the control plane can dynamically reallocate the hardware nodes to guarantee the performance. It is shown in the experiment that the end-to-end latency of accessing memory node is reduced by 21% based on the resource node reconfiguration.

SDN-enabled disaggregated DC

The SDN-enabled reconfigurable disaggregated DC architecture comprising hardware nodes, nanoseconds optical switches, SDN controller, and orchestrator is depicted in Fig. 1. In the data plane, there are three kinds of hardware nodes, including CPU, GPU, and memory nodes. All the hardware nodes are divided into *N* racks, and each rack consists of *I* CPU nodes, *J* GPU nodes, and *K* memory nodes (*N*=*I*+*J*+*K*), respectively. The *N* hardware nodes in a rack are connected by an intra-rack NOS (RNOS), while the *i*-th CPU node/ *j*-th GPU node/ *k*-th memory node in each rack is connected by the *i*-th CNOS/ *j*-th GNOS/ *k*-th MNOS (*i*=1,⋯ *I*; *j*=1,⋯ *J*; *k*=1,⋯ *K*). Based on this flat network interconnection, the CPU node can access the GPU/memory node in the same rack with only a single hop, and at most two hops in a different rack with the shortest path. The functional blocks of CPU node are illustrated in Fig. 2(a). The CPU node receives the instruction from the control plane to send network statistics by means of a customized agent that enables gRPC-based SDN control at the hardware devices. There is also a minimal local memory in the CPU node for the running operation system and necessary data caching. Once accessing the memory node, the read/writing instruction of a logical memory address is sent to PCIe processing module at first. The memory mapping table from logical to physical address is stored in the memory management unit (MMU). According to the MMU information, the physical memory address of the target data is added by the packet encoder, and the instruction is encoded as an optical packet. Fig. 2(b) shows functional blocks of the memory node. The received instruction from the CPU node is processed by the memory controller (MC). If the target data is stored in the cache space, then is directly accessed and send back to the CPU node. Otherwise, the MC need to access the target memory address of on-board memory resource via the DMA engine. The optical flow control protocol is applied in the NOS based disaggregated architecture to solve the packet contention among hardware nodes[8]. In the case of packet contention, the packet with higher priority is forwarded by the NOS, while the others are blocked. The ACK/NACK signal is generated by switch controller and sent back to source hardware node for releasing or retransmitting the packet. More details about the structure of NOS can be found in [7].

The orchestrator, which resides on top of the control plane, is responsible for setting up the Disaggregated Compute Nodes (DiCN) based on application request. Each DiCN consists of variable kinds/amounts of hardware nodes, and can run multiple applications. The SDN controller receives the instruction coming from the orchestrator through the Northbound interface (NBI), which implements the Transport API (T-API) protocol[9]. According to the application requirements, the Provisioning Manager (PM) relies on the Topology Manger and the Path Computation modules to select the hardware nodes to compose the DiCN and their interconnection, then it triggers the configuration of hardware nodes through the gRPC-based Southbound interface (SBI). This configuration is enabled in the data plane via the SDN Agent of each node. During applications runtime, the agents collect statistics of the hardware nodes and send them to the Monitoring Manager (MM)

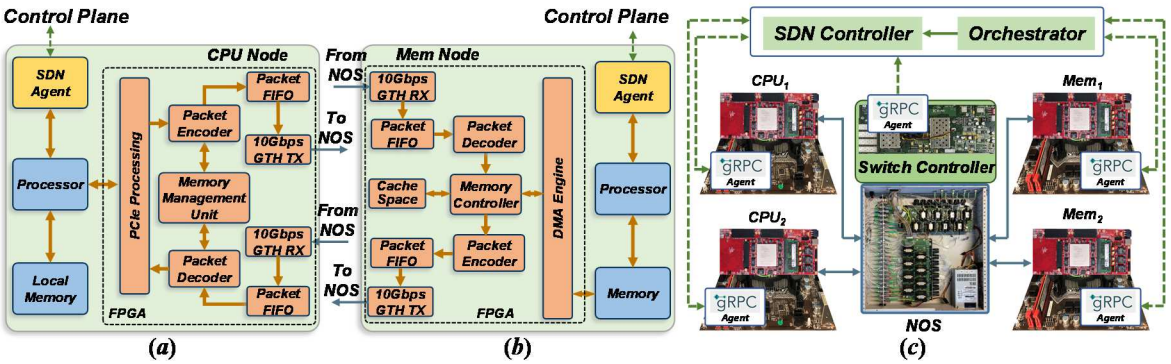

Fig. 2: Functional blocks of (a) CPU node and (b) memory node. (c) Experimental setup.

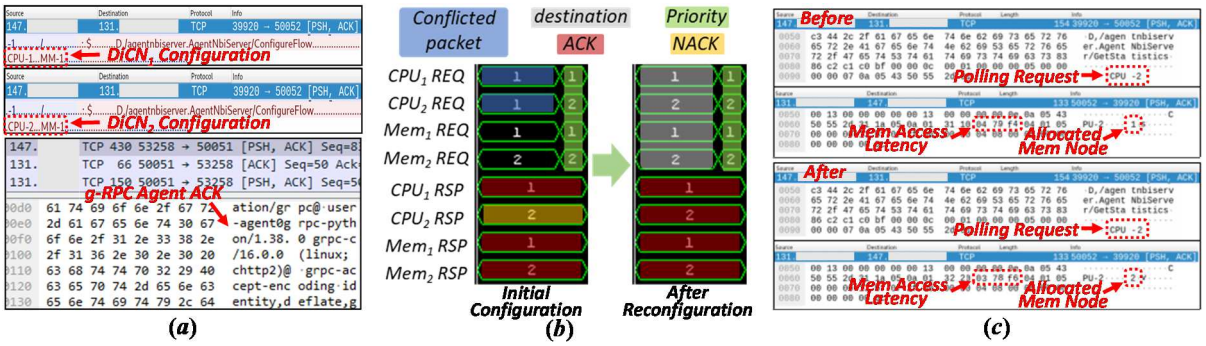

Fig. 3: (a) Packets of DiCN configuration from control plane. (b) Request processing of the switch controller before and after reconfiguration. (c) Monitored statistics of the data plane before and after reconfiguration.

of the controller (via the SBI). If detecting the performance degradation, the reconfiguration manager (RM) dynamically reallocates hardware nodes of DiCN. Both the orchestrator and SDN controller are customized for the disaggregated DC infrastructure. This is because existing technologies cannot cope with the DiCN setup, management, and reconfiguration nor with its stringent operational requirements (such as monitoring latency and scalability).

Experimental setup and results

The experimental setup to demonstrate the SDN-enabled disaggregated DC architecture is shown in Fig. 2(c). The data plane consists of 2 CPU nodes (CPU_1 and CPU_2), 2 memory nodes (Mem_1 and Mem_2), and one 4-port NOS. All the hardware nodes are implemented based on a micro-board and a Xilinx UltraScale board, connecting to by the PCIe Gen3 bus. The CPU and memory of micro-board is applied as hardware nodes, and the processing modules are implemented on the FPGA. Each hardware node is equipped with two 10Gb/s SFP+ transceivers to connect to the NOS and the other connects to the NOS switch controller (Xilinx Vertex-7). The lossless 4-port NOS is a broadcast and select switch based on SOA gates and amplifiers. The processing clock frequency of all the hardware nodes is synchronized by the switch controller for fast clock and data recovery. The CPU node processes data with a 64-bit width at 156.25MHz. The payload length of optical packet is set as 64 bytes as the typical length of cache line in current computer architecture.

The SDN controller sends periodic polling requests for monitoring network statistics. The packets from control plane to configure DiCNs and acknowledgement signal from SDN agent are shown in Fig. 3(a). It is found that the orchestrator takes ≈0.98ms to send the instruction to controller, and 1.2ms for the controller to receive the network statistics. It is shown that two DiCNs are set up sharing the same memory node (Mem_1): $DiCN_1$ includes CPU_1 and Mem_1, and $DiCN_2$ consists of CPU_2 and Mem_1. The requests (REQ) of memory node access from two DiCNs and the processing of switch controller are illustrated in Fig. 3(b). The REQ priority of CPU_1 is 1 while the REQ priority of CPU_2 is 2 (smaller number is higher priority). As the configured memory node of two DiCNs is the same, both CPU nodes send REQ with a destination of Mem_1 to the switch controller. With a higher priority, CPU_1 packet is successfully forwarded, while CPU_2 packet is blocked and the switch controller sends a response (RSP) of NACK to CPU_2 for packet retransmission. The retransmission costs higher network latency, degrading the performance of application running in the $DiCN_2$. The monitored statistics is illustrated in Fig. 3(c), in which the hexadecimal "F" is used to represent the decimal point in gRPC packets. It is shown that, under the initial configuration (Mem_1 is shared by CPU_1 and CPU_2), the end-to-end latency of accessing memory node is 479.4ns for the $DiCN_2$ and 378.6ns for $DiCN_1$. The monitored statistics trigger the RM of SDN controller to reallocate Mem_2 to CPU_2, thus the $DiCN_2$ is reconfigured as Mem_2 and CPU_2, as shown in Fig. 3(c).

After the hardware node reconfiguration, the CPU_1 and CPU_2 send the REQ to the switch controller with destination Mem_1 and Mem_2, respectively, as shown in Fig. 3(b). All packets from CPU_1 and CPU_2 are successfully forwarded to the memory nodes, and only ACK signals are sent back to CPU nodes. Fig. 3(c) shows the monitored statistics after reconfiguration (Mem_2 is allocated to CPU_2). The end-to-end latency of accessing memory node is reduced to 378.6ns for $DiCN_2$. Thus, benefiting from the automatic dynamic resource reconfiguration, an end-to-end latency improvement of 21% is achieved in the disaggregated data plane network.

Conclusions

We have experimentally demonstrated an SDN-enabled reconfigurable disaggregated DC architecture based the NOS. The implemented prototype can flexibly reconfigure allocated hardware nodes based on network statistics. Based on the monitoring and dynamic resource reallocation, the latency of accessing memory node achieves an improvement of 21%.

References

[1] W. Mulia, N. Sehgal, S. Sohoni, J. Acken, C. Stanberry, and D. Fritz, “Cloud Workload Characterization,” IETE Technical Review, vol. 30, no. 5, pp. 382-397, 2013.

[2] R. Lin, Y. Cheng, M. D. Andrade, L. Wosinska and J. Chen, "Disaggregated Data Centers: Challenges and Trade-offs," in IEEE Communications Magazine, vol. 58, no. 2, pp. 20-26, 2020.

[3] P. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker, “Network requirements for resource disaggregation,” in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), Georgia, pp. 249–264, 2016.

[4] S. Han, N. Egi, A. Panda, S. Ratnasamy, G. Shi, and S. Shenker, “Network support for resource disaggregation in next generation datacenters,” in 12th ACM Workshop on Hot Topics in Netwworks (HotNets-XII), Maryland, 2013.

[5] W. Cao and L. Liu, "Hierarchical Orchestration of Disaggregated Memory," in IEEE Transactions on Computers, vol. 69, no. 6, pp. 844-855, 2020.

[6] V. Mishra, J. L. Benjamin and G. Zervas, "MONet: heterogeneous Memory over Optical Network for large-scale data center resource disaggregation," in IEEE/OSA Journal of Optical Communications and Networking, vol. 13, no. 5, pp. 126-139, 2021.

[7] X. Guo, F. Yan, X. Xue, B. Pan, G. Exarchakos and N. Calabretta, "QoS-aware data center network reconfiguration method based on deep reinforcement learning," Journal of Optical Communications and Networking, vol. 13, no. 5, pp. 94-107, 2021.

[8] W. Miao, S. Di Lucente, J. Luo, H. Dorren, and N. Calabretta, "Low latency and efficient optical flow control for intra data center networks," Optics express, vol. 22, no. 1, pp. 427-434, 2014.

[9] TAPI – Open Transport Configuration and Control (OTCC), https://wiki.opennetworking.org/display/OTCC/TAPI, accessed on May 2021