MC2E: The Environment for Interdisciplinary Research

R. SMELIANSKY
Computational Mathematics and Cybernetic Department
Moscow State University
Lomonosov Moscow State University, 1 Leninskiye Gory, Moscow, 119991,
RUSSIA

Abstract: - The Meta Cloud Computing Environment (MC2E) project is a Russian-Chinese project dedicated to the study of methods and means of building an informational and computational environment for scientific computational experiments and interdisciplinary research, based on the federal principle of management. Such a federation was supposed to be a specialized, heterogeneous ecosystem of cloud data processing centers (DC), high-performance computer installations united by telecommunication resources. The article provides a brief overview of the main results of this project.

Keywords: High Performance Computing, Supercomputer, Cloud, Data-Center

1 Introduction

Today's interdisciplinary research in various fields of science involves collaboration of multiple research teams, unique scientific instruments and large demands in computational resources. Such collaboration requires an informational, computational and communicational infrastructure specifically tuned for each project. Efforts to create such infrastructure in a traditional way (a local DC with domain-specific software) cause a number of problems:

1. It requires significant financial and material investments, since each new experiment needs specific software adjusted by highly qualified IT-specialists. The problem becomes more complicated if such experiments are performed by independent research teams, because such teams often have different internal business processes, specialize in different subject areas, have their own hardware and software preferences and may be located far from each other;
2. At the initial stage of a project the requirements to the project infrastructure are known roughly and overestimated. This could lead to waste the efficiency of investments;
3. A lot of difficulties arise when scientific data is distributed and used by different teams simultaneously. Data that is needed for one team could be acquired by another. And without a specialized system that manages infrastructure such cases are hard to solve;
4. The groups of researchers from different projects may already have tools, software for processing, collecting and storing data. Creating or mastering new ones for a project is usually unacceptable. Therefore, it is necessary to provide the possibility to bring into the environment already existing developments.

The main instrument for numerical experiments and simulation has been High Performance Computing (HPC). Computational resources for HPC are provided by supercomputers, mainframes and servers clusters. The general trend today is the usage of supercomputers and HPC installations. However, a trend analysis at TOP500.org [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**] suggests that the number of applications is growing faster than the number of supercomputers and HPC installations. At the same time, we can see the rapid growth in the popularity of cloud computing, the usage of Data Centers Networks (DCN) to increase the power of cloud computing platforms. A good example is EGI Association [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**].

These two kind computational platforms have a different computational capability but they also have big differences in their load. Most applications will run faster on a supercomputer than on a server cluster in DC. However, it may turn out that the total delay of the application in the queue plus the execution time on HPC-Supercomputer (HPC-S) would be more than execution time plus waiting time in the queue on HPC cloud server cluster (HPC-C). We will call this total delay as program time in system (PTS criterion).

MC2E project mission was to study how to develop an environment that would allow easy way to create an informational and computational infrastructure, which meet the listed above specifics of a certain interdisciplinary project. One of the actual research topic and still weakly explored problems of this mission is the integration two pretty different HPC environments – supercomputers and DC Clouds. These environments vary in many ways: differences in the level of resource management in the computational environment in use, by the virtualization technique, by the composition of the parameters and the specification form of the request for program execution, by scheduling and resource allocation policy. On-demand clouds could help solve this problem by offering virtualized resources customized for specific purposes. Cloud platform offers more flexibility and convenience for researchers in compare with HPC-S. But in any case the HPC-S and HPC-C platforms heterogeneity makes it hard to switch automatically between them if some platform becomes highly loaded. So in order to change the target platform, researchers need to spend time and resources adjusting their software for the new API.

Another problem on the way to the integration of HPC-S and the HPC-C is how to choose the proper computer installation for MPI program execution in heterogeneous integrated environment: HPC-S or HPC-C? Other words for every MPI program in the queue of the environment make a decision where it should be executed more effectively in term of PTS criterion in the current resource amount/configuration. On the way of solution to this integration problem we need to justify the hypothesis that the sharing of physical resources by several MPI programs (several tenants) in HPC-C environment will reduce the total execution time of these programs, i.e. this time will be less than the sum of the execution times of each program separately.

One more problem is the ability of the environment aggregates the resources of DC network (DCN). At this point the key problem is feasibility the Bandwidth-on-Demand (BoD) service problem. This service should develop/allocate, under request, the channels between two or more DC with the appropriate QoS parameters and total throughput for transmitting specified amount of data at particular interval of time through TCP/IP transport network. It should be emphasized that such a service does not imply a dedicated channel between the interacting parties. Moreover, it should be dynamically created through the aggregation of existing network resources.

Here the overview the main results of MC2E project are presented in the following structure. We will shortly present the architecture of the MC2E environment (section 2); the experimental exploring of DC network influence on CPU utilization in the clouds and on resources sharing ability for MPI programs (section 3); new approaches to the prediction of the MPI programs execution time on a certain set of computer installations to determine a proper choice of order and place for program execution (section 4); and approaches to BoD service development (section 5).

2 MC2E Architecture

The MC2E environment architecture is based on the following principles:

1. The infrastructure is a federation of locations called federates with local computing, storing and networking resources. Federation administrates all resources (CPU, memory, network, software) provided by federates;
2. Resources of the same federate can be shared between different projects simultaneously;
3. All physical resources are virtualized;
4. Resources on a user level have a high level of abstraction. Usage of such resources should not require strong qualification from system administrator;
5. Experiments' results could be saved. Saved results could be used by other research teams to reproduce or continue the experiment;
6. The federation provides data processing as a virtualized service.

On the figure 1 the MC2E architecture is layout with responsibilities allocation between the international project participants.

An example of the federate computational resource could be an HPC cluster, DC, supercomputer under unique administration. And each federate has its own policy which regulates federate resource allocation to the users. Infrastructures that are built as federation of heterogeneous computational resources are already used in many existing projects. Several such projects are designed to perform experiments in computer networking. For example, the GENI project (Global Environment for Network Innovations) [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**] that was initiated by the US National Scientific Foundation (NSF) is a virtual laboratory aimed to provide an environment for networking experiments on an international scale. Today more than 200 US universities contribute to the GENI project. Another project which supported by NSF is FABRIC [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**]. FABRIC is an adaptive programmable research infrastructure for computer science and science applications. Similar but less known projects are Ophelia [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**] (supported by the EU) and Fed4Fire [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**] (supported by 17 companies from 8 countries).

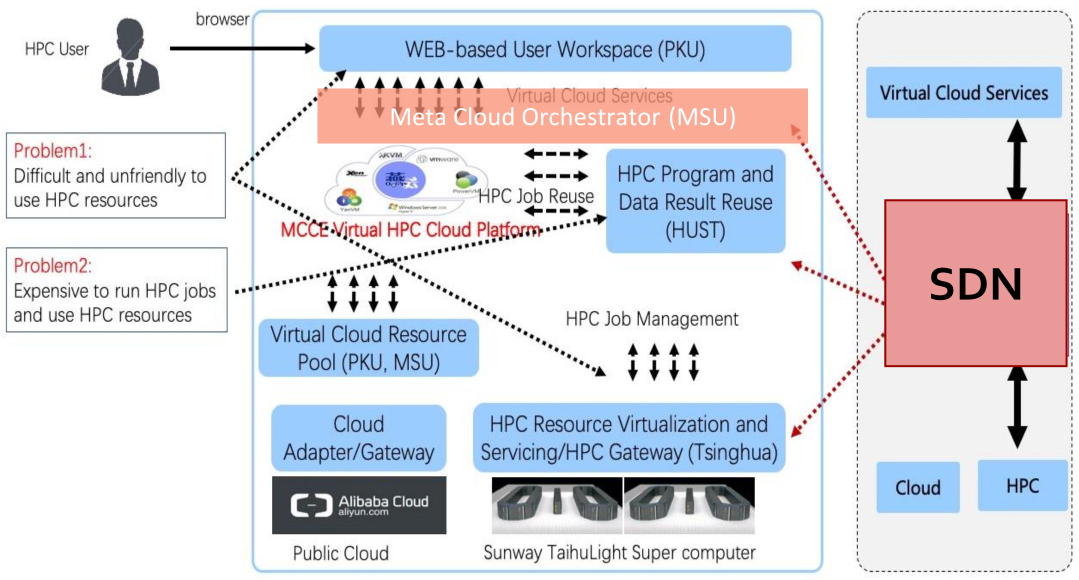

HPC User
browser
WEB-based User Workspace (PKU)
Virtual Cloud Services
Meta Cloud Orchestrator (MSU)
Problem1: Difficult and unfriendly to use HPC resources
Problem2: Expensive to run HPC jobs and use HPC resources
HPC Job Reuse
HPC Program and Data Result Reuse (HUST)
MCCE Virtual HPC Cloud Platform
HPC Job Management
Virtual Cloud Resource Pool (PKU, MSU)
Cloud Adapter/Gateway
HPC Resource Virtualization and Servicing/HPC Gateway (Tsinghua)
Alibaba Cloud aliyun.com
Public Cloud
Sunway TaihuLight Super computer
Virtual Cloud Services
SDN
Cloud
HPC

Fig. 1. MC2E Architecture

There are some other ones provide environments for computational experiments regardless of their domain [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**], [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**], [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**].

However, these projects have several following drawbacks:

1. Weak integration between HPC-S and HPC-C, which does not allow you to automatically select an effective computational resource from the available;
2. Drawbacks in service chaining to perform experiments and bring already existed services/tools into the new environment;
3. Resource planning doesn't take into account possible services scaling;
4. The lack of DCN control and management services like monitoring and BoD services.

The basic technologies for MC2E project to develop a virtual infrastructure for interdisciplinary research project as a flexible and economically effective way were Software-Defined Networking (SDN) and Network Function Virtualization (NFV). These technologies allow increase the level of resource abstraction, enable coordinated resource optimization and automatize infrastructure management. Instead of providing individual resources, users receive complete virtual infrastructures (computational power, communication channels and storage systems) with guaranteed performance and QoS based on the service level agreement (SLA).

Actually there were two cloud environments used for experiments in MC2E project: Docklet [10] and Cloud Conductor (C2) [11]. The goal of Docklet is to provide Personal Development Workspace in the Cloud solution [53]. It covers all the SaaS, PaaS and IaaS layer of cloud computing architecture. The basic of Docklet is LXC vcluster [54], but not Docker container. For Docklet users, what they face directly is their Workspace. They use browser to do software development, debugging and testing, etc. using tools Docklet provide, working in a high layer. With the help of Docklet, research groups can easily virtualize their small scale data centers, creating virtualized clusters, and then providing users a customizable Workspace in the cloud. Users only need a modern browser to visit their own Workspace located in the enterprise's Intranet from anywhere, at any time. They can do works like online editing source codes, debugging, testing, managing data files, analyzing data, visualizing results, etc.

The C2 Platform architecture is based on the reference implementation ETSI NFV MANO model and provides the full VNF life-cycle support: initialization, configuration, execution and deinitialization. This description, called TOSCA-template [11], is all that needed to set a VF. TOSCA template includes the structure of a cloud application, application management policies, OS image and scripts to start, to stop and to configure the application that implements the VF. The C2 platform assumes that a cloud administrator should provide the TOSCA-template as a zip or tar archive with a predetermined structure.

The proposed federation-based environment has the following advantages:

1. Easy locating, setup and scaling resources across a variety of services in minutes;
2. Developing application as a chain of multiple services based on NFV technology;
3. Merging infrastructures from different research teams and adjust access policies;
4. Automated resource planning to perform user requests based on access policies and SLA;
5. Extensive application description environment, that allows to abstract away low-level system details;
6. A decentralized resource accounting system for settlements between project participants;
7. Wide possibilities for experiments tracing and monitoring;

8. Increased efficiency of network virtualization with SDN, that allows to adjust virtualized network channels for each particular experiment;
9. Common specification language that is necessary for transferring existed research software into MC2E environment.

The main MC2E architecture components or subsystems are listed below:

1. Meta-Cloud – orchestrate user applications, allocate and schedule them between federates;
2. Interface – provides a unified API for users to submit their applications and for federate administrator to manage and control the resources of the federate;
3. Networking – regulates network resource usage and provides Capacity-on-Demand service;
4. Monitor – performs resource monitoring and clearance for all federates in MC2E;
5. Quality of Service, Administration Control and Management – enforces resource usage policy, provides QoS based on user requirements and guarantees resource reliability.

These components are described in details in [12] General MC2E workflow looks as follows:

1. By unified MC2E interface a user send his application and data to the front-end server;
2. The front-end server invokes Meta-Cloud scheduler and monitor to choose a federate for application execution;
3. Meta-Cloud analyze the queue and predicts application execution time and data transmission time for all available federates;
4. Based on the prediction Meta-Cloud chooses the federate that will minimize the total of application in system (data transmission time + queue waiting time + execution time);
5. Meta-Cloud call the Networking for channel development to the destination federate and sends application and its data;
6. Federate executes the application and returns results to the user;
7. In the case of a federate failure, an application will be migrated by Meta-Cloud QoS to another federate.

The following resources were used for the experiments in MC2E project:

1. Supercomputer Lomonosov-2 [13];
2. HPC computer installation Polus - IBM Power 8 [14];
3. HPC computer installation BlueGene – IBM Power PC [15];
4. HPC computer installation MBC-10П Tornado [16];
5. mini DC: head server – Intel Xeon CPU E5-2650 v4 @ 2.20GHz with 48 cores with 64 Gb RAM and 6 workers – Intel Xeon CPU E5-2667 v4 @ 3.20GHz with 16 cores with 32 Gb RAM, HD – 3.5 TB, Ubuntu 18.04.4 LTS; each physical link had 10 Gbps;
6. DC Peking University [17];
7. VMs (4 cores, 16 GB) in Alibaba Cloud DC.

3 Cloud as HPC environment

Clouds are less powerful than specialized server clusters or supercomputers [18]. Nevertheless they are becoming more popular as a platform for HPC due to the low cost and easy to access. Several papers [19], [20] have shown that one of the main performance bottlenecks in HPC-clouds issues from communication delays within the DC network. While supercomputers use fast interconnections like [21], [22] HPC-clouds mostly rely on usual Ethernet networks. This performance bottleneck could also lead to CPU underutilization with network-intensive applications, since such applications may spend a lot of time waiting for their messages to pass through the network.

One of the important results of MC2E project is the verification and justification of the hypothesis that network-intensive HPC-applications could share CPU cores among each other with negligible performance degradation. Such behaviour could be used to improve CPU utilization and to increase the effectiveness of

HPC-application execution. The hypothesis was validated in a cloud environment with HPC benchmark – NAS Parallel Benchmarks (NPB) [23] on mini-DC (the parameters see above) with QEMU/KVM hypervisor 64 virtual machines (VMs) (Ubuntu 16.04, 1 vCPU, 1024 Mb RAM). MPI version was 3.2. Head server contained 16 VMs, other servers contained 8 VMs per each. Average RTT between different VMs was near 400 µs. Bandwidth between VMs was at the same server – 18.2 Gbps, on different servers – 5.86 Gbps.

In this experiment, the network bandwidth influence on CPU utilization was explored. Sequentially 5 NPB MPI programs with 2, 4, 8, 16, 32, 64 MPI processes on separate VM were run on network with three bandwidths: 100 Mbps, 1Gbps, 10Gbps. As it should be seen on Fig. 2 when the number of MPI processes increases, CPU usage drops, because different MPI processes run on different virtual machines and data is transferred over the network between the different physical servers and so the delay increases. Also, CPU usage drops when MPI program run in one physical server (2, 4 and 8 CPUs). This CPU usage decrease allows share the same CPU between different MPI programs.

In another experiments series the ability different HPC-applications to share CPU cores was explored. The experiment methodology was as follow: sequentially 5 pairs of NPB MPI programs (each pair contained two identical programs) on (2, 4, 8, 16, 32, 64) VMs (N MPI processes in each MPI program). To evaluate the ability MPI programs to share CPU resources the queue metric, see Fig. 3, was used, where pure time is execution time without resources sharing; sharing time is execution time when two MPI programs use the same CPUs and cores. It is not hard to see if value of queue metrics is more than 1 therefore two programs run simultaneously take less time to complete than in sequential order. According to the Fig. 3 even in the cloud with slow network (100 Mbps) we can get up to 20 percent execution time acceleration.

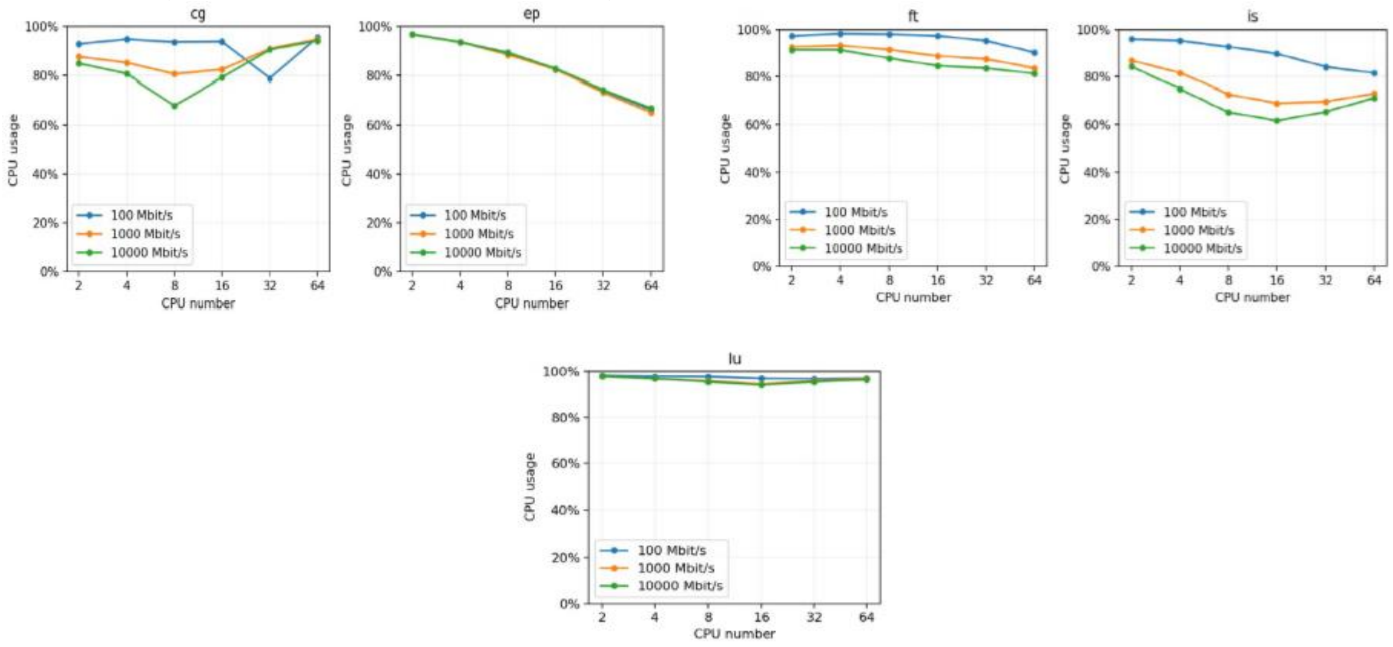
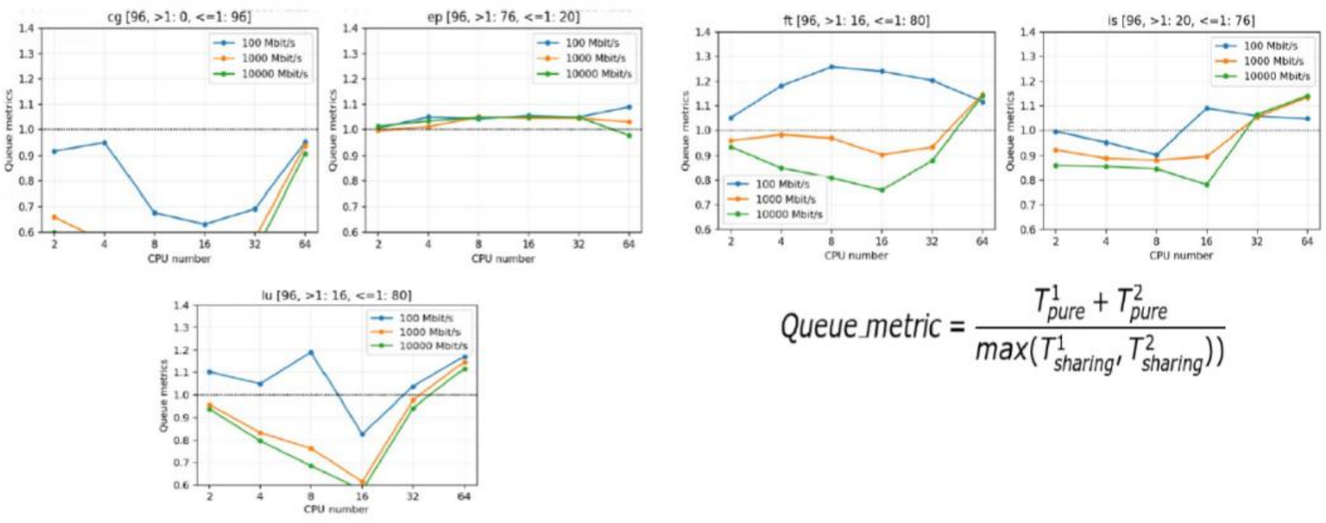


Fig. 2. CPU utilization for NPB

However as it shown on Fig.3 not all MPI programs can effectively share resources with other MPI programs.



$$Queue\_metric = \frac{T^1_{pure} + T^2_{pure}}{max(T^1_{sharing}, T^2_{sharing})}$$

Fig. 1. Queue metric

4 MPI program execution time prediction

As it was already said one of the main criteria for the effectiveness of a cloud computing environment for HPC applications is the time spending by MPI program in this environment (PTS criterion). This time consists of the time spent by program in the queue (waiting time) and the program execution time (execution time). These values depend on the resources allocation algorithms of the cloud environment (CE) (mapping virtual computer installations to physical ones) and the queue discipline, taking into account the heterogeneity of physical computer installations.

Look at the «path» of a program from the coming in the CE up to getting the result of its execution:

1. User forms a set of input data: the text of the program, initial values of the program input parameters, program running resources like number of MPI processes or the requested resources (CPU, RAM), a special script to compile program;
2. User send a set of input data to the CE through a single portal of the federation. The input data from the portal come to the orchestrator of the CE, which is responsible for implementing the virtual infrastructure and executing the programs in it;
3. The orchestrator responsible for federation resources allocation and has a unique algorithm to do this. This algorithm determines the best order and the best federate and physical computer installation inside federate for program execution, in sense of the PTS effectiveness criterion. When computer installation is selected this program is sent to there;
4. Each federate has its own queue of programs. The arrived program is processed by the local resources scheduler and will start running at a certain time defined by the local scheduler;
5. When the program execution is completed the results is returned to the orchestrator which in its term sends it to the users in the proper form.

From the path above, it can be seen that step 3 (meta-scheduling) and step 4 (local scheduling) are optimization points by criterion PTS. The important part of MC2E project was the research and development the algorithm to choose the most effective computer installation in the federation for the received MPI program based on the minimum execution time criterion. Within MC2E project the algorithms for program execution time prediction on a certain set of computer installations based on the execution history of the program on different computer installations were developed (detailed description of the algorithms see [24]).

The problem of predicting the execution time of a program on a computer is well known and is a classical one. For example program execution time on the specific computer installation, as well as the waiting time in the queue, can be predicted based on the histories of its running on this computer installation [25], [26], [27], [28]. Many extrapolation algorithms can be proposed for that like [29], [30], or regression [31], or more complex algorithms like the ensemble of decision trees (Random Forest) [32]. The main disadvantage of these algorithms is that they applicable only to the same computer installation. But the point of the question in MC2E was a program execution time prediction on a certain set of computer installations. These are the computer installations whose characteristics meet the requirements of the virtual infrastructure. Of course, the algorithms mentioned above can be used to estimates programs execution time on the several computer installations. However, this requires a history of running of this program on each computer installation from the set. It is unlikely this information will be available.

The logic for choice of computer installation based on the histories of program execution can be described as follows. One of the well-known algorithms [25], [26], [27], [28] is applied to program execution histories to estimate program execution time on each of the computer installation from a certain set of computer installations. One of the widely used the form of such history could be a trace of program execution [33]. Based on the estimates, one can either developing scheduling for a group of programs, or follow a greedy strategy and send each program to the computer installation that has the minimum execution time. However, the usage any of the above approaches, we need all the execution histories of all programs on each of the computers in the set under consideration.

One of the important and new results of the MC2E project is the development the MPI program execution time prediction method which allows relaxing this requirement: to predict the program execution time on several computer installations, only some running histories of this program on them are sufficient (the detailed description of the method see [24]). Moreover, it is not necessary to run each program on each computer installation.

The main idea of proposed new approach to MPI program execution time prediction problem is based on the association that the problem under consideration is very similar to the problem solved in the recommendation systems or recommender system [34]. A recommender system (RS), or a recommendation system (sometimes replacing 'system' with a synonym such as platform or engine), is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item [35]. The examples of the items could be movies, books or any other goods. So recommender system restore, predict the relationships between users and items based on some user estimations, preferences.

RS system has a rating matrix where the rows (or columns) correspond to movies, books, or goods, and the columns (or rows) correspond to users. This matrix is often sparse, since there are a lot of users and items, and users can't physically evaluate all the items under consideration. The RS system tries to predict the preferences of each user for each item, based on individual user ratings for some items. So, in other words, RS system has to fill in the empty entries in the rating matrix. In these terms consider the following analogy: users are the computer installations, items are the programs, and user ratings are execution times. Thus, computer installations “evaluate” programs and the smaller the rating (execution time), the better.

In the proposed approaches, the problem of the program execution time prediction is reduced to the problem of filling empty entries in the matrix “Programs-Computers” built for a given set of programs and a given set of computer installations. In the entries of this matrix there is an execution time a certain program with certain sets of arguments corresponding to a specific computer installation.

It should be clear that accuracy of the prediction depends on the number of program running histories on computer installations from a certain set. There were proposed two approaches to the program execution time prediction problem on a certain set of computer installations. The first one is based on computer installations grouping based on the Pearson correlation [36]. It was shown that this method reasonably apply in the case of a "densely" (dense matrix) filled matrix «Programs-Computers» (at least 95% of entries are filled).

The second one was developed for the sparse «Programs-Computers» matrix and relies on decomposition of that matrix into vector representations of computer installations and programs, so-called embeddings. An embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embeddings make it easier to do machine learning on large inputs like sparse vectors representing words. Ideally, an embedding captures some of the semantics of the input by placing semantically similar inputs close together in the embedding space [37]. In the paper [24] it is shown how to use embedding of program and embedding of computer installation to predict the program execution time on specific computer installation. There was explored the decomposition technique [38] to the matrix «Programs-Computers» to calculate embeddings.

Also, there was considered ensembles from algorithms Ridge regression, grouping computer installations based on Pearson correlation, and matrix decomposition in MC2E project. It was shown that the ensemble of algorithms is more resistant to outliers than other algorithms and gives the best results on dense matrices. All proposed algorithms have been deeply experimentally explored on MPI benchmarks and OpenMP benchmarks [39], [40] (see Table 1).

Experiments with testing data from Table 1 showed that an ensemble of algorithms with a small percentage of empty entries in «Program-Computer» matrix – up to 52% – makes a better prediction compared to all other algorithms. Also, accordingly to the experiments, the ensemble and ALS show good results even in the presence of outliers in the source testing data sets. In case that percentage of empty entries is much more than 50%, ALS algorithm demonstrated the best prediction. Thus, for dense matrices, it is better to use an ensemble of algorithms, for sparse ones-matrix decomposition, in particular the ALS algorithm.

It is important to emphasize that the proposed approach to predicting program execution time requires a minimal set of data about the program, which is usually available on all modern computer installations. Another important advantage of the proposed approach is that a result of matrix decomposition is the embeddings of Programs and computer installations of dimension 1. This fact allows one set up the total order as on a set computer installations as on a set of programs what significantly help to properly select the computer installation with effective execution time. For the more data about prediction methods testing see [24].

Name	**Number of computer installations**	**Number of programs**	**Benchmark type**
MPIL2007	163	12	MPI [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**]
MPIM2007	396	13	MPI [**Σφάλμα! Το αρχείο**

			προέλευσης της αναφοράς δεν βρέθηκε.]
ACCEL_OMP	25	15	OpenMP [**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**]

Table 1. Data for testing algorithms

5 Bandwidth on Demand service

As it was mentioned in the introduction one of the problems was studied in MC2E project was the ability of HPC-C environment aggregates the resources of DCN. The environment intended for interdisciplinary scientific research projects has to have very flexible and powerful network resource allocation, scheduling and administration mechanisms. Otherwise, the overhead for DCN network resources should be very high and the load on the DCN network resource will be very volatile. At this point the key problem is feasibility the Bandwidth-on-Demand (BoD) service problem. This service should develop/allocate, under request, the channels between two or more DC with the appropriate QoS parameters and total throughput to transmity a specified amount of data at particular interval of time through TCP/IP transport network. It should be emphasized that such a service does not imply a dedicated channel between the interacting parties. Moreover, it should be dynamically created through the aggregation of existing network resources. It was implied that a request for BoD service is possible under the contract between the user and network carrier.

Two kinds of user flows were under consideration in the project. The background flows – the user flows that duration is significantly greater than the duration of flows resulting from BoD service request and occupy all period of observation. The urgent flow – the user flow for which BoD service request was made.

In the paper [41] the problem BoD service development is considered in details. Here the reasoning layout of the approach proposed in the cited paper is presented. The implementation of BoD services can be divided into two components: route aggregation and fair distribution of user traffic flows among these routes accordingly to the SLA of the contract.

Under the term “route aggregation” it means a service that can use for a user data flow transmission between a pair of DCs several different routes in data communication network (further just network) at the same time. The task of route aggregation does not address the issue of SLA quality of service that what BoD should do. To implement route aggregation, several problems have to be solved.

The first one is how many and what routes should be aggregated to meet the BoD service SLA. It should be clear that the aggregated routes have to have the minimum intersections. This constraint comes from the specifics of the operation of congestion control algorithms on network transport level. The number of edge-disjoint paths between two vertices in a graph is defined by the Menger theorem [42]. This theorem states that the largest number of edge-disjoint routes from vertex u to vertex v in the graph is equal to the smallest number of edges in the <u,v> cut of that graph.

The absence of the route intersection is not always a critical point. For example, if the physical links on the intersections have a sufficient available bandwidth, than there is no bottleneck. However, it is possible that there are no alternative disjoint routes between source and destination in the network topology. In this case, the problem can be reduced to the previous one by transforming the graph of the network topology in such a way that the edge corresponding to the physical link with a high bandwidth is replaced by several edges with a lower bandwidth. An alternative solution could also be to search for routes with the least number of intersections, as MCMF [43] does.

After find the value k – the number of disjoint routes, the network topology graph can be processed by the special algorithm to identify k routes between the source and the destination. It was found that not any algorithm is suitable for this purpose. For example, the greedy algorithm [44] will not be solution of the problem. The proper choice is MCMF [45], which reduces original problem to the maximum flow in the network problem. Follow this way the set of routes with sufficient available resources to provide the BoD service will be identified in a network topology.

The next problem is how to use simultaneously the resources of these routes to transmit user flow, i.e. to do the route aggregation. There are a large number of protocols and technologies that can help with this problem. In [41] following the 5-tier TCP/IP model, these protocols were considered and the main requirements for them were formulated. At any of these levels, you have to deal with multipath protocols, which allow you to divide a single application flow on the several transport subflows each uses a single path.

There are two basic approaches to multipath routing: static and dynamic. The MPTCP is a static approach [45] involving a priori allocation of a certain number of transport subflows among which data stream segments are distributed. The dynamic approach e.g. FDMP [46] involves the dynamic allocation of a subflow at the request of a transport agent, depending on the correspondence of the total allocated subflows throughput to the application demand.

At the network level, transport flow balancing techniques such as ECMP [47], MPLS-TE [48] together with the RSVP resource reservation protocol [49] can be applied. However, ECMP has one constraint: the routes should have the same cost (e.g. have the same lengths in case of hop count metric), that is not true in general case for k disjoint routes discussed above. Therefore, to balance flows, it is more profitable to look towards unequal-cost multipath (UCMP), where route cost can be varied.

In the case of the link layer, the main constraint for all link layer aggregation protocols is to use only those routes that pass through the same network devices. Most network equipment for working with Ethernet networks supports both static configuration of link aggregation and dynamic control like the LACP, PAgP protocols [50]. And again as at network level the problem flows' distribution problem arises. Similar to LACP channel aggregation techniques can be found for other types of networks, although they can have a completely different physical nature.

After the route aggregation the problem of Application Flow Load Distribution (AFLD problem) among identified routes was considered. The AFLD problem was divided on resource estimation problem and resource distribution problem. The solution of the first one brings us the answer whether there is enough available capacity on the identified routs to meet the SLA BoD contract? If it is so the second one comes – how application flow load should be distributed between these routes?

To solve this problem, a mathematical model was developed for the simultaneous transmission of data for a given set of contracts over several routes. Based on this model, the AFLD problem was formulated as discrete-time integer linear programming problem (ILP) [12]. Solution AFLD problem in ILP form gave the answers for the following questions: is there enough available capacity on the identified routes to accommodate all urgent flows? What urgent flows can be accommodated? How much of the available capacity on the identified routes each accommodated urgent flow can occupy?

On the basis of Juniper VMX routers, a prototype of the above approaches to the implementation of BoD service based on VPN technology was built, which was successfully tested between DC at Moscow State University and DC of Peking University. In April 2021, a pilot project for the implementation of BoD service between data centers in Moscow and Novosibirsk was successfully completed [55].

6 Conclusion

The main results of MC2E international project are presented. This project was targeting on study of the development an environment for academic interdisciplinary research. MC2E is built as a federation of local computing units called federates. Each federate can consist on an HPC cluster, DC, a supercomputer, a scientific installations covered by data communication network. The advantages of MC2E include:

- High level of resource control and flexible capabilities to define virtual environments;
- High quality of resource scheduling and utilization providing efficient scientific services by PST criterion ;
- It relieves a user from tedious system administration tasks and it also specifies a unified way to describe a DC (or an HPC cluster) service life cycle.

MC2E can be applied in different areas, such as educational activities of research institutes and universities, interdisciplinary research, international research collaboration, increasing resource utilization in DCs, popularizing supercomputer usage in different research areas, shared data usage by multiple organizations.

Beside listed above this research presents the experimental justification the hypothesis that you can get acceleration of MPI programs executions time when you run in the cloud several MPI programs simultaneously. The experiments demonstrated up to 20 percent execution time acceleration.

The problem solution was developed how to choose the proper computer installation for MPI program execution in heterogeneous environment like MC2E project based on PST criterion. For that a new approach to predict the MPI program execution time on a certain computer installation was proposed, even it was not executed on it. Two algorithms were constructed and analyzed: an algorithm based on computer installations grouping based on the Pearson correlation (for dense Program-Computer matrix) and an algorithm based on matrix decomposition technique, which allows to obtain vector representations (embeddings) of the Program and computer installations (for spare Program-Computer matrix).

It is important to emphasize that the proposed approach to predicting program execution time requires a minimal set of data about the program, which is usually available. Another important advantage of the proposed approach is that a result of matrix decomposition is the embeddings of MPI programs and computer installations have dimension 1. This fact allows one set up the total order as on a set computer installations as on a set of programs what significantly help to properly select the computer installation with effective execution time. As a hypothesis it was proposed in the project that execution time prediction techniques one can apply not only to MPI programs.

The approach to the BoD service development was proposed. This approach was divided into the route aggregation problem and the flow load distribution problem. Various implementation options for the route aggregation problem were analyzed according to the network parameters, the desires of the service provider and the capabilities of the network equipment. The problem flow load distribution among aggregated routes was solved in the form as ILP problems.

It would be naive to believe that the presented results fully cover the solutions to all the problems that arise when creating such environments. For example, the automation of programs lunching for execution on different computer installations in the environment integrating HPC-C and HPC-S resources, the coordinated management of data and network resources in real time, analytics, management and security in such environments [51], the problems of accounting for consumed resources, mutual settlements between federation members, the use of edge computation technology [52].

Acknowledgemen

The author would like to express his deep appreciation to all members of MC2E project joint force:

Lomonosov Moscow State University (Russia): Vitaly Antonenko - senior research fellow; Anatoly Bahmurov – senior research fellow, Ivan Petrov, Andrew Chupakhin, Alexey Kolosov – Ph.D. students; Gleb Ishelev – master student;
Peking University (China): Xiangqun Chen – professor (Principal Investigator); Mei Hong – Professor; Donggang Cao - research professor, Junming Ma – Ph.D student;
Tsinghua University (China): Wenlai Zhao - researcher;
Huazhong University of Science & Technology (China): Min Chen - professor;

This work is supported by Russian Ministry of Science and Higher Education, grant #05.613.21.0088, unique ID RFMEFI61318X0088 and the National Key R&D Program of China (2017YFE0123600).

References

[1] Meuer, H., et al. "The Top500 project." URL: http://www.top500.org/ (2019).
[2] Kranzlmüller, D., de Lucas, J. M., & Öster, P. (2010). The european grid initiative (EGI). In Remote instrumentation and virtual laboratories (pp. 61-66). Springer, Boston, MA.
[3] Hwang, T. (2017, March). NSF GENI cloud enabled architecture for distributed scientific computing. In 2017 IEEE Aerospace Conference (pp. 1-8). IEEE.
[4] Baldin, Ilya and Nikolich, Anita and Griffioen, James and Monga, Indermohan and Wang, Kuang-Ching and Lehman, Tom and Ruth, Paul. "FABRIC: A National-Scale Programmable Experimental Network Infrastructure," IEEE Internet Computing, Vol.23, 2020
[5] Dewar, R. G., MacKinnon, L. M., Pooley, R. J., Smith, A. D., Smith, M. J., & Wilcox, P. A. (2002). The OPHELIA Project: Supporting Software Development in a Distributed Environment. In ICWI (pp. 568-571).
[6] Fed4Fire project: https://www.fed4fire.eu/the-project/ (2019)
[7] Grossman, R. L., Gu, Y., Mambretti, J., Sabala, M., Szalay, A., & White, K. (2010, June). An overview of the open science data cloud. Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (pp. 377-384). ACM.
[8] Bal, H. E., Bhoedjang, R., Hofman, R., Jacobs, C., Langendoen, K., Rühl, T., & Kaashoek, M. F. (1998). Performance evaluation of the Orca shared-object system. ACM Transactions on Computer Systems (TOCS), 16(1), pp. 1-40.
[9] Brun, R., Urban, L., Carminati, F., Giani, S., Maire, M., McPherson, A., ... & Patrick, G. (1993). GEANT: detector description and simulation tool (No. CERN-W-5013). CERN.
[10] Donggang Cao, Bo An, Peichang Shi, Huaimin Wang, Providing Virtual Cloud for Special Purposes on Demand in JointCloud Computing Environment, *JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY*, 32(2):211-218, Mar 2017
[11] Antonenko Vitaly, Smeliansky Ruslan, Ermilov Alexander, Plakunov Artem, Pinaeva Nadezhda, Romanov Andrey C2: General Purpose Cloud Platform with NFV Life-cycle Management// 2017 IEEE 9th International Conference on Cloud Computing Technology and Science, pp. 353-356
[12] Antonenko Vitaly, Chupakhin Andrey, Kolosov Alexey, Smeliansky Ruslan, Stepanov Evgeniy On HPC & Cloud Environments Integration. In G.Bocewicz, J.Pemoera, V.Toporkov (eds) Performance Evaluation Models for Distributed Service Networks. Springer. 2020
[13] https://parallel.ru/cluster/lomonosov2.html
[14] http://hpc.cmc.msu.ru/polus
[15] http://hpc.cmc.msu.ru/
[16] http://www.jscc.ru/resources/hpc/
[17] https://iwork.pku.edu.cn/
[18] Netto, M. A., Calheiros, R. N., Rodrigues, E. R., Cunha, R. L., & Buyya, R. (2018). HPC cloud for scientific and business applications: Taxonomy, vision, and research challenges. ACM Computing Surveys (CSUR), 51(1), 8.
[19] Gupta, A., Faraboschi, P., Gioachin, F., Kale, L. V., Kaufmann, R., Lee, B. S., ... & Suen, C. H. (2016). Evaluating and improving the performance and scheduling of HPC applications in cloud. IEEE Transactions on Cloud Computing, 4(3), pp. 307-321.

[20] Gupta, A., & Milojicic, D. (2011, October). Evaluation of hpc applications on cloud. In 2011 Sixth Open Cirrus Summit (pp. 22-26). IEEE.

[21] Infiniband in supercomputer systems. https://www.businesswire.com/news/home/20181112005379/en/Mellanox-InfiniBand-Ethernet-Solutions-Accelerate-Majority-TOP500

[22] Gigabit Ethernet in supercomputer systems. https://www.mellanox.com/solutions/high-performance-computing/top500.php

[23] NAS Parallel Benchmarks. https://www.nas.nasa.gov/publications/npb.html

[24] A. Chupakhin, A. Kolosov, A. Bahmurov, V. Antonenko, G. Ishelev. "Application of recommender systems approaches to the MPI program execution time prediction". Proc. of 3rd International Conference "Modern Network Technologies-2020" (MoNeTec-2020), Moscow, Oct. 27-29, 2020.

[25] Gibbons, R. A historical application profiler for use by parallel schedulers. *In:* Proceedings of the Job Scheduling Strategies for Parallel Processing. Springer (1997),

[26] Kapadia, N.H., Fortes, J.A., Brodley, C.E.: Predictive application performance modeling in a computational grid environment. Proceedings of the Eighth International Symposium on High Performance Distributed Computing, pp. 47–54. IEEE (1999),

[27] Li, H., Groep, D., Templon, J., Wolters, L.: Predicting job start times on clusters. In: CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid, 2004, pp. 301–308,

[28] Mohr, B., Wolf, F.: Kojak—a tool set for automatic performance analysis of parallel programs. In: Euro-Par 2003 Parallel Processing, pp. 1301–1304. Springer (2003)

[29] Iverson, M.A., Özgüner, F., Potter, L.: Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. IEEE Trans. Comput. 48(12), 1999, pp. 1374–1379

[30] Liu, X., Chen, J., Liu, K., Yang, Y.: Forecasting duration intervals of scientific workflow activities based on time-series patterns. In: Proceedings of the IEEE Fourth International Conference on eScience, 2008, eScience '08, pp. 23–30 (2008)

[31] Ridge Regression[Online] Available: https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Ridge_Regression.pdf [Accessed: 21-Jun-2020]

[32] Random forest algorithm [Online] Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm [Accessed: 21-Jun-2020]

[33] Smeliansky R. L. Model of distributed computing system operation with time doi: 10.1134/s0361768813050046 // *Programming and Computer Software*. — 2013. — Vol. 39, no. 5. — pp. 223–241

[34] https://www.coursera.org/specializations/recommender-systems

[35] https://en.wikipedia.org/wiki/Recommender_system#:~:text=A%20recommender%20system%2C%20or%20a,would%20give%20to%20an%20item

[36] Pearson's Correlation Coefficient [Online] Available: https://link.springer.com/referenceworkentry/10.1007%2F978-1-4020-5614-7_2569 [Accessed: 21-Jun-2020]

[37] https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture

[38] Cheng CM., Jin XQ. (2018) Matrix Decomposition. In: Alhajj R., Rokne J. (eds) Encyclopedia of Social Network Analysis and Mining. Springer, New York, NY

[39] MPI2007 datasets [Online] Available: https://spec.org/mpi2007/results/mpi2007.html [Accessed: 24-Jun-2020],

[40] Accel OpenMP dataset [Online] Available: https://spec.org/accel/results/accel.html [Accessed: 24-Jun-2020]

[41] Stepanov E. P., Smeliansky R. L. On bandwidth on demand problem // Proceedings of the 27th International Symposium Nuclear Electronics and Computing (NEC'2019). — Vol. 2507. — CEUR-WS Budva, Montenegro, 2019. pp. 402–407.

[42] Böhme, Thomas, Frank Göring, Jochen Harant. "Menger's theorem." *Journal of Graph Theory* 37.1 2001, pp. 35-36

[43] Stepanov, E., R. Smeliansky. "On Analysis of Traffic Flow Demultiplexing Effectiveness" 2018 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC). IEEE, 2018

[44] Navin Kukreja, Guido Maier, Rodolfo Alvizu, Achille Pattavina. SDN based automated testbed for evaluating multipath TCP. In IEEE International Conference on Communication, ICC 2015, London, United Kingdom, June 8-12, 2015, Workshop Proceedings, pp. 718–723, 2016.

[45] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, Mark Handley. How hard can it be? designing and implementing a deployable multipath tcp. In Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pp. 399–412, San Jose, CA, 2012. USENIX.

[46] Evgeny Chemeritskiy, Evgeny Stepanov, Ruslan Smeliansky. Managing network resources with flow (de) multiplexing protocol. *Mathematical and Computational Methods in Electrical engineering*, 53:35–43, 2015

[47] Chiesa, Marco, Guy Kindler, Michael Schapira. "Traffic engineering with equal-cost-multipath: An algorithmic perspective." IEEE/ACM Transactions on Networking (TON) 25.2 (2017): 779-792.

[48] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, Requirements for Traffic Engineering Over MPLS, RFC 2702, Sep. 1999.

[49] Awduche, Daniel, et al. "RSVP-TE: extensions to RSVP for LSP tunnels." (2001) https://tools.ietf.org/html/rfc3209

[50] Irawati, Indrarini Diah, Sugondo Hadiyoso, Yuli Sun Hariyani. "Link Aggregation Control Protocol on Software Defined Network." International Journal of Electrical and Computer Engineering 7.5 (2017): 2706.

[51] Burke J. What is the role of machine learning in networking? https://searchnetworking.techtarget.com/answer/What-is-the-role-of-machine-learning-in-networking

[52] Smelyansky R. Hierarchical edge computing // International conference proceedings Modern Network Technologies, MoNeTec-2018:. — Москва, 2018. pp. 97–105.

[53] Bo An, Xudong Shan, Zhicheng Cui, Chun Cao and Donggang Cao, Workspace as a Service: an Online Working Environment for Private Cloud, 2017 IEEE Symposium on Service-Oriented System Engineering (SOSE), San Francisco, 2017, pp. 19-27.

[54] Donggang Cao, Peidong Liu, Wei Cui, Yehong Zhong, Bo An, Cluster as a Service: a Resource Sharing Approach for Private Cloud, Tsinghua Science and Technology, 2016, 21(6): pp. 610-619

[55] Bandwidth on Demand Service Demo based on SDN controller RunOS https://youtu.be/XjggLW1LKKg