Network Powered by Computing

Conference Paper · November 2022

DOI: 10.1109/MoNeTec55448.2022.9960771

Ruslan. L. Smeliansky
Lomonosov Moscow State University

Network Powered by Computing

Ruslan Smeliansky
Dept. of Computational Mathematics and Cybernetics
Moscow State University
Moscow, Russia
smel@cs.msu.su

***Abstract*—The paper presents the concept of Network Powered by Computing (NPC)—Computing infrastructure, which is the convergence of data networks with computing installations like DC, Edge, HPC. This concept is based on an analysis of the properties of modern applications and the requirements for the computational infrastructure for them. The functional architecture of NPC and the main problems on the way of its feasibility are described.**

***Keywords*—computational infrastructure, serverless computing, computer resources planning**

I. Introduction

Organization of infrastructure for computation is one of the key point of modern civilization. Therefore it is important to understand the main trends and prospects of its development, to understand what problems will need to be solved.

II. Modern application properties

The main driving force of the development of infrastructure for computing, its operating environment, programming tools have always been the needs of applications, which today suite can be summarized as follows:

- **Distributed**—it's a modern application—this is a set of interacting functions/services that run in parallel on several computers connected by NPCs data transmission network (hereinafter as DTN);
- **Self–sufficiency**—application is no longer represented by code and source data only, it is accompanied by a description of the structure of the interconnection of the components that make up the application (hereinafter application services), determination of the required level of their productivity explicitly formulated requirements for computing and network resources, data storage and access to data resources, intended timeframes for computing and data transmission in the form of a service level agreement (Service Level Agreement—SLA), application start procedures. This description is written in a special language, an example of which one can be the TOSCA language [1] (hereinafter such a description will be called Application Operation Specification—AOS).
- **Real–time mode**—applications are sensitive to delays and have the limitation for response time;
- **Elasticity**—the performance of the application changes automatically without interruption of its operation in accordance with the requirements of the SLA and the current load on it [2];
- **Cross–platform**—application is independent of the software and hardware environment;
- **Interaction and Synchronization**—Combining the results of different application components is not dependent on their location, but is determined only by AOS;
- **Update–friendly**—updating application components should not affect its operation.

III. Computational Infrastructure Requirements

The computational infrastructure for such applications must meet the following requirements:

- **Behavior predictability**—predictability of delays associated with computations, data transfer and access to data during the application operation, in order to manage application's execution accordingly to the requirements of the SLA;
- **Security**—it does not pose unacceptable risks to the application and its data like Confidentiality, Integrity, Availability;
- **Availability, Reliability and Fault Tolerance**—the infrastructure should be robust enough to ensure a high level of availability and operability of its services, application components, recovery of lost data in case of failures and attacks, to react in real time by changes in topology, traffic flows and shape routing to ensure the fulfillment of SLA requirements;
- **Efficiency and Fairness**—the infrastructure must ensure that the application runs, delivers and processes its data by infrastructure resources, reliably, without impairing other applications and their traffic;
- **Virtualization**—virtualization of all types of infrastructure resources (computing, network, storage);
- **Scalability**—scalability of application performance regardless of its data location, services and intensity of data flow, without stopping application operation;
- **Serverless**—the infrastructure should automatically place application components in a way that allows them to interact according to the application stricture, and in a way that ensures that the SLA requirements of the application are met, while minimizing infrastructure resources utilization [3].

In order for the NPC to meet the requirement of efficiency and predictability and serve as the computational infrastructure for applications, in the above sense (everywhere below the App.), its behavior, functioning must meet the requirements, namely:

- predictability of time of execution of application components and their interaction time (data transfer) according to AOS;
- availability of a variety of virtualized network functions (VNF hereinafter) and based on machine

learning algorithms for distribution, balancing, shaping, filtering and other traffic engineering (TE) methods on DTN channels;

- reliable isolation of control plane and data plane in DTN from errors in network equipment, as well as isolation of different data flows in these planes.

For predictability of the characteristics of data transfer between application components, it is necessary to:

- set and guarantee fluctuation ranges of end–to–end delay and jitter in DTN;
- guarantee the probability of packet loss in the DTN at the level corresponding to the SLA application;
- make the usability of the available bandwidth of DTN channels be maximal (mass overuse of resources is prohibited, such as flooding);
- exclude the unpredictable transmission delays caused by DTN, such as packet delays due to failure of order, retransmission, overload feedback, etc.

Techniques and methods for predicting the execution time of services and applications are discussed in detail in [4] and we will not consider them here.

IV. NPC Functional architecture

The computational infrastructure with properties above, we will call Network Powered Computing (NPC)—it is a software–driven infrastructure, which is a tight software–driven integration of various computers with a high–speed DCN. Such an NPC is a fully manageable, programmable, virtualized infrastructure. In other words, the NPC becomes the computer!

The NPC organization should be based on the federative principle. Each federate has its own administration and possess an independent authority in whose jurisdiction there is a certain amount of computing, telecommunication, storage resources. The federate transfers part of these resources to the Federation, which forms and monitors a unified policy for their use.

Here is a summary of what a functional NPC architecture would look like. Its main layers are the layer of applications, application services and network functions (ASNF layer), layer of NPC infrastructure control (NPCIC): computing, networking, storing, resources layer (NPCR) and E2E orchestration, administration and management layer (OAM) responsible for orchestration, administration and management of NPC infrastructure.

The functionality of the ASNF layer is application representation development: it's code and it's data, it's Application Operation Specification (AOS) representing application services (AS) and virtualized network functions (VNF) necessary for the operation of the application, specification of the data transmission network between application components, formation SLAs for AS based on SLA for the application as a whole.

The NPCIC functionality provides planning and assignment of the application components to NPC resources, in accordance with the AOS and the prediction of the computation time, data transfer, define QoS resource requirements according to the application SLA, collection and aggregation data on the current state of resources; create an overlay network according to AOS (topology, QoS channels, security management).

The NPCR functionality provides a unified representation of the state of heterogeneous resources (computing — cloud, HPC; storage, network), monitoring their current states and predicting their states for the nearest future.

The task of the OAM is to provide orchestration of the interaction of the application components in accordance with the AOS, collection of data on the resources consumption by the application components, security functions in terms of managing the NPC, support for the operation and administration of the NPC.

The basis for building NPCs is formed by the technologies of software–defined networks (SDN) and network functions virtualization (NFV). Taking into account that the scaling range of network functions is huge and works in real time, the NPC will require low time complexity algorithms to optimize resources planning and resource allocation. And given the working speed, as for data transmission networks as for computer installations, it becomes clear that only sub–optimal solutions to the emerging optimization problems will be available based on ML methods.

The functional architecture of the NPC described above is shown in the Fig.1 below.

The functionality of its main layers has already been briefly reviewed above: ASNF, NPCIC, NPCR and OAM. Here we will dwell on the description of their interaction. AOS can have two types of components: network functions for managing data flows (traffic) and application services for data processing and computing services. The first type of components (network functions) is placed either in the DTN control plane (applications of SDN controller) or directly in network devices. Examples are NAT, Firewall, BRASS, balancers, shapers, etc. Components of the second type—application services—are placed in a virtualized form (on virtual machines or containers) or directly on computing resources (servers, Edges data centers of various levels, HPC installations) of federates (in the figure below they are shown in the form of racks).

Please keep in mind that the application programmers is not required to foresee and explicitly insert the necessary network functions or their chains into their application. VNFs can be automatically integrated into AOS applications by means of ASNF, just as compilers or application libraries do, “plug in” the necessary functionality into the application code.

When AOS application services interact through the DTN, the SDN controller of the overlay network “catches” the request for data transfer, accesses the distributed ledger (DL) of the overlay network tunnels to find the proper tunnel. If there is no one in DL, then the SDN controllers apply to the control center for application services in the NPCIC layer. There, with the help of E2E OAM orchestrator (end–to–end orchestrator) and NPCIC data transmission, the wanted tunnel is constructed.

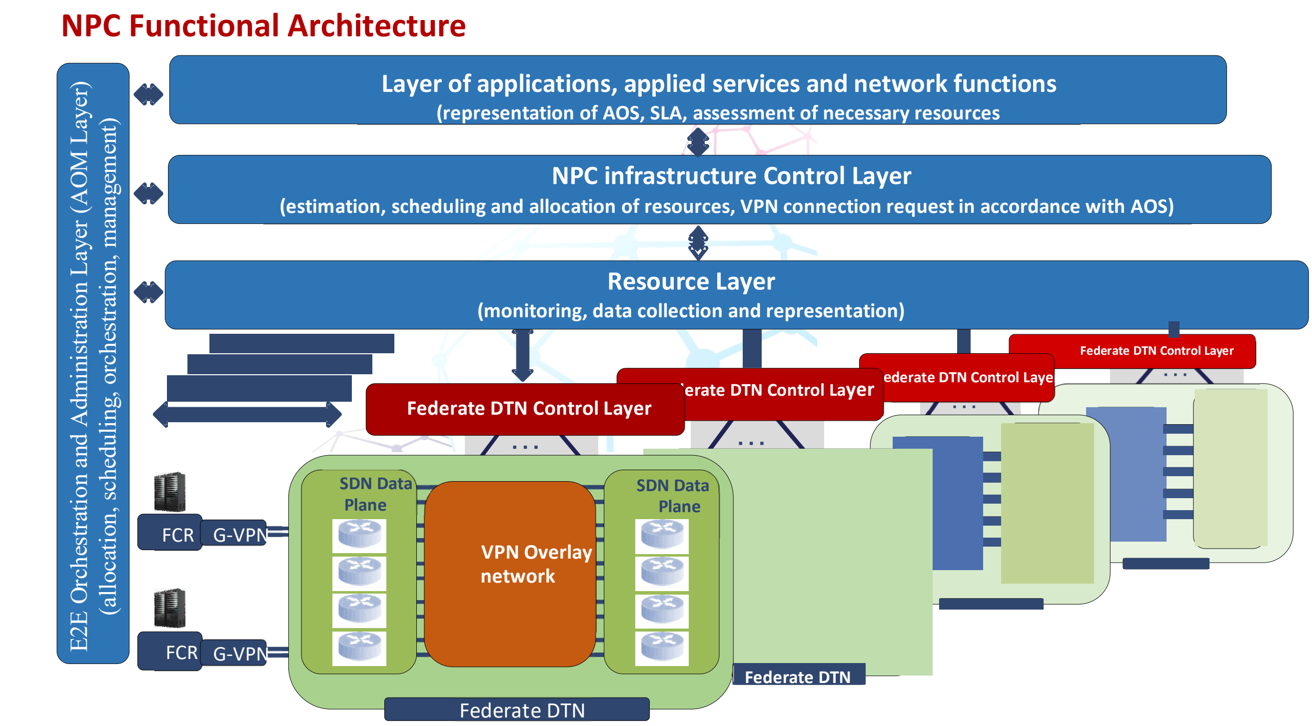

Fig. 1. Functional architecture of Network Powered by Computing

Such NPC can run applications in three modes—pro-active, active, and mixed. In pro–active mode, application services are loaded in advance on the federate's resources in passive state (in the form of code and data on external or internal memory). In this case, when running the application, it is only necessary to activate the required application services according to AOS on those computing resources that will ensure SLA compliance with the specific call to a specific application. It should be clear from the above that different references to the same application may have different SLAs. Same application but with different SLA is treated as different one.

The active mode involves loading the code of the necessary application/computing services in accordance with the AOS on the computing resources of the federates on demand in such a way as to ensure compliance with the SLA of the application. Mixed mode involves a combination of proactive and active modes, i.e. some of the application services and network functions are already pre-installed and are only being activated, the rest ones are loaded and activated upon request.

V. The Major Problems Statements

To meet the above requirements, the NPC must be able to solve a wide range of tasks. For example, in NPCIC and OAM blocks, that essentially form the NPC[1] control circuit, it is necessary to define: method of distribution (distributed vs centralized) of computing resources between threads of computing / application services and requests to them in a given NPC mode of operation (proactive, active or mixed); method for optimal control of data flows in the interaction of computing/application services, method for managing resource monitoring, prediction of the state of overlay network channels, selection of the optimal virtual channel of the overlay network with best QoS to meet the requirements of the SLA of the application, congestion control management, minimization of end–to–end delay, bandwidth monitoring, scheduling flows in queues, etc., scaling the NPC control plane and the data plane when changing the NPC scale, optimal channel routing of the overlay DTN, fair distribution of channel bandwidth, balancing data flow between computing/application services, allocation of channel bandwidth on demand [5]. In the NPCR layer: resource state scan frequency, data presentation format.

Within the framework of the article, it is impossible to cover the entire range of problems that need to be solved in order to implement the formulated concept of the NPC. Here we only present statements for the most important problems:

1. Optimal distribution of the chain of application services on the NPC resources;

2. Distribution of a given set of services for projective mode of operation of the NPC.

Let's denote

$\Gamma = (V, A)$, where

$V = CN \cup SN \cup P$, where

CN – set of NPC computational nodes,

SN – set of NPC VPN gateways,

P – set of Γ poles.

$A = \{(v_i, v_j) \mid v_i, v_j \in V\}$—multiple channels of overlay network.

$Q\ (l_{vi,vj}, \Delta t) \mid = (B, D, L, J)$ is the function defined on A, where

- Δt—interval of time;

[1] The space of the paper does not allow to describe in details NPCIC and OAM interaction.

- $B = (\hat{b}, \overline{b})$—bandwidth of $l_{vi,vj}$ in terms: $\hat{b}$ —average and $\overline{b}$ —maximum on Δt;
- $D = (\underline{d}, \hat{d}, \bar{d})$ – delay on Δt in terms: $\underline{d}$ – minimum, $\widehat{d}$ —average, $\bar{d}$- maximum RTT;
- L —percentage of lost packets;
- $J = \hat{j}, \overline{j}$ —jitter Δt in terms: $\hat{j}$ —average, $\overline{j}$ — maximum.

$CN = \{cn_i = <cr, m, h>\}$, where $cr, m, h \in N$ – set of integers

cr – number of cores (possible with characteristics);

m – amount of RAM;

h – storage size.

$P = \{p_i\}$, where p_i - source of request/application stream, which is characterized by the function of distributing the probability of requests/applications each with its SLA. Consider that the same request/application but with different SLAs are different requests/applications. The same p_i can be a source of requests for different services. Each request is characterized by an application ID and a specific SLA—execution time plus result delivery time.

AS—set of application services, each characterized by the required computing resources, memory resources, storage resources *(cr, m, h)*.

VNF—set of virtual network functions, each characterized by required computing resources, memory resources, storage resources *(cr, m, h)* and which can be found in AOS.

Application service or network function is always allocated to one *cn*.

$W = \{w_i = (s^i_1, \ldots, s^i_k)\}$, where $s^i_j \in AS \cup VNF$, $s^i_j = <cr, m, h>$—chain of application services (SFC);

Let's introduce the function $ET: (AS \cup VNF) \times CN \rightarrow R$ – set of rational numbers, understood as estimation of element execution time from $AS \cup VNF$ on the certain $cn_i \in CN$. This function can be represented as a matrix, where columns correspond to elements from $AS \cup VNF$, and rows correspond to elements from *CN*.

In these terms, the problem of the optimal distribution of SFC on NPC can be formulated as follows:

It is required to construct a mapping $F: \boldsymbol{W} \rightarrow \Gamma$ for a given set P in such a way, that

1. Meet the SLA requirements for all w_i from W with given $p_i \in P$

2. Under the condition of minimizing the objective function, for example, in the following form:

$$F = min \sum_1^{|CN|} \left[\alpha \frac{\overline{c}_i}{c_i} + \beta \frac{\overline{s}_i}{s_i} + \gamma \left(\left(\frac{\overline{c}_i}{c_i} - \Theta \right)^2 + \left(\frac{\overline{s}_i}{s_i} - \Delta \right)^2 \right) \right] \quad (1)$$

where:

α, β, γ—constant values;

c_i, s_i—cn_i resources are used

$\overline{c}_i, \overline{s}_i$— cn_i resources and queue length averaged over usage time;

Θ, Δ—used resources of the entire NPC, averaged over time;

$(cn_i)_w$ is a path in NPC correspond to SLA(w).

It is required to find the distribution of the AOS component in such a way as to minimize the objective function, i.e. in the table representing the *ET* function, you need to add the application service ID in those positions that correspond to the appropriate resources.

The problem of distribution of application services over the resources of the NPC for the proactive mode of operation can be formulated as follows. Let's have given Γ, *AS*, *W* and *P*. It's required to build a matrix *X* with dimension $|X| = |AS| \times |CN|$ where $x_{ij} = 1$, if s_i can be located on cn_j, subject to the following conditions:

1. The constraints of none cn_j and none of $(v_i,v_j) \in A$, incident to cn_j, from Γ are violated;

2. $\forall\ w_i \in W$, SLA applications always be met for any $p_j \in P$.

It is clear that for both problems we must first of all prove for the existence of the solutions and define the conditions for solutions existence.

Given the intensity of request flows in modern networks, the speed of data transfer and computation, it is clear that classical optimization methods will not work to solve the problems listed above. The most appropriate mathematical technique seems to be the multi–agent optimization (MA) technique. The application of this technique is considered in two variants, which are compared with the centralized approach. **Centralized approach** assumes the presence of a control center and that each agent forms its own local state–status. The Control Center collects the status of each agent, makes a decision based on the optimization policy and sends each agent a management impact. Another possible option—is the NPC of the **interconnected agents**. In this case, each agent knows its local state. Information exchange is limited by neighboring agents only. Based on local and neighbor–based information, each agent decides on the optimal strategy. The second option – this is **NPC independent agents**. In this way each agent knows its local state. Each agent judges the management strategy and actions of other agents based on their experience. Agent implements management solutions according to its local optimization strategy and based on its observations.

The size of the control domain, which is understood as a subgraph of the NPC topology, plays a great importance when using MA–approach in control. Experiments with the use of MA–optimization for the router have shown that by adjusting the domain size it is possible to achieve the optimal combination of convergence and quality optimum solution of the routing problem.

VI. Organization of NPC Computing Resources

From the point of view of the organization the computing node (CN) could be as Edges [5] as standalone supercomputer or HPC installation. Existing Data Center construction approaches: high demands to quality of communication channels service, to ensure availability of service; very high capital construction costs of a centralized Data Center. Significant problems of traditional DC are scaling and low level of resource utilization due to the lack of a centralized management system and orchestration system.

The advantages of building a NPC based on Edges over the traditional approach have been discussed in detail in [6] and characterized by: reduction of transport requirements by proximity of the service copy to the final consumer; reducing the cost of organizing a Data Center due to the absence of the need to build a centralized Data Center; efficient scaling through the use of a centralized cloud platform; increasing the efficiency of the network due to a centralized management and orchestration system and the proximity of the service to the client. The problems of organizing the control plane and the data plane in Edges are in many ways similar to those that were already listed above for the NPC Federate DTN Control Layer (see Fig.1). The main difference—the decision–making speed should be much higher.

As is well known, for maximum efficiency of program execution, a specific set of hardware and their configuration is required. Several attempts have already been made to implement the approach of dynamically adjusting the architecture for the application, i.e. see [7]. Currently, to meet this need, it is proposed to use the resource disaggregation approach [8]. Its essence is as follows. Data center use a monolithic server model for more than 20 years, where each server has a motherboard that hosts all types of hardware resources, generally including processor, memory chips, storage devices and network cards. Resource disaggregation involves splitting the server's hardware resources into stand–alone devices connected to the network, to which applications can access remotely. Applications must be provided with virtualized and secure access to hardware resources, and data centers should support these applications with tools that ensure their good performance.

Server in Data Center rack is no longer considered an indivisible resource. The rack consists of a collection of heterogeneous processing hardware: CPU general-purpose, GPU, FPGA co-processors, solid state drivers, disk, tape subsystems, tensor processing, neuromorphic and other types of processors. All this equipment is connected by high-speed NPC, for example, photonic. Today, the switch with a capacity of 26.5 Tbps—is a reality.

Resource disaggregation improves resource efficiency and application performance through more flexible workload distribution and choice of means of data processing tools, that best suits the processing algorithm. The optimal allocation of resources in a disaggregated data center depends on its topology and the placement of workloads.

In a disaggregated Data Center, rack hardware changes must be transparent to virtual machines. In such a disintegrated Data Center hardware infrastructure, the hypervisor's job is to hide all the nuances of working with such heterogeneous hardware and provide a consistent abstraction of resources for virtual machines used by Data Center users. The key enabling factor for disaggregation will be the NPC—to maintain good performance at the application layer, it becomes critically important that the NPC provide low latency communications even as the traffic load increases due to disaggregation.

The article introduced the concept of service-oriented network of data processing and transmission Network Powered by Computing (NPC)—computing technology based on the convergence of data networks with computing tools (DC, Edge, HPC centers). This concept implements the approach of modern computational infrastructure - a network in which the means of data processing and transmission are in close convergence. This network implements a well-known slogan—"The network is a computer" and is based on the analysis of the properties of modern applications and the resulting requirements for computational infrastructure for them. The functional architecture of NPC and the main problems on the way of its implementation are described. It is shown that the modern computational infrastructure is an NPC that integrates heterogeneous disintegrated computing tools connected by a data transmission network with software control. The issue of organizing the ASNF, which, together with the OAM and the NPCIC, is essentially an analogue of the operating environment—an analogue of the traditional operating system. But this is an independent large topic that requires a separate publication.

Acknowledgment

This work is performed within the scientific program of National Center of Physics and Mathematics (the project "National center for supercomputer architecture research"), under the Contract No 96–2022/174 dated Jun 01, 2022 with RFNC–VNIIEF.

References

[1] Topology and Orchestration Specification for Cloud Applications. http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html

[2] Elasticity. https://wa.aws.amazon.com/wellarchitected/2020-07-02T19-33-23/wat.concept.elasticity.en.html

[3] "What Is Serverless Computing?". ITPro Today. 2021–12–13. Retrieved 2022–03–23.

[4] R. Smeliansky, "MC2E: The environment for interdisciplinary research," Engineering Letters, vol. 6, pp. 40–54, 2021.

[5] https://www.youtube.com/watch?v=TuzDDZT4NL4

[6] R. Smeliansky, "Hierarchical edge computing," 2018 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC), 2018, pp. 1–11, doi: 10.1109/MoNeTeC.2018.8572272.

[7] A. V. Kalyaev, Mnogoprocessornye sistemy s programmiruemoj arhitekturoj [Multiprocessor systems with programmable architecture]. Moscow, Russia: Radio i svyaz', 1984.

[8] Yuxin Cheng, Rui Lin, Marilet De Andrade, Lena Wosinska, and Jiajia Chen. Disaggregated Data Centers: Challenges and Tradeoffs, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden, http://kth.diva-portal.org/smash/get/diva2:1292615/FULLTEXT01