Switches for HIRE: Resource Scheduling for Data Center In-Network Computing

Extended Abstract

Marcel Blöcher[†] Lin Wang[◇,†] Patrick Eugster[♯,†,‡] Max Schmidt[†]

[†]*TU Darmstadt, Germany* [◇]*Vrije Universiteit Amsterdam, The Netherlands*

[♯]*Università della Svizzera italiana, Switzerland* [‡]*Purdue University, USA*

1. Motivation

Over the past decades data center (DC) network appliances have become increasingly programmable. Originally benefitting the prototyping, testing, and deployment of more flexible and novel network(-wide) services and protocols, this trend has been more recently exploited for benefitting more specific applications and services. By supporting certain specific computations "in the network" on the path between data sources and sinks, individual distributed systems concerns like agreement [14, 6] or caching [19, 15], and even high-level application components such as for machine learning [32, 24], can be handled in a much accelerated fashion, ushering in a new era of in-network computing (INC).

Despite the proliferation of exciting INC applications, running multiple INC applications on the same network, or even having multiple tenants using the same INC application remains under-explored. Existing DC resource management frameworks (RMFs) use a model of resource pools of isolated compute containers and the resource scheduling problem is complex already. Throwing INC resources into the mix adds new challenges and significantly exacerbates existing ones: (1) INC resources such as programmable ASICs and NPUs are **highly heterogeneous** [HET] in terms of not only processing power, but also programming models [9, 26]; (2) INC resources are relatively scarce compared to server resources, demanding **interchangeable resources** [ALT] as fallbacks; (3) INC-enabled jobs impose **fine-grained locality** [LOC] constraints regarding the underlying network topology, with dependencies between servers and INC appliances; (4) INC resources exhibit **non-linear sharing** [NOL], as contrary to "complete" isolation on servers, partial INC resources may be shared by multiple tenants or INC service(s) [30]. These new challenges render the existing RMFs inapplicable, motivating a new RMF design for INC-enabled DCs.

Given the huge potential of INC in a wide spectrum of current applications and the ever-increasing interest of INC deployments in large-scale DCs, an effective RMF for INC would be a key enabler for the democratized use of INC.

2. Limitations of the State of the Art

So far, no work has studied the resource management problem for INC. Related works mainly fall into three categories:

Individual INC services. As mentioned, many individual INC services have been proposed for facilitating networking [16, 1, 18] and other [14, 15, 20, 33] tasks leveraging the high performance and programmability of network appliances. However, all these efforts focus on single use cases, leaving aside the problem of coordinating usage of potentially scarce and heterogeneous resources on network appliances among multiple INC scenarios, applications, and users.

Multitenancy for network appliances. Some recent works [12, 30] have explored the potential of enabling multitenancy support on a single network appliance, e.g., a P4 [2] switch or a smart network interface controller. However, such efforts are limited to device-level sharing, i.e., co-locating multiple INC services on the same network appliance. None of them have considered how to coordinate the use of INC resources on network appliances at a network-global level.

Data center resource management. There is a plethora of work on resource management and scheduling in DCs [10, 5, 13, 27, 3, 22, 29, 7, 8, 17]. However, none of them are able to meet the requirements for INC resource management due to one or more of the following limitations: supporting only homogeneous resources, single schedules (no alternatives), linear resource sharing, and domain-specific scheduling (e.g., GPU scheduling for deep learning workloads [11, 21, 23, 31]).

3. Key Insights

This paper presents Holistic INC-aware Resource managEr (HIRE), the first-of-its-kind RMF design for DCs supporting INC. At a high level, HIRE proposes a novel resource model and develops a corresponding flow-based scheduler. HIRE's design is driven by the following important insights:

Insight 1: A new resource abstraction is required to capture INC resources. Most existing resource abstractions for DCs focus on server resources and thus use a simple fixed-slot resource abstraction. Others consider also the network bandwidth allocation and propose an abstraction such as virtual cluster, virtual over-subscribed cluster, or tenant application graph. These abstractions are not able to fully capture INC resources due to the [ALT] and [NOL] challenges mentioned. Therefore, HIRE proposes a two-level resource model based on the notion of *composite* where users describe resource demands in a concise way and HIRE's resource model handles the interchangeability and non-linearity and transforms com-

posite requests into scheduler-friendly forms automatically.
Insight 2: The implementation details of INC services should be hidden from the application/user. As discussed, network appliances show high heterogeneity in their hardware architectures and thus, deploying the same INC service on different network appliances may require different domain (programming language/compiler/architecture) expertise. Moreover, INC services typically involve multiple network appliances interacting with each other in complex ways. As a result, setting up an INC service correctly is a heavy and error-prone task. Considering all these factors, HIRE proposes to use INC service *templates* recorded together with their implementation details in a store ([HET]). HIRE thus only exposes high-level interfaces for applications/users to specify application requests following the provided templates and to add new templates.
Insight 3: Flow-based scheduler design can be leveraged to encode the scheduling requirements of INC. HIRE adopts a *flow-based scheduler design* and performs the requirement encoding via novel designs on the flow network. Flow-based schedulers have been shown to have comparable scheduling quality compared with other centralized scheduler approaches, while still achieving outstanding scalability. This flow-based approach provides high flexibility for encoding the complex INC scheduling requirements ([ALT], [LOC], and [NOL]) compared to other centralized scheduling approaches.

4. Main Artifacts

Our paper provides the following main artifacts:
A two-level resource model design. The proposed resource model allows a user to submit a job in the form of *composite resource request* (`CompReq`) consisting of a set of composites derived from the composite template from the *composite template store* (`CompStore`). Users can specify the configuration for each of the composites in a `CompReq`, and also the way these composites are related. Once submitted, a `CompReq` is transformed into a *polymorphic resource request* (`PolyReq`). A `PolyReq` considers the different implementations (from details stored in the `CompStore`) of each composite in the `CompReq` and provides more detailed resource demands of the job, incorporating resource alternatives and non-linear resource usage.
A flow-based scheduler design. HIRE adopts a flow-based scheduler design inspired by Firmament [10], taking the `PolyReq`s generated by the resource model as input. However, HIRE incorporates several new designs into the flow network and the cost model to support the INC scheduling requirements. In particular, HIRE's flow network includes a *shadow network* for the physical network topology to encode the INC resources as well as the locality constraints. HIRE also includes new node types for choosing each job's alternatives with efficient approximations for combinatorial costs. Finally, HIRE uses methods for efficient propagation and filtering for performing resource matching when dealing with [NOL] and [HET].
A DC simulator implementation. We implement a simulator for DCs with INC including the HIRE flow-based scheduling logic in 13K lines of Scala code and run trace-driven simulations of different setups with 4394 servers and 845 switches.
An evaluation methodology for INC scheduling. HIRE is the first RMF capable of scheduling server and INC resources. To show the benefits of our novel custom solution, we retrofit a broad set of schedulers for different scheduling methods for comparison, including queue-based best-effort (Kubernetes [4]), delay scheduling using dominant resource fairness (Yarn [28]), random sampling using a variant of power of two choices (Sparrow [22]), and another flow-based scheduler with a multi-dimensional resource model (CoCo/Firmament [10, 25]). We run these in two modes (and applying some further modifications): (a) concurrent request mode mimics users who submit two variants of an INC enabled job, one with INC resources and one without (applying server fallback) where upon successful scheduling of one the other is dropped; (b) timeout request mode mimics users who submit first only the INC variant, and submit the corresponding server fallback if the INC variant is not served before a timeout. To better understand the impact of INC resources on the scheduling problem, we report metrics including placement latency, resource utilization, network detours, and rates of satisfied INC demands.

5. Key Results and Contributions

Overall, HIRE achieves the highest success rates in serving INC resources in all experiments, outperforming the best performing retrofitted existing schedulers by $8-30\%$, while at the same time performing allocations that result in least network detours (improving by $\sim 20\%$) for INC communication.

In summary, this paper makes the following contributions:

- We identify the main challenges in resource management for INC-enabled DCs ([HET], [ALT], [LOC], and [NOL]).
- We present a novel two-level resource model for capturing INC and other resources in DCs, as well as the model transformation rules between the two levels.
- We propose a new flow-based scheduler for INC scheduling, addressing all challenges mentioned.
- We implement our design and conduct thorough experiments to validate its performance using real-world workload traces.

Our work is first-of-its-kind. Although much work remains to be done to achieve fully holistic resource management for *all* heterogeneous resources (e.g., GPUs, TPUs, FPGAs, programmable network appliances), we believe HIRE presents a promising step in that direction.

6. Why ASPLOS

This paper presents an RMF for joint scheduling of INC and server resources, manifesting crosscutting research comprising programming abstractions for the resource model, scheduler design influenced by operating systems research, and INC being itself an emerging network technology.

References

[1] Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minlan Yu, and Michael Mitzenmacher. PINT: probabilistic in-band network telemetry. In *ACM SIGCOMM*, pages 662–680. ACM, 2020.

[2] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review (CCR)*, 44(3):87–95, 2014.

[3] Eric Boutin, Jaliya Ekanayake, Wei Lin, Bing Shi, Jingren Zhou, Zhengping Qian, Ming Wu, and Lidong Zhou. Apollo: Scalable and coordinated scheduling for cloud-scale computing. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 285–300, 2014.

[4] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, omega, and kubernetes. *Communications of the ACM*, 59(5):50–57, 2016.

[5] Carlo Curino, Djellel Eddine Difallah, Chris Douglas, Subru Krishnan, Raghu Ramakrishnan, and Sriram Rao. Reservation-based scheduling: If you're late don't blame us! In *ACM Symposium on Cloud Computing (SoCC)*, pages 2:1–2:14, 2014.

[6] Huynh Tu Dang, Daniele Sciascia, Marco Canini, Fernando Pedone, and Robert Soulé. Netpaxos: Consensus at network speed. In *ACM Symposium on Software Defined Networking Research (SOSR)*, 2015.

[7] Pamela Delgado, Diego Didona, Florin Dinu, and Willy Zwaenepoel. Kairos: Preemptive data center scheduling without runtime estimates. In *ACM Symposium on Cloud Computing (SoCC)*, pages 135–148, 2018.

[8] Carlo Fuerst, Stefan Schmid, Lalith Suresh, and Paolo Costa. Kraken: Online and elastic resource reservations for cloud datacenters. *IEEE/ACM Transactions on Networking (ToN)*, 26(1):422–435, 2018.

[9] Jiaqi Gao, Ennan Zhai, Hongqiang Harry Liu, Rui Miao, Yu Zhou, Bingchuan Tian, Chen Sun, Dennis Cai, Ming Zhang, and Minlan Yu. Lyra: A cross-platform language and compiler for data plane programming on heterogeneous asics. In *ACM SIGCOMM*, pages 435–450, 2020.

[10] Ionel Gog, Malte Schwarzkopf, Adam Gleave, Robert NM Watson, and Steven Hand. Firmament: fast, centralized cluster scheduling at scale. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, page 99, 2016.

[11] Juncheng Gu, Mosharaf Chowdhury, Kang G. Shin, Yibo Zhu, Myeongjae Jeon, Junjie Qian, Hongqiang Liu, and Chuanxiong Guo. Tiresias: A GPU cluster manager for distributed deep learning. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 485–500, Boston, MA, 2019.

[12] David Hancock and Jacobus E. van der Merwe. Hyper4: Using P4 to virtualize the programmable data plane. In *ACM International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, pages 35–49, 2016.

[13] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar, and Andrew Goldberg. Quincy: fair scheduling for distributed computing clusters. In *ACM Symposium on Operating Systems Principles (SOSP)*, pages 261–276, 2009.

[14] Xin Jin, Xiaozhou Li, Haoyu Zhang, Nate Foster, Jeongkeun Lee, Robert Soulé, Changhoon Kim, and Ion Stoica. Netchain: Scale-free sub-rtt coordination. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 35–49, 2018.

[15] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. Netcache: Balancing key-value stores with fast in-network caching. In *ACM Symposium on Operating Systems Principles (SOSP)*, pages 121–136, 2017.

[16] Changhoon Kim, Parag Bhide, E Doe, H Holbrook, A Ghanwani, D Daly, M Hira, and B Davie. In-band Network Telemetry (INT) Dataplane Specification. `https://p4.org/assets/INT-current-spec.pdf`, 2016.

[17] Jeongkeun Lee, Myungjin Lee, Lucian Popa, Yoshio Turner, Sujata Banerjee, Puneet Sharma, and Bryan Stephenson. Cloudmirror: Application-aware bandwidth reservations in the cloud. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2013.

[18] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. HPCC: high precision congestion control. In *ACM SIGCOMM*, pages 44–58. ACM, 2019.

[19] Ming Liu, Liang Luo, Jacob Nelson, Luis Ceze, Arvind Krishnamurthy, and Kishore Atreya. Incbricks: Toward in-network computation with an in-network cache. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 795–809, 2017.

[20] Zaoxing Liu, Zhihao Bai, Zhenming Liu, Xiaozhou Li, Changhoon Kim, Vladimir Braverman, Xin Jin, and Ion Stoica. Distcache: Provable load balancing for large-scale storage systems with distributed caching. In Arif Merchant and Hakim Weatherspoon, editors, *USENIX Conference on File and Storage Technologies (FAST)*, pages 143–157, 2019.

[21] Kshiteej Mahajan, Arjun Singhvi, Arjun Balasubramanian, Varun Batra, Surya Teja Chavali, Shivaram Venkataraman, Aditya Akella, Amar Phanishayee, and Shuchi Chawla. Themis: Fair and efficient GPU cluster scheduling for machine learning workloads. In *USENIX Symposium on Network Systems Design and Implementation (NSDI)*, 2020.

[22] Kay Ousterhout, Patrick Wendell, Matei Zaharia, and Ion Stoica. Sparrow: distributed, low latency scheduling. In *ACM Symposium on Operating Systems Principles (SOSP)*, pages 69–84, 2013.

[23] Yanghua Peng, Yixin Bao, Yangrui Chen, Chuan Wu, and Chuanxiong Guo. Optimus: an efficient dynamic resource scheduler for deep learning clusters. In Rui Oliveira, Pascal Felber, and Y. Charlie Hu, editors, *ACM European Conference on Computer Systems (EuroSys)*, pages 3:1–3:14, 2018.

[24] Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan R. K. Ports, and Peter Richtárik. Scaling distributed machine learning with in-network aggregation. *CoRR*, abs/1903.06701, 2019.

[25] Malte Schwarzkopf. *Operating system support for warehouse-scale computing*. PhD thesis, PhD thesis. University of Cambridge Computer Laboratory, 2015.

[26] Hardik Soni, Myriana Rifai, Praveen Kumar, Ryan Doenges, and Nate Foster. Composing dataplane programs with μp4. In *ACM SIGCOMM*, pages 329–343, 2020.

[27] Alexey Tumanov, Timothy Zhu, Jun Woo Park, Michael A. Kozuch, Mor Harchol-Balter, and Gregory R. Ganger. Tetrisched: global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters. In *ACM European Conference on Computer Systems (EuroSys)*, pages 35:1–35:16, 2016.

[28] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *ACM Symposium on Cloud Computing (SoCC)*, page 5, 2013.

[29] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at google with borg. In *ACM European Conference on Computer Systems (EuroSYs)*, page 18, 2015.

[30] Tao Wang, Hang Zhu, Fabian Ruffy, Xin Jin, Anirudh Sivaraman, Dan R. K. Ports, and Aurojit Panda. Multitenancy for fast and programmable networks in the cloud. In *USENIX Symposium on Hot Topics in Cloud Computing (HotCloud)*, 2020.

[31] Wencong Xiao, Romil Bhardwaj, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, Zhenhua Han, Pratyush Patel, Xuan Peng, Hanyu Zhao, Quanlu Zhang, Fan Yang, and Lidong Zhou. Gandiva: Introspective cluster scheduling for deep learning. In Andrea C. Arpaci-Dusseau and Geoff Voelker, editors, *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 595–610, 2018.

[32] Zhaoqi Xiong and Noa Zilberman. Do switches dream of machine learning? toward in-network classification. In *ACM Workshop on Hot Topics in Networks (HotNets)*, pages 25–33, 2019.

[33] Zhuolong Yu, Yiwen Zhang, Vladimir Braverman, Mosharaf Chowdhury, and Xin Jin. Netlock: Fast, centralized lock management using programmable switches. In *ACM SIGCOMM*, pages 126–138, 2020.