Network Working Group
Internet-Draft
Intended status: Informational
Expires: 24 April 2025

S. Randriamasy
Nokia Bell Labs
L. M. Contreras
Telefonica
J. Ros-Giralt
Qualcomm Europe, Inc.
R. Schott
Deutsche Telekom
21 October 2024

Joint Exposure of Network and Compute Information for Infrastructure-Aware Service Deployment
draft-rcr-opsawg-operational-compute-metrics-08

Abstract

Service providers ~~are starting to~~ deploy computing capabilities across their network(s) for hosting applications such as distributed AI workloads, AR/VR, vehicle networks, and IoT, among others. In this network-compute environment, knowing information about the availability and state of the underlying communication and compute resources is necessary to determine both the ~~proper~~ adequate deployment location of the applications and the most suitable server instances on which to run them. ~~Further, t~~This information ~~is~~ might be used ~~by~~ in numerous use cases with different interpretations. This document proposes an initial approach towards a common exposure scheme for metrics reflecting compute and communication capabilities.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://giralt.github.io/draft-rcr-opsawg-operational-compute-metrics/draft-rcr-opsawg-operational-compute-metrics.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-rcr-opsawg-operational-compute-metrics/.

Source for this draft and an issue tracker can be found at https://github.com/giralt/draft-rcr-opsawg-operational-compute-metrics.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

a mis en forme : Surlignage

Table of Contents

1. Introduction 3
2. Conventions and Definitions 4
3. Problem Space and Needs 4
4. Use Cases 7
4.1. Distributed AI Workloads 7
4.2. Open Abstraction for Edge Computing 9
4.3. Optimized Placement of Microservice Components 10
5. Production and Consumption Scenarios of Compute-related Information 10
5.1. Producers of Compute-Related Information 10
5.2. Consumers of Compute-Related Information 11
6. Considerations about Selection and Exposure of Metrics . . . 11
6.1. Considerations about Metrics 11
6.2. Decision Dimensions for Metrics Selection 12
6.3. Abstraction Level and Information Access 14
6.4. Distribution and Exposure Mechanisms 15
6.4.1. Metric Distribution in Computing-Aware Traffic Steering (CATS) 15
6.4.2. Metric Exposure with Extensions of ALTO 15
6.4.3. Exposure of Abstracted Generic Metrics 16
6.5. Examples of Resources 16
6.5.1. Network Resources 16
6.5.2. Cloud Resources 17
7. Study of the Kubernetes Metrics API and Exposure Mechanism . 18
7.1. Understanding the Kubernetes Metrics API and its Exposure Mechanism 18
7.2. Example of How to Map the Kubernetes Metrics API with the IETF CATS METRICS Distribution 19
7.3. Available Metrics from the Kubernetes Metrics API 21
8. Related Work 24

9. Guiding Principles 25
10. GAP Analysis 25
11. Security Considerations 26
12. IANA Considerations 26
13. References 26
13.1. Normative References 26
13.2. Informative References 27
Acknowledgments 28
Authors' Addresses 28

1. Introduction

Operators are starting to deploy distributed computing environments in different parts of their networks that must support a variety of applications with different performance needs such as latency, bandwidth, compute power, storage, energy, etc. This translates in the emergence of distributed compute resources (both in the cloud and at the edge) with a variety of sizes (e.g., large, medium, small) characterized by distinct dimensions of CPUs, memory, and storage capabilities, as well as bandwidth capacity for forwarding the traffic generated in ~~and~~ , out, and among ~~of~~ the corresponding compute resource.

The proliferation of the edge computing ~~paradigm~~ further increases the potential footprint and heterogeneity of the environments where a function or an application can be deployed, resulting in different unitary cost per CPU, memory, and storage. This increases the complexity of deciding the location where a given ~~function or~~ application should be ~~best~~ deployed or executed as a function of a set of target objectives (QoE, cost, etc.). ~~On the one hand, this~~Such decision should be jointly influenced by the available resources in a given computing environment and~~, on the other,~~ by the capabilities of the network ~~path~~ connecting the traffic source with the destination.

Network and compute-aware application placement and service selection has become of utmost importance in the last decade. The availability of such information is taken for granted by the numerous service providers and bodies that are specifying them. However, distributed computational resources often run different implementations with different understandings and representations of compute capabilities, which poses a challenge to the application placement and service selection problems. While standardization efforts on network capabilities representation and exposure are well advanced, similar efforts on compute ~~capabilitites~~capabilities are in their infancy.

This document proposes an ~~initial~~ approach towards a common understanding and exposure scheme for metrics reflecting compute capabilities. It aims at leveraging existing work in the IETF on compute metrics definitions to build synergies. It also aims at reaching out to working or research groups in the IETF that would consume such information and have particular requirements.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and

Commenté [MB1]: I think I understand what you intend, but this is not new per se (think about distributed resources in PoPs, service platforms, etc.).

Commenté [MB2]: Not sure I got the link with the «proliferation of edge computing..»

Commenté [MB3]: No need to repeat each time app/function.

Commenté [MB4]: Not resources as well?

Commenté [MB5]: As multiple paths can be used even within the same network

Commenté [MB6]: I would soften this

Commenté [MB7]: Can we have an example here? Thanks.

a mis en forme : Surlignage

"OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Problem Space and Needs

With the emergence of ~~a new generation of~~ applications with stringent performance requirements (e.g., distributed AI training and inference, driverless vehicles, and virtual/augmented reality) the need for ~~advanced~~ solutions that can model and manage both compute and communication resources has become essential to manage ~~and optimize~~ the performance of these applications. Today's networks connect compute resources deployed across a continuum, ranging from data centers (e.g., cloud computing) to the edge (e.g., edge computing). While the same architecture principles apply across this continuum, ~~in~~ this ~~draft~~ document ~~we~~ focuses on the deployment of services at the edge, involving the cooperation of different actors---namely, network operators, service providers, and applications---in a heterogeneous environment.

In what follows, we use ~~the~~ a simplified service lifecycle ~~of a service~~ to understand the problem space and guide the analysis of the capabilities that are lacking in ~~today's~~ current protocol interfaces needed to enable these new services.

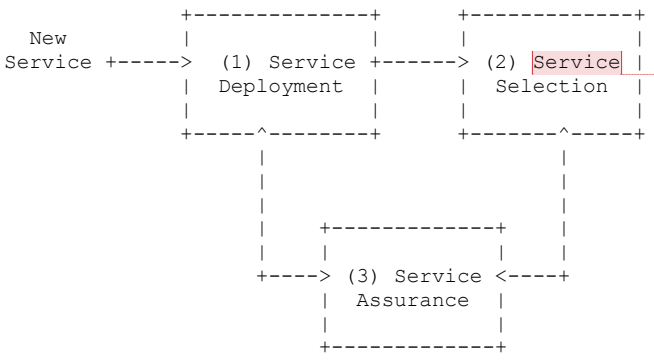


Figure 1: Service lifecycle.

At the edge, compute nodes are deployed near communication nodes (e.g., co-located in a 5G base station) to provide computing services that are close to user devices with the goal to (1) reduce latency, (2) increase communication bandwidth, (3) increase reliability, (4) enable privacy and security, (5) enable personalization, and (6) reduce cloud costs and energy consumption. Services are deployed on the communication and compute infrastructure through a phased lifecycle that generally involves a service _deployment stage_, a _service selection_ stage, and a _service assurance_ stage, as shown in Figure 1.

Commenté [MB8]: Not used in the doc. Can be deleted.

Commenté [MB9]: Covered by «manage»

Commenté [MB10]: Can the «edge» be defined?

Commenté [MB11]: That is?

a mis en forme : Surlignage

Commenté [MB12]: I guess this is about «service instance selection».

Commenté [MB13]: To be defined. Do you mean routers/switches?

Commenté [MB14]: Not sure how this is met in this setup.

(1) Service deployment~~. :*~~ This stage is carried out by the service provider and involves the deployment of a new service (e.g., a distributed AI training/inference, an XR/AR service~~, etc.~~) on the compute and communication infrastructures. The service provider needs to properly size the amount of compute and communication resources assigned to this new service to meet the expected user demand. The decision on where the service is deployed and how many resources are requested from the infrastructure depends on the levels of Quality of Experience (QoE) that the provider wants to guarantee to the users of the service. To make a proper deployment decision, the provider must have visibility on the resources available within the infrastructure, including compute (e.g., CPU, GPU, memory and storage capacity) and communication (e.g., link bandwidth and latency) resources. For instance, to run a Large Language Model (LLM) with 175 billion parameters, a total aggregated memory of 350GB and 5 GPUs may be needed ~~\cite{~~[llm_comp_req~~}.~~]. The service provider needs an interface to query the infrastructure, extract the available compute and communication resources, and decide which subset of resources are needed to run the service.

(2) Service instance selection~~. :*~~ This stage is initiated by the user, through a client application that connects to the deployed service. There are two main actions that must be performed in the service selection stage: (2.a) \textit{compute node selection} and (2.b) \textit{path selection}. In the compute node selection step, as the service is generally replicated in N locations (e.g., by leveraging a microservice architecture), the application must decide which of the service replicas it connects to. This decision depends on the compute properties (e.g., CPU/GPU availability) of the compute nodes running the service replicas. On the other hand, in the path selection decision, the application must decide which path it chooses to connect to the service. This decision depends on the communication properties (e.g., bandwidth and latency) of the available paths. Similar to the service deployment case, the application needs an interface to query the infrastructure and extract the available compute and communication resources, with the goal to make informed node and path selection decisions. Note that in some scenarios, the network or service provider can make node and path selection decisions in lieu of the application. It is also important to note that, ideally, the node and path selection decisions should be jointly optimized, since in general the best end-to-end performance is achieved by jointly taking into account both factors. In some cases, however, such decisions may be owned by different players. For instance, in some network environments, the path selection may be decided by the network operator, ~~where~~whereas the compute node selection may be decided by the application or the service provider. Even in these cases, it is crucial to have an ~~proper~~ interface (for both the operators and the application) to query the available compute and communication resources from the system.

(3) Service assurance~~. :*~~ Due to the stringent Quality of Experience (QoE) requirements of edge applications, service assurance (SA) is also essential. SA continuously monitors service performance to

a mis en forme : Surlignage

a mis en forme : Surlignage

Commenté [MB15]: I would indicate that multiple instances are available. May be add a pointer to the CATS framework.

Commenté [MB16]: The app does not have a control on the path selection, in general. At most, it ca, select an interface but not beyond.

Commenté [MB17]: Is this echoing what is deployed or a target workflow?

Commenté [MB18]: App client/server?

a mis en forme : Surlignage

ensure that the distributed computing and communication system meets the ~~applicable~~ target Service Level Objectives (SLOs). If the SLOs are not met, corrective actions can be taken by the service provider, the application, or the network provider. The evaluation of SLO compliance needs to consider both computing metrics (e.g., compute latency, memory requirements) and communication metrics (e.g., bandwidth, latency). Corrective actions can include both new service placement and new service selection tasks. For instance, upon detecting that a certain compute node is overloaded, increasing the compute delay above the corresponding SLO threshold, the application can reinvoke service node selection (2.a) to migrate its workload to another less utilized compute node. Similarly, upon detecting that a certain communication link is congested, increasing the communication delay above the corresponding SLO threshold, the application can

reinvoke service path selection (2.b) to move the data flow to another less congested link. If SA detects that there are not enough compute or communication resources to guarantee the SLOs, it can also invoke service placement (1) to allocate additional compute and communication resources.

Table 1 summarizes the problem space, the information that needs to be exposed, and the stakeholders that need this information.

Action to take	Information needed	Who needs it?
(1) Service placement	Compute and communication	Service provider
(2.a) Service instance selection: compute node selection	Compute and communication	Network provider, service provider or application
(2.b) Service selection: path selection	Communication	Network provider or application
(3) Service assurance	Compute and communication	Network provider, service provider or application

Table 1: Problem space, needs, and stakeholders.

4. Use Cases

Commenté [MB19]: Some of these use cases can be simplified and replaced with a reference to the CATS use case I-D.

4.1. Distributed AI Workloads

Generative AI is a technological feat that opens up many applications such as holding conversations, generating art, developing a research paper, or writing software, among many others. Yet this innovation comes with a high cost in terms of processing and power consumption. While data centers are already running at capacity, it is projected that transitioning current search engine queries to leverage

generative AI will increase costs by 10 times compared to traditional search methods [DC-AI-COST]. As (1) computing nodes (CPUs, GPUs, and NPUs) are deployed to build the edge cloud leveraging technologies like 5G and (2) with billions of mobile user devices globally providing a large untapped computational platform, shifting part of the processing from the cloud to the edge becomes a viable and necessary step towards enabling the AI-transition. There are at least four drivers supporting this trend:

* Computational and energy savings: Due to savings from not needing large-scale cooling systems and the high performance-per-watt efficiency of the edge devices, some workloads can run at the edge at a lower computational and energy cost [EDGE-ENERGY], especially when considering not only processing but also data transport.

* Latency: For applications such as driverless vehicles which require real-time inference at very low latency, running at the edge is necessary.

* Reliability and performance: Peaks in cloud demand for generative AI queries can create large queues and latency, and in some cases even lead to denial of service. Further, limited or no connectivity generally requires running the workloads at the edge.

* Privacy, security, and personalization: A "private mode" allows users to strictly utilize on-device (or near-the-device) AI to enter sensitive prompts to chatbots, such as health questions or confidential ideas.

These drivers lead to a distributed computational model that is hybrid: Some AI workloads will fully run in the cloud, some will fully run in the edge, and some will run both in the edge and in the cloud. Being able to efficiently run these workloads in this hybrid, distributed, cloud-edge environment is necessary given the aforementioned massive energy and computational costs. To make optimized service and workload placement decisions, information about both the compute and communication resources available in the network is necessary too.

Consider as an example a large language model (LLM) used to generate text and hold intelligent conversations. LLMs produce a single token per inference, where a token is a set of characters forming words or fractions of words. Pipelining and parallelization techniques are used to optimize inference, but this means that a model like GPT-3 could potentially go through all 175 billion parameters that are part of it to generate a single word. To efficiently run these computational-intensive workloads, it is necessary to know the availability of compute resources in the distributed system. Suppose that a user is driving a car while conversing with an AI model. The model can run inference on a variety of compute nodes, ordered from lower to higher compute power as follows: (1) the user's phone, (2) the computer in the car, (3) the 5G edge cloud, and (4) the datacenter cloud. Correspondingly, the system can deploy four different models with different levels of accuracy and compute requirements. The simplest model with the least parameters can run in the phone, requiring less compute power but yielding lower accuracy. Three other models ordered in increasing value of accuracy and computational complexity can run in the car, the edge, and the

Commenté [MB20]: This one and the one right after may be conflicting

cloud. The application can identify the right trade-off between accuracy and computational cost, combined with metrics of communication bandwidth and latency, to make the right decision on which of the four models to use for every inference request. Note that this is similar to the resolution/bandwidth trade-off commonly found in the image encoding problem, where an image can be encoded and transmitted at different levels of resolution depending on the available bandwidth in the communication channel. In the case of AI inference, however, not only bandwidth is a scarce resource, but also compute.

4.2. Open Abstraction for Edge Computing

~~Modern a~~Applications such as AR/VR, V2X, or IoT, require bringing compute closer to the edge in order to meet strict bandwidth, latency, and jitter requirements. While this deployment process resembles the path taken by the main cloud providers (notably, AWS, Facebook, Google and Microsoft) to deploy their large-scale datacenters, the edge presents a key difference: datacenter clouds (both in terms of their infrastructure and the applications run by them) are owned and managed by a single organization, whereas edge clouds involve a complex ecosystem of operators, vendors, and application providers, all striving to provide a quality end-to-end solution to the user. This implies that, while the traditional cloud has been implemented for the most part by using vertically optimized and closed architectures, the edge will necessarily need to rely on a complete ecosystem of carefully designed open standards to enable horizontal interoperability across all the involved parties.

As an example, consider a user of an XR application who arrives at his/her home by car. The application runs by leveraging compute capabilities from both the car and the public 5G edge cloud. As the user parks the car, 5G coverage may diminish (due to building interference) making the home local Wi-Fi connectivity a better choice. Further, instead of relying on computational resources from the car and the 5G edge cloud, latency can be reduced by leveraging computing devices (PCs, laptops, tablets) available from the home edge cloud. The application's decision to switch from one domain to another, however, demands knowledge about the compute and communication resources available both in the 5G and the Wi-Fi domains, therefore requiring interoperability across multiple industry standards (for instance, IETF and 3GPP on the public side, and IETF and LF Edge [LF-EDGE] on the private home side).

4.3. Optimized Placement of Microservice Components

Current applications are transitioning from a monolithic service architecture towards the composition of microservice components, following cloud-native trends. The set of microservices can have associated Service Level Objectives (SLOs) that impose constraints not only in terms of the required computational resources dependent on the compute facilities available, but also in terms of performance indicators such as latency, bandwidth, etc., which impose restrictions in the networking capabilities connecting the computing facilities. Even more complex constraints, such as affinity among certain microservices components could require complex calculations for selecting the most appropriate compute nodes taken into consideration

both network and compute information.

5. Production and Consumption Scenarios of Compute-related Information

From the standpoint of the network operator and the service provider, understanding the scenarios of production and consumption of compute and communication-related information is essential. By leveraging this combination, it becomes possible to optimize resource and workload placement, leading to significant operational cost reductions for operators and service providers, as well as enhanced service levels for end users.

Commenté [MB21]: This is difficult to assess, but optimization from a technical standpoint are worth to investigate.

5.1. Producers of Compute-Related Information

The information relative to compute (e.g., processing capabilities, memory, or storage capacity~~, etc.~~) can be structured in two ways. On one hand, the information corresponding to the raw compute resources; on the other hand, the information of resources allocated or utilized by a specific application or service function.

The former is typically provided by the management systems enabling the virtualization of the physical resources for a later assignment to the processes running on top. Cloud Managers or Virtual Infrastructure Managers (VIMs) are usually the entities that manage these resources. These management systems offer APIs to access the available resources in ~~the~~ a computing facility. ~~Thus, i~~It can be expected that these APIs can also be used for consuming such information. Once the raw resources are retrieved from the various compute facilities, it is possible to generate topological network views including such resources, as proposed in [I-D.llc-teas-dc-aware-topo-model].

Regarding the resources allocated or utilized by a specific application or service function, two situations apply: (1) The total allocation of resources, and (2) the allocation per service or application. In the first case, the information can be supplied by the virtualization management systems described before. For the specific per-service allocation, it can be expected that the specific management systems of the service or application are capable of providing the resources being used at run time, typically as part of the allocated ones. In this last scenario, it is also reasonable to expect the availability of APIs offering this information, even though they can be specific to the service or application.

a mis en forme : Surlignage

5.2. Consumers of Compute-Related Information

The consumption of compute-related information is relative to the different phases of the service lifecycle (Figure 1). This means that this information can be consumed in different points of time and for different purposes.

The expected consumers can be both external or internal to the network. As external consumers, it is possible to consider external application management systems requiring resource availability information for service function placement decision, workload migration in the case of consuming raw resources, or requiring information on the usage of resources for service assurance or

service scaling, among others.

As internal consumers, it is possible to consider network management entities requiring information on the level of resource utilization for traffic steering (e.g., as done by the Path Selector in [I-D.~~ldbc~~ietf-cats-framework]), load balance, or analytics, among others.

6. Considerations about Selection and Exposure of Metrics

One can distinguish the topics of (1) which kind of metrics need to be exposed and (2) how the metrics are exposed. The infrastructure resources can be divided into (1) network and (2) compute related resources. This section ~~intends to give~~provides a brief outlook regarding these resources ~~for stimulating additional discussion with related work going on in other IETF working groups or standardization bodies~~.

6.1. Considerations about Metrics

The metrics considered in this document are meant to support decisions for selection and deployment of services and applications. Further iterations of this document may consider additional lifecycle operations such as assurance and relevant metrics.

The abovementioned operation may also involve network metrics that are specified in a number of IETF documents such as RFC 9439 ~~[I-D.ietf-alto-performance-metrics]~~, which itself leverages on RFC 7679. The work on compute metrics at the IETF, on the other hand, is in its first stages and merely relates to low-level infrastructure metrics such as in [RFC7666]. However:

* Decisions for service deployment and selection may further involve decisions that require an aggregated view, for instance, at the service level.

* Deciding entities may only have partial access to the compute information and actually do not need to have all the details.

Compute metrics and their acquisition and management have been addressed by standardization bodies outside the IETF such as NIST and DMTF, with the goal to guarantee reliable assessment and comparison of cloud services. A number of public tools and methods to test compute facility performances are made available by cloud service providers or service management businesses (e.g., see [UPCLOUD] and [IR]). However, for the proposed performance metrics, their definition and acquisition method may differ from one provider to the other, making it thus challenging to compare performances across different providers. The latter aspect is particularly problematic for applications running at the edge where a complex ecosystem of operators, vendors, and application providers is involved and calls for a common standardized definition.

6.2. Decision Dimensions for Metrics Selection

Once defined, the compute metrics are to be selected and exposed to management entities acting at different levels, such as a centralized controller or a router, taking different actions such as service

a mis en forme : Surlignage

a mis en forme : Surlignage

a mis en forme : Surlignage

Commenté [MB22]: For what purpose?

a mis en forme : Surlignage

placement, service selection, and assurance, with decision scopes ranging from local compute facilities to end-to-end services.

Upon exploring existing work, this ~~draft~~ document proposes to consider a number of "decision dimensions" reflecting the abovementioned aspects in order to help identifying the suitable compute metrics needed to take a service operation decision. This list is initial and is to be updated upon further discussion.

Dimensions helping to identify needed compute metrics:

Dimension	Definition of the dimension	Examples
Target operation	What operation the metric is used for	Monitoring, benchmarking, service placement and selection
Driving KPI(s)	KPI(s) assessed with the metrics	Speed, scalability, cost, stability
Decision scope	Granularity of metric definition	Infrastructure node/cluster, compute service, end-to-end application
Receiving entity	Function receiving the metrics	Router, centralized controller, application management
Deciding entity	Function using the metrics to compute decisions	Router, centralized controller, application management

Table 2: Dimensions to consider when identifying the needed compute metrics.

The "value" of a dimension has an impact on the characteristic of the metric to consider. In particular:

* The target operation: determines the specific use case for the metric, such as monitoring, benchmarking, service placement, or selection, guiding the selection of relevant metrics.

* The driving KPI(s): leads to selecting metrics that are relevant from a performance standpoint.

* The decision scope: leads to selecting metrics at a relevant granularity or aggregation level.

* The receiving entity: impacts the dynamicity of the received metric values. While a router likely receives static information to moderate overhead, a centralized control function may receive more dynamic information that it may additionally process on its own.

* The deciding entity: computes the decisions to take upon metric values and needs information that is synchronized at an appropriate frequency.

Metric values undergo various lifecycle actions, primarily acquisition, processing, and exposure. These actions can be executed through different methodologies. Documenting these methodologies enhances the reliability and informed utilization of the metrics. Additionally, detailing the specific methods used for each approach further increases their reliability. The table below provides some examples:

Lifecycle action	Example
Acquisition method	telemetry, estimation
Value processing	aggregation, abstraction
Exposure	in-path distribution, off-path distribution

Table 3: Examples of lifecycle actions documented on metrics.

6.3. Abstraction Level and Information Access

One important aspect to consider is that receiving entities that need to consume metrics to take selection or placement decisions do not always have access to computing information. In particular, several scenarios may need to be considered, among which:

* The consumer is an ISP that does not own the compute infrastructure or has no access to full information. In this case, the compute metrics will likely be estimated.

* The consumer is an application that has no direct access to full information while the ISP has access to both network and compute information. However, the ISP is willing to provide guidance to the application with abstract information.

* The consumer has access to full network and compute information and wants to use it for fine-grained decision making, e.g., at the node/cluster level.

* The consumer has access to full information but essentially needs guidance with abstracted information.

* The consumer has access to information that is abstracted or

detailed depending on the metrics.

These scenarios further drive the selection of metrics upon the above mentioned dimensions.

6.4. Distribution and Exposure Mechanisms

6.4.1. Metric Distribution in Computing-Aware Traffic Steering (CATS)

The IETF CATS WG has explored the collection and distribution of computing metrics in [I-D.~~ldbc~~ietf-cats-framework]. ~~In their deployment considerations, the authors~~The document consider three deployment models for the location of the service selection function: distributed, centralized and hybrid. For these three models, the compute metrics are, respectively:

* Distributed among network devices directly.

* Collected by a centralized control plane.

* Hybrid where some compute metrics are distributed among involved network devices, and others are collected by a centralized control plane.

In the hybrid mode, the draft says that some static information (e.g., capabilities information) can be distributed among network devices since they are quite stable. Frequent changing information (e.g., resource utilization) can be collected by a centralized control plane to avoid frequent flooding in the distributed control plane.

The hybrid mode thus stresses the impact of the dynamicity of the distributed metrics and the need to carefully sort out the metric exposure mode with respect to their dynamicity.

The section on Metrics Distribution also indicates the need for required extensions to the routing protocols, in order to distribute additional information such as link latency and other information not standardized in these protocols, such as compute metrics.

6.4.2. Metric Exposure with Extensions of ALTO

The ALTO protocol has been defined to expose an abstract network topology and related path costs in [RFC7285]. ALTO is a client-server protocol exposing information to clients that can be associated to applications as well as orchestrators. Its extension RFC 9240 allows to define entities on which properties can be defined, while [I-D.contreras-alto-service-edge] introduces a proposed entity property that allows to consider an entity as both a network element with network related costs and properties and a element of a data center with compute related properties. Such an exposure mechanism is particularly useful for decision making entities which are centralized and located off the network paths.

6.4.3. Exposure of Abstracted Generic Metrics

In some cases, whether due to unavailable information details or for the sake of simplicity, a consumer may need reliable but simple guidance to select a service. To this end, abstracted generic metrics may be useful.

One can consider a generic metric that can be named 'computingcost' and is applied to a contact point to one or more edge servers such as a load balancer, for short an edge server, to reflect the network operator policy and preferences. The metric "computingcost" results from an abstraction method that is hidden from users, similarly to the metric "routingcost" defined in [RFC7285]. For instance, "computingcost" may be higher for an edge server located far away, or in disliked geographical areas, or owned by a provider who does not share information with the Internet Service Provider (ISP) or with which the ISP has a poorer commercial agreement. 'computingcost' may also reflect environmental preferences in terms, for instance, of energy source, average consumption vs. local climate, location adequacy vs. climate.

One may also consider a generic metric named 'computingperf', applied to an edge server, that reflects its performance based on measurements or estimations by the ISP or combination thereof. An edge server with a higher "computingperf" value will be preferred. "computingperf" can be based on a vector of one or more metrics reflecting, for instance, responsiveness, reliability of cloud services based on metrics such as latency, packet loss, jitter, time to first and/or last byte, or a single value reflecting a global performance score.

6.5. Examples of Resources

6.5.1. Network Resources

Network resources relate to the traditional network infrastructure. The next table provides examples of some of the commonly used metrics:

Kind of Resource
QoS
Latency
Bandwidth
RTT
Packet Loss
Jitter

Table 4: Examples of network resource metrics.

Commenté [MB23]: Isn't this covered by the loss/delay/jitter/ ?

6.5.2. Cloud Resources

Cloud resources relate to the compute infrastructure ~~infrastructure~~. The next table provides examples of some of the commonly used metrics:

Resource	Type	Example
CPU	Compute	Available CPU resources in GHz
Memory	Compute	Available memory in GB
Storage	Storage	Available storage in GB
Configmaps	Object	Configuration and topology maps
Pods	Object	Current list of active pods
Jobs	Object	current list of active jobs
Services	Object	Concurrent services

Table 5: Examples of cloud resource parameters.

7. Study of the Kubernetes Metrics API and Exposure Mechanism

An approach to develop IETF specifications for the definition of compute and communication metrics is to leverage existing and mature solutions, whether based on open standards or de facto standards. On one hand, this approach avoids reinventing the wheel; on the other, it ensures the specifications are based on significant industry experience and stable running code.

For communication metrics, the IETF has already developed detailed and mature specifications. An example is the ALTO Protocol [RFC7285], which provides RFCs standardizing communication metrics and a detailed exposure mechanism protocol.

Compute metrics, however, have not been thoroughly studied within the IETF. With the goal to avoid reinventing the wheel and to ensure significant industry experience is taken into account, in this section we study the Kubernetes Metric API. Kubernetes is not only a de facto standard to manage containerized software in data centers, but it is also increasingly being used by telecommunication operators to manage compute resources at the edge.

7.1. Understanding the Kubernetes Metrics API and its Exposure Mechanism

Figure 2 shows the Kubernetes Metric API architecture. It consists of the following components:

Pod. A collection of one or more containers.

Cluster. A collection of one or more nodes.

HPA, VPA and 'kubectl stop'. Three different applications that

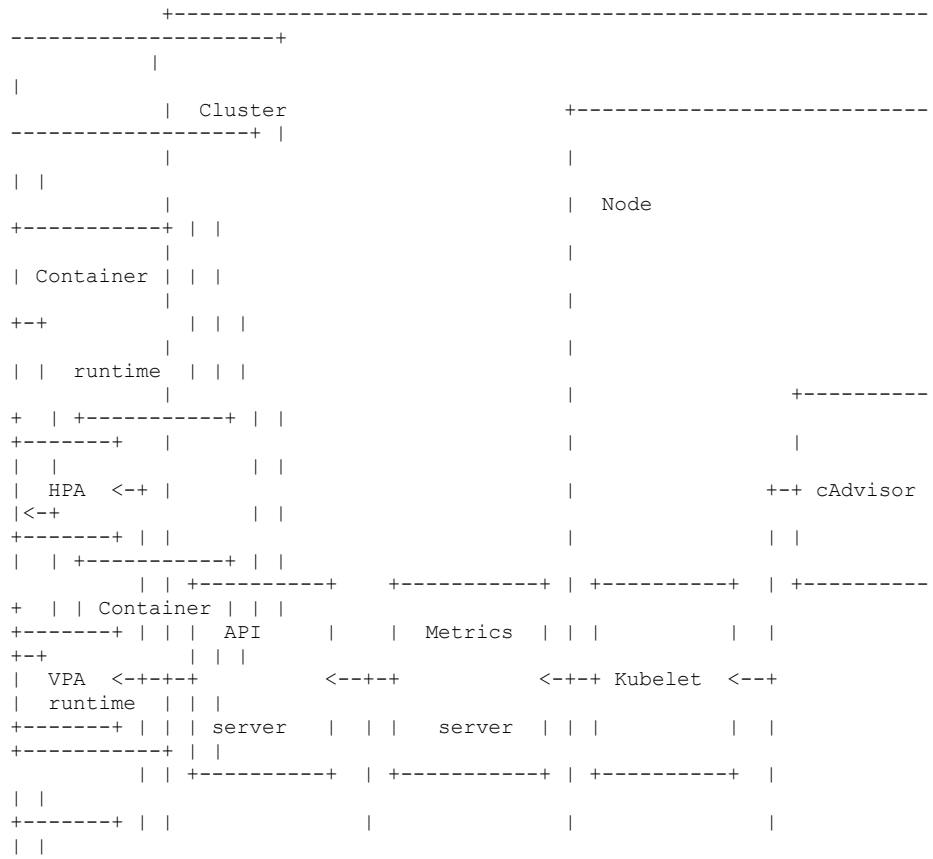
serve as examples of consumers of the Metrics API. The HorizontalPodAutoscaler (HPA) and VerticalPodAutoscaler (VPA) use data from the metrics API to adjust workload replicas and resources to meet customer demand. 'kubectl stop' can be used to show all the metrics.

cAdvisor. Daemon for collecting metrics (CPU, memory, GPU, etc.) from all the containers in a pod. It is responsible for aggregating and exposing these metrics to kubelet.

Kubelet. Node agent responsible for managing container resources. It includes the ability to collect the metrics from the cAdvisor and making them accessible using the /metrics/resource and /stats kubelet API endpoints.

Metrics server. Cluster agent responsible for collecting and aggregating resource metrics from each kubelet.

API Server. General server providing API access to kubernetes services. One of them corresponds to the Metrics API service. HPA, VPA, and 'kubectl top' query the API server to retrieve the metrics.



```
|kubectl| | |                |                |                | +----------
+            | |
| top   <-+ |                | +-----------+ |                | |  Other
|            | |
+-------+   |                | |  Other    | |               +-+   pod
|            | |
            |               +-+           | |                 |   data
|            | |
            |                  |  data     | |                 +----------
+            | |
            |                  +-----------+ |
| |
            |                               +----------------------------
-------------------+ |
            |
|
            +------------------------------------------------------------
---------------------+
```

Figure 2: Collection and exposure of metrics using the Kubernetes Metrics API.

7.2. Example of How to Map the Kubernetes Metrics API with the IETF CATS METRICS Distribution

In this section, we describe a mapping between the Kubernetes Metrics API and the IETF CATS metric dissemination architecture, illustrating ~~and~~ an example of how a de facto standard widely used in production systems can be adapted to support the CATS metrics framework.

To describe the mapping, we take the centralized model of the CATS metrics dissemination framework introduced in [I-D.~~ldbe~~ietf-cats-framework], which we include in Figure 3 for ease of reading. (Similar mappings can be created with the distributed and hybrid models also introduced in this Figure 3)

```
               :       +-------+
               :<------| C-PS  |<-----------------------------------+
               :       +-------+ <-------+              +---------+  |
               :          ^             |           +--|CS-ID 1 |  |
               :          |             |           |  |CIS-ID 1|  |
               :          |   +-----------------+   |  +---------+  |
               :          |   |    C-SMA       |---|Service Site 2|
               :          |   +-----------------+   |  +---------+  |
               :          |   |CATS-Forwarder 2|   +--|CS-ID 1 |  |
               :          |   +-----------------+      |CIS-ID 2|  |
   +---------+  :          |             |              +---------+  |
   | Client |  :  Network |   +----------------------+              |
   +---------+  :  metrics |   | +--------+           |              |
        |       :         +-----| C-NMA |           |      +------+
        |       :          |   | +--------+           |      |C-SMA|<-+
   +-----------------+ <---+   |                      |      +------+  |
   |CATS-Forwarder 1|---------|                      |         ^     |
   +-----------------+         |      Underlay        |         |     |
               :              |    Infrastructure    |     +---------+|
               :              |                      |     |CS-ID 1 ||
               :              +----------------------+  +--|CIS-ID 3||
```

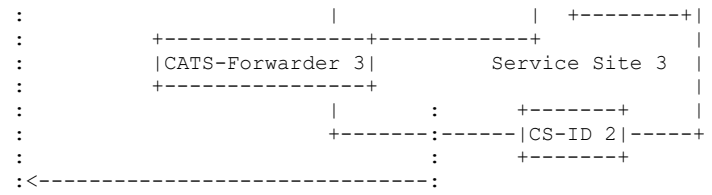


Figure 3: Collection and exposure of metrics using the CATS Centralized Model. (Taken from [I-D.~~ldbe~~ietf-cats-framework])

Commenté [MB24]: Need to map service contact instance

The following table provides the mapping:

IETF CATS component	Kubernetes Metrics API component
CIS-ID	Cluster API
C-SMA	cAdvisor
C-NMA	Other components outside Kubernetes
C-PS	Other components outside Kubernetes
CATS Service Site	Node or cluster
CATS Service	One or more clusters distributed on several locations

Table 6: Example of how to map the Kubernetes Metrics API with the IETF CATS Architecture.

Note that while in Kubernetes there are multiple levels of abstraction to reach the Metrics API (cAdvisor -> kubelet -> metrics server -> API server), they can all be co-located in the cAdvisor, which can then be mapped to the C-SMA module in CATS.

7.3. Available Metrics from the Kubernetes Metrics API

Commenté [MB25]: Would be great to assess which ones are useful/meaningful for CATS.

The Kubernetes Metrics API implementation can be found in staging/src/k8s.io/kubelet/pkg/apis/stats/v1alpha1/types.go as part of the Kubernetes repository (https://github.com/kubernetes/kubernetes):

In this section we provide a summary of the metrics offered by the API:

Nodel-level metric	~~Decription~~Description
nodeName	Name of the node
ContainerStats	Stats of the containers within this node
CPUStats	Stats pertaining to CPU resources

MemoryStats	Stats pertaining to memory (RAM) resources
NetworkStats	Stats pertaining to network resources
FsStats	Stats pertaining to the filesystem resources
RuntimeStats	Stats about the underlying containers runtime
RlimitStats	Stats about the rlimits of system

Table 7: Summary of the Kubernetes Metric API: Node-level metrics.

Pod-level metric	Description
PodReference	Reference to the measured Pod
CPU	Stats pertaining to CPU resources consumed by pod cgroup
Memory	Stats pertaining to memory (RAM) resources consumed by pod cgroup
NetworkStats	Stats pertaining to network resources
VolumeStats	Stats pertaining to volume usage of filesystem resources
FsStats	Total filesystem usage for the containers
ProcessStats	Stats pertaining to processes

Table 8: Summary of the Kubernetes Metric API: Pod-level metrics.

Container-level metric	Description
name	Name of the container

CPUStats	Stats pertaining to CPU resources
MemoryStats	Stats pertaining to memory (RAM) resources
AcceleratorStats	Metrics for Accelerators (e.g., GPU, NPU, etc.)
FsStats	Stats pertaining to the container's filesystem resources
UserDefinedMetrics	User defined metrics that are exposed by containers in the pod

Table 9: Summary of the Kubernetes Metric API: Container-level metrics.

For more details, refer to https://github.com/kubernetes/kubernetes under the path staging/src/k8s.io/kubelet/pkg/apis/stats/v1alpha1/types.go.

8. Related Work

Some existing work has explored compute-related metrics. It can be categorized as follows:

* *References providing raw compute infrastructure metrics*:
  - [I-D.contreras-alto-service-edge] includes references to cloud management solutions (e.g., OpenStack, Kubernetes) that administer the virtualization infrastructure, providing information about raw compute infrastructure metrics.
  - [NFV-TST] describes metrics related to processor, memory, and network interface usage.
* *References providing compute virtualization metrics*:
  - [RFC7666] defines several metrics as part of the Management Information Base (MIB) for managing virtual machines controlled by a hypervisor. These objects reference the resources consumed by a particular virtual machine serving as a host for services or applications.
  - [NFV-INF] provides metrics associated with virtualized network functions.
* *References providing service metrics including compute-related information*:
  - [I-D.dunbar-cats-edge-service-metrics] proposes metrics associated with services running in compute infrastructures. Some of these metrics do not depend on the infrastructure behavior itself but on the topological location of the compute infrastructure.
* *Other existing work at the IETF CATS WG*:

- [I-D.ldbc-cats-framework] explores the collection and distribution of computing metrics. In their deployment considerations, they consider three models: distributed, centralized, and hybrid.

9. Guiding Principles

The driving principles for designing an interface to jointly extract network and compute information are as follows:

* *P1. Leverage existing metrics across working groups to avoid reinventing the wheel.* For instance:

  - RFC 9439 ~~([I-D.ietf-alto-performance-metrics])~~ leverages IPPM metrics from RFC 7679.

  - Section 5.2 of [I-D.du-cats-computing-modeling-description] considers delay as a good metric, since it is easy to use in both compute and communication domains. RFC 9439 also defines delay as part of the performance metrics.

  - Section 6 of [I-D.du-cats-computing-modeling-description] proposes representing the network structure as graphs, similar to the ALTO map services in RFC 7285.

* *P2. Aim for simplicity, while ensuring the combined efforts don't leave technical gaps in supporting the full lifecycle of service deployment and selection.* For instance:

  - The CATS working group covers path selection from a network standpoint, while ALTO (e.g., RFC 7285) covers exposing network information to the service provider and the client application. However, there is currently no effort being pursued to expose compute information to the service provider and the client application for service placement or selection.

10. GAP Analysis

From this related work it is evident that compute-related metrics can serve several purposes, ranging from service instance instantiation to service instance behavior, and then to service instance selection. Some of the metrics could refer to the same object (e.g., CPU) but with a particular usage and scope.

In contrast, the network metrics are more uniform and straightforward. It is then necessary to consistently define a set of metrics that could assist to the operation in the different concerns identified so far, so that networks and systems could have a common understanding of the perceived compute performance. When combined with network metrics, the combined network plus compute performance behavior will assist informed decisions particular to each of the operational concerns related to the different parts of a service lifecycle.

11. Security Considerations

TODO Security

12. IANA Considerations

This document has no IANA actions.

13. References

13.1. Normative References

[I-D.du-cats-computing-modeling-description]
Du, Z., Yao, K., Li, C., Huang, D., and Z. Fu, "Computing Information Description in Computing-Aware Traffic Steering", Work in Progress, Internet-Draft, draft-du-cats-computing-modeling-description-03, 6 July 2024, <https://datatracker.ietf.org/doc/html/draft-du-cats-computing-modeling-description-03>.

[I-D.ietf-alto-performance-metrics]
Wu, Q., Yang, Y. R., Lee, Y., Dhody, D., Randriamasy, S., and L. M. Contreras, "Application-Layer Traffic Optimization (ALTO) Performance Cost Metrics", Work in Progress, Internet-Draft, draft-ietf-alto-performance-metrics-28, 21 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-alto-performance-metrics-28>.

[I-D.ldbc-cats-framework]
Li, C., Du, Z., Boucadair, M., Contreras, L. M., and J. Drake, "A Framework for Computing-Aware Traffic Steering (CATS)", Work in Progress, Internet-Draft, draft-ldbc-cats-framework-06, 8 February 2024, <https://datatracker.ietf.org/doc/html/draft-ldbc-cats-framework-06>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/rfc/rfc7285>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

13.2. Informative References

[DC-AI-COST]
"Generative AI Breaks The Data Center - Data Center Infrastructure And Operating Costs Projected To Increase To Over $76 Billion By 2028", Forbes, Tirias Research Report , 2023.

[EDGE-ENERGY]
"Estimating energy consumption of cloud, fog, and edge computing infrastructures", IEEE Transactions on Sustainable Computing , 2019.

[I-D.contreras-alto-service-edge]
Contreras, L. M., Randriamasy, S., Ros-Giralt, J., Perez, D. A. L., and C. E. Rothenberg, "Use of ALTO for Determining Service Edge", Work in Progress, Internet-Draft, draft-contreras-alto-service-edge-10, 13 October 2023, <https://datatracker.ietf.org/doc/html/draft-contreras-alto-service-edge-10>.

[I-D.dunbar-cats-edge-service-metrics]
Dunbar, L., Majumdar, K., Mishra, G. S., Wang, H., and H. Song, "5G Edge Services Use Cases", Work in Progress, Internet-Draft, draft-dunbar-cats-edge-service-metrics-01, 6 July 2023, <https://datatracker.ietf.org/doc/html/draft-dunbar-cats-edge-service-metrics-01>.

[I-D.llc-teas-dc-aware-topo-model]
Lee, Y., Liu, X., and L. M. Contreras, "DC aware TE topology model", Work in Progress, Internet-Draft, draft-llc-teas-dc-aware-topo-model-03, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-llc-teas-dc-aware-topo-model-03>.

[IR] "Cloud Performance Testing Best Tips and Tricks", n.d., <https://www.ir.com/guides/cloud-performance-testing>.

[LF-EDGE] "Linux Foundation Edge", https://www.lfedge.org/ , March 2023.

[LLM_COMP_REQ]
"Serving OPT-175B, BLOOM-176B and CodeGen-16B using Alpa", n.d., <https://alpa.ai/tutorials/opt_serving.html>.

[NFV-INF] "ETSI GS NFV-INF 010, v1.1.1, Service Quality Metrics", 1 December 2014, <https://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/010/01.01.01_60/gs_NFV-INF010v010101p.pdf>.

[NFV-TST] "ETSI GS NFV-TST 008 V3.3.1, NFVI Compute and Network Metrics Specification", 1 June 2020, <https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/008/03.03.01_60/gs_NFV-TST008v030301p.pdf>.

[RFC7666] Asai, H., MacFaden, M., Schoenwaelder, J., Shima, K., and T. Tsou, "Management Information Base for Virtual Machines Controlled by a Hypervisor", RFC 7666, DOI 10.17487/RFC7666, October 2015, <https://www.rfc-editor.org/rfc/rfc7666>.

[UPCLOUD] "How to benchmark Cloud Servers", May 2023, <https://upcloud.com/resources/tutorials/how-to-benchmark-cloud-servers>.

Acknowledgments

The work from Luis M. Contreras has been partially funded by the European Union under Horizon Europe projects NEMO (NExt generation Meta Operating system) grant number 101070118, and CODECO (COgnitive, Decentralised Edge-Cloud Orchestration), grant number 101092696.

Authors' Addresses

S. Randriamasy
Nokia Bell Labs
Email: sabine.randriamasy@nokia-bell-labs.com

L. M. Contreras
Telefonica
Email: luismiguel.contrerasmurillo@telefonica.com

Jordi Ros-Giralt
Qualcomm Europe, Inc.
Email: jros@qti.qualcomm.com

Roland Schott
Deutsche Telekom
Email: Roland.Schott@telekom.de