CLOUD CONTINUUM

▪ Edge–Cloud Continuum

IEEE COMPUTER SOCIETY

IEEE

VOL. 2 • NO. 1 • 2022

www.computer.org/publications/cloud-continuum

ComputingEdge

Your one-stop resource for industry hot topics, technical overviews, and in-depth articles.

Cutting-edge articles from the IEEE Computer Society's portfolio of 12 magazines.

Unique original content by computing thought leaders, innovators, and experts.

Keeps you up to date on what you need to know across the technology spectrum.

Subscribe for free
www.computer.org/computingedge

IEEE COMPUTER SOCIETY™

IEEE

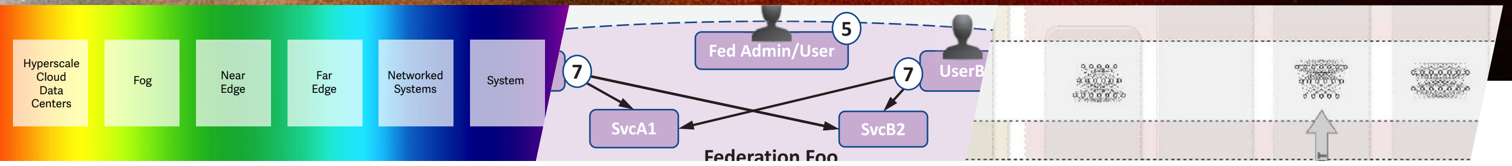
CLOUD CONTINUUM

IEEE COMPUTER SOCIETY computer.org

Circulation: *Cloud Continuum* is published by the IEEE Computer Society. IEEE Headquarters, Three Park Avenue, 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; voice +1 714 821 8380; fax +1 714 821 4010; IEEE Computer Society Headquarters, 2001 L Street NW, Suite 700, Washington, DC 20036.

VOL. 2 · NO. 1 · 2022

CLOUD CONTINUUM



The Edge, the Cloud, and the Continuum

Mazin Yousif, *Gainwell Technologies*

If we consider the cloud as the compute/data backend, and the edge the environment where the data sources are, we can imagine a huge space between them. That is what we refer to as the continuum, which can be single hop or multi-layer multi-hop involving very diverse technologies. This is true regardless of whether the edge is big or small, is computationally-intensive or computationally-limited, is physically in close proximity to the cloud, or is really far away from a cloud—pick your other favorite characterizations here.

Given the vast varieties of edge deployments, the majority requiring some sort of network connectivity to a cloud, you can imagine the enormity of the design space the edge-cloud continuum encompasses. Also, imagine the countless use cases, whether in industry, government, education, just plain consumers, and so on, that we can bring to bear. Hence, this publication will explicitly aim to span the full edge-to-cloud continuum.

What features of the design space make it so vast and complicated? Let's dive in with more details here. Without much effort, we can define at least six major considerations.

The first consideration is the edge, and how to classify an edge. We also hear about far-edge and near-edge conditions. An edge can be as small as a little sensor or camera detecting drops of rain or as huge as an oil and gas refinery, which can be as big as a small city. So, size matters here.

The second is the specific use case the edge is used for. For example, edge requirements for preventive maintenance are different from those for video analytics, or those for connected workers or those for traffic management systems, as the key drivers from latency, bandwidth, availability and reliability are different. Additionally, the industry vertical may add specific nuances to the above-mentioned example use cases.

The third is the architecture of the edge. This architecture varies depending on the complexity of the use case, the size of the edge, the network connectivity, the software layers and more.

The fourth is the network connectivity at the edge as well as between the edge and the cloud. Examples include wired, cellular, WiFi, LoRA, BLE, Sigfox, and many more.

The fifth is the appropriate and optimal software layers to deploy at the edge to support the underlying use case.

The sixth is the most appropriate and optimal analytics/ML/AI/DL that need to be used at the edge for the specific use case.

I can go on and on, but I think the above made my point clear. The many ways that exist to analyze and subdivide the design space add to the confusion about the terms "edge" and "fog" and how best to navigate the space between them.

One way to help bring order to this confusion is to consider the effects of scale. Let's look at a few example use cases, starting with small and simple ones and proceeding up to large and complex ones. Preventive maintenance for a pump in a manufacturing plant or refinery provides an example at the low end of this scale. We can expand this to include any set of connected devices in a manufacturing plant for a medium-scale example in terms of numbers and complexity. At the high-end would-be examples involving consumers or automated transportation (for example, connected drones). Other complex examples include autonomous supply chains, industry 4.0, autonomous and connected cars, smart agriculture, smart healthcare systems, and smart city use cases with services including transportation, utilities, parking, and energy. Through these examples I am trying to convey the essential point that although there are tons of use cases, they share features that can be addressed through design.

As a result of the impact of these technologies, there is high interest from all aspects of the community including research, professionals, educators, government, business, and more. For example, looking at the trend for Edge Computing in Google Analytics (Figure 1), you see the growth is amazing; not quite a hockey stick, but close to it. It is not that difficult to check industry market analysts on their Edge computing market projections.

The above and more are the reasons we launched this new edge-cloud continuum publication—*Cloud Continuum*.

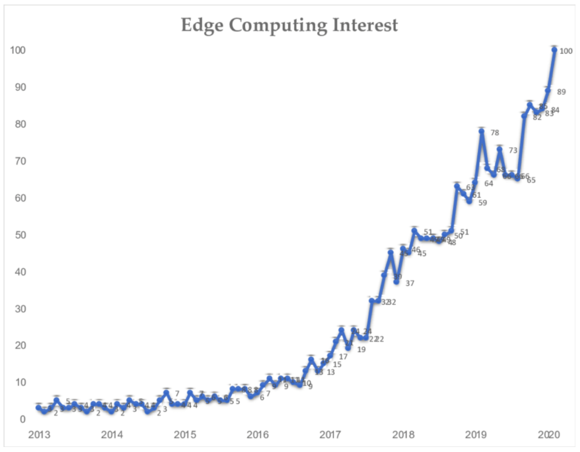


FIGURE 1. Trend for edge computing (https://www.researchgate.net/figure/Edge-computing-interest-Google-Trends_fig1_341037496).

THE PUBLICATION'S EDITORIAL BOARD

I have asked many of the editors who served on the Editorial Board of *IEEE Cloud Computing* that ran from 2014–2018 to join me in this new publication. Each will oversee one segment or area along the continuum: Joe Weinman (Digital Strategy and Economics), Alan Sill (Frontiers in Software, Architecture and Standards), Rajiv Ranjan (Blue Skies Research), Omer Rana (AI/ML for Cloud Continuum), Keith Jeffery and Lutz Schubert (International Research Directions), and Beniamino Di Martino (Cloud Continuum Interoperability). We will add other editors and columns in due time.

INNOVATIONS AND RESEARCH IN THE CONTINUUM

One goal for this publication is to further spur innovations and research for the continuum. Although the cloud is more than a decade old, it is still in a growing stage. I say it is a toddler. Adding topics related to the continuum between edge and cloud deployments and use cases as spelled out above, we see that the combination is ripe for innovation, invention, and research.

Key areas that this publication will cover include: (a) how best to represent an application to the cloud continuum; (b) how to optimize deployment of an application instance across the cloud continuum; (c) how, and to what extent, does the end-user control/steer the execution of the application across the cloud continuum; (d) availability, security, privacy across the cloud continuum; (e) application development environments for the cloud continuum; (f) data management across the cloud continuum; (g) economics of the cloud continuum; (h) how the cloud continuum can support the UN Strategic development goals; (i) interoperability across the cloud continuum; (j) the role of and the right AI/ML/DL in the cloud continuum; (k) how to manage data lifecycle in the cloud continuum; (l) scenarios to move the data where compute is and compute where the data is; and many more. In fact, I can enumerate beyond the letter (z).

CURRENT ISSUE

In this issue, which is intended to be an overview of the edge-cloud continuum, we have two original columns and one reprint article. Joe looks at the tradeoffs along the edge-cloud continuum and Alan introduces the "Frontiers" column with an invited article on the IEEE 2302-2021 standard on Intercloud Interoperability and Federation.

As the intention of this publication is to be a one stop shop for thought leadership on the edge-cloud continuum, we are also including reprints of select peer-reviewed articles and thought-provoking pieces from the Computer Society Digital Library. Specfically for this issue, we are including an article from the 13th IEEE/ACM International Conference on Utility and Cloud Computing that was held in 2020. Its title is "Scission: Performance-driven and Context-aware Cloud-Edge Distribution of Deep Neural Networks," and looks at the advantages of partitioning and distributing Deep Neural Networks across end-devices, edge resources and the cloud—basically the continuum.

EDITOR: Joe Weinman, joeweinman@gmail.com

DEPARTMENT: DIGITAL STRATEGY AND ECONOMICS

Trade-offs Along the Cloud–Edge Continuum

Joe Weinman

A range of options exist to implement today's digital architectures. Although an oversimplification, we can think of these options along a spectrum: At one extreme is a single instance "data center," which might be an enterprise data center or a colocation facility, or perhaps even a single server or rack containing a database or application. Adjacent to that option is a set of perhaps dozens of hyperscale cloud facilities, with perhaps hundreds of thousands of servers, typically geographically distributed. As we continue through various layers of "fog," we hit the near edge, which might include facilities in major metros, and then the far edge, which might include computing nodes located on every street corner, or spaced several per mile along a traffic corridor such as a superhighway. These then typically tie to endpoint elements, such as devices or things—for example, smart meters, irrigation sensors, smartphones, smart TVs, or connected vehicles. Sometimes individual processors will be aggregated into a system, such as an autonomous vehicle with a hundred or more microprocessors, or perhaps its entertainment system, or a network of systems, such as a factory including its robots, quality inspection systems, materials handling systems, and fire/smoke detection sensors.

Along the spectrum from one single instance data center to a trillion or more connected endpoints, some would argue that on-demand, pay-per-use, consolidated hyperscale facilities (the "cloud", the "center," or the "core") are the best choice to solve today's demanding IT challenges; others would say that a decentralized edge is the only solution. The truth is that there are benefits and disadvantages to each, and finding the right weighting of components by considering the trade-offs along the cloud-fog-edge continuum (Figure 1) will typically be the best choice for many business and technology challenges.

We will focus here on the general characteristics of cloud vs. edge. Each of the accepted elements of a "cloud" can generate benefits. For example, on-demand resource provisioning in the presence of variable demand generates benefits through right-sized capacity, since too many resources leads to waste and thus unnecessary costs, and too few resources drives opportunity costs through the missed revenue, profit, or other benefits associated with the application that is executing on that capacity.[1] Finer granularity of that capacity for shorter intervals can lead to less waste; just as it's cheaper to buy a glass of wine than the chateau, and less costly to run an appliance for a shorter period of time.[2] In a computing context, granularity gets finer as we move from monolithic systems to servers to cores to virtual machines to containers to microservices. Pay-per-use can lead to benefits even with a unit cost premium, since although one may pay more for resources that *are* employed, one *doesn't* pay for resources that are *not* used, unlike with fixed, owned capacity.[3]

However, all of these characteristics—on-demand provisioning, fine resource granularity, shorter intervals, and pay-per-use pricing—are feasible at the edge, even if they originated with the cloud. Thus, we must look to other characteristics to understand the trade-offs between cloud and edge. We also must consider atypical variations. For example, the center may be a single facility, such as a solitary enterprise data center serving global demand from customers, employees, and partners. However, the edge also may be a single location, such as a stand-alone facility that does image processing for railroad car maintenance deep in a rural area. Typically, though, we think of the cloud/core/center as a few, hyperscale, consolidated facilities; and the edge as a globally geo-dispersed set of smaller locations.

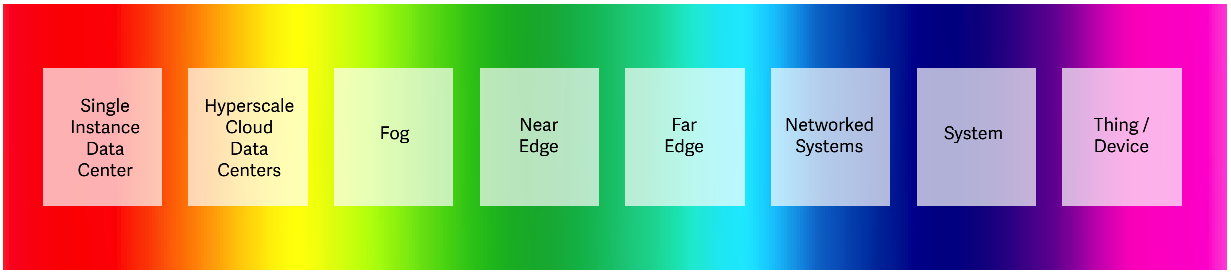


FIGURE 1. The cloud-fog-edge continuum.

SERVICE AND DATA PROXIMITY

The edge is in proximity to where sensor data is collected and used. On the other hand, centralized facilities can be in proximity to data and related application components or services. Proximity can be unhelpful, as in the case of a localized disaster such as a hurricane or flood that destroys, say, an original and its only replica, but is generally good, for example, to reduce data transport costs, minimize bandwidth requirements, and minimize latency.

Edge

The edge is in close proximity or even co-located with the user devices and things that include sensors that collect data and the actuators that causally interact with the real world. This includes smart phones, tablets, smart TVs, connected soccer balls, connected autonomous and non-autonomous vehicles, drones, robots, flexible manufacturing cells, video surveillance systems, irrigation sensors, connected trash receptacles, and so on. As video typically drives the majority of data collection, and resolutions and frame rates increase, local processing can have numerous advantages.

Cloud

The cloud is in close proximity to—in fact, likely co-located with—other applications, services, and/or single-instance or master replicas of cloud-based data lakes or data warehouses (or lakehouses). It is also likely to be in close proximity to major backbone network transport and interconnection facilities, compared to many edge locations that may be constrained by local wireline, wireless, or satellite access network capacity and capability. Proximity is not just a function of geographic distance, but network latency, which can be impacted by routing, congestion, bandwidth limitations, and outages.

LATENCY

Edge

Latency from endpoints to the cloud may be too long for many applications. In the worst case, a remote cloud may be 150 milliseconds or more round trip from a given endpoint. Thus, an application that needs time at the endpoint, time at the remote server or servers, and time to traverse the distance between them may take too long to meet user needs or application requirements. Especially for applications such as connected autonomous vehicles communicating with each other or with pedestrians or roadside infrastructure, low latency is critically important. A few hundred milliseconds of additional latency to communicate, perhaps via multiple round trips, may literally be a life-or-death matter.

Many other applications can also benefit from low latency at the edge. These may be stateful and perhaps require local data which may be ephemeral or long-lived, say, to authenticate user access to certain services. Or they may be ephemeral and event-driven, say, image processing a license plate for upload to a criminal tracking or toll collecting database. Multiple image sensors may need to efficiently and rapidly collaborate to correctly identify the vehicle. In any event, performant interactions between sensors, actuators, processors, and data may be critical.

Cloud

However, latency between various application components, microservices, or data spanning a distributed edge may also be an issue. Applications and data have different characteristics, and thus the advantages and disadvantages of the cloud can differ. A single monolithic application running in a legacy or high-performance computing environment can benefit from tight coupling between modules. For example, such an application might exploit a hypercube or fat-tree network topology implemented in a high-performance backplane in a single physical system.

But even a distributed architecture in a hyperscale cloud can have performance optimized via latency reduction enabled by proximity. For example, today, a typical application will be architected as a set of microservices interacting with each other across APIs. Having all connected microservices co-located at a single cloud location or rack can minimize inter-service latency, and thereby maximize performance, throughput, and response time.

At an even higher level, a typical workflow may consist of dozens of these applications or services. For example, the workflow to process a customer order may involve image capture, data validation against a customer, product, and/or geographic database, payment authorization, entry into a manufacturing planning system, and so forth. Even a few milliseconds extra delay may not seem like much, until one realizes that this may be multiplied by hundreds or millions of orders.

LOCAL VS. GLOBAL

We live in a diverse world. There are different countries, on different continents, with different climates and geological characteristics. Even within any country, there are subdivisions such as provinces, states, and special administrative regions. Then there are urban, suburban, and rural areas. Many localities have unique risks, challenges, economies, cost structures, environmental priorities, and legal and regulatory frameworks.

Edge

Rather than a one-size fits all approach globally, edge resources can vary to fit local requirements. For example, in a city, edge nodes can be very dense and support 5G mmWave communications and high data volume of multiple Terabits per second per square kilometer. Backhaul can be high-capacity fiber. In a more rural area, a trade-off can be made between cost, coverage, and density, perhaps using 4G LTE Advanced or 5G mid-band wireless with its better propagation characteristics and larger radius of coverage. Instead of fiber backhaul, integrated access and backhaul or a wireless or microwave mesh can be used for coverage. Points of presence can be adapted to local conditions—for example, degree of ruggedness against heat or humidity.

Cloud

Cloud services can take advantage of the unique benefits of a specific region, and "broadcast" those benefits around the world. For example, data centers in Iceland can be more green than in other locations, because they can be powered by geothermal heat and cooled much of the year by the ambient environment, leading to a net-zero carbon footprint. These data centers can run applications and services supporting needs around the world, for the limited marginal cost of networking. This is advantageous compared, to, say, a data center local to its users that is powered by coal-generated electricity and also needs to power its cooling systems.

DATA SOVEREIGNTY, CONTROL, AND PRIVACY

There are a variety of unique, innovative ideas for siting processing, storage, and/or networking resources. For example, Facebook had examined solar-powered planes,[4] Google had explored hot-air balloons with Project Loon,[5] and floating or platformed offshore data centers are in service.[6,7] However, most data centers are located on land, within the jurisdiction of nation-states with particular societal objectives and constraints embodied in laws, statutes, regulations, and executive orders.

Edge

Resources located at the edge can conform to regulations such as those that mandate that no personal data can leave a given geographic region, a boon for user privacy. In the case of three letter government agencies or military groups, the ideal approach is not just to have a firewall between the compute and data resources, but to air-gap them entirely to preserve security. Such control, privacy, and sovereignty are thus best maintained via local resources, such as the edge and/or endpoints.

Of course, connected edge resources can still be attacked. For example, a steel mill in Germany suffered severe damage after hackers infiltrated its control systems, thus letting a furnace overheat and causing physical destruction.[8] Even air-gapped resources can be attacked. The vector to carry out an attack on a uranium processing facility in Natanz, Iran, was the Stuxnet virus, brought in on a USB drive.[9]

Cloud

Resources located in centralized fashion can be subpoenaed and/or forensically analyzed by the appropriate legal authority. For example, the U.S. National Security Agency has had a program called "Prism" that allegedly enabled the NSA to have direct access to servers belonging to Google, Facebook, and Apple.[10] This enabled direct government access to many types of data such as search queries and the contents of email and files. In the case of "keyword warrants," a government can broadly request information on anyone who has used specific search terms, and for "geofence warrants," a cloud or network service provider may have to share information concerning anyone within a given geographic region where a crime was committed. Although there may be diverse opinions on the benefits or concerns of such actions, from a technology perspective it is clear that having such data within a jurisdiction makes it easier to legally access.[11]

DATA MANAGEMENT PRACTICALITY

Sufficiently large datasets may be too large to store in their entirety at the edge, or too large to transport in their entirety to the cloud.

Edge

Endpoints are the first location for data capture via sensors, and the edge is typically the first stop on data's journey to other destinations, such as peer endpoints or to the cloud. We say "typically the first stop," because endpoints may be configured into a peer-to-peer mesh that packets traverse before they reach the edge. The volume of data that is captured may far exceed the capacity of access or edge networks to transport in a useful amount of time, requiring compression or selection of relevant subsets of the data.

Cloud

Consider the size of the index that Google builds as it crawls the web to enable search query processing. It has been estimated to be "hundreds of billions" of web pages amounting to "well over 100,000,000 gigabytes."[12] It is obviously not possible to store this much data in today's popular endpoints, such as smartphones with 128GB of total storage, or even today's edge nodes. As device and edge capacity grow, so will the size of databases such as that index, meaning that this constraint is unlikely to ever vanish.

DATA REPLICATION COSTS

The larger that a collection of data is, and the more times that it is replicated, the higher the costs of storing and managing that data. On the other hand, if the data can successfully be partitioned, total storage capacity requirements and costs won't vary very much between a few locations that aggregate the data and a multitude of locations with small portions of the data and few replicas.

Edge

The degree of replication vs. partitioning determines the overhead—if any—associated with edge storage. As an example, consider the storage requirements for 100GB of personal data such as family movies and photos. It requires 100GB whether stored in a smartphone, at the edge, or in the cloud, if only one copy is stored. On the other hand, replicating the same content multiple times takes a lot of aggregate storage. For example, there are tens of thousands of different types of devices running Android, for a total of over 3 billion devices.[13] Depending on the version of the OS, it may need 10, 20, or even 30GB on each device , leading to a total storage of roughly 100,000,000,000GB, or 100 exabytes.

Cloud

Suppose that instead of Android, we consider the scores of petabytes for Google's search index. If we tried to store it in every one of the 3 billion Android endpoints, or the tens of billions of connected devices, it would obviously be very costly, if there were even that much storage capacity in the world.

It is thus self-evident that massive data sets are unlikely to be replicated unnecessarily—there must be a compelling reason such as performance, control, or privacy to drive such replication.

STATISTICAL EFFECTS ON RESOURCE UTILIZATION

Demand for processing typically varies over time for most applications and services. Workloads as diverse as ecommerce and connected vehicles have circadian rhythms, since very few people shop online or drive in the middle of the night—although these patterns may change as, say, underutilized autonomous vehicle resources become pressed into service to make deliveries. Demand also varies based on events, such as the Olympics, or seasonal factors, such as tax-filing deadlines, Singles' Day, or Black Friday. Generally speaking, the coefficient of variation—a measure of variance relative to a given mean level of demand—becomes lower as the number of independent aggregate workloads is increased—indicating that the aggregate demand is smoother, that is, has less variability. Such statistical multiplexing effects—such as serving diverse, uncorrelated demand from a pool of common resources—display well-known behaviors as the number of demands increases.[14]

Edge

Single workloads typically have great variability, whether it is playing Wordle once a day for a few minutes, or doing a monthly backup for eight hours.

The edge is typically an aggregation point for multiple workloads, such as image processing across multiple quality control inspection points in a factory or video stream processing of license plates or traffic congestion, or intermittently connected endpoints, such as vehicles connecting to base stations or other highly dense edge processing facilities. Interestingly, the $1/\sqrt{n}$ measure of coefficient of variation of the aggregate of n workloads implies that the reduction in variablity is steepest when n is smallest.[15] In other words, not that many independent workloads need to be aggregated to achieve substantial capacity benefits.

Cloud

Hyperscale facilities that aggregate the most independent workloads have the lowest demand variability, because peaks in some workloads correspond to valleys in others. Of course, not all real-world workloads are independent and uncorrelated, due to correlative factors such as circadian and seasonal rhythms. However, there will still clearly be benefits compared to the variability of a single workload, and, in the case of hyperscale clouds unused capacity can easily be filled with deferrable or discretionary work, incented via dynamic pricing.

STATISTICAL EFFECTS ON PROFITABILITY

There is a direct relationship between resource utilization and profitability, therefore there are trade-offs between cloud and edge that depend on their capacity, their cost structure, and the statistics of workloads running on their compute and network resources.[16] For example, a facility that only runs at 50% utilization must recover not just the cost of a used resource, but also the cost of the matching unused resource. In such a case, if a server costs, say, $1 to run per day, including leasing, depreciation, installation, maintenance, power, and so on, it must be billed out at $2 per day just to break even, not including sales, general, and administrative costs. A facility that runs at 100% utilization can make money, conceptually, only charging $1.01 per day. Of course, other factors, such as powering down or otherwise idling unused servers, consolidating workloads on virtualized servers, storage hierarchies, volume discounts, better PUE (Power Usage Effectiveness), etc., impact these calculations, but the general principle remains.

The statistics of storage utilization also may make a difference. Sometimes storage requirements are monotonically nondecreasing, as more and more data is collected. Sometimes they vary up and down, as with a test/dev workload.

In any event, the greater the workload variability, the greater the aggregate variability. The greater the number of workloads of a given variance, the smoother the aggregate behaves.

Edge

Edge resources can achieve higher utilization levels than most endpoints, but not as high as clouds, as described above. However, "profitability" may not be a major concern. A factory with a private edge and perhaps private 5G may have lower utilization than a cloud, but also does not need to spend money to fund a cloud provider's profits. A city may choose to deploy edge resources to help with public safety, public health, emergency services, or to address the digital divide. A user deploying a residential gateway in their home is probably more concerned with functionality than with "making a profit" on family members Internet usage.

Cloud

Clouds and their hyperscale facilities that multiplex massive numbers of workloads and then also fill in any valleys with deferrable and discretionary workloads will achieve the highest possible utilization levels. According to one estimate, cloud servers operate at 65% utilization, whereas on-premises—such as facilities that are likely to combine fewer, less diverse workloads—are more likely to operate at 12-18% utilization.[17]

STATISTICAL MULTIPLEXING EFFECTS ON RESOURCE AVAILABILITY

Automobiles typically have a 25% safety margin on tires (the spare tire in the trunk in addition to the four pressed into service) and a 5 to 10% safety margin on fuel (the reserve capacity after the meter shows "empty"). Similarly, data centers and networks are typically overprovisioned to provide a safety margin above and beyond what might be expected during "normal" peak demand. This can help ensure service continuity even during unexpected spikes, or during the failure of some components. The degree of overprovisioning can depend on the variability of demand, the criticality of the workload, the supply chain availability of resources and other factors.[18]

Edge

Edge resources disaggregate the overcapacity, or safety margin. Each resource must be engineered to allow for unplanned capacity issues or component failures, the same way that every car must carry its own spare tire. This is the case even though the vast majority of these tires are never needed at any given time.

On the other hand, if edge resources are configured into a mesh and can share capacity, then their geographic proximity can reduce these requirements, the same way that a neighborhood can get by with only one homeowner owning, say, a chain saw, and sharing it with neighbors when needed.

Cloud

Clouds require much less capacity to meet the same safety margins. The reason is that a demand spike in a few workloads is likely to be complemented by a demand trough in a few other workloads and near-average demand in others. (Both the analysis of cloud overhead capacity requirements and edge overhead capacity requirements assumes that the workloads are independent and uncorrelated, an assumption which is sometimes violated due to circadian, weekly, or seasonal correlations).

Clouds can also share capacity, either by running excess workloads in a different location or even a different cloud.[19]

RESILIENCE AND SECURITY

Clouds or edge nodes from a single given provider or using a given stack are subject to systemic outages—for example, due to a software bug, human error in updating configurations,[20] or an unexpected clash of algorithms. Often, however, clouds and edge nodes differ in their resilience in the face of different events or failure modes.

Edge

Edge nodes may be located in a server room, a corner of the factory, a street corner, along a traffic corridor, or on a remote farm or oil pipeline. This makes them difficult to physically secure, because there are too many locations to guard, and often, they are space-constrained.

On the other hand, loss of one or even several edge nodes is likely to be a drop in the bucket in terms of total processing capacity for an edge network.

However, services offered may be impacted substantially within the localized service area of those lost nodes. For example, loss of nodes near the corner of Elm and Main St. may impact vehicle collision avoidance at that intersection.

Cloud

Cloud data centers and other centralized facilities can and do have excellent physical security. They typically have few windows, and have man traps, video surveillance, and even armed guards, sometimes from special services.

On the other hand, a large data center is a large target for a flood, hurricane, tornado, lightning strike, power outage, or terrorist attack.

In other areas, cloud security can have advantages. For example, the high access bandwidth of these facilities can better defend against Distributed Denial of Service attacks. The largest such attack as of this writing was a 3.47 Terabit per second DDoS attack from 10,000 servers, and was successfully stopped by a major cloud provider.[21]

There are numerous other factors that may affect the balance of trade-offs, such as provisioning, operations, configuration, administration, monitoring, maintenance, and management of architecture components. However, the bottom line is that various applications and organizations may benefit more from centralization or more from decentralization. That said, the exponential growth in data capture at endpoints is inexorably tilting that balance to the edge.

REFERENCES

1. Weinman, Joe. *Cloudonomics: The business value of cloud computing*. John Wiley & Sons, 2012.
2. ibid
3. ibid
4. Coldewey, Devin. "Facebook permanently grounds its Aquila solar-powered internet plane," *TechCrunch*, June 26, 2018. https://techcrunch.com/2018/06/26/facebook-permanently-grounds-its-aquila-solar-powered-internet-plane/
5. Singh, Manish. "Alphabet shuts down Loon internet balloon company," *TechCrunch*, January 21, 2021,. https://techcrunch.com/2021/01/21/google-alphabet-is-shutting-down-loon-internet/
6. "Offshore Cryptofarm." https://offshore.farm/
7. Roach, John. "Microsoft finds underwater datacenters are reliable, practical and use energy sustainably," *Microsoft Innovation Stories*, September 14, 2020. https://news.microsoft.com/innovation-stories/project-natick-underwater-datacenter/
8. "German Steel Plant Suffers Significant Damage from Targeted Attack," *Trend Micro*, January 12, 2015. https://www.trendmicro.com/vinfo/fr/security/news/cyber-attacks/german-steel-plant-suffers-significant-damage-from-targeted-attack
9. Terdiman, Daniel, "Stuxnet delivered to Iranian nuclear plant on thumb drive," *CNET*, April 12, 2012. https://www.cnet.com/news/stuxnet-delivered-to-iranian-nuclear-plant-on-thumb-drive/
10. Greenwald, G., and McAskill, E., "NSA Prism program taps in to user data of Apple, Google and others," *The Guardian*, June 6, 2013. https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data
11. Brewster, Thomas, "Exclusive: Government Secretly Orders Google To Identify Anyone Who Searched A Sexual Assault Victim's Name, Address Or Telephone Number," *Forbes*, October 4, 2021. https://www.forbes.com/sites/thomasbrewster/2021/10/04/google-keyword-warrants-give-us-government-data-on-search-users/
12. "How Search Works," *Google*, https://www.google.com/search/howsearchworks/how-search-works/organizing-information/
13. Simon, Michael. "Why 64GB isn't enough space for Android phones anymore," *PCWorld*, June 10, 2020. https://www.pcworld.com/article/399260/why-64gb-isnt-enough-space-for-android-phones-anymore.html
14. Weinman, Joe. "The economics of computing workload aggregation: capacity, utilization, and cost implications." *IEEE Cloud Computing* 4.5 (2017): 6-11.
15. Weinman, Joe. "Smooth operator: The value of demand aggregation." (2011).
16. Weinman, Joe. "The economics of computing workload aggregation: capacity, utilization, and cost implications." *IEEE Cloud Computing* 4.5 (2017): 6-11.
17. Barr, Jeff, "Cloud Computing, Server Utilization, & the Environment," *AWS News Blog*, June 5, 2015. https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment/
18. Weinman, Joe. "The economics of computing workload aggregation: capacity, utilization, and cost implications." *IEEE Cloud Computing* 4.5 (2017): 6-11.
19. Weinman, Joe. "Intercloudonomics: Quantifying the Value of the Intercloud." *IEEE Cloud Computing* 2.5 (2015): 40-47.
20. "Summary of the December 24, 2012 Amazon ELB Service Event in the US-East Region." https://aws.amazon.com/message/680587/
21. Tung, Liam, "Microsoft: Here's how we stopped the biggest ever DDoS attack," ZDNet, January 31, 2022. https://www.zdnet.com/article/microsoft-heres-how-we-stopped-the-biggest-ever-ddos-attack/

The IEEE 2302-2021 Standard on Intercloud Interoperability and Federation

Seamless Cloud-agnostic Interoperability Management for Resource Discovery, Access and Trust

Craig A. Lee, *Federation Partners, LLC, lee@keyvoms.org*
Robert Bohn, *NIST, robert.bohn@nist.gov*
Martial Michel, *Data Machines Corp., martialmichel@datamachines.io*

EDITOR'S NOTE

Frontiers in Software, Architecture, and Standards

This column in *Cloud Continuum* is titled "Frontiers in Software, Architecture, and Standards" or "Frontiers" for short. It will cover emerging design and implementation patterns being used to lay the software, architecture, and standards groundwork for new functional capabilities in the cloud-to-edge continuum. In it, we'll use articles, interviews, and contributions from leading proponents for new cloud tools to explain how they are being developed and designed to work together and the capabilities they provide. Wherever possible, we'll focus on areas that include opportunities for open source development and community involvement in carrying out these advances.

I'm excited to include as the first article in this series a high-level description from Craig Lee, Robert Bohn, and Martial Michel of the newly published *IEEE Standard on Intercloud Interoperability and Federation* (SIIF), IEEE 2302-2021, completed this past December and already gathering attention and implementations. I've followed this important area of work as it has been developed over the past several years and think that this new standard represents a real advance in both of the areas addressed in its title. The formal descriptions of these topics are laid out in easily understood terms in this article, and a standards basis for them is both highly desirable and necessary as a building block for future developments. I look forward to further adoption and refinement of these concepts in working software in the near future.

We are pursuing and interested in other topics with potential high impact in the areas covered by this publication, including service mesh design, ingress and security controls, toolkits and frameworks for multi-scale systems, and technology developments that can help to tame and control the energy hunger of large-scale computing. If you have contributions to make in any of these areas, or in other topics that meet the focus above, please write to the Editor (Mazin Yousif; mazin@computer.org) or to me directly at alan.sill@standards-now.org so we can engage on these topics. Submissions should be technically accurate and written in a style that is informative to experts, focusing on the highlights, but readable and accessible by non-experts interested in cloud technologies.

We look forward to hearing from you and to the new directions to be explored in this feature.

This standard enables generalized federation, supports cloud-to-cloud interoperability, and provides an evolutionary pathway for federation systems.

The need to collaborate is fundamental. DropBox and Google Docs are two simple examples of collaboration based on managing access to documents in a shared, centralized repository. However, there are many other collaboration use cases that involve i*nherently distributed* environments. Such collaboration requirements can be addressed using *federation*. The IEEE 2302-2021 Standard on Intercloud Interoperability and Federation[1] was developed to enable federation for data and service sharing in heterogeneous distributed environments, including edge computing, sensor networks, and the Internet of Things. Furthermore, general federations can be used to federate resources at any level in the system stack, whether they are bare-metal or virtualized

The term federation gets used informally to mean different things. Sometimes it denotes just a vague notion of "sharing" by some undefined means. Other times federation is used to denote what is actually just the a*ggregation* of data. For this IEEE standard, the term federation strictly means the instantiation of *virtual trust domains*. These trust domains constitute purpose-built, *federation instances* where the discovery and access of shared resources are strictly controlled by well-defined identities, attributes, authorizations, and policies.

The need for this kind of collaboration has been long recognized. Earlier work used the term Secure Virtual Enclaves[2] to denote a collaboration environment for distributed application objects. During the early 2000s, the term Virtual Organization was coined to denote shared environments that collaborators could join and share resources.[3,4] With the advent of cloud computing, cloud federation was recognized as a high-priority collaboration requirement in the *NIST Cloud Computing Technology Roadmap*.[5] Pursuant to this need, NIST and the IEEE launched a joint federation project. NIST extended the *NIST Cloud Computing Reference Architecture*[6] to become the *NIST Cloud Federation Reference Architecture*.[7] IEEE subsequently defined the IEEE 2302-2021 standard based on the NIST model.

In the course of this work, it became clear that the security enclaves and virtual organizations of previous work were fundamentally *federation instances*, where a federation instance is a *virtual administrative domain*. It is a domain that can be administered like any other domain, except that it is virtual and not part of any one organization. A federation instance can also be called a *distributed trust domain*. It is a trust domain with a perimeter that is distributed and virtual, that is, software-defined. This standard codifies the fundamental model and core API for this approach.

To motivate the reader, the next section describes a number of end-user application domains that fundamentally need federation to provide definite societal benefits. This is paired with a short review of federation-related systems and how they compare to the IEEE 2302-2021 approach. After this, basic federation terms and concepts are introduced prior to introducing the *Federation Hosting Service (FHS) model* and its APIs. For true, wide-spread adoption to occur, a number of additional supporting capabilities need to be incorporated in the overall architectural approach. These are described in the subsequent section, prior to ending with the Conclusions.

APPLICATION DOMAINS

Many application domains involve inherently distributed environments but cannot be fully and effectively realized because of the lack of standardized federation tooling. Here are some high-level examples:

- *Cloud Bursting:* Occurs in situations where the capabilities of a primary cloud are expanded by using an external cloud. One motivation for cloud bursting is a datacenter with a private cloud needs to use additional cloud resources from a public cloud when its capacity or capabilities are exceeded. However, in other circumstances, access to specialized hardware or localized data, alleviating resource constraints, and distributing costs to additional resources are the driving factors. The motivation for using federation is to uniformly manage access and authorization at the cloud infrastructure level. A concrete use case described below.
- *International Disaster Response:* Effective large-scale disaster response requires governments, multi-national organizations, and officials in the affected regions to securely share information for accurate situational awareness. Outside of the disaster region, information sharing is critical for the effective planning and delivery of relief supplies. Inside the region, relief agencies must coordinate with the first responders even though the in-region infrastructure will be degraded. This secure information sharing can be managed as a federation instance where data access is based on stakeholder authorizations.
- *Virtual Mission Infrastructures:* Modern organizations must collaborate with each other and with external mission partners for their

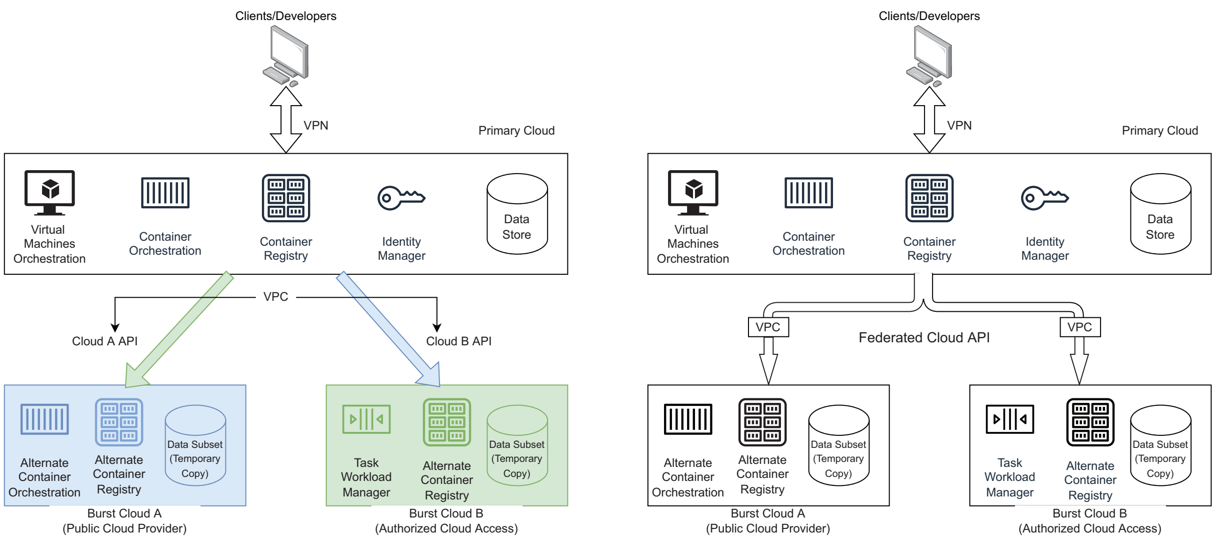

Clients/Developers
VPN
Primary Cloud
Virtual Machines Orchestration
Container Orchestration
Container Registry
Identity Manager
Data Store
VPC
Cloud A API
Cloud B API
Alternate Container Orchestration
Alternate Container Registry
Data Subset (Temporary Copy)
Burst Cloud A (Public Cloud Provider)
Task Workload Manager
Burst Cloud B (Authorized Cloud Access)
Federated Cloud API

FIGURE 1. Using federation to manage cloud bursting.

intelligence and business goals. This encompasses the acquisition and dissemination of data from different data producers and consumers. Federation directly supports the notion of virtual mission infrastructure whereby mission partners can access the data needed based on authorizations regardless of source.

› *National Strategic Computing Reserve:* The National Strategic Computing Reserve[8] is an initiative being developed by the National Science and Technology Council. The motivation is to have a connected reserve of compute resources that is available during times of national emergency. This strategic reserve could support efforts such as the COVID-19 HPC Consortium for work in bioinformatics, epidemiology, and molecular modeling. The planned NSCR Implementation and Operations have specifically identified the need for dynamic federation across NSCR stakeholders to manage the integration and use of resources.
› *Federated Analytics:* The Confidential Computing Consortium has identified a number of use cases for Trusted Execution Environments (TEE) and is developing a tool set to support them. The *Federated Analytics* use case[9] involves a remote user that exports data that is encrypted until it enters the hardware-protected TEE memory. The analytics are executed and the results are encrypted and returned to the remote user. The authorizations to discover and use such remote TEE analytics can be managed as part of a federation.

CLOUD BURSTING USE CASE

A cloud bursting solution for a given cloud is tied to that burst cloud's computation orchestrator, data storage, and other cloud-specific components under the service provider control. Figure 1 (left) illustrates where a client/developer has access to a primary cloud with its specific Virtual Machines Orchestration mechanism to support Container deployment (Orchestration + Registry), its own Identity Management and Data Store. This primary cloud owner has negotiated access to "Burst Cloud A" from a public cloud provider, as well as "Burst Cloud B" from another entity.

Currently, a primary cloud owner has to implement case-by-case methods for its client-specific requirements on Burst Clouds A and B such that they appear transparent to the user. A Virtual Private Connection (VPC) is created with both clouds. Identity is negotiated by the primary cloud in order to enable access and billing of resources, and the containerized application is transferred from the primary cloud's container registry to the burst cloud's own registry. Finally, in order to limit cost, only a known subset of data that the application would use is copied over to the remote cloud. When an application is set to be executed on those burst clouds, they need to be run using the orchestration mechanism specific to each cloud, container orchestration or task workload. Each one of these operations is dependent on the burst cloud, and requires a specific implementation using that cloud's service level API.

With cloud federation, in Figure 1 (right), a common set of APIs governs the interactions between the primary cloud and each burst cloud, which alleviates implementing burst cloud-specific solutions. The primary cloud owner can use the Cloud Federation API to initiate each burst cloud interaction, such that each burst cloud represents it in its own local API. This simplifies the implementation and integration to additional potential burst clouds.

FEDERATION-RELATED SYSTEMS

Given the fundamental need for distributed collaboration, a number of federation-based and federation-related systems have been built and are operational. These are, however, limited point solutions and narrow in purpose. They include

- *R&E Federations:* Many Research & Education federations exist to serve academic requirements. The most prominent examples are *InCommon*[10] and *eduGAIN*.[11] InCommon works by maintaining the *metadata*, which is a flat file of Identity Providers and Service Providers. Academic participants can download the latest metadata for Identity and Service Providers that can be used. eduGAIN is a global academic interfederation service. eduGAIN works by consolidating multiple national metadata registries into an international metadata registry. Many other federation-related systems exist (EGI, EOSC, Gaia-X, eduRoam) but cannot be covered here for lack of space.
- *Dataverse:* Dataverse[12] was developed to enable institutions and projects to share data through a proxy approach. Institutions deploy a Dataverse proxy and configure it to peer with other institution's proxies. Data owners can register their data and make it discoverable. Dataverse has an extremely rich API for doing all manner of operations on data of all types, shapes and sizes. The proxy approach that the IEEE 2302-2021 model uses has strong parallels to that of Dataverse. Hence, 2303-2021 has many of the same flexibility for supporting different deployment and governance requirements.
- *OpenStack:* The OpenStack Keystone v3 service[13] has basic support for the federation of cloud infrastructure services. This was developed with the goal of supporting hybrid-cloud business models, and allowing one OpenStack cloud to cloud-burst into another. The approach enables an administrator to configure their Keystone to *federate-in* and *federate-out* with other specific OpenStack installations. Federate-out enables the local Keystone to act as an Identity Provider for a remote Keystone. Federate-in enables users from remote OpenStack installations to invoke services on the local OpenStack.
- *Secure Production Identity Framework for Everyone (SPIFFE):* The SPIFFE model,[14] developed by the Cloud Native Computing Foundation, manages cryptographic identities at the service mesh level within trust domains. The SPIFFE model supports identity federation across trust domains by exchanging trust bundles of trusted identity information. Of course, managing general federations will require SPIFFE to be augmented to manage resource discovery, access, policies, and so on.

A key observation here is that all of the application domains mentioned would benefit from *purpose-built*, *on-demand*, federations. Rather than one "flat" federation, purpose-built feds can define their own "rules of the road" governing what the members can do to achieve common goals. As a distributed trust domain, purpose-built federations can have their own identity credentials that are meaningful, appropriate, and trusted. The *ad hoc* federation of trust domains requires the interoperation of identity silos that were not designed or intended to do so. In this case, *attribute release* becomes a common problem since it may not be appropriate to expose some user attributes in external environments. Having a standards-based approach to general federation wherein membership identity, resource discovery and access can be uniformly managed is a great advantage.

The data sharing capabilities of Dataverse are a tremendous achievement, but the ability to manage remote analytics on data, such as, services, would benefit many application domains. While federating cloud infrastructure services is a significant use case by itself, federation can be done at any level in the system stack and for arbitrary, application-level services. This is even true for service mesh and zero-trust architectures where a limited form of federation has been implemented. However, service mesh-based systems will still need egress/ingress points to interact with larger, distributed environments.

In all of these cases, a general, standards-based approach to managing security boundaries and trust domains—federations—wherein the discovery and access to services for both data access and analytics can be managed, offers great benefit. To this end, the IEEE 2302-2021 is a modular, proxy approach that allows a wide variety of deployment and governance models, whereby a broad range of application domains can be supported.

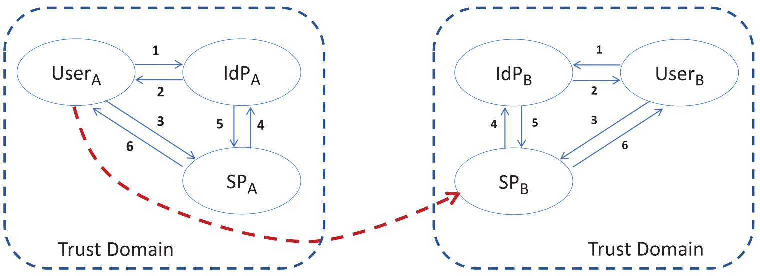


FIGURE 2. Federation in a nutshell.

BASIC FEDERATION CONCEPTS

Figure 2 illustrates the federation challenge in a nutshell. Each trust domain has an Identity Provider (IdP) that issues credentials to Users. Those Users present their credentials to a Service Provider (SP) when requesting service. The SP validates the credentials with the IdP before rendering service.

How, then, can a UserA discover and invoke a service SPB in a different trust domain? How can SPB manage its discoverability in other trust domains? Once discovered and invoked, how can SPB validate UserA's credentials and make a proper access decision? These are the requirements federation addresses.

Figure 3 illustrates the conceptual federation

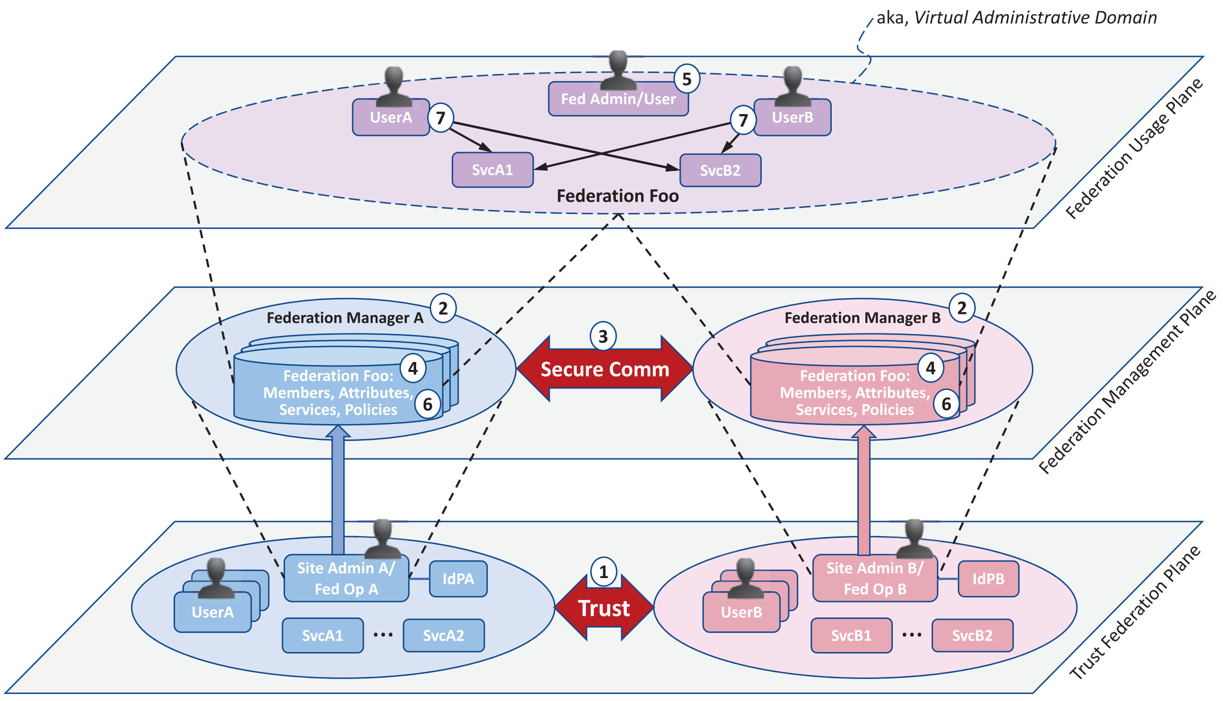


FIGURE 3. The NIST Cloud Federation Reference Architecture. (Source: NIST SP 500-332.)

approach defined in the NIST Cloud Federation Reference Architecture (CFRA) in three planes. In the bottom *Trust Federation Plane,* two or more trust domains exist. To achieve common goals, these domains decide to collaborate. To do so, they must establish *trust (1).* The exact criteria of trust depend on the nature of the common goals, how the collaboration will be operated, and the participant's risk tolerance. Once this is done, each site can deploy a *Federation Manager (2)* in the conceptual *Federation Management Plane*. Based on the trust among the participants, these Federation Managers establish *Secure Communication (3).* This communication enables the Federation Managers to maintain consistent state for *federation instances*—illustrated here as *Federation Foo (4).* Federation Foo can be said to exist in the *Federation Usage Plane*. Every federation instance has at least one member that is the *Federation Administrator,* or *Fed Admin (5)*, who is a user from one of the original trust domains. This Fed Admin can grant Foo membership and authorizations to other users. Members that are ServiceOwners can register service endpoints and define the discovery and access policies. All of this information is consistently maintained in the Federation Managers *(6).* Finally, Foo members can discover and invoke services, based on their authorizations in Foo *(7)*.

From a conceptual perspective, each federation instance can be considered a *Virtual Administrative Domain*. It is an administrative domain, much like any other. It has an administrator and policies that define its "rules of the road". It is virtual in that it does not "belong" to any one physical organization. The term *Distributed Trust Domain* can also be used. It is a trust domain where identity credentials are trusted and access decisions can be made, but it is distributed across the participant's infrastructures.

All of this is to give the user—a federation member—a consistent view and ability to interact with federation resources, as illustrated in Figure 4. After authentication to a federation, the user can use services and access data based on their authorizations, regardless of who owns the resource. Resource owners, however, retain unilateral control over the discovery and access policies for their resources.

Managing security perimeters and trust boundaries is a fundamental concept and the essence of the FHS model. Likewise, Zero Trust Architectures (ZTAs)[15] are based on a set of design tenets to define and to manage trust boundaries. These design tenets map neatly to federations such that the notion of *Zero Trust Federations* is a natural extension.[16]

THE FHS MODEL AND API

As a reference architecture, the NIST CFRA is inherently conceptual. Its purpose was to thoroughly understand and organize the roles and entities in the federation design space. The IEEE 2302-2021 SIIF casts the conceptual Federation Manager model into a concrete *Federation Hosting Service (FHS) model* and its associated *APIs*, as illustrated in Figure 5.

Each FHS has three APIs:

- **The FHS Operator API:** A FHS has a *FHS Operator,* in much the same way a web server has an operator. The FHS Operator starts the FHS and ensures that it stays up and running The FHS Operator configures the FHS for peering with other FHSs. As a privileged operation, the FHS Operator grants *FedAdmin* membership to specific users.
- **The FHS Member API:** All members interact with the FHS through the Member API. There are three, pre-defined member roles associated with three attributes: *FedAdmin*, *ServiceOwner*, and simply *Member*. A FedAdmin can create new federation instances, grant membership in that instance, define additional attributes, grant them to members, and connect with external monitoring tools. ServiceOwners can register service endpoints and define their discovery and access policies based on the defined federation attributes. While the general Member attribute is pre-defined, the FedAdmin can define any number of additional attributes whereby the roles and authorizations of any member can be properly managed. All members can query the FHS for the services available in their instance, based on the service discovery policies. Once discovered, the services can be invoked, based on the access policies.
- **The FHS-FHS API:** To provide a consistent view of a federation instance, the FHSs must communicate through this API. When services are registered or attributes defined, this information

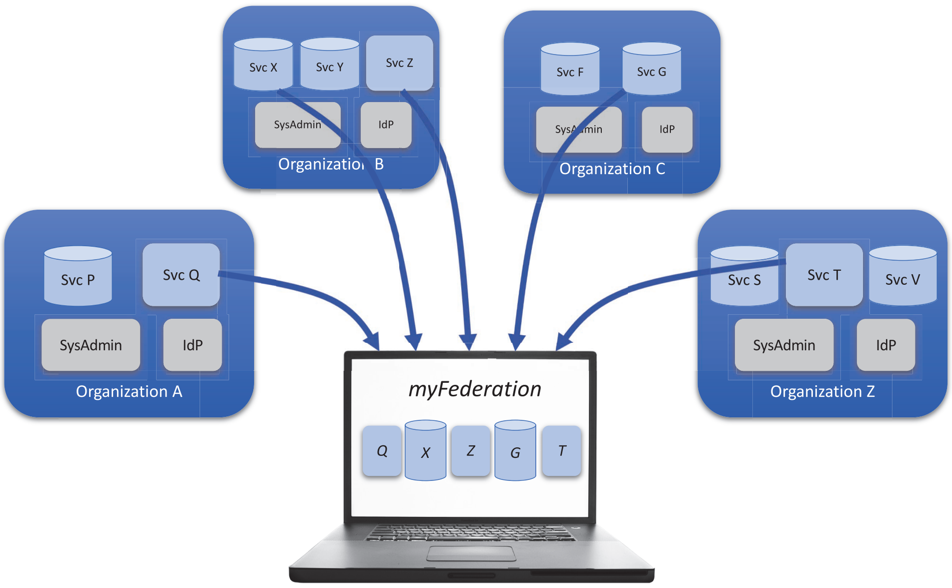


FIGURE 4. The user's perspective.

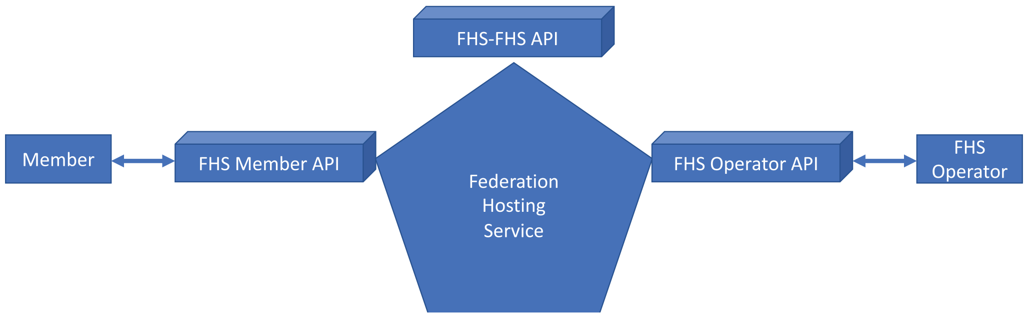


FIGURE 5. Federation hosting service reference model.

is propagated. While the SIIF defines this API, it does not define the topology of this communication. Monitored data can also be collocated on a per-federation basis, which is supported through this API.

The FHS design is essentially a set of peering *API Gateways* that provides a distributed trust domain. An FHS proxies service calls in much the same way as an API Gateway. Likewise, the endpoints made available to members are encoded versions of the endpoints registered. As discussed in the CFRA, this approach allows a wide variety of deployment and governance models to be used. Entire federations could be hosted on a single FHS that everybody trusts. Alternatively, each participating organization could deploy and configure their own FHS, which serves as their ingress point to larger federated environments. By way of analogy, a set of FHSs can be considered the "railroad tracks" on which any number of federation instances can run.

The API defined in the SIIF only covers the most fundamental "core" operations to provide a viable federation capability. It relies on many operations being manually managed by the FHS Operator or the FedAdmins. As federations become more large-scale, involve money changing hands, or require regulatory compliance, the API will need to be expanded. In the SIIF, this was identified as four *Federation Capability Levels*:

- **Level 1 (The Core):** The core API supports the basic operations described above.
- **Level 2:** This includes direct support for accounting, billing, and auditing.
- **Level 3:** This adds support for legal agreements and regulatory compliance.
- **Level 4:** Finally, this supports automation across the federation lifecycle.

While these levels are used to form a neat hierarchy, they actually involve a related set of underlying capabilities that need to be built out. This constitutes an evolutionary path not only for IEEE 2302 but for other federation systems, as well.

BUILDING OUT THE CAPABILITIES

The core API has standardized on a flexible, proxy-based design. The next step to FCL 2 is simply *accounting* and *billing*. The core API supports the acquisition and collation of basic monitoring data on a per-federation basis. However, additional "back-end" tools will have to be built that can keep track of who's doing what, resource usage, and so on. Such accounting data will have to be evaluated using a *cost structure* whereby bills can be periodically presented to consumers and timely payments can be made.

All federations are predicated on the establishment of trust. For simple deployments, it is certainly possible to establish trust "out of band." That is to say, FHS operators that trust each other can manually configure their servers to establish a secure, communication channel. However, for larger scale deployments at FCL3 and FCL4, trust management will have to be more formalized and automated, where possible.

One existing, relevant tool is the OIDC Federation Specification.[17] This standard defines a protocol whereby an OIDC installation can traverse a trust chain presented by another OIDC installation with the goal of identifying a *common trust anchor*. If a common trust anchor is found, trust is said to be established. At this point, the OIDC installations exchange JSON Web Key Sets after which all communication can be cryptographically signed and encrypted. While this standard is very useful, it does not address the issue of what criteria defines trust, or what aspects of trust a trust anchor may denote.

To define semantically what "trust" means, specific sets of trust criteria can be managed in a Trust Framework.[18] Here, the satisfaction of any given criteria is denoted by a cryptographically signed document, called a Trustmark, that is issued by a Trustmark provider. Trustmark recipients can store these documents in a Trust Interop Profile. To establish trust, different Trustmark subsets can be presented to a Trustmark Relying Party. After the Trustmarks have been validated, those trust criteria have been established.

Trustmarks could be used to directly support *legal agreements* and *regulatory compliance* at FCL3. A *Trustmark* could be used to denote the terms of a legal contract and that it has been signed. Trustmarks could likewise be used to denote regulatory compliance. While well-defined resource discovery and access policies are an integral part of the FHS model, observing large-scale, real-world policies will require that (1) all federation identities and attributes carry the necessary semantics, and (2) the policies necessary to enforce regulatory requirements are properly implemented and deployed.

In general, it may be possible that different regulatory policies may be conflicting or ambiguous. Such situations are, however, outside of the scope of the federation infrastructure and must be resolved by the regulatory agencies involved.

While things like the OIDC Federation Spec and trustmarks help automate trust, further automation can be done at FCL 4. From a technical perspective, it is entirely possible that a commercial cloud provider could operate fleets of on-demand FHSs, in much the same way that Content Distribution Networks are provided on-demand now. Federations could be instantiated on-demand through a set of web menus, or under program control. These federations could have a global footprint by transparently using the cloud provider's backbone network among their data centers.

At any capability level, the federation user experience cannot be ignored. Widespread adoption will take place when end users are presented with an intuitive, easy-to-use interface. Several options are possible for federations. A user could authenticate to a federated environment through a *web portal*. In this case, a web server sits in front of an FHS proxy. The widgets populated on a web page are determined by a user's authorizations with a specific federation instance. Another possibility is a *virtual desktop*. Here a remote desktop server sits in front of the proxy. Similarly, when the user authenticates, the icons and browsable files that are populated on the desktop are determined by the user's authorizations. Yet another possibility is a *Jupyter Notebook*. Again, when the user authenticates, the data sets, analysis tools, documents, and so on, that are accessible in the notebook are determined by the user's authorizations.

CONCLUSIONS

The IEEE 2302-2021 model, API, and capability levels were derived to standardize key architectural concepts that are the basis for flexibly implementing a wide range of federated environments. A number of application domains have been identified, along with additional capabilities that must be built-out to achieve real-world federations, at scale.

By providing a standardized method for supporting inherently distributed collaborations, a strategic intent of IEEE 2302-2021 is to jump-start a new technology market segment around federation tooling and provisioning. This will enable vendors to build interoperable tools with standardized interfaces. Cloud providers will be able to become on-demand *federation providers*. The ultimate goal is to establish an economically self-sustaining marketplace for federation capabilities.

ACKNOWLEDGMENTS

Certain commercial entities, equipment, or materials may be identified in this article in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

REFERENCES

1. IEEE, Standard on Intercloud Interoperability and Federation (SIIF), IEEE 2302-2021, December 2021, https://standards.ieee.org/ieee/2302/7056.
2. D. Shands, R. Yee, J. Jacobs and E. J. Sebes, "Secure Virtual Enclaves: supporting coalition use of distributed application technologies," *Proceedings DARPA Information Survivability Conference and Exposition*. DISCEX'00, vol. 1, 2000, pp. 335-350, doi: 10.1109/DISCEX.2000.825037.
3. I. Foster, "The anatomy of the grid: enabling scalable virtual organizations," *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2001, pp. 6-7, doi: 10.1109/CCGRID.2001.923162.
4. J. Cummings et al., "Beyond Being There: A Blueprint for Advancing the Design, Development, and Evaluation of Virtual Organizations", *NSF Workshop Report*, 2008.
5. L. Badger et al., NIST US Government Cloud Computing Technology Roadmap, Volume I: High Priority Requirements to Further USG Agency Cloud Computing Adoption, *SP* 500-293, 2014, http://dx.doi.org/10.6028/NIST.SP.500-293.
6. F. Liu, et al., The NIST Cloud Computing Reference Architecture, *NIST Special Publication* 500-292, September, 2011.
7. C. Lee, R. Bohn, M. Michel: The NIST Cloud Federation Reference Architecture, *NIST* SP 500-332, 2020, https://doi.org/10.6028/NIST.SP.500-332.
8. National Science and Technology Council, National Strategic Computing Reserve: A Blueprint, https://www.nitrd.gov/pubs/National-Strategic-Computing-Reserve-Blueprint-Oct2021.pdf
9. Confidential Computing Consortium: Confidential Computing: Hardware-Based Trusted Execution for Applications and Data. https://confidentialcomputing.io/white-papers July, 2020.
10. InCommon, https://incommon.org.
11. eduGAIN, https://edugain.org.
12. The Dataverse Project, https://dataverse.org.
13. The OpenStack Foundation, OpenStack Keystone v3, https://docs.openstack.org/keystone/latest.
14. Cloud Native Computing Foundation, Solving the Bottom Turtle, https://spiffe.io/book.
15. Rose, et al., Zero Trust Architecture, *NIST* SP 800-207, Draft 2, 2020, https://csrc.nist.gov/publications/detail/sp/800-207/final.
16. Lee, C.A., Federated Security, OGC 20-027, http://docs.opengeospatial.org/per/20-027.html, January, 2021.
17. R. Hedberg, ed. *OpenID Connect Federation* 1.0 - draft 12, https://openid.net/specs/openid- connect-federation-1_0.html, 2020.
18. M. Moyer et al., Enabling Scalable Multidimensional Trust in Heterogeneous Distributed Systems with Machine-Readable Trustmarks, 2016, https://trustmark.gtri.gatech.edu/wp-content/uploads/2017/09/trustmark-white-paper.pdf.

PROCEEDINGS ARTICLE

This article originally appeared in the 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing

Scission: Performance-driven and Context-aware Cloud-Edge Distribution of Deep Neural Networks

Luke Lockhart and Blesson Varghese, *Queen's University Belfast, UK, {llockhart04, b.varghese}@qub.ac.uk*

Paul Harvey and Pierre Imai, *Rakuten Mobile, Japan, {paul.harvey, pierre.imai}@rakuten.com*

Peter Willis, *British Telecommunications plc, UK, peter.j.willis@bt.com*

Partitioning and distributing deep neural networks (DNNs) across end-devices, edge resources and the cloud has a potential twofold advantage: preserving privacy of the input data, and reducing the ingress bandwidth demand beyond the edge. However, for a given DNN, identifying the optimal partition configuration for distributing the DNN that maximizes performance is a significant challenge. This is because the combination of potential target hardware resources that maximizes performance and the sequence of layers of the DNN that should be distributed across the target resources needs to be determined, while accounting for user-defined objectives/constraints for partitioning. This paper presents `Scission`*, a tool for automated benchmarking of DNNs on a given set of target device, edge and cloud resources for determining optimal partitions that maximize DNN performance. The decision-making approach is context-aware by capitalizing on hardware capabilities of the target resources, their locality, the characteristics of DNN layers, and the network condition. Experimental studies are carried out on 18 DNNs. The decisions made by* `Scission` *cannot be manually made by a human given the complexity and the number of dimensions affecting the search space. The benchmarking overheads of* `Scission` *allow for responding to operational changes periodically rather than in real-time.* `Scission` *is available for public download*[*]

Keywords- edge computing; deep neural network; DNN partitioning;

I. INTRODUCTION

Deep Neural Networks (DNNs) are integral to image, video or speech recognition applications[1,2]. A DNN is a sequence of multiple layer types, such as convolution, activation or pooling, that have varying computational requirements. The output size of each layer depends on the layer type and configuration.

Recently, distributed execution of the DNN across the cloud and resources at the edge of the network, within the edge computing paradigm[3,4,5], has been found to be beneficial[6,7]. The advantages offered by using the edge are privacy preservation, reduced ingress bandwidth demand, and reduced inference times. A distributed execution approach could either execute the entire DNN on the edge if there are sufficient compute resources available or act as a pre-filter (partially processed) for the input data before it is sent in the WAN to the cloud. The edge can[*] also be an aggregation point in use-cases, for example, a network of drones or cameras that are linked to an edge resource.

978-0-7381-2394-3/20/$31.00 ©2020 IEEE
DOI 10.1109/UCC48980.2020.00044

* https://github.com/qub-blesson/Scission

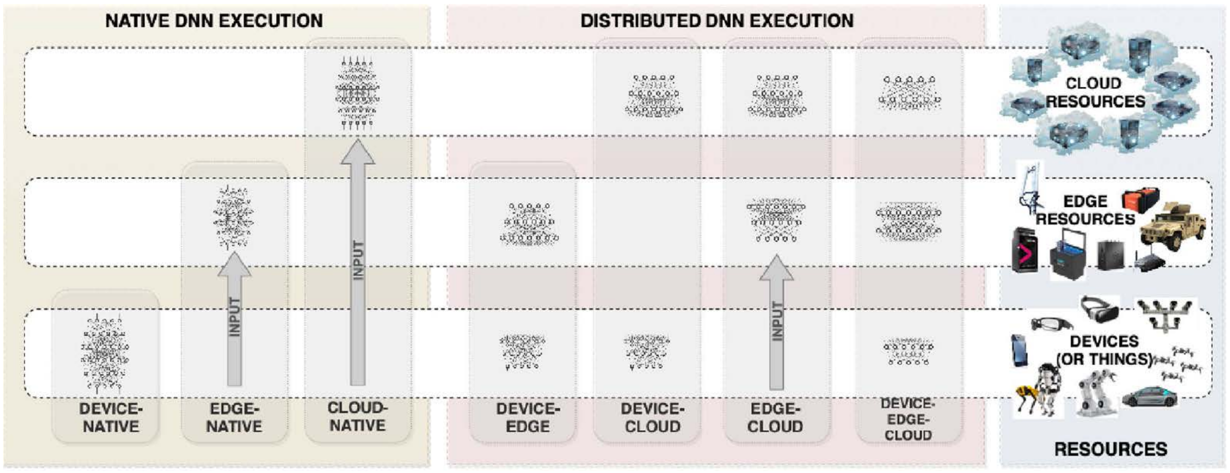


FIGURE 1. Native and distributed DNN execution options on a three-tier resource pipeline (devices may also be called 'things').

It has also been demonstrated that for data streams, the frame drop rate can be reduced at the edge when compared to the cloud[8]. Additionally, resources at the edge may be powered by the main lines and may have relatively more compute capabilities than the end device, thereby providing opportunities for executing large DNNs while being sufficiently accurate.

Leveraging the edge provides numerous possibilities for distributing a DNN in addition to those when only using the device and the cloud[6,7]. These possibilities as shown in Figure 1 are: (i) edge-native execution of the DNN, (ii) distributed execution across the edge and the device, or (iii) distributed execution across the cloud, edge and device. In edge-native execution, all the layers of the DNN will run on the edge and in distributed execution, a specific sequence of layers will run on each resource. However, *for any given DNN, identifying the execution approach that maximizes its performance* is not a trivial challenge. This is because the following three associated questions need to be addressed:

Q1. *Which combination of potential target hardware resources maximizes performance?* This question requires the identification of whether native or distributed execution approaches are best suited for a given DNN on a set of resources, comprising the device, edge, and cloud. Also, if there are multiple device, edge or cloud choices, which target resource(s) should be selected for deploying the DNN.

Q2. *Which sequence of layers should be distributed across the target resource(s) for maximizing DNN performance?* DNNs can have a large number of layers with varying computational requirements and output sizes. For distributed execution, the layers at which a DNN is partitioned for optimal performance needs to be identified. This cannot be done manually because there are DNNs that could have a large number of layers. For example, DNNs such as NASNetLarge has 1041 layers and InceptionResNetv2 has 782 layers. In addition, DNNs cannot be partitioned at all layers (will be discussed in Section II). An ad hoc distribution of a DNN that arbitrarily selects the sequence of layers would result in under-performing DNNs.

Q3. *How can the performance of DNNs be optimized given user-defined objectives or constraints?* Although addressing Q1 and Q2 will provide an ideal partition of a DNN for a given set of hardware resources, they may not be optimal when user-defined objectives or constraints are taken into account. For example, although a cloud native execution approach may be ideal for maximizing the performance of a DNN, an application owner may want to run a specific sequence of layers on the edge for enhancing data privacy or reducing the volume of output data sent to the cloud. If an edge resource has to undergo maintenance, then an administrator may require the DNN to be redistributed across the cloud and the device, which would need a different partition configuration.

To address the above challenge and associated questions, this paper makes the following *research contributions*:

1. Proposes Scission, a tool for automated benchmarking of DNNs on a given set of target device, edge and cloud resources for determining the optimal partition for maximizing DNN performance.
2. Develops the underpinning benchmarking approach of Scission that collects benchmark data by executing the DNNs on all target resources and subsequently identifies whether a native or distributed execution approach is most suited for the DNN (addresses Q1). For distributed execution, it identifies the optimal resource pipeline and partitions measured by the lowest end-to-end latency (compute time on resources and the communication time between resources) of the DNN by: (a) pairing the most computationally intensive layers with capable resources to minimize compute latencies, and at the same time (b) selecting layers with the least amount of output data as potential end layers of a partition to minimize communication latencies (addresses Q2). Thus the decision-making approach in Scission is context-aware by capitalizing on the hardware capabilities of the target resources, their locality, the characteristics of DNN layers, and network condition.
3. Provides a querying engine that has less than a 50 millisecond overhead to ensure that user-define constraints or objectives can be taken into account for determining optimal partitions that maximize the performance of distributed DNNs (addresses Q3).
4. An experimental study to demonstrate that Scission can facilitate: (a) DNN partitioning under different network conditions, (b) DNN partitioning under different input data sizes, (c) DNN partitioning under user-defined constraints, (d) DNN partitioning for comparing different target resource pipelines, and (e) the identification of the top *N* DNN partitions that maximize performance. It is observed that ideal DNN partitioning needs to be context and data-driven and it is impossible to determine optimal partitions manually. Scission achieves these and is a valuable tool for deploying context-aware and distributed DNNs in an cloud-edge environment.

The remainder of this paper is organized as follows. Section II provides a background to the DNN models considered in this paper and presents an overview of the underpinning methodology for benchmarking, decision-making and querying of Scission. Section III presents the results obtained from an experimental study on Scission. Section IV presents related work. Section V concludes this paper by presenting avenues for future research.

II. SCISSION

This section firstly provides a background to DNNs and the types of DNNs that are considered in this paper, which is followed by observations that led to the development of Scission. The architecture, the underlying benchmarking approach, the context-aware decision-making process, and finally the querying capability of Scission are then presented

A. Background

A DNN is a sequence of layers and is a general term that covers all neural networks with multiple hidden layers (that is multiple layers between the input and output layers)[1,2]. A DNN may consist of different layers and the most common types are as follows: 1) Fully-connected layers connect every neuron to all neurons in the previous layer with the aim of preforming high-level reasoning. 2) Convolution layers convolve the input to produce feature maps of inputs with the aim of learning features. 3) Pooling layers apply a pre-defined function (maximum or average) to down sample the input. 4) Activation layers apply non-linear functions and the most commonly used is the rectified linear unit (ReLu). 5) A Softmax layer is generally used for classification with the aim of generating a probability distribution over the possible classes.

In this paper, 18 DNNs as shown in Table I are considered. The table presents the size of a trained model and its corresponding weights, the total number of layers in the DNN (including input and output layers), the number of valid points for partitioning, and the type of the DNN. These models are explored in the context of Keras†, an open source neural network library that runs on TensorFlow‡. These models are trained on the ImageNet database[18].

Two categories of DNNs are considered, namely linear and branching. In a linear DNN, the neural network is sequential - the input of one layer is connected to the next. This results in a singular path between the first and last layers as seen in Figure 2a. Figure 3 shows the execution time and the output data size of the 23 different layers of VGG16, an example linear model (executed on the 'Cloud' resource shown in Table II). It is noted that the layers have varying execution time and the output sizes of the layers vary.

If a linear DNN that has N layers needs to be distributed across two resources, then a partitioning approach would need to create two partitions of the DNN. The first partition would consist of a sequence of the first x layers and the second partition would consist of $N - x$ layers. The output of the xth layer would need to be provided as an input layer for the second partition. DNNs naturally lend themselves to distributed execution as their segmented structure provides rational points to partition. There are $N - 2$ potential partitioning points (rather than $N - 1$) since a partition configuration in which the first partition comprises only the first layer, results in the second partition containing a duplicated input layer. Figure 2a provides an example of a linear DNN that is distributed across a resource pipeline consisting of the device, edge, and cloud. The red connectors show the valid partitioning points in the linear model.

On the other hand, in a branching DNN, a layer may be connected to more than two layers which results in parallel paths between the first and last layers. Partitioning a model in a parallel region can lead to synchronization issues and may add additional communication overhead as multiple metadata outputs will need to be transferred from one resource to another[8]. Therefore, layers within a branch are grouped together as a block of layers and treated as a single entity. This reduces the number of partitioning points (for example, the ResNet50 DNN has 177 layers, but only 23 valid partition points as shown in Table I). Figure 4 shows the execution time and the output data size of the 25 different entities (layers and blocks are identified by the layer numbers) of ResNet50, an example

TABLE 1. Pre-trained DNN models from Keras used in this paper; Type: L - linear, B - branchings.

DNN Model	Size(MB)	Layers	Partition points	Type
Xception [9]	88	134	13	B
VGG16 [10]	528	23	21	L
VGG19 [10]	549	26	24	L
ResNet50 [11]	98	177	23	B
ResNet101 [11]	171	347	40	B
ResNet152 [11]	232	517	57	B
ResNet50V2 [11]	98	192	15	B
ResNet101V2 [11]	171	379	15	B
ResNet152V2 [11]	232	556	15	B
InceptionV3 [12]	92	313	18	B
InceptionResNetV2 [13]	215	782	60	B
MobileNet [14]	16	93	91	L
MobileNetV2 [15]	14	157	65	B
DenseNet121 [16]	33	429	21	B
DenseNet169 [16]	57	597	21	B
DenseNet201 [16]	80	709	21	B
NASNetMobile [17]	23	771	4	B
NASNetLarge [17]	343	1041	4	B

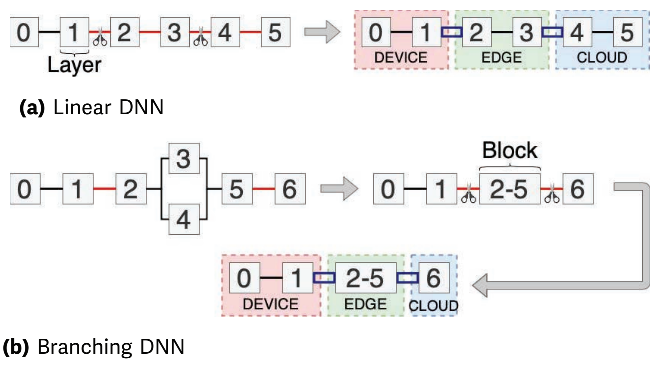


FIGURE 2. An example of partitioning a linear and branching DNN across the entire resource pipeline comprising device, edge and cloud. An example of a block is shown in Figure 2b (Layers 2-5). Red connectors between layers/blocks are valid partitioning points. Blue connectors are inter-resource communication.

† https://keras.io
‡ https://www.tensorflow.org/

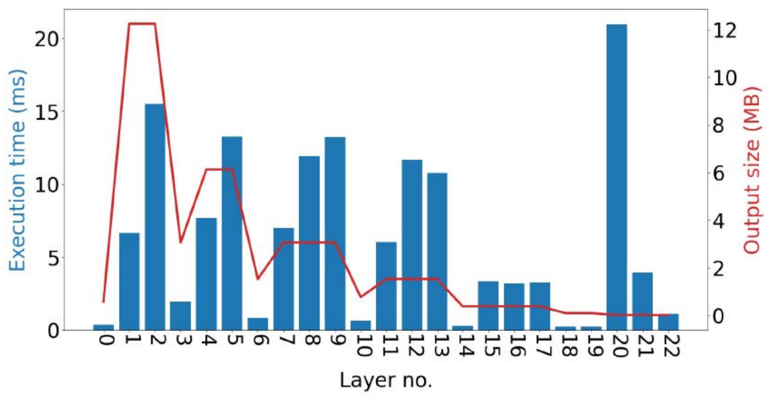


FIGURE 3. Average execution time (of five runs) and output data size of each layer of VGG16 on a 'Cloud' resource (refer Table II).

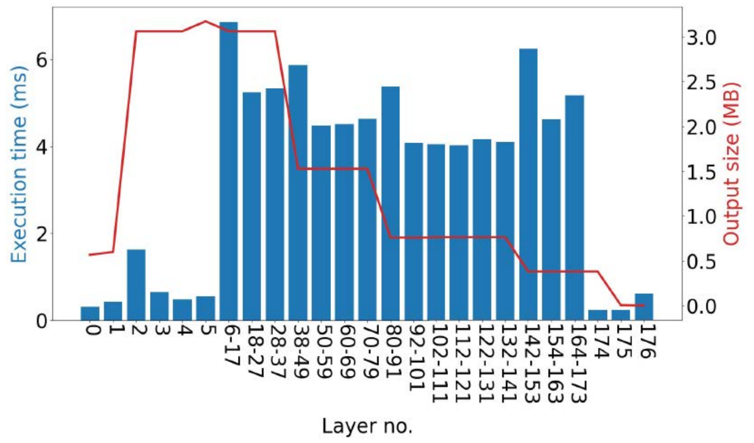


FIGURE 4. Average execution time (of five runs) and output data size of each layer of ResNet50 on a 'Cloud' resource (refer Table II).

branching model. Again the variable execution time and output data size of layers/blocks are noted. Figure 2b shows an example of a branching DNN that is distributed across the device, edge and cloud. Layers 2-5 are considered as a single block.

The above highlights that DNNs may have a large number of layers and may take the form of linear or branching models. The execution time of individual layers and the output size vary for each layer. If a DNN needs to be distributed across multiple resources, it would be impossible to manually determine efficient partition configurations. This is due to the potentially complex structure of a DNN and a large search space arising from the combination of partitioning points, target hardware resources, and optimization criteria. Therefore, an automated approach for DNN partitioning is required.

B. Motivation

`Scisssion` proposed in this paper is designed on the following six practical observations to make it widely applicable for maximizing the performance of DNNs:

(i) *DNN partitioning must account for multiple resource tiers in cloud-edge continuum.* Many options for distributing large DNNs become available as more resource tiers between the cloud-edge continuum become accessible for computing. The approach for identifying optimal partitions of DNNs should scale across resource tiers. This paper considers the device, edge and cloud tiers.

(ii) *DNN partitioning must be based on empirical data obtained from the underlying hardware rather than based on estimates.* A large body of existing research estimates optimal partitions by relying on predicted performance on a given resource by making assumptions of the target hardware platform. However, modern hardware is known to have complex processor and memory architectures that sometimes results in a non-linear relationship between performance and the amount of resource[19]. Hence, empirical data based partitioning will be more reliable than alternate approaches.

(iii) *DNN partitioning must be able to identify a set of performance efficient partitions.* This is important because the most efficient DNN partitions may only have a negligible improvement over the other partitions, which may be more practical due to organizational or geo-political reasons.

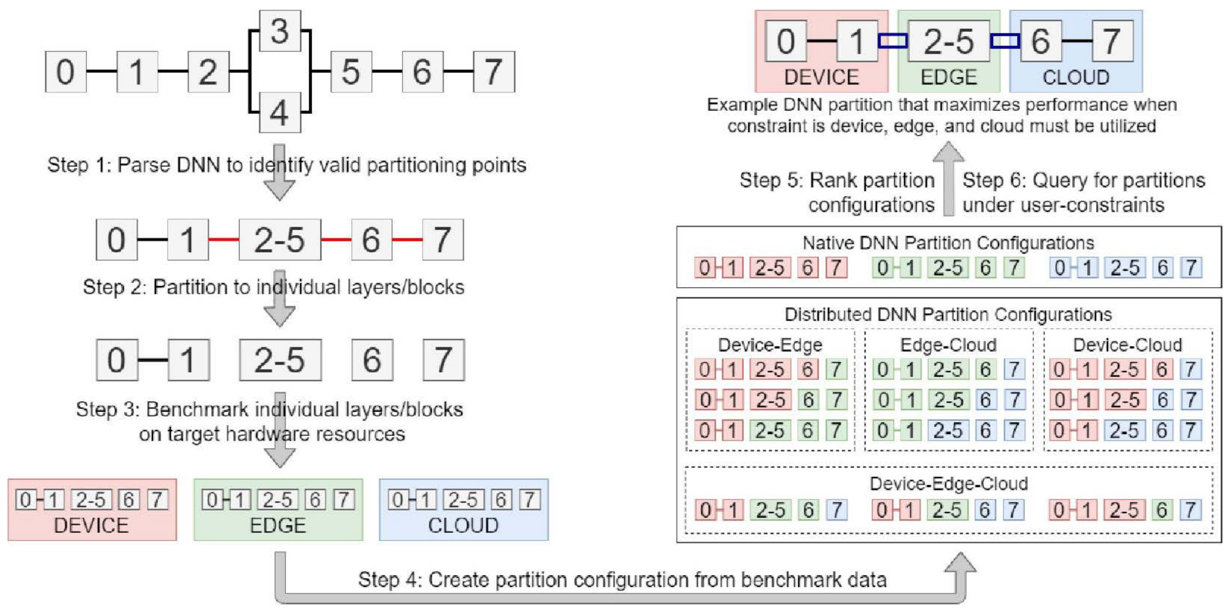

Step 1: Parse DNN to identify valid partitioning points
Step 2: Partition to individual layers/blocks
Step 3: Benchmark individual layers/blocks on target hardware resources
DEVICE
EDGE
CLOUD
Step 4: Create partition configuration from benchmark data
Native DNN Partition Configurations
Distributed DNN Partition Configurations
Device-Edge
Edge-Cloud
Device-Cloud
Device-Edge-Cloud
Step 5: Rank partition configurations
Step 6: Query for partitions under user-constraints
Example DNN partition that maximizes performance when constraint is device, edge, and cloud must be utilized

FIGURE 5. The underlying six-step benchmarking and partitioning methodology of `Scission`.

(iv) *DNN partitioning needs to be context-aware across multiple dimensions.* Identifying performance efficient partitions is not only dependent on DNN layer characteristics and output data. Performance is affected by hardware capabilities of the target platform, resource locality, load and failures, and network condition between resources. These dimensions need to be taken into account.

(v) *DNN partitioning must account for user-defined objectives or constraints.* A performance efficient partition obtained by optimizing against the dimensions described above may not always be ideal. A human must be able to specify constraints as input to the partitioning process. For example, an application administrator may want a particular sequence of layers to be executed on an end device for retaining intermediate data of a few layers on the device although it affects the overall end-to-end latency.

(vi) *Practical DNN partitioning must be rapid.* Variations in network conditions and changes to resource workloads may affect the optimal partition points of a DNN. For example, the available bandwidth to a drone may increase as it navigates away from a low coverage area. This may result in the DNN to be partitioned from device-native (which may be less energy efficient) to be distributed across the device-edge-cloud. This repartitioning needs to occur with low overheads to be advantageous in real-world use (the worst case prediction for `Scission` only takes 0.05 seconds on the cloud).

C. Partition Methodology

The six step methodology for automated partitioning adopted by `Scission` is shown in Figure 5 and described below:

Step 1: Parse the DNN to find valid partitioning points. As presented in Section II-A the DNN is parsed to identify valid partitioning points. For a linear DNN this is straightforward, where as for a branching DNN, the parallel paths need to be identified. Layers within the parallel path are considered as a single entity, referred to as block. As shown in Figure 5 the red connectors show the valid partitioning points.

Step 2: Partition into individual layers/blocks. This step ensures that the DNN is partitioned into distinct sub-models with individual layers or blocks for the purposes of benchmarking. It should be noted that each sub models requires an input layer to facilitate the processing of the output from the previous layer.

Step 3: Benchmark each layer/block on target hardware resources. In this step, given a set of target hardware resources, such as the device, edge, or cloud, each layer/block is benchmarked five times. The average execution time and the output data size is recorded. The 18 DNNs shown in Table I are considered in this paper.

Step 4: Create partition configurations from benchmark data. The benchmark data comprises the average execution time of each layer/block. The communication overhead to transfer output data across different resources is calculated from user-provided data, such as the average bandwidth available. This data is used to exhaustively develop partition configurations such that the end-to-end latency (compute and communication overheads) of all combinations of layers/blocks paired to different resources are known.

Two types of partition configurations are considered by `Scission`, namely native and distributed as shown in Figure 1. Native partition configurations are those in which all layers/blocks execute on a single resource (for example, device-native, edge-native, or cloud-native). Distributed partition configuration are those in which the DNN collaborates across multiple resources by executing the layers/blocks on multiple resources (for example, distributed execution across device-edge, device-cloud, and device-edge-cloud).

Step 5: Rank partition configurations. Once all partition configurations have been generated, they are ranked. The ranking may be generated by optimizing against end-to-end latency (additional objectives, such as minimum data transfer across resources, or a combination of these can be provided in Step 6). The Top *N* partition configurations are presented to the user.

Step 6: Query `Scission` for partition configurations given user-defined constraints. `Scission` interacts with the user by not only providing the default rankings produced in Step 5, but also accepting user-defined constraints provided as queries. The example shown in Figure 5 is the result of executing the query for the fastest DNN partition configuration that collaborates between all (device, edge and cloud) resources. Queries are not limited to only minimizing execution latency or bandwidth. For example, they may be constructed to:

- Apply bandwidth constraints (for example, the edge resource must not transfer more than 1MB to the cloud).
- Apply execution time constraints (for example, the execution time on the device must not exceed 1 second, or 30% of the overall execution time must be on the edge).
- Include or exclude resources (for example, distribution must not include the cloud, or execution must be edgenative).
- Specify layer/block execution locations (for example, Layer 7 must execute on the edge).

The Top *N* partition configurations are presented to the user. More complex queries can be provided to `Scission`. Examples include: (i) Find the partition configuration that results in the lowest execution latency, but the device and edge must not transfer more than 1MB. (ii) Find partition configuration that has the lowest inter-resource data transfer, but *n* layers are executed on the edge. (iii) Find partition configuration with lowest end-to-end latency and does not use the cloud and at least half of the layers/blocks must be executed on the device.

III. EXPERIMENTAL STUDIES

This section presents the experimental test bed and software set up and is followed by the results obtained from `Scission`.

A. Setup

Experiments are carried out on hardware resources shown in Table II to reflect a range of resources typically used. Two edge resources are employed with different hardware characteristics. Two cloud resources are used with and without a GPU.

To emulate real world network performance, `Scission` uses the average network latency and bandwidth for: (i) 3G (1.6 Mbps upload and 67ms network latency)[§], (ii) 4G (12.4Mbps upload and 55ms network latency)[§], and (iii) wired home fibre broadband (20Mbps upload and 20ms network latency)[¶]. A network latency of 25ms and a bandwidth of 50Mbps is assumed for all edge-cloud connections. Results reported are average of five experimental runs.

§ https://bit.ly/3hrGk4N
¶ https://bit.ly/2EfHjqr

TABLE II. Specification of the target hardware resources used.

Resource	CPU Arch.	CPU freq. (GHz)	CPU cores	RAM (GB)	GPU	OS
Device	ARMv8	1.5	4	4	N/A	Raspbian Buster
Edge (1)	AMD64	4.5	2	4	N/A	Ubuntu 18.04 LTS
Edge (2)	AMD64	3.7	4	8	N/A	Ubuntu 18.04 LTS
Cloud	AMD64	4.5	8	32	N/A	Ubuntu 18.04 LTS
Cloud	AMD64	4.5	8	32	Nvidia GTX 1070	Ubuntu 18.04 LTS

The `Scission` tool is implemented in Python and requires Tensorflow 2.0+ to be installed. Tensorflow is an end-to-end open source machine learning platform, which is used as the back end to run the pre-trained DNNs provided by Keras. NumPy is used for processing multi-dimensional arrays that are produced as layer outputs.

`Scission` makes two assumptions. Firstly, the communication overheads can be calculated as network latency + data size/bandwidth. The second assumption is that the total inference time of a model is the sum of the execution times of individual layers or blocks. This assumption has been validated in previous research [20,21]

B. Results

The experimental results obtained from `Scission` are exhaustive and discussing them entirely is outside the scope of this paper. However, the experiments and results to demonstrate the following five capabilities of `Scission` are considered in this paper: 1) DNN partitioning under different network conditions, 2) DNN partitioning under different input data sizes, 3) DNN partitioning under userdefined constraints, 4) DNN partitioning for comparing different target hardware resource pipelines, and 5) the top *N* DNN partitions. Sample results for executions on VGG19, ResNet50, MobileNetV2, InceptionV3 and DenseNet169 are presented. All experiments in this paper use a 150KB size input image unless otherwise stated.

The results from the above capabilities address Q1: 'Which combination of potential target hardware resources maximizes performance?' that was posed in Section I, but is specifically considered by the fourth capability. Similarly, all five capabilities will determine the best sequence of layers (or partition configuration) to address Q2: 'Which sequence of layers should be distributed across the target platform for maximizing the DNN performance?' The third capability specifically addresses Q3: 'How can the performance of DNNs be optimized given user-defined objectives or constraints?'

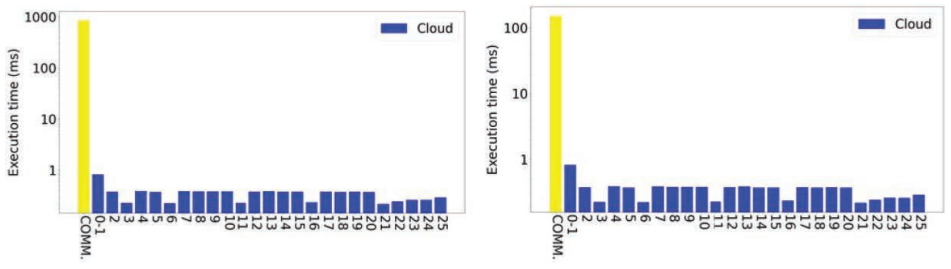
The time taken by the `Scission` partitioning methodology (overhead) is shown in Table III using an input image of size 150KB. The time on the cloud, edge and device is proportional to the number of layers in the DNN model. Gathering benchmark data for the DNNs on the device takes the most time as expected. However, if a dedicated device were to be utilized, then the DNN will only need to be benchmarked once in an offline manner. The edge resources can benchmark all DNNs (except NASNetLarge) in under half a minute. Given these overheads, the methodology cannot be used in a highly transient environment, but can be used to respond to operational changes periodically. The querying time on the benchmark data is just under 50 milliseconds if two resource (cloud-edge-device) pipelines are considered.

TABLE III. Overhead (in seconds) in benchmarking DNNs using the `Scission` partitioning methodology.

DNN Model	Cloud	Cloud	Edge (1)	Edge (2)	Device
Xception	2.95	2.11	7.07	6.78	36.10
VGG16	3.85	1.93	8.52	9.83	44.88
VGG19	2.98	1.92	8.43	11.86	51.82
ResNet50	3.12	1.92	6.65	5.27	29.91
ResNet101	6.03	5.19	12.42	9.55	57.67
ResNet152V2	9.37	7.86	18.41	14.17	82.11
ResNet50V2	3.31	2.77	6.59	5.27	33.30
ResNet101V2	5.97	4.96	11.43	9.28	52.27
ResNet152V2	8.85	7.49	17.27	14.23	77.99
InceptionV3	4.93	4.36	9.46	7.48	43.46
InceptionResNetV2	11.67	10.14	22.64	18.01	105.73
MobileNet	1.66	1.52	3.60	2.97	14.09
MobileNetV2	2.63	2.48	4.70	3.64	22.37
DenseNet121	5.88	5.45	10.67	8.22	48.40
DenseNet169	8.26	7.73	14.64	11.31	66.66
DenseNet201	9.94	9.24	17.78	14.09	82.34
NASNetMobile	10.40	9.95	17.42	13.02	77.45
NASNetLarge	18.69	13.99	38.14	35.25	172.27

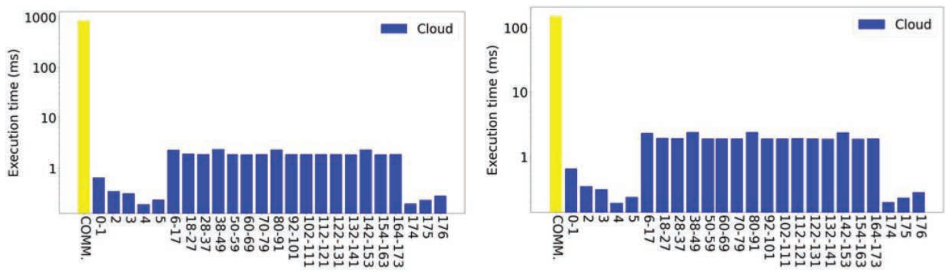
1. *DNN partitioning under different network conditions:* The results obtained from `Scission` highlight that DNN partitioning is affected by different network conditions (the optimal partitions for the same DNN may be different under different network conditions).

   Figure 6 and Figure 7 show that the lowest end-to-end latency execution of VGG19 and



(a) 3G **(b)** 4G

FIGURE 6. DNN partition of VGG19 with lowest end-to-end latency for different network conditions.



(a) 3G **(b)** 4G

FIGURE 7. DNN partition of ResNet50 with lowest end-to-end latency for different network conditions.

ResNet50, respectively, under 3G and 4G conditions would be obtained if the DNN is cloud-native. This is because the cloud resource in terms of its execution performance is much faster than the device and edge resource utilized in this experiment. The communication overhead of 800ms of sending the image from the device to the cloud does not offset the compute performance obtained on the resource.

However, Figure 8 demonstrates the end-to-end latency of MobileNetV2 (that has sub-second execution performance when it is device-native) under 3G and 4G conditions. In the 3G context, the DNN has the least inference time when the DNN is device-native. However, in the 4G context, given a lower latency network, the DNN is performance efficient when it is cloud-native. The above highlights the capability of `Scission` to identify optimal DNN partitions under different network conditions.

2. *DNN partitioning under different input data sizes:* If the input image size were increased from 150KB to 170KB, then for ResNet50 under 3G conditions, a device-native execution is determined by `Scission` to be performance efficient as shown in Figure 9. This is in contrast to a cloud-native execution that `Scission` identifies as performance efficient for a 150KB input image size (Figure 7a).
3. *DNN partitioning under user-defined constraints:* Figure 10 and Figure 11 are exemplars of performance efficient distributed execution of the DNN when the constraint imposed is that the entire resource pipeline must be employed. The results are shown for 3G and 4G network conditions for VGG19 and ResNet50. The difference in the optimal DNN partition is immediately evident. For example, the optimal partition configuration for VGG19 in a 3G network is: device executes Layers 0-23, edge executes Layer 24 and cloud executes Layer 25 (refer Figure 10a). However, in a 4G network, the optimal partition configuration is: device executes Layers 0-6, edge executes Layers 7-22, and cloud executes Layers 23-25 (refer Figure 10b).

4. *DNN partitioning for comparing different target hardware resource pipelines:* Two examples from InceptionV3 and DenseNet169 highlight that `Scission` can compare target hardware resource pipelines specified by a user for identifying which resource pipeline is performance efficient.

 Figure 12 considers the execution of InceptionV3 when the edge resource must be used in a pipeline and the device is connected to the edge resource via a wired connection for two different edge resources. Although the edge-native execution of InceptionV3 on Edge (1) and Edge (2) differed only by 0.07 seconds, the DNN partition configuration when the resource pipeline has Edge (1) and Edge (2) is different. The DNN partition is sensitive to the hardware capabilities of different resources in the pipeline. Since these are subtle, it would not be evident to a human, and therefore demonstrates the value of a tool, such as `Scission`.

 Figure 13 and Figure 14 considers the distributed execution of InceptionV3 and DenseNet169, respectively, for the entire resource pipeline (device, edge and cloud) when a device is connected to the edge through a wired connection for two different

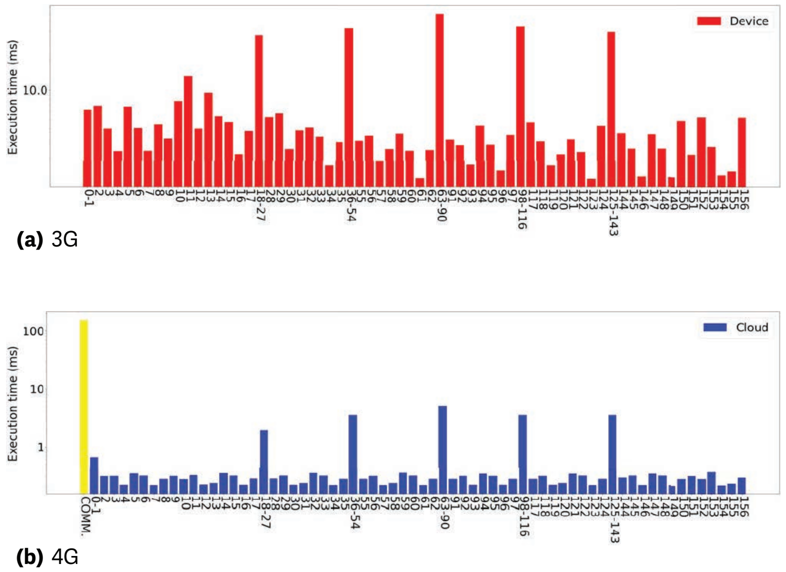


FIGURE 8. DNN partition of MobileNetV2 with lowest endto-end latency for different network conditions.

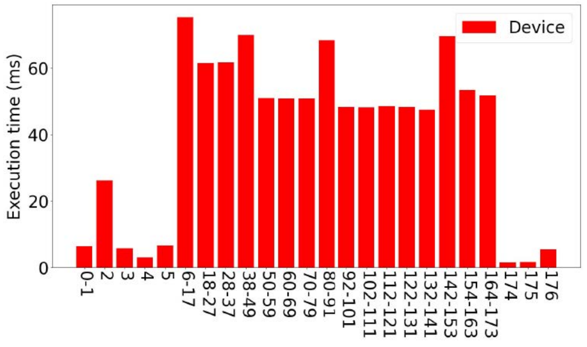


FIGURE 9. DNN partition of ResNet50 with lowest endtoend latency in a 3G network when input data size is 170KB (instead of 150KB).

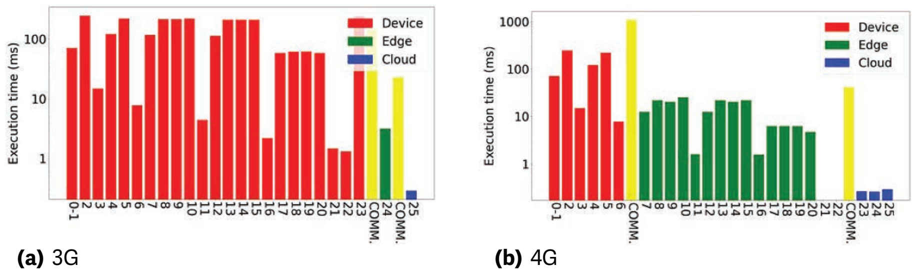
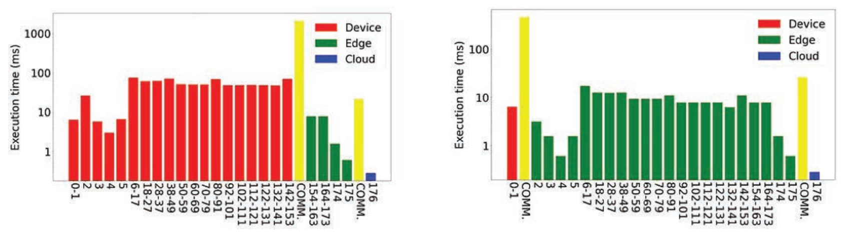
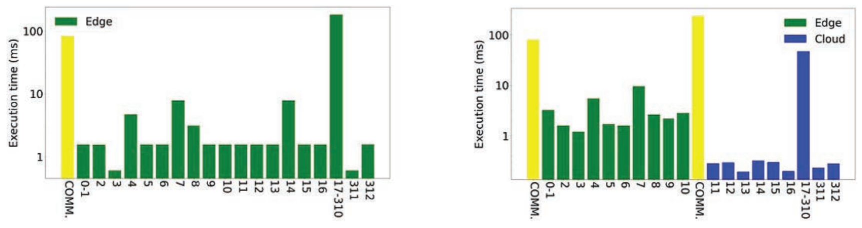


FIGURE 10. DNN partition of VGG19 with lowest end-to-end latency when the constraint imposed is that the device, edge and cloud must be used in different networks.



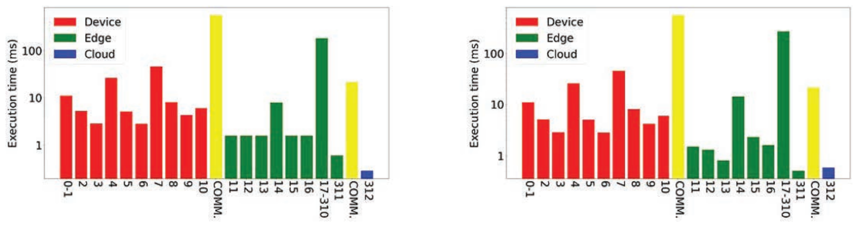
(a) 3G **(b)** 4G

FIGURE 11. DNN partition of ResNet50 with lowest end-toend latency when the constraint imposed is that the device, edge and cloud must be used in different networks.



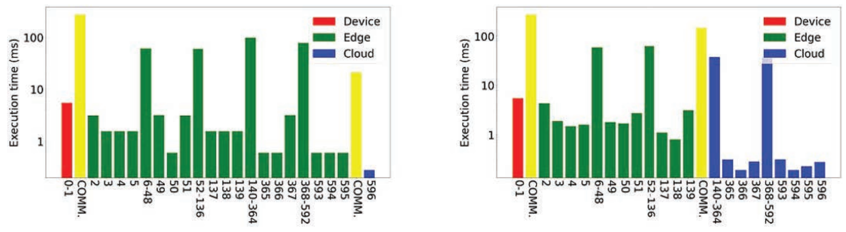
(a) Edge (1) **(b)** Edge (2)

FIGURE 12. Lowest latency executions of InceptionV3 when the edge must be used in a wired network with the device.



(a) Edge (1) **(b)** Edge (2)

FIGURE 13. Lowest latency executions of InceptionV3 when the device, edge and cloud (entire resource pipeline) must be used in a wired network with the device.



(a) Edge (1) **(b)** Edge (2)

FIGURE 14. Lowest latency executions of DenseNet169 when the device, edge and cloud (entire resource pipeline) must be used in a wired network with the device.

edge resources. For the same resource pipelines it is noted that for InceptionV3 there is no change to the partition configuration whereas for DenseNet169 the partition configuration changes.

5. *Top N performance-efficient DNN partitions:* `Scission` provides a list of potential candidate DNN partitions. Table IV shows the top three partitions with lowest endto-end latency of ResNet50 for four different distributed pipelines that use a wired network between the device and the edge.

   Figure 15 shows the DNN partitions with the first and second lowest end-to-end latencies for ResNet50 when Edge(1) must be used in the resource pipeline. The fastest partition requires offloading the most layers to the cloud, resulting in an end-to-end latency of 0.237 seconds, transferring a total of 0.785MB across the resources. On the other hand, the second DNN partition is an edge-only execution that has an end-to-end latency of 0.248 seconds, and only requires the input 150KB to be transferred to the edge. The benefit of the second partition is that it uses a much lower bandwidth than the first partition. `Scission` thus provides a user with a list of potential DNN configurations each of which might benefit in different scenarios.

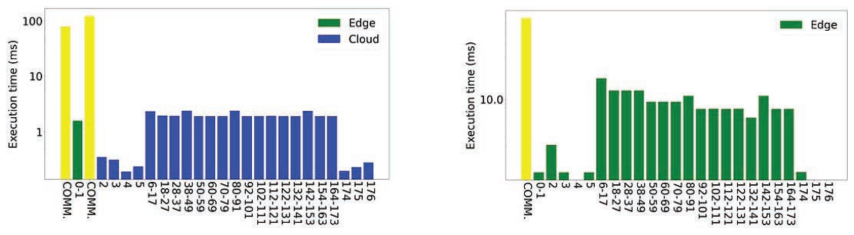
C. SUMMARY

The following are observations from the results:

1. `Scission` takes between 1.52 and 38.14 seconds for generating the benchmark data for different DNNs on the cloud and edge. Due to this overhead `Scission` would be more appropriate for responding to operational changes periodically rather than in real-time.
2. DNN partitioning is affected by different network conditions. Although a cloud-native DNN execution was beneficial for some of the examples (VGG19 and ResNet50), it was noted that MobileNetV2 presented the possibility of both a device-native and cloud-native execution for 3G and 4G networks respectively.
3. A slightly larger input data of 170KB over 150KB changes the DNN partition of ResNet50. This highlights the potential sensitivity of DNN partitioning to data sizes. These are subtle and not quickly evident to manual inspection.
4. User constraints, such as requiring the use of the entire resource pipeline, affects DNN

TABLE 4. Top 3 DNN partitions with the lowest end-to-end latency for ResNet50 across different distributed resource pipelines.

Layers			**End-to-end latency (s)**	**Total data transfer (MB)**
Device	Edge(1)	Cloud(GPU)		
Device-Edgepipeline				
0-1	2-176	-	0.446	0.634
0-91	92-176	-	0.944	0.831
0-175	176	-	0.979	0.008
Device-Cloudpipeline				
0-1	-	2-176	0.339	0.635
0-91	-	92-176	0.920	0.803
0-101	-	102-176	0.996	0.803
Edge-Cloudpipeline				
-	0-1	2-176	0.237	0.785
-	0-175	176	0.269	0.159
-	0-153	154-176	0.319	0.552
Device-Edge-Cloudpipeline				
0-1	2-175	176	0.468	0.643
0-1	2-153	154-176	0.517	1.036
0-1	2-163	164-176	0.523	1.036



(a) First rank **(b)** Second rank

FIGURE 15. Top two DNN partitions with lowest end-to-end latency for ResNet50 when Edge (1) must be used and the device is connected to the edge via a wired network.

partitions. The sequence of layers on the device, edge and cloud change for different networks, such as VGG19 and ResNet50. These cannot manually be identified.

5. Variation in the edge hardware characteristics affects DNN partitioning. For InceptionV3 it was noted that using two different edge resources did not result in different partition configurations. However, the difference in performance between the two edge resources resulted in different configurations for Densenet169.
6. Obtaining a set of ranked configurations can help maximize performance in different scenarios. For example, the fastest partition with the lowest end-to-end latency for ResNet50 when a certain edge resource is to be utilized has more layers on the cloud. The second fastest partition is edge-native (suitable for enhanced privacy). The results that can be observed on `Scission` are exhaustive. The above is only a subset of those observations arising from the experimental results. The need for such a tool is essential as more complex DNNs are appearing and is required to optimally leverage the edge and maximize the performance of distributed DNNs.

IV. RELATED WORK

Many applications have been demonstrated to benefit from using the edge either by running them natively on the edge, across the cloud and edge, or across the cloud, edge and device[22,23,24]. DNNs are an example application that can be executed natively on a single resource, such as a end user device, or on the edge or cloud, or in a distributed manner across multiple resources[6]. DNN partitioning is one approach that is essential for the distributed execution of DNNs[6,7]. This has gained prominence with the upcoming paradigms in distributed systems, such as edge computing[3,4,5], because by using an edge resource a series of layers of the DNN can be executed closer to the input data source, thereby reducing the ingress bandwidth demands and end-to-end latency in a resource rich environment.

There are two main methodologies that have been considered in DNN partitioning for inference (DNN partitioning for training is not considered in this paper). The first is DNN layer distribution and the second is DNN sub-model distribution. *DNN layer distribution* refers to the distribution of a sequence of layers on to a resource by assuming that the resource has access to the entire pre-trained model and weights[25,26,27] (this methodology will be further considered).

DNN sub-model distribution on the other hand refers to slicing the DNN model for different resources and does not require the entire model, rather it only requires the metadata relevant to the slice of the model being executed[28,29]. IONN introduces the concept of incremental offloading in which a DNN is partitioned and incrementally uploaded on to an edge server so as to enable partial execution of the DNN even before the entire DNN is available on the edge server[28]. DeepX partitions the DNN model into several sub-models, which are then distributed to the edge[29].

However, DNN layer distribution is a less intrusive method than DNN sub-model distribution as it does not require the DNN to be modified. Regardless, both methodologies require the identification of valid and optimal partitioning points for deploying optimal DNN partitions across resources, given the numerous combinations that may be possible. `Scission` is positioned as a tool to be used by system and network administrators for maximizing distributed DNN performance using the edge. Therefore, the design decision is one that is less intrusive and can be broadly applied.

Approaches adopted for determining optimal DNN partitions are: (i) Profiling and estimation-based, (ii) integer linear programming-based (ILP), (iii) structural modificationbased, and (iv) benchmarking-based approaches.

Profiling and estimation-based approaches are popular and aim to estimate the performance against metrics, such as end-to-end latency, energy or a combination, for each layer type in the DNN. Four examples of this approach are presented.

Neurosurgeon is one example in which a regression-based method is used for estimating optimal partitions between a device and the cloud[25]. This is achieved by building models on the performance of individual types of layers and their configuration. DeepWear similarly uses a similar approach to train prediction models to estimate latency and energy consumption of four popular layer types and their parameter combinations across a wearable and its paired device[30]. The models are also trained with device-specific latencies and energy prediction models. Musical Chair is another profiling and estimation-based approach that develops behavioral models that are trained to estimate the latency and memory usage of specific layer configurations[31]. Couper is another such approach in which the end-toend latency of each potential partition is verified on a set of resources and then assumed as a direct correlation to hardware capability for other resource configurations[8]. These approaches generally work well within the space they are trained for. However, if a new layer type/configuration or hardware emerges, then the estimation models will not be accurate. In addition, many of these approaches make assumptions regarding the execution behavior of different layers on the underlying hardware. It is not entirely possible to accurately model the execution profiles on complex hardware architectures.

ILP-based approaches have also been considered for DNN partitioning. Within the context, the partitioning problem is formulated as an ILP problem with the aim to find an optimal partition that minimizes the inference latency and maximizes accuracy[26,32]. ILP techniques can be time consuming.

Structural modification-based approaches can efficiently partition DNNs, but in an intrusive manner. It can be achieved realistically only by modifying underlying libraries of existing frameworks or by writing bespoke code for DNNs. However, these approaches provide a finegrained control over DNN partitioning. Examples include DeepThings[33] and MoDNN[34]. DeepThings utilizes fuse tile partitioning, in which a DNN is not partitioned horizontally (based on layers), rather they are partitioned vertically to reduce resource footprint[33]. MoDNN is developed for distributing DNNs across different nodes of the same cluster[34]. Three approaches are presented: (i) for partitioning the convolutional layers, biased one-dimensional partitioning, (ii) for partitioning the weights, modified spectral co-clustering (the fully connected layers are dependent on weights), and (iii) for partitioning sparse fully connected layers, fine-grain cross partition are proposed.

Benchmarking-based approaches are proposed so that actual measurements or observations are made on the target hardware resource. No assumptions are made of the underlying hardware or performance of the layers on the hardware and therefore are more reliable. In these approaches, benchmarking data of the DNN on the hardware is first obtained. Then during deployment, a snapshot of the operational environment (for example, load on the network and compute resource) is taken and the optimal partition is calculated. This approach is minimally intrusive, requires no modification to the code, and is a pragmatic solution in the complex space of DNNs with many layers (and layer types and configurations) and the availability of diverse hardware resources, although they cannot operate in real-time and can only be used periodically. `Scission` proposed in this paper is therefore positioned as a benchmarking-based approach. This approach is used by LAVEA for distributed video analytics[35].

Other approaches, such as approximation-based are also considered in the literature[27], but are not considered here.

V. CONCLUSIONS

This paper presented `Scission`, a tool for automated benchmarking of DNNs on a given set of target device, edge and cloud resources for determining the optimal partition for maximizing DNN performance. `Scission` is underpinned by a benchmarking approach that determines the combination of potential target hardware resources and the sequence of layers that should be distributed for maximizing distributed DNN performance while accounting for user-defined objectives. `Scission` relies on empirical data and does not estimate performance by making assumptions of the target hardware or the DNN layers. Experimental studies were carried out on 18 different DNNs to demonstrate that `Scission` is a valuable tool for obtaining context-aware and performance efficient distributed DNNs. `Scission` can also make decisions that cannot be manually made by a human due to the complexity and number of dimensions affecting the search space.

Limitations and Future Work: Since `Scission` relies on exhaustive benchmarking and search it cannot be used in scenarios that need to account for rapid changes (failures or variance) given the time overhead. Nonetheless, it would prove useful in scenarios where accuracy of the partition configuration is important. Meta-heuristic optimization will be considered to rapidly respond to network congestion or resource failure. Other metrics, such as monetary costs and performance improvements, as well as trade-offs that exist among performance gain and costs, and optimal partitioning and responsiveness of the approach will be considered. The offering of `Scission` as a service will be integrated within a standard orchestration framework to monitor and partition DNNs. The current work assumes that partitioning a given DNN is beneficial and does not account for whether the performance gain may be relatively low. `Scission` can be further extended to determine whether an alternate DNN can be selected for performance gain instead of partitioning a given DNN.

ACKNOWLEDGMENT

Dr Blesson Varghese is supported by a Royal Society Short Industry Fellowship to British Telecommunications plc, UK and by funds from Rakuten Mobile, Japan.

REFERENCES

1. Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, p. 436–444, 2015.
2. J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks*, vol. 61, p. 85–117, 2015
3. B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and Opportunities in Edge Computing," in *Proceedings of the IEEE International Conference on Smart Cloud*, 2016, pp. 20–26.
4. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
5. M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, 2017.
6. Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.
7. X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Communications Surveys Tutorials*, 2020.
8. K. J. Hsu, K. Bhardwaj, and A. Gavrilovska, "Couper: DNN Model Slicing for Visual Analytics Containers at the Edge," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 11 2019, pp. 179–194.
9. F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1800–1807.
10. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2015.
11. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
12. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.
13. C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inceptionv4, Inception-ResNet and the Impact of Residual Connections on Learning," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, p. 4278–4284.
14. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.
15. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
16. G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269.
17. B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "earning Transferable Architectures for Scalable Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.
18. J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
19. A. Lastovetsky and R. R. Manumachu, "New Model-Based Methods and Algorithms for

Performance and Energy Optimization of Data Parallel Applications on Homogeneous Multicore Clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, p. 1119–1133, 2017.

20. M. Xu, F. Qian, and S. Pushp, "Enabling Cooperative Inference of Deep Learning on Wearables and Smartphones," *CoRR*, vol. abs/1712.03073, 2017. [Online]. Available: http://arxiv.org/abs/1712.03073
21. C. Xia, J. Zhao, H. Cui, X. Feng, and J. Xue, "DNN-Tune: Automatic Benchmarking DNN Models for Mobile-Cloud Computing," *ACM Transactions on Architecture and Code Optimization*, vol. 16, no. 4, 2019.
22. J. McChesney, N. Wang, A. Tanwer, E. de Lara, and B. Varghese, "DeFog: Fog Computing Benchmarks," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, p. 47–58.
23. N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A Framework for Edge Node Resource Management," *IEEE Transactions on Services Computing*, 2017.
24. N. Wang, M. Matthaiou, D. S. Nikolopoulos, and B. Varghese, "DYVERSE: DYnamic VERtical Scaling in Multi-tenant Edge Environments," *Future Generation Computer Systems*, vol. 108, pp. 598–612, 2020.
25. Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge," in *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 615–629.
26. H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "JALAD: Joint Accuracy and Latency-aware Deep Structure Decoupling for Edge-Cloud Execution," in *Proceedings of the 24th IEEE International Conference on Parallel and Distributed Systems*, 2018, pp. 671–678.
27. C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic Adaptive DNN Surgery for Inference Acceleration on the Edge," in *Proceedings of the IEEE Conference on Computer Communications*, 2019, pp. 1423–1431.
28. H. J. Jeong, H. J. Lee, C. H. Shin, and S. M. Moon, "IONN: Incremental Offloading of Neural Network Computations from Mobile Devices to Edge Servers," in *Proceedings of the ACM Symposium on Cloud Computing*, 2018, p. 401–411.
29. N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, "DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices," in *Proceedings of the 15th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2016, pp. 1–12.
30. M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu, "DeepWear: Adaptive Local Offloading for On-Wearable Deep Learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 2, pp. 314–330, 2020.
31. R. Hadidi, J. Cao, M. Woodward, M. S. Ryoo, and H. Kim, "Musical Chair: Efficient Real-Time Recognition Using Collaborative IoT Devices," 2018.
32. A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "JointDNN: An Efficient Training and Inference Engine for Intelligent Mobile Cloud Computing Services," *IEEE Transactions on Mobile Computing*, 2019.
33. Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2348–2359, 2018.
34. J. Mao, X. Chen, K. W. Nixon, C. Krieger, and Y. Chen, "MoDNN: Local Distributed Mobile Computing System for Deep Neural Network," in *Proceedings of the Design, Automation Test in Europe Conference Exhibition*, 2017, pp. 1396–1401.
35. S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: Latency-Aware Video Analytics on Edge Computing Platform," in *Proceedings of the IEEE 37th International Conference on Distributed Computing Systems*, 2017, pp. 2573–2574.

IEEE COMPUTER SOCIETY JOBS BOARD

Evolving Career Opportunities Need Your Skills

Explore new options—upload your resume today

www.computer.org/jobs

Changes in the marketplace shift demands for vital skills and talent. The IEEE Computer Society Jobs Board is a valuable resource tool to keep job seekers up to date on the dynamic career opportunities offered by employers.

Take advantage of these special resources for job seekers:

- JOB ALERTS
- TEMPLATES
- WEBINARS
- CAREER ADVICE
- RESUMES VIEWED BY TOP EMPLOYERS

No matter what your career level, the IEEE Computer Society Jobs Board keeps you connected to workplace trends and exciting career prospects.

IEEE COMPUTER SOCIETY

IEEE