HEPS Virtual Cloud Desktop System Based on OpenStack, Design and Implementation

Jiping Xu[a,*] **and QingBao Hu**[a]

[a] *Computing Center, Institute of High Energy Physics, Chinese Academy of Sciences, 100049, Beijing, P.R.China*

E-mail: xujp@ihep.ac.cn, huqb@ihep.ac.cn

High Energy Photon Source (HEPS) will generate a large amount of experimental data for diverse scientific analyses. The traditional method of using local computing environments for data downloading and analysis by users can no longer meet the growing experimental demands. This paper proposes a virtualization-based HEPS cloud desktop system for 3D data imaging and crystal scattering experiments in HEPS. Its features require high-quality image display, high-performance GPU computing, and image rendering, but the experimental site generally lacks these conditions. Therefore, it is particularly important to utilize the resources of a computing cluster to provide a virtual cloud desktop. First, we introduce the basic situation and experimental characteristics of HEPS, as well as the research motivation behind the virtual cloud desktop system. Then, we provide a detailed description of the architecture, service mode, authentication system, GPU usage, and present the design concept of the heterogeneous resource mixed scheduling strategy for the virtual cloud desktop. Finally, we demonstrate the actual application effect of the virtual cloud desktop system in the light source experiment, which highlights its superiority and good prospects in the field of synchrotron radiation sources.

International Symposium on Grids and Clouds (ISGC) 2023,
19 - 31 March 2023
Academia Sinica Taipei, Taiwan

*Speaker

1. Introduction

HEPS is a high-performance synchrotron radiation source with an electron energy of 6 GeV and an emittance of less than or equal to 0.06 nm·rad. It mainly consists of an accelerator, beamlines, and experimental stations, as shown in Figure 1. HEPS is internationally advanced, with its main performance indicators ranking among the top in the world.

Figure 1: HEPS Site

As a fourth-generation synchrotron radiation source, HEPS possesses the world's highest spectral brightness and is expected to provide over 5,000 hours of experimental time per year. In the first stage, the 14 beamlines will generate tens of petabytes of raw experimental data every month. On March 14, 2023, HEPS linear accelerator successfully accelerated its first electron beam, marking another important milestone in the construction of the HEPS facility.

HEPS offers a wide range of experimental scenarios, including dozens of experimental methods such as Extended X-ray Absorption Fine Structure (EXAFS) and X-ray Absorption Near Edge Structure (XANES). Synchrotron radiation is the most advanced in fields such as physics, chemistry, materials science, life science, and medicine. At the same time, the diverse experimental scenarios and research fields also pose new challenges for services such as computing and storage.

In recent years, cloud computing technology[1] has experienced rapid development. In complex computing environments, cloud computing can significantly improve the utilization of computing resources[2]. It can provide services with less management work, enabling rapid deployment and elastic configuration of computing, including computing, storage, and networking[3]. Through virtualization technology, cloud computing effectively integrates software and hardware resources and is being increasingly applied in the field of high-energy physics. The Institute of High Energy Physics (IHEP) of the Chinese Academy of Sciences has built the IHEP cloud platform based on the open-source cloud computing management software OpenStack[4] and KVM[5]. For computing-intensive services such as high-energy physics experiments, the platform provides a computing cluster composed of virtual machines[6]. The virtual machine computing cluster uses Puppet[7] for automated deployment of the software environment required for scientific computing, and uses the job management system HTCondor[8] to schedule and manage user jobs. The European Organization for Nuclear Research (CERN) uses OpenStack cloud to support physical computing and infrastructure services at its sites. CERN virtual machines provide nodes for users to perform computing-intensive data processing, WLCG[9], and development services. CERN's private cloud provides data volume services that are provided by Ceph and NetApp storage [10]. OpenStack is widely used internationally to support physical computing and infrastructure services at sites.

2. Motivation

The characteristics of imaging, crystal scattering, and artificial intelligence experiments in HEPS are as follows:

- High requirements for image display quality: CT image and other experimental data require high-resolution display to observe sample details.
- Large storage capacity required: HEPS experimental datasets can reach hundreds of TB to PB, usually in unstructured formats such as text, images.
- Large memory capacity required: The experimental software uses random access memory (RAM) to process images.
- High-performance GPU computing and image processing required: CT image reconstruction and rendering, deep learning algorithms, and other algorithms require a large number of matrix multiplication and accumulation floating-point operations.
- Closed-source experimental software: There are many commercial software for experimental analysis, such as VG Stdio Max, Avizo, etc., which cannot be integrated into self-developed software frameworks.

The traditional method of using local computing environment for data downloading and analysis cannot meet the growing experimental demands. It is crucial to utilize the resources of computing clusters, and provide virtual cloud desktop services and high-performance GPU computing services. However, existing remote desktop software has shown unsatisfactory results in testing for synchrotron radiation experiments. It is difficult to meet the experimental requirements for imaging and experiments requiring 3D rendering, mainly due to slow rendering speed, lagging in converting 3D perspectives, insufficient resolution and frame rate leading to unclear observations, and other issues. On the other hand, GPU computing resources are fewer than CPU, and the scattered tower machine model in various laboratories is not conducive to unified management and full utilization of computing resources. It is necessary to implement intelligent scheduling for GPU and a one-card-multi-use model.

This paper presents the implementation of HEPS Virtual Cloud Desktop system (HEPS-VCD), which enables users to access the cloud-based operating system interface clearly and conveniently. It provides high-resolution and high-frame-rate display effects, and more efficient CPU and GPU usage methods, which can be used for high-performance computing, image display, and image rendering to improve the efficiency of experimental analysis.

3. Design and Implementation of the HEPS-VCD

3.1 Framework of HEPS-VCD

We integrate Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) to provide Desktop as a Service (DaaS), which includes computing clusters, interactive computing platforms, experimental analysis software, and operating system interfaces.DaaS provides a virtual desktop environment that can be accessed and used by users over the internet without the need to install and maintain software and hardware locally. Administrators deploy the desktop environment in the cloud and provide a range of remote access and management tools, allowing users to complete various tasks and operations over the internet. The advantages of DaaS include improved flexibility and scalability, reduced device costs and management complexity, and enhanced security and data protection capabilities.

We have applied cloud desktop in the field of scientific research, which serves as a window for HEPS users to access experimental data and conduct analysis. Users first authenticate their identity by logging into the service website, and then apply for cloud desktops with different configurations, including CPU, memory, disk, GPU, etc., based on their needs. Upon receiving the request, the underlying OpenStack performs tasks such as creating virtual machines, injecting Metadata, initializing the operating system, configuring the desktop streaming system, and provides pass-through GPU according to the requirements. The initialization of an operating system includes creating users and groups, switching login users, mounting storage systems,such as Lustre and HUAWEI OceanStor. Additionally, it involves launching experimental analysis software, configuring the computing environment. The initialization of the operating system includes mounting the storage system and authentication, such as Lustre, HUAWEI OceanStor, as well as starting experimental analysis software and configuring the computing environment. The experimental data of the storage system is generated during the Data Acquisition (DAQ) process at the HEPS beamline station. According to security policies, this data can only be accessed within the IHEP intranet. The overall architecture of cloud desktop services is illustrated in Figure 2.

3.2 Design of Authentication System for HEPS-VCD

After logging into the HEPS service website using their IHEP unified authentication account, users can apply for and create cloud desktops and virtual machines themselves. We use a token-based authentication mechanism to verify the identity of users. After logging in, the server generates a random token and returns it to the client. The client carries the token in subsequent requests to prove its identity.

The workflow of token-based authentication is as follows:

1. The HEPS user service website sends the unified authentication information of the current logged-in user to the token server.
2. The server verifies the user's information. If it is correct, it generates a token and returns it to the client.
3. After receiving the token, the client stores it locally, usually in a cookie or local storage.
4. The client combines the token, streaming server IP, and port number into a URL and accesses the cloud desktop streaming server through the browser.
5. 5.The streaming server receives the request, calls an external authentication API to obtain the user's identity, and verifies the validity of the token and access permissions. If the verification passes, it allows the user to access. The streaming server queries the IP address

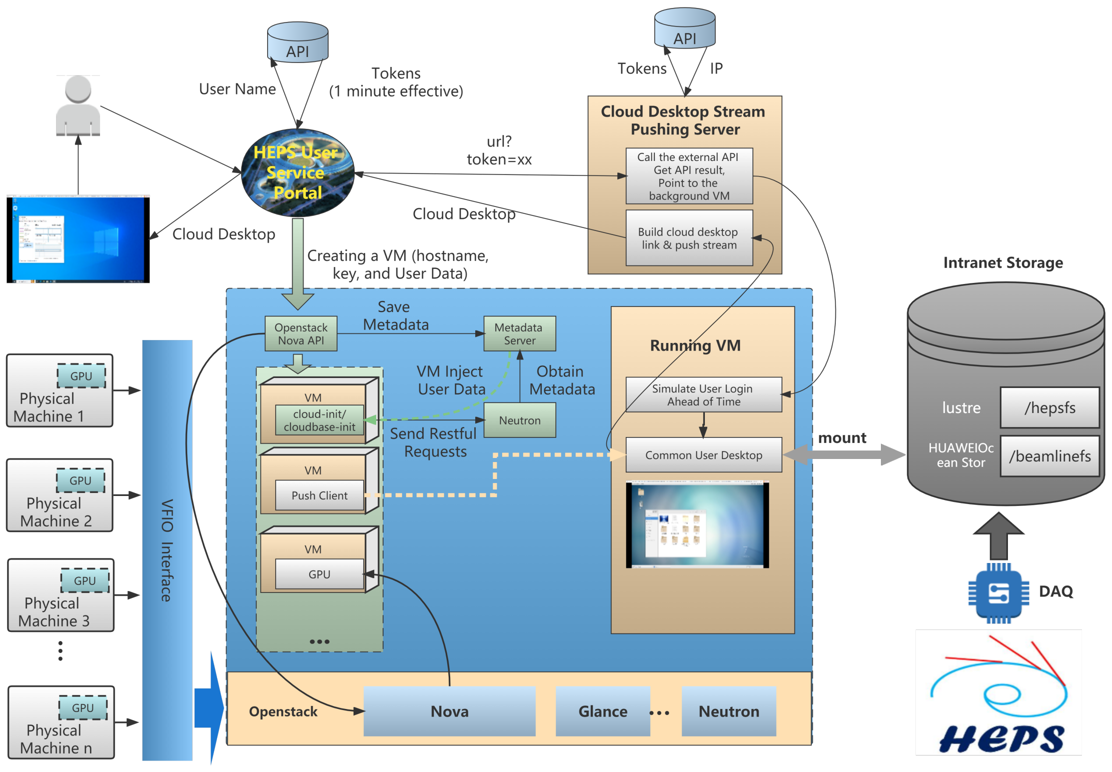


Figure 2: Framework of HEPS-VCD

of the corresponding virtual machine for the user and streams the cloud desktop service of the virtual machine to the browser page.

6. The user can enter the Linux or Windows operating system interface without logging in.

3.3 Service Model of HEPS-VCD

Users can apply for cloud desktop services on the HEPS service website. The backend system schedules computing resources, creates virtual machines, and starts the cloud desktop streaming service. Once completed, users can click the "Connect Cloud Desktop" button on the service website, and a new page will automatically open in the browser, which is the operating system page of the cloud desktop.

The cloud desktop image includes multiple versions of Windows and Linux operating systems. Users can customize the cloud desktop image according to their experimental needs. The cloud desktop image is customized differently for different experimental stations. Different experimental stations also have different requirements for cloud desktop configuration, such as whether high-performance GPU computing is required. The data collected by HEPS experimental stations is stored on a distributed storage system. Currently, we have Lustre, HUAWEI OceanStor 9950. The cloud desktop will mount the specified storage disk and access stored data according to the user's permissions. Users can load and analyze experimental data using pre-installed data analysis software, and write the analysis results back to their personal storage directory,as shown in Figure3. Both raw data and experimental results can be downloaded from the portal website.

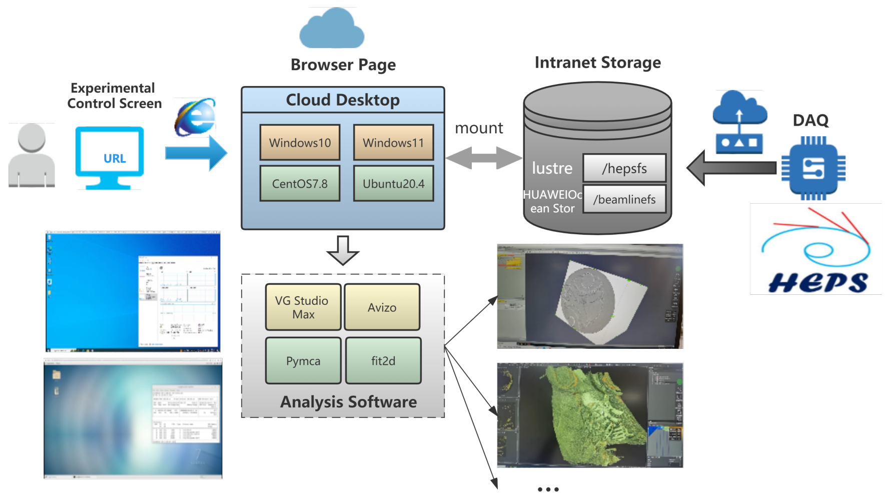


Figure 3: Using Analytics Software in Cloud Desktop

Users can also analyze experimental data using software frameworks or interactive computing platforms on the cloud desktop, supporting data analysis environments such as HEPSCT, Cumpy, Tomopy, Alphafold, as shown in Figure 4. The software framework is based on its own software and hardware foundation, abstracting computing hardware resources and providing standard calling interfaces for upper-layer applications. The frontend of the interactive computing platform is based on JupyterLab, providing users with software, algorithm development, and data processing environments through web browsers, combining the computing platform with the software framework to provide data analysis services for users.

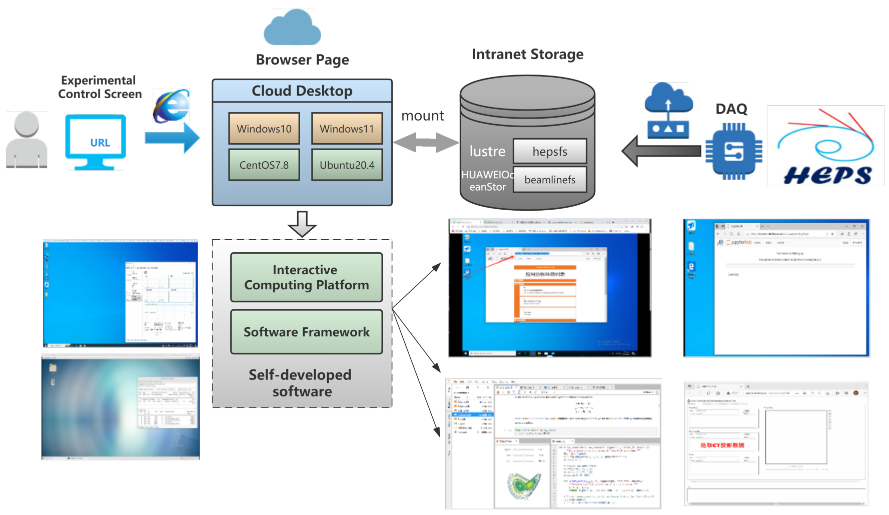


Figure 4: Using Custom Software Framework and Computing Platform in Cloud Desktop

3.4 GPU in HEPS-VCD

In traditional virtualization environments, virtual machines usually can only use virtual graphics card devices, which have limited performance and functionality and cannot meet the needs of high-performance graphics applications. However, GPU pass-through technology can directly map physical graphics cards to virtual machines, allowing virtual machines to directly access the hardware resources of the graphics card, thereby obtaining better graphics performance and user experience.

The GPU pass-through technology requires a CPU that supports IOMMU[11] and a graphics card that supports SR-IOV[12]. The advantage of GPU pass-through is that it can provide better graphics performance and user experience, while also improving the flexibility, security, and reliability of virtualized environments.GPU pass-through requires virtualization software support, such as KVM, Xen, and others. These virtualization software need to provide corresponding APIs and management tools to allocate physical graphics cards to virtual machines for use. We use the open source cloud computing platform (OpenStack) to provide virtualization management services.

Interface-based GPU pass-through involves the following steps:

1. Enable IOMMU on the OpenStack compute node.
2. Insert a DP emulator into the host's graphics processing GPU to avoid downclocking and sleeping.
3. Detach the graphics card device from the physical machine, disable the physical machine driver used by the graphics card, create a GPU audio device blacklist, and then add the graphics card device to the Virtual Function I/O (VFIO) module, allowing the device to use the VFIO driver and be added to the virtualizable column.
4. Configure the Nova-related services of OpenStack control and compute nodes, including filter scheduler and the PCI device section, and add the devices configured with VFIO to the OpenStack PCI available list for the Nova component to call and allocate.
5. Create a virtual machine and install the corresponding GPU driver.

3.5 Resource Hybrid Scheduling Strategy in HEPS-VCD

Currently, OpenStack's GPU scheduling only considers whether the number of GPUs meets the virtual machine's requirements, and OpenStack cannot obtain detailed information about GPU memory and cores. If GPUs are allocated based solely on their quantity, task failures may occur due to insufficient GPU memory. To address the issue of insufficient granularity in GPU scheduling resource reporting, this paper proposes a component that describes detailed GPU status information to expand OpenStack's API support for GPUs. This custom resource is named System Resource Monitoring (SRM) and includes information on the number of GPUs, GPU memory size, core count, and bandwidth on each node. An SRM is created for each node, and these SRMs are registered in the OpenStack database for persistent storage. The scheduler can query the SRMs of each node from the API server to obtain a global view of GPU status information.

Figure 5 illustrates the process of the controller updating GPUs. First, the controller creates an SRM for the current node and then enters a loop. Every interval time, it checks the status of the GPUs and aggregates GPU information, then updates the local SRM status information. When users apply for GPU resources, they can add metadata definitions for fine-grained resource requests, including GPU quantity, core count, memory, and other information. During the compute node scheduling phase, the custom scheduler queries the latest GPU status information from the API server to provide fine-grained GPU information for the next scheduling decision.

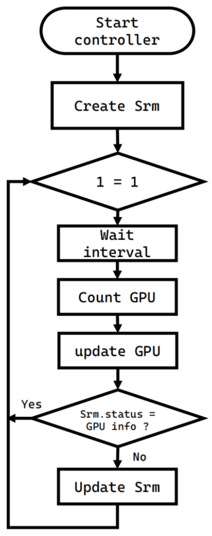


Figure 5: System Resource Monitoring

In the case of mixed deployment of CPU and GPU nodes, the OpenStack scheduler cannot perceive heterogeneous resources, leading to resource competition. To address this, this paper proposes a fine-grained heterogeneous resource mixed scheduling method that comprehensively considers the distribution of heterogeneous resources on nodes and hardware status. First, using OpenStack's custom resources and controllers to collect detailed GPU status information on each computing node and provide GPU status to the scheduling algorithm. Second, the scheduling filter algorithm is improved by adding custom GPU information filtering. Finally, we need to improve the rating algorithm of the scheduler. Nodes are divided into CPU nodes and GPU nodes based on whether they provide GPU, and virtual machines are divided into CPU-type virtual machines and GPU-type virtual machines based on whether they apply for GPU.

3.5.1 Improving Filtering Algorithm

During the filtering stage of the scheduler, in addition to the original filtering based on CPU and memory, the proposed method in this paper also checks the virtual machine's metadata definition for fine-grained GPU resource requests. By comparing the requested resources of the virtual machine with the available resources on the GPU node, the nodes that meet user requirements are added to the schedulable list. Under the GPU/CPU mixed deployment scheduling strategy, the improved filtering algorithm proposed in this paper is shown in Figure 6:

It selects the nodes that meet both CPU and GPU requirements and adds them to the schedulable list. This fine-grained filtering algorithm ensures that virtual machines are scheduled to the most suitable nodes, improving the efficiency of resource utilization.

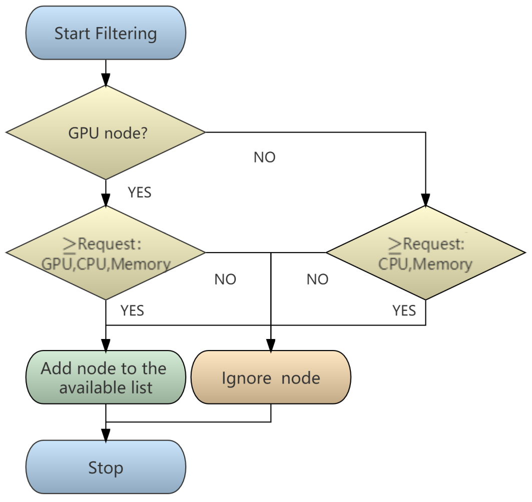


Figure 6: Improvement of Filtering Algorithm

1. The scheduler obtains the metadata definition of the virtual machine and determines whether it is a GPU-type or CPU-type. If it is a CPU-type application, it jumps to step 2. If it is a GPU-type virtual machine, it jumps to step 3.

2. The scheduler uses the default filtering strategy, Filter Scheduler, to check whether the available resources of the node meet the requirements of the virtual machine, such as CPU, memory, and GPU quantity. If the requirements are met, it proceeds to step 4. If not, it goes to step 5.

3. Initially, the scheduler filters out the nodes that do not meet the CPU and memory requirements. Subsequently, the scheduler retrieves the requested GPU information from the metadata definition of the virtual machine, queries the SRM information of all GPU nodes from the API server, and checks whether the idle GPU status on each GPU node meets the request. If it does, proceed to step 4; otherwise, proceed to step 5.

4. If the node meets the requirements of the virtual machine, the scheduler adds the current node to the list of available nodes and exits.

5. If the node does not meet the requirements of the virtual machine, the scheduler ignores the current node and exits.

Filter selects the nodes that meet both CPU and GPU requirements and adds them to the schedulable list. This fine-grained filtering algorithm ensures that virtual machines are scheduled to the most suitable nodes, improving the efficiency of resource utilization.

3.5.2 Improving Scoring Algorithm

To select the most suitable node, the scheduler needs to score the filtered nodes and bind the highest-scoring node to the virtual machine. In the case of mixed deployment of CPU and GPU resources, our scoring strategy is to prioritize heterogeneous resource nodes, ensuring that CPU-type virtual machines are scheduled to CPU nodes first. When CPU node resources are insufficient, they can be scheduled to idle CPU on GPU nodes to improve cluster resource utilization.

For CPU nodes, this paper uses the least Requested Score (LRS) to represent the score of the current node for CPU-type virtual machines. The calculation method of LRS is shown in Equation (1):

$$LRS = \left(\sum_{i=1}^{N} \frac{c_i - r_i}{c_i} * w_i \right) * M \tag{1}$$

Here, $N = 1, 2$ represents the two types of resources, CPU and memory. c represents the maximum capacity of resources on the node, r represents the number of requested resources on the node, and w represents the weight of the resources, which is assumed to be 0.5 by default. M represents the maximum score of the node, and the score range of the node is $[0, M]$.

When scheduling GPU applications, we scores the GPU nodes based on the requirements of the GPU-type virtual machine. The performance score of the GPU card on the node is defined as S_card, where cc represents the normalized GPU computing power, bw represents the normalized bandwidth, and cl represents the normalized clock. The weights $w1$, $w2$, and $w3$ are all assumed to be $1/3$. The scoring formula is shown in Equation (2). A higher S_card indicates a higher performance of the card, which makes the current scoring algorithm prefer GPU with better performance.

$$S_{card} = (cc * w1 + bw * w2 + cl * w3) \tag{2}$$

The similarity between the idle resource vector of the current node and the requested resource vector of the virtual machine is defined as $S_{similarity}$. The number of idle GPU cards on each node is $Card_{free}$, the size of the memory on each card is $Card_{mem}$, and the number of cores on each card is $Card_{core}$. The number of idle CPU cores on the node is CPU_{free}, and the idle memory is Mem_{free}. The idle resource vector N on the node can be represented as:$N = (Card_{free}, Card_{mem}, Card_{core}, CPU_{free}, Mem_{free})$. If the virtual machine requests $RCard_{num}$ GPU cards, with $RCard_{mem}$ memory size per card and $RCard_{core}$ cores per card, as well as $RCPU_{num}$ CPU cores and $RMem_{num}$ memory size, then the requested resource vector of the virtual machine can be represented as: $R = (RCard_{free}, RCard_{mem}, RCard_{core}, RCPU_{num}, RMem_{num})$. The formula for calculating the similarity, as shown in Equation (3).

$$S_{similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|} \tag{3}$$

$S_{similarity}$ makes the current scoring algorithm prefer nodes with the least remaining resources. The scoring formula for GPU nodes is shown in Equation (4) where S_{card} represents the inherent properties of the GPU card and $S_{similarity}$ represents the degree of fit between the node resources and the virtual machine requested resources.

$$Gpu_{score} = S_{card} \times S_{similarity} \tag{4}$$

Pseudocode for the hybrid scheduling scoring algorithm as shown in algorithm1:

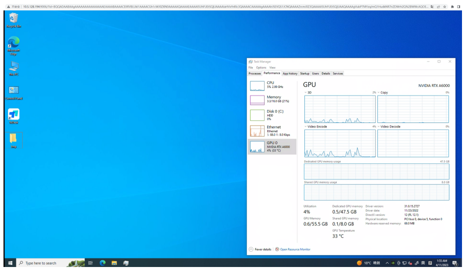
Algorithm 1: Hybrid Scheduling Scoring Algorithm

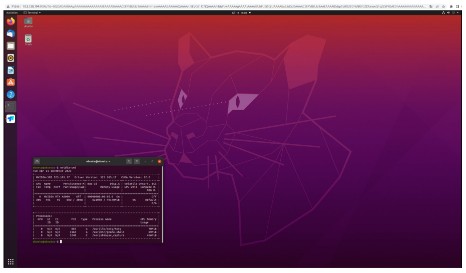
```
input  : Each node of the CPU, storage and GPU extension information
output : The binding information between computing nodes and virtual machines
1  R=getVMsRequest(VM) // Obtain the resources requested by the VM
2  if isGPUVMs(R) then
       // Check whether the VM applies for GPU resources
3      return GPUScore(R) // The VM requests the GPU and return the GPU
           score of the current node
4  else
5      if nodeHasGPU(nodes) then
           // Check whether the node is GPU node
6          return PriorityScore(CPUScore(R,node)) // CPU VM score in GPU node
7      else
8          return PriorityScore(CPUScore(R,node)) // CPU VM score in CPU node
9      end
10 end
11 candidateNode = getMaxScoreNodes(nodes);
12 bind(candidateNode,pod)// Bind VM to node with the highest score
```

3.6 Application Effect of Cloud Desktop in Synchrotron Radiation Light Source Experiments

HEPS is primarily applied in the fields of physics, materials science, life science, and other related areas. It can be utilized to investigate the properties of matter, design of materials, and structure of biomolecules. Currently, the HEPS virtual cloud desktop system is in the development and testing phase, with a cluster consisting of three hosts, including three NVIDIA RTX A6000 graphics cards on the compute nodes. In the planning of the HEPS virtual cloud desktop project, there will be approximately 250 physical machine and 500 virtual machine for providing cloud desktops. The HEPS virtual cloud desktop includes multiple versions of Windows, CentOS, and Ubuntu, and the Figure 7 below shows the effect of the cloud desktop.

HEPS is currently in its construction phase, and we conducted tests on a cloud desktop at the Beijing Synchrotron Radiation Facility (BSRF) using VG Studio Max and Avizo software. In synchrotron radiation experiments, VG Studio Max can be used for processing X-ray diffraction and CT scan data, enabling 3D visualization and analysis. For example, VG Studio Max can be used in protein crystallography experiments to convert X-ray diffraction data into 3D crystal structures, and in materials science experiments to analyze and simulate properties by processing CT scan data to visualize the 3D structure of materials. Figure 8 (a) depicts the scene of experimental personnel using VG Studio Max for experimental analysis on a cloud desktop. The interface of Avizo is more complex, providing more functionalities and customization options, suitable for processing more complex data. Figure 8 (b) below shows a scene of experimental analysis using Avizo on the cloud desktop by the experimental personnel.


(a) Windows 10


(b) Ubuntu 20.04

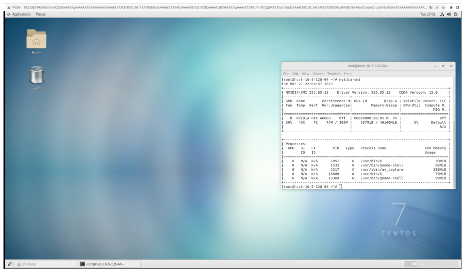
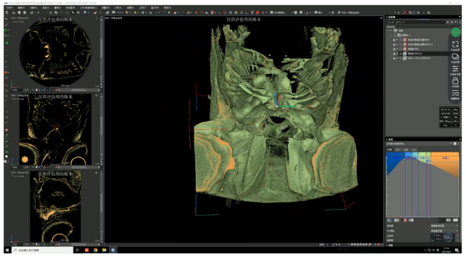

(c) CentOS 7.8

Figure 7: Cloud Desktop


(a) VG Studio Max

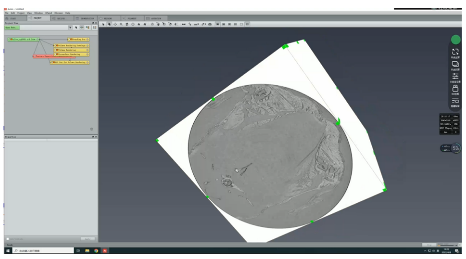

(b) Avizo

Figure 8: Experimental Analysis Scene

4. Conclusion

In this paper, we designed a virtual cloud desktop system based on Openstack using virtualization technology and tailored it to the characteristics of HEPS experiments. We proposed a new virtual cloud desktop service mode for HEPS and developed a user authentication system. We managed and allocated GPU devices uniformly in virtualization and proposed a heterogeneous resource scheduling algorithm.Furthermore, we analyzed the system in the context of practical applications in light source experiments and evaluated the performance of experimental analysis software on the system. Our conclusion is that the virtual cloud desktop system demonstrates clear advantages in terms of computing resource utilization while ensuring normal operation of synchrotron radiation experimental analysis. The centralized resource management of the cloud desktop provides the same level of operational convenience as local data analysis for users, and offers more reasonable resource allocation strategies and stronger computing and analysis capabilities. Currently, the system is still in the testing and development phase with a relatively small scale. In the next step, more HEPS scientists will join the implementation process and continuously improve and refine the system in large-scale usage, such as enhancing cloud desktop file transfer function, improving resolution and frame rate, and optimizing dynamic scheduling algorithms.

References

[1] Foster I, Zhao Y, Raicu I, et al. Cloud computing and grid computing 360-degree compared [C]//2008 grid computing environments workshop. IEEE, 2008: 1-10.

[2] Botta A, De Donato W, Persico V, et al. Integration of cloud computing and internet of things: a survey [J]. Future generation computer systems, 2016, 56: 684-700.

[3] Liu Y, Liu X, Zhou W, et al. A Survey on Resource Management in Cloud Computing: Taxonomy, Challenges, and Future Directions [J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(4): 839-857.

[4] Sefraoui O, Aissaoui M, Eleuldj M, et al. Openstack: toward an open-source solution for cloud computing [J]. International Journal of Computer Applications, 2012, 55(3): 38-42.

[5] Kivity A, Kamay Y, Laor D, et al. kvm: the linux virtual machine monitor [C]//Proceedings of the Linux symposium: volume 1. Dttawa, Dntorio, Canada, 2007: 225-230.

[6] Li H, Cheng Y, Huang Q, et al. Integration of openstack cloud resources in bes iii computing cluster [C]//Journal of Physics: Conference Series: volume 898. IOP Publishing, 2017: 062033.

[7] Loope J. Managing infrastructure with puppet: configuration management at scale [M]. O'Reilly Media, Inc, 2011.

[8] Erickson R A, Fienen M N, McCalla S G, et al. Wrangling distributed computing for high-throughput environmental science: An introduction to htcondor [J]. PLoS computational biology, 2018, 14(10): e1006468.

[9] Bird I, Carminati F, Mount R, et al. Update of the computing models of the wlcg and the lhc experiments [R]. 2014.

[10] Bell T, Bompastor B, Bukowiec S, et al. Scaling the cern openstack cloud [C]//Journal of Physics: Conference Series: volume 664. IOP Publishing, 2015: 022003.

[11] Gupta A, Vetter J S. IOMMU: An Effective Remedy for Security Vulnerabilities and Performance Overheads in Modern I/O Devices [J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(9): 2058-2072.

[12] Dong F, Wu J, Wang X, et al. SR-IOV based Virtualized NICs: Architecture, Design and Implementation [J]. Journal of Network and Computer Applications, 2015, 55: 105-115.