Published in final edited form as:

IEEE Netw. 2020 ; 34(6): 259–265. doi:10.1109/mnet.011.2000180.

Compute-Less Networking: Perspectives, Challenges, and Opportunities

Boubakr Nour,

Beijing Institute of Technology

Spyridon Mastorakis,

University of Nebraska–Omaha

Abderrahmen Mtibaa

University of Missouri-St. Louis

Abstract

Delay-sensitive applications have been driving the move away from cloud computing, which cannot meet their low-latency requirements. Edge computing and programmable switches have been among the first steps toward pushing computation closer to end-users in order to reduce cost, latency, and overall resource utilization. This article presents the “compute-less” paradigm, which builds on top of the well known edge computing paradigm through a set of communication and computation optimization mechanisms (e.g., in-network computing, task clustering and aggregation, computation reuse). The main objective of the compute-less paradigm is to reduce the migration of computation and the usage of network and computing resources, while maintaining high Quality of Experience for end-users. We discuss the new perspectives, challenges, limitations, and opportunities of this compute-less paradigm.

Introduction

The current Internet ecosystem is witnessing a massive growth of the number of connected devices and the volume of generated content, which forms what we know as the Internet of Things (IoT). This IoT growth leads to the execution of computational tasks and data processing that may be computationally intensive in nature. IoT devices may have limited resources, therefore, they may not be able to execute these tasks, offloading them to a cloud where the computation is powerful and resources are in abundance. Yet, the user and application requirements are changing; emerging applications (e.g., augmented reality, autonomous driving) have ultra low-latency constraints. As a result, such applications need to receive the requested content and/or task execution results within their latency constraints, which is not feasible when the content and the computing resources reside on distant clouds. To tackle this challenge, edge and fog computing [1] have been proposed as paradigms for moving services from the cloud to the edge of the network (i.e., closer to end-users), so that tasks are offloaded from end-user devices to edge servers [2].

In addition to edge and fog computing, other attempts include the integration of networking and computing to improve response times and the utilization of network and computing resources. Different Internet Engineering Task Force (IETF) research groups, such as the Computing in the Network Research Group (COINRG) [3] and the Routing Area Working Group (RTG WG) [4], have explored various possibilities to integrate computing with networking. Existing frameworks [3, 5] have explored the applicability of the “computing in the network” concept and the existing challenges and issues. Furthermore, researchers have designed architectures [4] that leverage both computing and networking in order to determine the optimal edge network among several edge networks across different geographic locations.

“*Compute-less*” networking refers to integrated network systems that utilize the minimum required network and computing resources for the completion of offloaded computational tasks. It builds on top of edge and in-network computing, capitalizing on their strengths, addressing their limitations, and contributing to their real-world, pervasive deployment. Compute-less networking can be enabled through the observation that the various offloaded tasks may have parts of the required computation in common. To this end, reusing (partially or fully) the results (final or intermediate) of already executed tasks for the execution of newly received tasks, a concept called *computation reuse* [6], can contribute to the minimization of the execution of duplicate computation.

In this article, we introduce the compute-less networking paradigm. Our contributions are as follows. We first provide a systematic definition of the compute-less paradigm along with associated computation optimization mechanisms and techniques to reduce resource utilization and task completion delays. Then we present the core design principles of the compute-less paradigm and highlight their benefits and trade-offs. Then we discuss various challenges related to an integrated, collaborative environment among stakeholders at the edge, which is enabled by the compute-less paradigm.

The remainder of this article is organized as follows. The following section reviews state-of-art remote computing paradigms and describes the challenges associated with each paradigm that compute-less can address. We then present a definition of the compute-less paradigm along with its design principles and benefits. Then we present various application use-cases that the compute-less networking paradigm can facilitate. Following that we discuss how different network paradigms and architectures can enable compute-less networking, and present various directions and issues for future research. Finally, we conclude our work.

Review of State-of-the-Art Remote Computing Paradigms

In this section, we provide a brief review of existing remote computing paradigms.

Cloud and Edge Computing

Cloud computing gained traction in mid-2000 offering on-demand computing resources to end-users through a “pay-as-you-go” model without requiring the active management of these resources by end-users. However, as increasingly delay-sensitive applications emerged, the network latency to remote clouds was prohibitive for these applications. Edge computing

was later proposed to migrate services and code to computing resources located at the edge of the network as a means to reduce execution times, load balance traffic, and minimize energy consumption.

How Can Compute-less Help?—Edge computing stakeholders today operate in a semi-centralized fashion, where task execution decisions are made independently from the network load. Edge computing also inherits the cloud computing operational practices. To this end, collaboration and sharing of computing resources, data, and execution results among stakeholders (e.g., service providers) is not typically allowed. This may not be a noteworthy issue in the context of cloud computing given the abundance of resources. However, at the edge, where computing resources are limited, collaborative approaches can increase the overall computing capacity. The compute-less paradigm aims to address such issues.

In-Network Computing

In-network computing has been, to some extent, a controversial research direction. Traditionally, computing capabilities have been offered by the communication endpoints (typically servers, but also clients to a certain extent). Recently, the community started paying considerable attention to the direction of in-network computing, which proposes the placement of computing capabilities in the network instead of the endpoints of communication. However, this direction has raised concerns within the community, since the primary job of network elements (e.g., network routers) is to forward traffic at line rates, therefore their resources should be fully dedicated to this purpose. With the emergence of recent trends in networking (e.g., edge computing, Information-Centric Networking, Software-Defined Networking), the community was provided with the proper mechanisms and appealing use-cases to make in-network computing a reality.

In-network computing resources may consist of either parts of the computing resources of network elements themselves or computing resources directly attached to network elements. About the former case, we acknowledge that the primary responsibility of network elements is traffic forwarding. However, when elements have available resources, they can perform “simple” functions based on incoming traffic (e.g., aggregate various measurements from IoT sensors in a building and return the average of these measurements to the requester). The critical question in this case would be “are we transforming network elements to middleboxes?” [7]. This is an aspect that needs careful consideration and further research to understand where the fine line between a network element and a middlebox is (if there is one). About the latter case, additional computing resources may be attached to network elements (e.g., one hop away from them). These resources may be able to offer pre-defined services (e.g., image annotation, matrix multiplication, real-time navigation) or execute code offloaded to them by end-users.

How Can Compute-Less Help?—In-network computing has been criticized for its limited scope and the limited set of applications that can benefit from it. Pushing computation to the network may introduce complexity for network elements, which would otherwise have to only forward network traffic. The compute-less paradigm aims to utilize

in-network computing for “simple” functions that will reduce the utilization of network and computing resources, while offering overhead assessment mechanisms to ensure that the complexity introduced is justified by the resulting amount of resources that the network and servers will save.

Computation Reuse

Computation reuse is a paradigm transparent to end-users requiring the availability of storage resources (to store previous computations) and efficient indexing and lookup mechanisms. Tasks can be divided into a set of subtasks forming an *execution dependency graph*. This graph represents a flow of which subtasks need to be executed first and pass their results as input to remaining subtasks. It also offers a fine granularity in terms of which subtasks of a task can be reused during the execution of another task (e.g., the subtask of multiplying parts of two matrices in the case of a graphics application). This approach may be suitable for applications that require the reuse of identical tasks/subtasks, where reusing “similar” but not identical tasks/subtasks may have considerable impact on the application performance (e.g., the results of the multiplication of slightly different matrices may be significantly different).

Computation reuse can also be based on the “similarity” between previously executed tasks/subtasks and tasks/subtasks to be executed [7, 8]. This approach may be suitable for applications, where reusing the results among tasks/subtasks that are different up to a certain degree may not severely impact the user Quality of Experience (QoE). For example, two images that are similar in the sense that they have slightly different backgrounds, but the same foreground, may yield the same results during object detection.

How Can Compute-Less Help?—The benefits of computation reuse are limited by how efficiently we can perform task profiling, creation of execution dependency graphs, estimation of the similarity between tasks, and lookups and storage of existing tasks/results. The compute-less paradigm employs both compute-aware and network-aware mechanisms to facilitate the reuse of computation and estimate its potential gain.

The Compute-Less Paradigm

In this section, we define and motivate the compute-less paradigm as well as present its design principles and associated challenges.

What is the Compute-Less Paradigm About?

Compute-less is a paradigm that builds on top and aims to improve the performance of the edge and fog computing paradigms through a set of communication and computation optimization mechanisms in the network and at the edge. These mechanisms aim to minimize task completion delays, resource utilization, and the task completion cost for both Internet Service Providers (ISPs) and edge service providers. At the same time, the compute-less paradigm envisions seamless, extensive collaboration among stakeholders such as end-users, ISPs, and service providers.

The mechanisms utilized by the compute-less paradigm include:

- Reducing the amount of computation traversing the network (e.g., through in-network computing, content summarization and aggregation).
- Minimizing unnecessary computation at the edge (e.g., by reusing the results of previously executed tasks).
- Assessing the overhead of the available optimization mechanisms to ensure performance improvements.

These mechanisms are realized and supported through context-aware networking, while any required state can be maintained both in the network and at the communication endpoints (e.g., edge servers). Figure 1 illustrates various compute-less mechanisms that may involve collaboration among stakeholders. For example, an ISP through its in-network computing resources identifies and forwards similar tasks to the same edge/fog node, enhancing the potential that this node will be able to reuse the results of previously executed tasks and reducing its resource utilization. Another example may be the collaboration between two service providers SP_1 and SP_2 to address cases where their computing resources are fully utilized. As a result, when the resources of SP_1 are fully utilized, SP_1 forwards received tasks to the resources of SP_2 for timely execution.

Why Yet Another Computing Paradigm?

While compute-less, edge, and fog computing share the same objective of bringing computing resources closer to end-users and reducing the overall task completion times, compute-less extends edge and fog computing to further reduce the utilization of computing resources and the task completion times. At the rate that edge and fog computing are being deployed by companies, cities, and ISPs coupled with the proliferation of IoT devices, available edge and fog computing resources may not scale quickly enough to accommodate the growing user/application demands [9]. At the same time, allocating more and more resources at the edge is costly (e.g., due to the need for cooling servers, additional space and maintenance requirements), essentially defeating the purpose of edge and fog computing. To this end, we propose the compute-less paradigm to fill this gap and work collaboratively with all stakeholders (e.g., service providers, ISPs, and end-users) to facilitate task scheduling, management, and computing while reducing the overall task completion time and resource utilization.

How Does Compute-Less Work?

To achieve the compute-less paradigm goals, edge service providers and ISPs work in a collective manner based on the following design directions.

Reduce the Load at the Edge: This can be achieved by executing tasks as close to end-users as possible (e.g., in-network or edge computing resources), and reducing the volume of computation using summarization or aggregation of similar tasks.

Compute-Aware Offloading Decisions: Networking elements perform forwarding operations based on information shared by both users and edge computing stakeholders. For instance, several tasks can be aggregated or clustered together to be offloaded to the same edge/fog node for efficient resource utilization and potentially faster execution.

Minimize the Execution of Duplicate Computation: When tasks reach an edge/fog node, (partial or full) reuse of the results of already executed similar tasks can deduplicate the computation execution and reduce the utilization of resources and the task execution times.

The realization and real-world deployment of these three directions come with different challenges that need to be addressed. These challenges are summarized below and are discussed in more detail later.

Assessment of the Benefits of Compute-Less and Its Associated

Overhead: Each of these directions requires additional processing, storage, and communication among different stakeholders. Often this overhead can exceed the benefits of the paradigm; therefore, an assessment and estimation of the trade-off between overhead and potential gain is needed.

Seamless Collaboration Among Stakeholders, Given the Limited Amount of Computing Resources at the Network Edge:

Different stakeholders, such as end-users, ISPs, and service providers, need to agree on protocols and incentive mechanisms (e.g., monetary, tit-for-tat) to enable seamless sharing of data and computation.

Security as a Built-In Design Component: Security is a vital component of the compute-less paradigm given the collaboration among different stakeholders and the sharing of data and computation.

Where Does Compute-Less Reside in the Spectrum of Computing Paradigm?

Figure 2 compares the communication and computation delays of different computing paradigms. The spectrum of computing paradigms ranges from remote processing on a distant cloud (i.e., high communication and low task execution delays, resulting in task completion times of several hundred milliseconds [10]) to edge computing (i.e., relatively low communication and moderate task execution delays, resulting in task completion times of tens to a few hundred milliseconds [10]) and running tasks locally at the end-user devices, which often exceeds the computing capabilities of the devices themselves. The compute-less paradigm achieves moderate to low utilization of network and computing resources and, in general, low task completion times. It utilizes edge servers that can be located at one-hop distances, but also a few hops away from users. In the former case, the servers can be accessed through direct links (e.g., LTE/5G, WiFi), while in the latter case, the servers may be interconnected through network elements (e.g., network routers). In addition to leveraging edge computing servers and in-network computing resources in a collaborative manner, compute-less features the reuse of previously executed computations, which can further reduce the computation burden at the edge and increase the capacity of available resources, while reducing the total execution time.

Use-Cases and Applications

The compute-less paradigm opens new perspectives and opportunities for applications and services that generate large amounts of data and/or result in large delays and overhead

if designed without the compute-less paradigm directives. Examples of such applications may include real-time computer vision, robotics, safety-critical systems, control systems, autonomous vehicles, and industrial machinery.

Command and Control Applications

Command and control is used in multiple applications that require sensing, monitoring, actuation, and fast and reliable control over a given mission or a set of actions. These systems consist of multiple actors/devices (including users, sensors, and robots among others) and may require the exchanges of several messages under strict latency and jitter requirements.

Application examples include military intelligence (e.g., surveillance, reconnaissance, electronic warfare), exploratory missions (e.g., find water in Mars), remote medicine (e.g., tele-surgeries), and services based on Unmanned Aerial Vehicles (UAVs) and autonomous vehicles. These applications (and more) have strict network requirements and require fast data analytics and reactive actions. While some of these applications utilize edge computing, the massive amount of data and computation required, especially in cases of crowded events such as concerts or the Super Bowl, make compute-less a great candidate for the realization of such applications. In the context of these applications, compute-less can reduce the load at the edge, enable fast processing, and reduce latency and jitter. For instance, the video streams of cameras broadcasting the Super Bowl can be aggregated and processed as a cluster of similar tasks (e.g., cameras sharing a common field-of-view or focusing on a single player).

Privacy-Preserving Services

Users, companies, and applications rely heavily on cloud vendors to secure their critical online data. This phenomenon has been motivated by the ease of executing complex tasks, such as big data analytics, on the cloud and utilizing different cloud services, such as data storage and business management platforms.

However, this ease is coupled with privacy concerns as private data may be traversing large parts of the public Internet and residing in third-party data centers. As an evolution of cloud computing, edge computing inherits the same challenge. On the other hand, the built-in security primitives of compute-less can enable privacy-preserving sharing of data and secure collaboration across stakeholders.

Time-Sensitive Applications

This category includes applications that require low-latency data processing beyond which the QoE of users deteriorates drastically. For example, mixed reality may include compute-intensive and potentially interactive applications that require low latency (e.g., augmented and/or virtual reality, object and/or face recognition). Figure 3 depicts a scenario of highly redundant object recognition tasks originated from different end-users. Compute-less will identify the similarity of these tasks, aggregate them, and forward them to the same edge/fog compute node so that the execution of duplicate computation is avoided and the results can be reused among tasks. Further examples may include:

- Gaming applications where compute-less offers methods that help similar computation (e.g., users playing the same game and sharing most of the objects) to be clustered and processed together as close as possible to the end-users. Therefore, gaming can benefit from sharing and reuse of computation across neighboring communities.
- Smart grid applications where big data is generated by numerous energy meters and microgrids, requiring low-latency processing on the fly to dynamically adapt to the load of the grid. In such cases, compute-less can reduce the data processing overhead, resource utilization, and response times.

Networking Platforms and Paradigms

Networking is in the core of the compute-less paradigm, since it impacts in-network computing operations and the adaptive and timely distribution of off loaded tasks for execution by computing resources [11]. Different networking platforms and paradigms can be adopted to enable the compute-less vision. In this section, we discuss the most relevant paradigms and explain how they can contribute to the realization of the compute-less paradigm.

Network Function Virtualization (NFV)

NFV proposes the virtualization of network functions (e.g., load balancing, intrusion detection, firewalls) that can be used for the realization of different communication services [12]. Furthermore, multiple functions can be chained together, so that complex functions are implemented. As mentioned previously, each computational task can be decomposed into a set of subtasks. Each subtask can be offered as a virtualized function, which can be deployed across different stakeholders (either within the network or at the edge servers) while being agnostic to specific deployment and hardware requirements. A task can be composed by chaining different subtask functions, making the reuse of similar subtasks/computation fine-grained. Finally, the compute-less security principles can be enhanced through the deployment of virtualized firewall and intrusion detection functions at strategic locations in the network.

Information-Centric Networking (ICN)

ICN proposes a network architecture that directly leverages application-defined, semantically meaningful names for communication purposes [13]. This name-based, stateful communication model can act as an enabler for the compute-less paradigm, allowing for the seamless and adaptive distribution of offloaded tasks toward different edge servers based on their resource utilization and availability over time [11]. Naming makes context about the offloaded tasks directly accessible to the network elements (e.g., routers) aiding in-network computing operations, while it also enables the seamless sharing of computation and data and the offloading of tasks across stakeholders. Furthermore, the built-in ICN security primitives enable the secure interactions across stakeholders and the verification of task results by users.

For example, let us consider an augmented reality application use-case, where image annotation tasks for images taken in the same room of a building are forwarded to the same edge server(s) through semantically meaningful naming conventions. An annotation task for an image taken in room 175 of the Computer Science (CS) department of a university U may have a name/image-annotation/ $U/CS/_room=175$, while an annotation task for an image taken in room 214 of the Electrical Engineering (EE) department at university U may have a name/image-annotation/ $U/EE/_room=214$. As a result, potentially similar tasks (e.g., images taken in the same room of a building) will be forwarded to the same set of servers, maximizing the likelihood of finding previously executed results to reuse. At the same time, if a previously selected edge server is out of available resources, tasks will be dynamically forwarded for execution toward any available server either within the same or across different edge networks.

Software-Defined Networking (SDN)

SDN features a logically centralized, programmable control plane, which can be utilized to install forwarding rules on network elements [14]. For the realization of the compute-less paradigm, such forwarding rules can facilitate the adaptive forwarding of tasks toward edge computing servers that have available resources or can reuse the results of previously executed tasks. Depending on the network architecture (IP-based or ICN-based), these rules will establish flows of traffic toward a set of edge servers. In IP, flows will be based on tuples of layer 2 to layer 4 header fields, while in ICN, flows will be based on semantically meaningful name prefixes.

Challenges and Future Research Directions

In addition to the challenges introduced above, in this section, we present a comprehensive list of challenges and research directions related to the compute-less paradigm.

Lightweight Clustering and Aggregation of Tasks:

Our design suggests that compute-awareness at the network layer can enable aggregation and clustering of similar tasks, constituting a key design feature of compute-less. However, such aggregation of tasks is challenging and requires:

- Profiling of applications and computational tasks.
- Similarity assessment to allow for on-the-fly forwarding decisions.
- Fast storage, processing, and forwarding of tasks.

When Does Computation Reuse Incur More Cost Than Benefit?

The reuse of computation is a fundamental principle of the compute-less design. However, storing the results of executed tasks indefinitely in the hope that they might get reused in the future is not feasible. To this end, we need mechanisms to estimate/predict the chances that the results of a task may be reused in the future as well as to what extent.

Furthermore, in some cases, executing a task from scratch incurs lower cost than searching and reusing parts of previous tasks. This can occur when we need to search through a large

data structure that stores previous tasks, while the task to be executed is not that complex. Another case is when the partial reuse gain is low enough compared to the lookup time to find a potential match. Efficient schemes are needed for decision making, while machine learning solutions can be utilized to achieve that based on a history of previously executed/stored tasks for each service.

In-Edge, Out-Container Sharing

Service providers may offer their services through containers over a shared edge infrastructure. In such cases, it is likely that some tasks or subtasks may be common among the deployed services. Through the collaboration and information sharing mechanisms enabled by the compute-less paradigm, subtasks/tasks can be shared and reused across service containers in order to eliminate duplicate computation. This direction requires further research on intelligent mechanisms to orchestrate the operation and communication among service containers.

Security and Privacy

Our compute-less paradigm introduces various security and privacy concerns including:

- Information sharing among involved stakeholders in a privacy-preserving manner.
- Protection of the edge computing resources from outsider attacks while preserving information sharing with other stakeholders.
- Establishing trust among stakeholders involved in compute-less in order to encourage collaboration and interactions.

Given that compute-less enables the offloading of computation from computationally weak devices to resourceful nodes either within the network or at the edge, offloaded computation needs to be “verifiable.” This enables end-users to independently verify the correctness of the received computation results. A potential line of future research may be to extend schemes that provide publicly verifiable cryptographic proofs of correctness [15]. Another line of future research may be the exploration of solutions based on trusted hardware frameworks (e.g., Intel SGX, ARM Trust-Zone) that will harden the execution of offloaded tasks by the computing resources.

Analytical Modeling and Optimization

Analytical modeling and evaluation of the compute-less design trade-offs is a vital direction of future research. Optimization techniques based on various constraints related to different stakeholders (e.g., users, service providers, infrastructure owners) can be utilized for:

- Task offloading and computation reuse.
- The allocation and management of computing and storage resources.
- Economic and other incentives that enable the collaboration among stakeholders.
- The scheduling and computing process within the network and at the edge.

The optimization techniques can be implemented in a centralized or in a distributed manner. Centralized approaches may be easier to deploy since the selected algorithms can be executed on a single server. Such approaches rely on the availability of the server itself, while they might not scale with the computation volume required by the optimization algorithms. To address this issue, there is a need for distributed approaches that can find a close to an optimal solution quickly, or at least in polynomial time. Stakeholders may collaborate through the mechanisms of the compute-less design to execute optimization algorithms locally and exchange necessary information in order to reach near-optimal solutions.

Machine learning and artificial intelligence are directions that need to be further investigated, since they can complement optimization frameworks for task offloading, computation reuse, and efficient utilization of resources. Deep reinforcement learning can also be used for the lightweight assessment of the similarity among previously executed and newly received tasks. Yet, the use of model-free, model-based reinforcement learning requires further investigation based on the network environment and requirements.

Conclusion

In this article, we introduced the compute-less networking paradigm, which aims to reduce the utilization of network and computing resources and provide fast response times as needed by delay-sensitive applications (e.g., autonomous driving, real-time data analytics, gaming). We discussed the fundamental design principles of compute-less networking along with their benefits, overhead, and trade-offs, as well as a number of challenges and directions for future research. We believe that compute-less networking will contribute to the realization and real-world, pervasive deployment of integrated networking and computing systems and collaborative environments that strive to minimize the utilization of resources for the completion of computational tasks offloaded by end-users.

Acknowledgements

The work of S. Mastorakis is partially supported by a pilot award from the Center for Research in Human Movement Variability and the NIH (P20GM109090), as well as a planning award from the Collaboration Initiative of the University of Nebraska system.

Biographies

Boubakr Nour [GS'17, M'20] (n.boubakr@bit.edu.cn) is currently pursuing his Ph.D. degree in computer science and technology at the Beijing Institute of Technology, Beijing, China. His research interests include next-generation networking and Internet. He was the recipient of the Best Paper Award at IEEE GLOBECOM (2018), and the Excellent Student Award at the Beijing Institute of Technology in 2016, 2017, and 2018 consecutively.

Spyridon Mastorakis (smastorakis@unomaha.edu) is an assistant professor in computer science at the University of Nebraska Omaha. He received his Ph.D. in computer science from the University of California, Los Angeles (UCLA) in June 2019. He also received an M.S. in computer science from UCLA in 2017 and a five-year diploma (equivalent to M.Eng.) in electrical and computer engineering from the National Technical

University of Athens (NTUA) in 2014. His research interests include network systems and protocols, Internet architectures (such as Information-Centric Networking and Named-Data Networking), and edge computing.

Abderrahmen Mtibaa (amtibaa@umsl.edu) is currently an assistant professor in the Department of Computer Science at the University of Missouri–St. Louis. Prior to that he held several research positions including visiting assistant professor in the Computer Science Department at New Mexico State University; research scientist at Texas A&M University; and postdoc in the School of Computer Science at Carnegie Mellon University. His current research interests include Information-Centric Networking, networked systems, social computing, personal data, privacy, IoT, mobile computing, pervasive systems, mobile security, and mobile opportunistic networks/DTN.

References

- [1]. Lin Let al., “Computation Offloading Toward Edge Computing,” *Proc. IEEE*, vol. 107, no. 8, 2019, pp. 1584–1607.
- [2]. Ullah R, Ahmed SH, and Kim B-S, “Information-Centric Networking with Edge Computing for IoT: Research Challenges and Future Directions,” *IEEE Access*, vol. 6, 2018, pp. 73 465–88.
- [3]. Kutscher D, Karkkainen T, and Ott J, “Directions for Computing in the Network,” Internet Engineering Task Force, Internet-Draft, 11. 2019, work in progress.
- [4]. Yizhou Let al., “Framework of Compute First Networking (CFN),” Internet Engineering Task Force, Internet-Draft, 11. 2019, work in progress.
- [5]. Król Met al., “Compute First Networking: Distributed Computing Meets ICN,” *Proc. ACM Conf. Information-Centric Networking*, 2019, pp. 67–77.
- [6]. Lee J, Mtibaa A, and Mastorakis S, “A Case for Compute Reuse in Future Edge Systems: An Empirical Study,” *Proc. IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2019, pp. 1–6.
- [7]. Guo P and Hu W, “Potluck: Cross-Application Approximate Deduplication for Computation-Intensive Mobile Applications,” *Proc. Int’l. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, vol. 53, no. 2. ACM, 2018, pp. 271–84.
- [8]. Guo Pet al., “Foggycache: Cross-Device Approximate Computation Reuse,” *Proc. Annual Int’l. Conf. Mobile Computing and Networking (MobiCom)*, ACM, 2018, pp. 19–34.
- [9]. Ahmed Eet al., “Bringing Computation Closer Toward the User Network: Is Edge Computing the Solution?” *IEEE Commun. Mag.*, vol. 55, no. 11, 2017, pp. 138–44.
- [10]. Ullah R, Rehman MAU, and Kim B-S, “Design and Implementation of an Open Source Framework and Prototype for Named Data Networking-based Edge Cloud Computing System,” *IEEE Access*, vol. 7, 2019, pp. 57 741–59.
- [11]. Mastorakis Set al., “ICedge: When Edge Computing Meets Information-Centric Networking,” *IEEE Internet of Things J*, 2020.
- [12]. Mijumbi Ret al., “Network Function Virtualization: State-of-the-Art and Research Challenges,” *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 1, 2015, pp. 236–62.
- [13]. Ahlgren Bet al., “A Survey of Information-Centric Networking,” *IEEE Commun. Mag.*, vol. 50, no. 7, 2012, pp. 26–36.
- [14]. Kim H and Feamster N, “Improving Network Management with Software Defined Networking,” *IEEE Commun. Mag.*, vol. 51, no. 2, 2013, pp. 114–19.
- [15]. Parno Bet al., “Pinocchio: Nearly Practical Verifiable Computation,” *IEEE Symposium on Security and Privacy*, IEEE, 2013, pp. 238–52.

Figure 1.

An overview of the compute-less paradigm. Collaboration occurs between ISPs and service providers at the edge to make compute-aware task forwarding and execution decisions.

Figure 2.

The spectrum of different computing paradigms. Trade-offs exist between task completion times (i.e., communication and computation delays) and resource utilization.

Figure 3.

Visitors take pictures of the statue of Nikola Tesla in Silicon Valley, CA in order to request information about this statue. The visitors provide “similar” inputs (pictures of the same statue through different angles) for tasks that will yield the same output (information about the statue). Subsequent tasks will result in duplicate computation, since the results of the first task can be reused.