Future Generation Computer Systems 81 (2018) 478–491

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

FastDesk: A remote desktop virtualization system for multi-tenant

Tao Song [a], Jiajun Wang [a], Jiewei Wu [a], Ruhui Ma [a,*], Alei Liang [a], Tao Gu [b], Zhengwei Qi [a]

[a] Shanghai Jiao Tong University, Shanghai 200240, PR China
[b] RMIT University, 124 La Trobe St, Melbourne VIC 3000, Australia

HIGHLIGHTS

- Present a remote desktop virtualization system for multi-tenant, namely FastDesk.
- Design a server-push streaming scheme to improve user experience & video quality.
- Propose a virtual desktop placement algorithm to reduce the waste of resource.
- Conduct a series of experiments to evaluate the performance of the FastDesk.

ARTICLE INFO

Article history:
Received 24 March 2017
Received in revised form 9 June 2017
Accepted 1 July 2017
Available online 5 August 2017

Keywords:
Remote desktop virtualization
Remote display
Virtualization system

ABSTRACT

Classical remote desktop systems are considered for a single host, thus hindering their applicability on virtualization environment. Remote desktop virtualization is rising in the recent years as a new advanced extension to virtualization technology. However, existing remote desktop virtualization solutions introduce weak performance in relation to response time, video quality, and cost saving. This paper proposes a novel remote desktop virtualization system for multi-tenant, namely FastDesk, which is based on a server-push mode streaming mechanism and a heuristic virtual desktop placement algorithm. Through extensive experiments, the results show that FastDesk outperforms other popular platforms in terms of bandwidth with less than 2 Mbps and 94% video quality. Meanwhile, it creates the minimal resource wastage compared with other virtual machine placement algorithms. Furthermore, the proposed FastDesk achieves low CPU utilization, low bandwidth and good scalability. At last, to widen its applicability, FastDesk has been implemented on VirtualBox for running 3D programs.

1. Introduction

Nowadays, Information Technology (IT) organizations find that the widening gap between the availability and demand of IT resources becomes a challenging issue. Personal desktop computers are ubiquitous in enterprises, educational institutions and governmental organizations, while the cost of maintenance and upgrade of these computers turns out to be enormous and unmanageable. Virtualization technology has been advocated as a realistic solution for resource consolidation, and it has demonstrated the cost saving benefit in these circumstances. It also provides many other benefits, including facilitating rapid deployment, offering enhanced flexibility and scalability [1,2].

Remote desktop virtualization, a new advanced extension of virtualization technology, is rising in the recent years as an alternative to classical desktop delivery [3]. It is a software technology that separates the desktop environment and associated applications from the remote client device. An end user only needs a thin client which handles display, keyboard and mouse combined with adequate processing power for graphical rendering and network communication. Moreover, the client no longer has to keep the user state and communicate with the server by using a remote protocol. This remote protocol allows graphical display to be virtualized, and transmits user input from client to server [4,5]. Thus, remote desktop virtualization offers a cost-efficient paradigm shift to ease management complexity since operating systems, applications and data are kept in a large data center. In addition, it is easy to troubleshoot and replace thin client since it is stateless. With the ever increasing popularity of cloud computing, remote desktop virtualization can be easily provided through cloud environments. Similar to the Software as a Service (SaaS) model, this approach is also referred to as Desktop as a Service (DaaS) [6].

Despite some works [3,6–8] have been contributed on remote desktop virtualization, technical challenges still exist. One key task in this field is to provide high fidelity display and good interactive experiences for end users, especially in multimedia application

* Corresponding author.
E-mail addresses: songtaotao429@163.com (T. Song), ruhuima@sjtu.edu.cn (R. Ma).

http://dx.doi.org/10.1016/j.future.2017.07.001
0167-739X/

which is typically integrated with modern desktop computing. Current remote display protocols such as Remote Framebuffer protocol (RFB) [9] and Remote Desktop Protocol (RDP) [10] are widely used in remote desktop virtualization systems [11]. They are mainly designed for slow-motion graphical applications, such as text editor, which imply minor changes with low frequency. On the one hand, RFB uses the client-pull model to update its screen. Nieh et al. [12] pointed out that the performance of the thin-client system is mainly affected by latency. In this case, the client-pull system will suffer more since it has to send two messages for each update and cannot effectively support high-motion scenarios such as video playback and real-time interactions [13]. On the other hand, the details of RDP are not open to the public. It is also worth noting that RDP uses the Microsoft proprietary solution such as Direct2D and Direct3D, which is not available in Linux and Mac OS. Therefore, the transport of multimedia data over these protocols is inefficient, requiring high bandwidth to ensure the delivery of all the frames to the client in time [14]. There are efforts [7,15] to support high fidelity display and improve user experience in the thin-client computing architecture, whereas these architectures are dependent on the underlying hardware or implementing specific hardware driver which is not suitable for a virtualization environment since Virtual Machine Monitor (VMM) provides an abstract layer for operating systems and applications.

In a remote desktop virtualization system, another key challenge is to improve cost saving as much as possible [16–18]. Virtualization technology provides the basis of statistical multiplexing to server resources in data centers. Typically, in the remote desktop virtualization system, all operating systems and applications are installed on Virtual Machines (VMs), which are located on a server or a cluster in a remote data center. Each virtual desktop maps into a VM. Under this scenario, finding the optimal placement strategy of VMs can effectively utilize server resources and maximize the profits. However, the VM placement is a kind of bin-packing problem, which is known as an NP-hard [18–20]. At present, the greedy heuristics [18–22], such as First-Fit (FF), Next-Fit (NF), Best-Fit (BF) etc., are popular. Nevertheless, these approaches are rough and hard to obtain maximized profit.

To address the above challenges, we propose FastDesk, a novel remote desktop virtualization system for multi-tenant, which is based on our previous work [23]. Combined to a server-push mode stream architecture and virtualization technology, FastDesk provides high quality display and good user interactive experience without modifying Guest Operating System (Guest OS) or requiring specific hardware. Meanwhile, to minimize the server resource wastage, it uses a nature-inspired heuristic algorithm to solve the VM placement problem. Furthermore, FastDesk supports multi-tenant management, which is a key feature in cloud computing services. Multi-tenant management contains resource allocation and performance monitoring, which transforms FastDesk into a DaaS system. It can also scale down the screen size to fit different devices. We implement FastDesk prototype based on Kernel-based Virtual Machine (KVM) [24], a popular virtualization system, and conduct extensive experiments. The results show that our system achieves 94% video quality in a 1024 × 768 display resolution for video playback in both Local Area Network (LAN) and Wide Area Network (WAN) environments, while classic remote systems only achieve at best around 20%. In a high interaction scenario, FastDesk achieves the shortest response time in WAN, outperforming other popular desktop virtualization systems. Furthermore, the proposed algorithm minimizes the waste of resources as compared with other classical VM placement heuristics.

To sum up, the main contributions of FastDesk as follows:

- We proposed a novel remote virtualization system for multi-tenant, which contains a server-push mode streaming mechanism and a heuristic virtual desktop placement algorithm.
- The results show, via a great amount of experiments, that the FastDesk achieves low CPU utilization, low bandwidth and good scalability.
- We have also demonstrated that the proposed system outperforms other popular platforms in terms of CPU utilization, bandwidth and video quality. Besides, our proposed virtual desktop placement algorithm always gets the minimal resource wastage against other classical heuristics.

The remainder of this paper is structured as follows: Section 2 explores some existing Remote Desktop Virtualization techniques and their placement algorithms. Section 3 elaborates on our architecture. Section 4 describes an evolutionary heuristic VM placement algorithm. Section 5 introduces our system implementations in details. Section 6 presents experimental results measuring our performance and compares them to other popular remote desktop virtualization systems. Finally, conclusion and future work are presented in Section 7.

2. Related work

Remote desktop virtualization is a thin-client computing pattern [3,25]. A common implementation of this system is that stateless thin clients interact with virtual desktops by means of network communication using a remote display protocol. Virtual desktops, which contain isolated operating system instances and related applications, are installed on VMs hosted by servers in a remote data center. Virtualization technology allows multiple VMs to run on a real server, which implies improving server resources utilization can increase system benefits. And the benefits rely heavily on VM placement strategy.

2.1. Remote desktop virtualization systems

Existing works propose many alternative designs. These solutions can be classified by the level of graphical output which are sent to the client. The X system [26] simply forwards high-level application display commands and leaves the graphical user interface processing to the client for remote display functionality. It is cost-efficient on the server side, but leads to complexity on the client side. Software on the client need to be updated frequently, which violates the goal of zero maintenance in desktop virtualization.

The high complexity of the client can be avoided by simply running graphical user interface on the server. Microsoft Remote Desktop [10], Citrix MetaFrame [27], PC-over-IP [28] and TeamViewer [29] are four famous commercial products used in remote desktop. They encode the display updates into low-level graphical commands and the client is simplified as an input–output device. The real application and display states are maintained on the server while only the transient soft state is on the client. Windows Remote Desktop and TeamViewer are typically used as remote tools to access one's Windows computer. TeamViewer and the desktop version of Windows both can have only one connection online. As for Windows Server, the number of simultaneous connections is configurable. But the remote access is just used to maintain the server, which is not a DaaS scenario. PC-over-IP is a closed-sourced protocol developed by Teradici. It is now used in VMware Horizon View [30] and Amazon WorkSpaces [31] as their DaaS solutions.

VNC [9] works at the framebuffer level and is independent from operating system. It reduces the display updates to raw pixel values and uses a demand-driven update protocol to send update from the server only in response to an explicit request from the client. Therefore, VNC decouples the processing of application display commands from display updates generation and increases the probability of the system across various client platforms. SPICE [32]

is a new protocol developed by Red Hat which is similar to VNC but has audio support. Furthermore, the SPICE protocol is built in Quick Emulator (QEMU), so it is suitable for DaaS. However, 3D support is still limited in SPICE.

Many strategies have been proposed leveraging the semantic information of desktop to improve the performance of remote interaction. THiNC [7] intercepts low-level video driver commands and adopts a push mode to interact with client. Although it supports native multimedia playback, it suffers from performance degradation over multimedia content encoding. Muse [8] uses a window-aware updating mechanism to reduce display update traffic and response latency. It applies a new RFB-based protocol that supports window display and dynamic region encoding. This solution allows user to only view the application which is currently in use and quickly switch to other application windows to adapt to small device screen size and network condition. As a virtual desktop can have various guest operating systems and the semantic information is lost in VMM, those strategies are inappropriate in desktop virtualization context.

Winter et al. [15] developed a thin-client system where graphical output is captured through a hardware frame-grabber. The dedicated frame-grabber device in the system can only support one user at a time, implying that the architecture is neither scalable nor flexible to support multi-users. Simoens et al. [33] developed a hybrid encoding system that applies a heuristic algorithm to determine the amount of motion in a frame in order to switch codec. Unfortunately, the intrinsic performance of the algorithm and the video encoding prevent this solution from providing high resolution real time streaming.

Besides, classic remote access systems suffer performance degradation due to inefficient mechanisms while the aforementioned remote access approaches improve performance and interaction experience by implementing standalone architectures. As a drawback, those approaches incur high CPU load and are not transparent to the operating systems; hence they are not qualified for a remote desktop virtualization system.

In this paper, FastDesk is built on our previous work [23], which proposed a streaming based remote desktop virtualization system to provide high quality display and good interactive experiences. However, the previous system aims to connect to a single host and ignores the DaaS scenario. In view of this, the new work extends the system to support multi-tenant cloud environment.

2.2. VM placement strategies

In a remote virtualization desktop system, virtualization provides a great deal of benefits such as flexibility in provisioning, but the real challenge is how to achieve virtual machine placement efficiently. A lot of literatures lay emphasis on virtual machine placement, which is well known as a bin-packing problem. In view of the NP-hard nature of the problem [18–20], these works can be divided into two categories: greedy heuristic algorithms [17–21,34,35] and evolutionary heuristic algorithms [16,36].

Chowdhury et al. [19] proposed multiple modified versions of First-Fit-Decreasing (FFD) and Worst-Fit-Decreasing (WFD) to perform VM placement. And Bobroff et al. [21] utilized Next-Fit(NF), First-Fit(FF) and Best-Fit(BF) heuristics to obtain the near highest profit. Man and Kayashima [17,18] proposed a Pattern-based Virtual Desktop Allocation (PBA) algorithm, which leverages the CPU usage patterns of the virtual desktops to find the suitable server for a virtual desktop so as to reduce the waste of resource. Li et al. [20] presented a novel greedy heuristics: BRP, a Resource Balancing VM Placement algorithm that aims to maintain a balanced resource utilization among different dimensions of one server and minimize the waste of resources. Besides, two famous open source cloud computing platforms also adopted greedy heuristic algorithms.

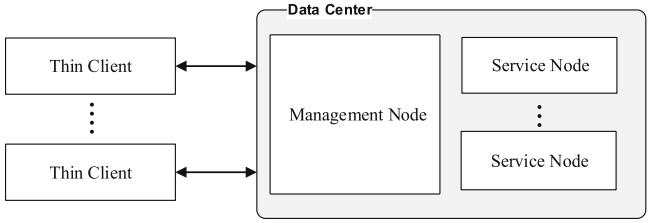


Fig. 1. High level architecture of FastDesk.

OpenStack [34] used the Chance algorithm, which randomly selects a server to place a new VM. Another option was Worst-Fit (WF) algorithm where a new VM chooses a server with the smallest load. Similar to OpenStack, Apache CloudStack [35] also used random placement solution but had some First-Fit-based VM placement algorithms. However, most of the introduced approaches only considered one-dimensional (i.e., CPU) bin-packing problems while VM placement problem is a multi-dimensional bin-packing problem [37]. Moreover, some of these solutions target to balance the resource workload among servers, which is a contrary objective to this paper.

With regard to evolutionary heuristics, Xu et al. [36] presented an improved genetic algorithm with fuzzy multi-objective evaluation. This approach conveniently combines possibly conflicting objectives. And Zhang et al. [16] proposed a unified genetic algorithm to solve virtual desktop placement problem which realizes the selection of data center and server simultaneously.

In this paper, we properly model the VM placement problem as an instance of the multi-dimensional bin-packing problem and propose a nature-inspired evolutionary heuristic method, modified Ant Colony Optimization (ACO) algorithm, to solve the VM placement problem to save the server resources as much as possible.

3. Architecture design

FastDesk can be used either in an internal network like lab and IT organizations, or as a DaaS platform. Fig. 1 shows the high-level architecture of FastDesk. The management node is a centralized control node in the system. The number of management nodes is not limited to one. It controls several service nodes. As the size of the system continues to grow, the number of management node can also be increased. Each service node is a physical machine that dynamically starts and stops multiple virtual machines according to the management server requests.

3.1. Service node

In order to provide high fidelity display and good user interactions in a desktop virtualization system, FastDesk service node combines virtualization with video-streaming. Fig. 2 displays an overview of such architecture. Thin clients are connected to the desktop virtualization server through Ethernet or Internet. Users' applications are executed in Guest OS virtualized by the server. The architecture of service node has the following features.

Transparency: On the server side, the display of a Guest OS is generated in the Guest OS's display driver and then rendered by a virtual display device. To be transparent to the Guest OS, FastDesk modifies the virtual display device that sits below the Guest OS such that FastDesk requires no modification of drawing functionality in the Guest OS, resulting in a simpler system that can work seamlessly with existing virtualization systems. FastDesk intercepts the displays rendered by the virtual display device and redirects them to the virtual loopback, instead of sending

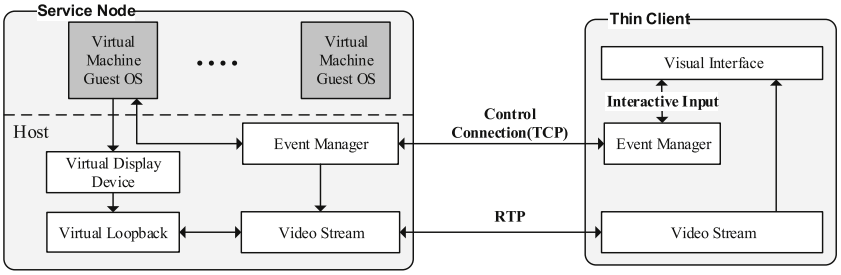


Fig. 2. Service node architecture.

them to the framebuffer. Virtual loopback is a module to create virtual video devices. With the virtual loopback, a process can read these devices as if they were ordinary video devices, which makes multimedia architecture easier compatible in virtualization environment.

Push Mode Streaming: The multimedia architecture is a video-stream pipeline mode. The video-stream in the server is mainly responsible for encoding the original display. Encoders such as H.264 [38] and WebM [39], have high performance in compression which reduces the network traffic to adapt to a low bandwidth network environment. To improve the response time of desktop interactions in the virtualized system, FastDesk uses a low-latency server-push display update mode which minimizes the synchronization costs between the client and the server. This choice was motivated by the traditional client-pull mode which requires client to send updating requests to server, therefore it does not fit a remote context. In fact, if video frames are generated faster than the rate at which client can send requests to server then the latency increases.

Diversified Displays: It becomes important to promise ubiquitous computing access in a cloud environment. The client may have different screen sizes and computing capabilities. To match these expectations, FastDesk decouples the original framebuffer size from the client display size. The display resizing is fully supported by the server which automatically resizes the display in the video-stream pipeline whenever a client reports a specific size.

3.2. Management node

The management node controls all the service nodes. Fig. 3 shows the components of management node and their interactions with service nodes.

The performance of service node can dramatically influence user experience. Limited CPU and memory resources may cause VM to react slowly. The screen will not update timely if network is congested. Therefore, it is critical for us to monitor all the CPU, memory and network utilization [40]. To facilitate the support of multi-tenant resource management, management node contains the following components.

Resource Monitor: It observes resource usage and availability of virtual machines on service node. The resource monitor periodically reports resource utilization queries to service nodes and records corresponding responses. The resource records of each node are to be provided to Resource Allocator.

Resource Allocator: To satisfy a user's request, a management node needs an allocator to allocate and place a virtual machine on an appropriate service node. The standard allocation and placement strategy must match two criteria: first, it meets Service-Level Agreement (SLA), for cloud providers commonly include an SLA in their contracts and penalties are applied when SLA violation occurs [41]. In this paper, we consider SLA violations as dissatisfaction of requirements and quality. To avoid SLA violations, resource reservation is set in each service node. Reserved resource is to be assigned to the virtual machines when it bursts into use. Second, the algorithm should be cost-efficient for cloud providers, which implies server resources utilization need to be increased as much as possible. Once a VM is started on a service node, the VM directly communicates with the client. So, the job of management node is only to accept the requests of creating new VMs and perform resource allocation and placement. The service nodes themselves will not interfere with each other.

4. VM placement algorithm

A remain problem is how to minimize the wastage of server resources, which depends heavily on the VM placement strategies. This section will cover a nature-inspired solution. After formulating the VM placement as a bin-packing problem, the proposed evolutionary heuristic algorithm is completely described.

4.1. Problem formulation

In the data centers, there are hundreds of servers in which VMs place to manage. The problem of VM placement is defined as a multiple dimensional vector bin-packing problem [37]. In this work, we use CPU and memory utilization as the two dimension. We do not count the network utilization assuming that the data centers have enough network bandwidth and FastDesk requires little bandwidth.

The following equations is used to calculate the wastage of the resources:

$$W_{pj} = \frac{T_{pj} - U_{pj}}{100}$$

$$W_{mj} = \frac{T_{mj} - U_{mj}}{100}$$

where W_{pj} denotes the CPU wastage of the jth server and W_{mj} denotes the memory wastage of the jth server. T_{pj} and T_{mj} denotes the threshold of the CPU and memory resources of the jth server. U_{pj} and U_{mj} represents the actual usage of the CPU and memory resources of the jth server.

Suppose we are given n VMs which is going to be placed on m servers. We make it that I as the set of n VMs and $i \in I, J$ as the set of m servers and $j \in J$. Let R_{pi} and R_{mi} denotes the CPU and memory demands of the ith VM. The binary variable y_i equals to 1 if the jth server is in use. And x_{ij} equals to 1 if the ith VM is assigned to the jth server. We can formalize the VM placement problem as:

$$\text{Min} \frac{\sum_{j=1}^{m} \left[y_j \times \frac{T_{pj} - \sum_{i=1}^{n} (x_{ij} \cdot R_{pi})}{\sum_{i=1}^{n} (x_{ij} \cdot R_{pi})} \right]}{\sum_{j=1}^{m} (y_j)}$$

$$\text{Min} \frac{\sum_{j=1}^{m} \left[y_j \times \frac{T_{mj} - \sum_{i=1}^{n} (x_{ij} \cdot R_{mi})}{\sum_{i=1}^{n} (x_{ij} \cdot R_{mi})} \right]}{\sum_{j=1}^{m} (y_j)}.$$

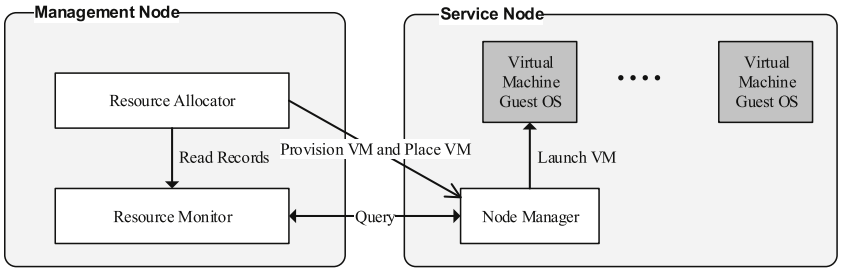


Fig. 3. Components in management node and interaction with service nodes.

Subject to:

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i \in I \tag{1}$$

$$\sum_{i=1}^{n} R_{pi} \cdot x_{ij} \leq T_{pj} \cdot y_j, \quad \forall j \in J \tag{2}$$

$$\sum_{i=1}^{n} R_{mi} \cdot x_{ij} \leq T_{mj} \cdot y_j, \quad \forall j \in J \tag{3}$$

$$y_j, x_{ij} \in \{0, 1\}, \quad \forall i \in I \text{ and } \forall j \in J. \tag{4}$$

Constraint (1) assigns a VM to one server. Constraint (2) and constraint (3) ensure that each server has enough CPU and memory resources to host the assigned VMs. Constraint (4) describes the problem.

4.2. Ant colony optimization algorithm

As the VM placement problem is formulated as a vector bin-packing problem, which is NP-hard [42]. We adopt a modified ACO heuristics algorithm to solve our problem. The pseudocode of the algorithm is presented in Algorithm 1.

Algorithm 1 The ACO Heuristics Algorithm

Require:
The list of VMs & their resources demand
The list of Servers & their resources
Output:
P_s: A Pareto set of mapping strategies
for t = 1 **to** iteration times **do**
  for i = 1 **to** N_a (number of ants) **do**
    Randomly sort the server list
    while {unplaced VMs} $\neq \emptyset$ **do**
      Get a server from the list
      for each VM fits into the server **do**
        Calculate η_{ij} (Equ. 5)
        Calculate P_{ij} (Equ. 8)
      end for
      Pick a random value q
      if $q < q_0$ **then**
        Exploitation
      else
        Exploration
      end if
    end while
    if S_c is not dominated by S_o in P_s **then**
      Add S_c to P_s
      for each S_o in P_s dominated by S_c **do**
        Remove S_o from P_s
      end for
    end if
  end for
  Update τ_{ij}^{local} (Equ. 9)
  Update $\tau_{ij}(t+1)$ (Equ. 10)
end for
return P_s

In each iteration, each ant tries to find a map from VMs to servers, that is to say, the paths from VMs to servers. Once an ant chooses the path, it will deposit pheromones along the path. Ants tend to choose the way with strong pheromone. The Pareto set contains the mapping results. There may be various mapping in the set as long as a mapping is not dominated by the others. Any one of the results is a valid solution to the VM placement problem.

A solution x dominates another solution y if both conditions are true [43,44]:

1. The wastage of solution x is not less than that of y in CPU and memory.
2. The wastage of solution x is strictly more than that of y in CPU or memory.

The key points of applying Ant Colony Optimization algorithm to the VM placement problem includes the following two aspects:

heuristic information: The ant chooses the path depends on both the pheromone and the heuristic information. The heuristic information only considers the wastage of servers, excluding the pheromone.

pheromone update: When finding path, ants tend to follow the pheromone. So we update the pheromone in each iteration to guide the ants to construct a result in the following iterations. Once an ant constructs a path from a VM to a server, or we come up with a Pareto set after each iteration, we update the local and global pheromone according to these information.

4.3. Constructing a solution

For each VM that can be chosen to fit into current server, we first calculate the desirability, aka the heuristic information, according to the server utilization showing in Eq. (5). The equation takes both CPU and memory wastage into consideration.

$$\eta_{ij} = \frac{1}{\sum_{j=1}^{m} W_{pj}} \times \frac{1}{\sum_{j=1}^{m} W_{mj}}. \tag{5}$$

Then we calculate the factor according to (6), where α is a fixed parameter determine the importance of pheromone. τ_{ij} refers to the pheromone on the path from VM i to server j. $q \in [0, 1]$ is a random value, compare it with a fixed parameter q_0. If q is less than q_0, we find the VM with the maximum factor, this process is called exploitation which is described in Eq. (7). Otherwise, we do the exploration process.

$$F_{ij} = \alpha \times \tau_{ij} + (1 - \alpha) \times \eta_{ij} \tag{6}$$

$$i = \begin{cases} \text{argmax}_{s \in \Omega_j} F_{sj}, & q < q_0 \\ s, & \text{otherwise.} \end{cases} \tag{7}$$

In exploration, each VM is assigned a probability, according to Eq. (8). The probability is the factor of VM i over the sum of all VMs.

Then we randomly pick a VM according to its probability.

$$P_{ij} = \begin{cases} \dfrac{F_{ij}}{\sum_{s\in\Omega_j} F_{sj}}, & i \in \Omega_j \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Once a placement solution is constructed and this solution is not dominated by any other solution in the Pareto set, we add this solution to the Pareto set. If there are some solutions in the Pareto set is dominated by this one, remove them from the set.

4.4. Local pheromone update

After each iteration, the local pheromone is updated according to Eq. (9). ρ is a fixed parameter refers to the pheromone trail decay rates.

$$\tau_{ij}^{local} = (1-\rho)\times\tau_{ij}(t) + \sum_{k=1}^{N_a}\Delta\tau_{ij}^{k}(t) \tag{9}$$

$\Delta\tau_{ij}^{k}(t)$ is the quantity of the pheromone of ant k:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{1}{\sum_{j=1}^{m} W_{pj}\times\sum_{j=1}^{m} W_{mj}}, & (i,j)\in S_k(t) \\ 0 & \text{otherwise} \end{cases}$$

$S_k(t)$ is the solution given by the ant k. The more resources are wasted on server j, the less pheromone will be left on the path. Hence in following iterations, this path will less likely to be chosen.

4.5. Global pheromone update

Similar to local pheromone update, the global pheromone is updated as well in each iteration according to Eq. (10). N_s means number of solutions. For each solution in the Pareto set, the pheromone is updated according to the mapping result on every path.

$$\tau_{ij}(t+1) = \tau_{ij}^{local} + \sum_{s=1}^{N_s}\Delta\tau_{ij}^{s}(t) \tag{10}$$

$\Delta\tau_{ij}^{s}(t)$ is the quantity of the pheromone of solution s:

$$\Delta\tau_{ij}^{s}(t) = \begin{cases} \dfrac{1}{\sum_{j=1}^{m} W_{pj}\times\sum_{j=1}^{m} W_{mj}}, & (i,j)\in S_s(t) \\ 0, & \text{otherwise.} \end{cases}$$

5. Implementation

This section elaborates the details of the FastDesk system implementation, including server-side, client-side and the remote display protocol.

We implemented a prototype server node based on QEMU-KVM[1] (version 1.2.0) and a client by extending a simple VNC Viewer[2] . Since FastDesk only hooks virtual display device under the Guest OS, no modification is required to applications and operating systems. The rendered images are written into a virtual loopback by using the standard Video4Linux interface. To support virtual loopback, we use a *v4l2loopback*[3] as a kernel module that creates virtual video devices. The FastDesk virtual loopback utilizes the *v4l2loopback* interfaces and multiplexes its resources. Fig. 4 shows the pipeline details of the video-stream. The server uses a display scaler to resize the output of an image in the virtual loopback in order to respond to the client size requirement. Then pipeline is directed to encoding. We choose the ×264 videocodec[4] , an open source H.264 encoder. Since we are encoding a live streaming environment, we have to minimize encoder delay. To reduce the latency of x264 codec , the *zerolatency* flag is enabled, which disables *rc-lookahead* and *b-frames*. Those two parameters are mainly used for offline encoding, as they incur higher latency. Once a frame is encoded, the pipeline starts to encapsulate it into Real-time Transport Protocol (RTP) packets, which are then sent to the client through UDP (User Datagram Protocol).

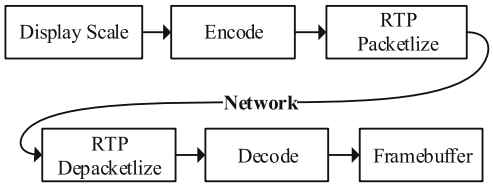


Fig. 4. Pipeline of video-stream.

The client reads the RTP packets received and pushes them to the decoder. Since video decoding is an extremely CPU-intensive task, especially at higher resolutions, we use hardware-accelerated video decoding to allow CPU to concentrate on other tasks. For the test client machine called Raspberry Pi, OpenMAX[5] is used to accelerate video decoding.

The communication between server and client is achieved through TCP (Transmission Control Protocol) connections. The event manager on the client handles interaction inputs such as keystrokes and mouse clicks and sends them to the server. The event manager on the server handles the request and forwards the commands to the video-stream, and the interactions to the Guest OS.

The remote display protocol is designed based on RFB. RFB operates in a client-pull updating mode. The display screen is updated and sent each time the client sends a *FramebufferUpdateRequest*. Since FastDesk adopts the server-push mode, the client no longer sends *FramebufferUpdateRequest* to the server. To support diversified displays in the server, an extension message *SetScreenSize* is added to the protocol. Whenever the client needs to update display size, *SetScreenSize* is sent from the client to the server.

6. Evaluation

To demonstrate the effectiveness of FastDesk, we first test different scenarios and evaluate the overall performance. We then conduct a comparison study with a number of widely used platforms such as X, VNC, Microsoft Remote Desktop, ThiNC, Spice and TeamViewer. Some commercial platforms like PCoIP, Citrix MetaFrame are not public available, and hence we did not compare them in this paper. Besides, we also compare the proposed nature-inspired evolutionary algorithm with other four classical heuristic VM placement methods. Finally, we evaluate the effectiveness of management component by analyzing potential SLA violation in multiple scenarios.

Since the Microsoft Remote Desktop Protocol is not open to public and there is no official RDP client on Linux, there are only some reverse-engineer work to use RDP in Linux. In our testbed, we use rdesktop[6] (version 1.8.3), a famous third-party RDP client on Linux, as the RDP client to test the performance of RDP. Some advanced features like support for Direct 2D, 3D and HD video

1 wiki.qumu.org/KVM.
2 www.realvnc.com/download/open.
3 github.com/umlaeute/v4l2loopback.
4 www.videolan.org/developers/x264.html.
5 www.khronos.org/openmax.
6 https://www.rdesktop.org.

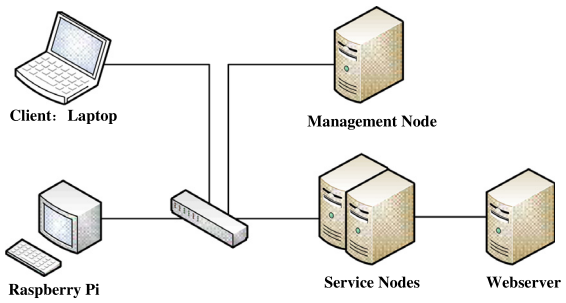


Fig. 5. Experimental testbed.

playback is lacking. Considering that we should not force users to a specific platform, we choose the open-source client rdesktop so the Linux users can still access to remote Windows computers.

6.1. Experimental setup

We conduct the experiments using an isolated network. Fig. 5 shows our testbed setup. The management node has a 2.66 GHz Intel Core i7-920 processor and 6GB of RAM. The service nodes are two identical machines both with a 2.67 GHz Intel Xeon X5650 processor and 8 GB of RAM. For all the clients, we use a 2.0 GHz Intel Core II laptop with 1Gbyte of RAM and a Raspberry Pi, a credit-card-sized single-board computer featuring a 700 MHz ARM11 processor and 512 MB of memory. They are applied to all the experiments. We consider two network environments: LAN and WAN. The LAN is a 100 Mbps network with ideal latency while the WAN is a 10 Mbps network with 66ms RTT emulated by ns-2[7] .

Since 100 Mbps network is commonly used, we define the LAN speed as 100 Mbps. If FastDesk performs well in 100 Mbps network, we conclude it has the same performance in Gigabit Ethernet.

We design four different scenarios with a 1024 × 768 display resolution to represent slow-motion, high-motion, low and high interactive scenarios, respectively. Since FastDesk is transparent to the Guest OS, we run both Linux and Windows Guest OS on FastDesk server in order to provide a fair comparison with other systems. The four scenarios are listed below.

- **Office:** Perform a sequence of actions in Openoffice on Linux platforms and Microsoft Office on Windows platforms. These actions include typing, creating objects, editing tables, etc.
- **Browsing Web:** Browse a sequence of 30 web pages containing a mix of texts and images. The browser we choose is Mozilla Firefox which is supported by both platforms. The web pages are saved on a local web server.
- **Photo Editing:** For the photo editing, we apply a sequence of filters such as blur, red eyes removal, sharpen. The GNU Image Manipulation Program and Adobe Photoshop are used on the Linux and Windows platforms, respectively.
- **Video Playback:** We play an H.264 codec movie clip at 24 frames per second (fps) in fullscreen. The video player used is MPlayer 1.0rc4 on both the Linux and Windows platforms. The original clip size is 853 × 480.

All scenarios are recorded with Xnee[8] by the client to ensure that clients provide the same inputs each time. Every scenario lasts at least 5 min and is tested 3 times. To perform scenarios on the Guest OS, we choose Ubuntu 12.04 for Linux, and Windows 7 for Windows platform.

To evaluate the application environment differences, we use common configuration options whenever possible. We set display to 32-bit color, RDP to LAN settings when in the LAN environment and WAN settings in the WAN environment. Any other settings remain unchanged.

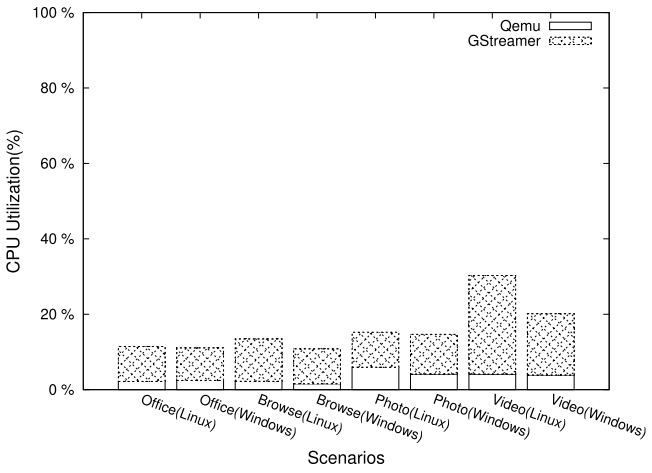


Fig. 6. CPU utilization in server.

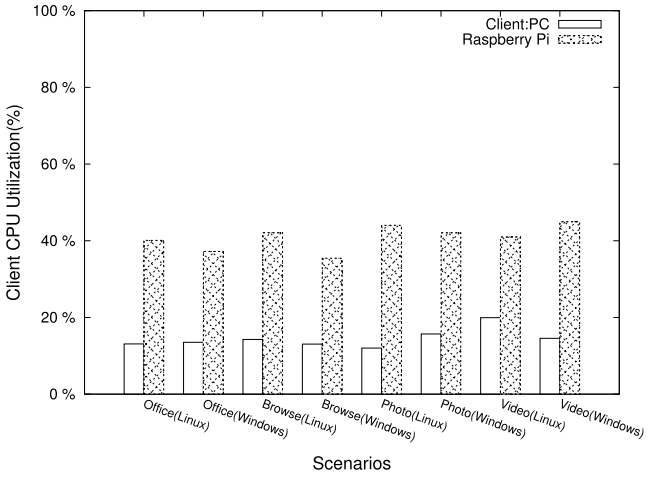
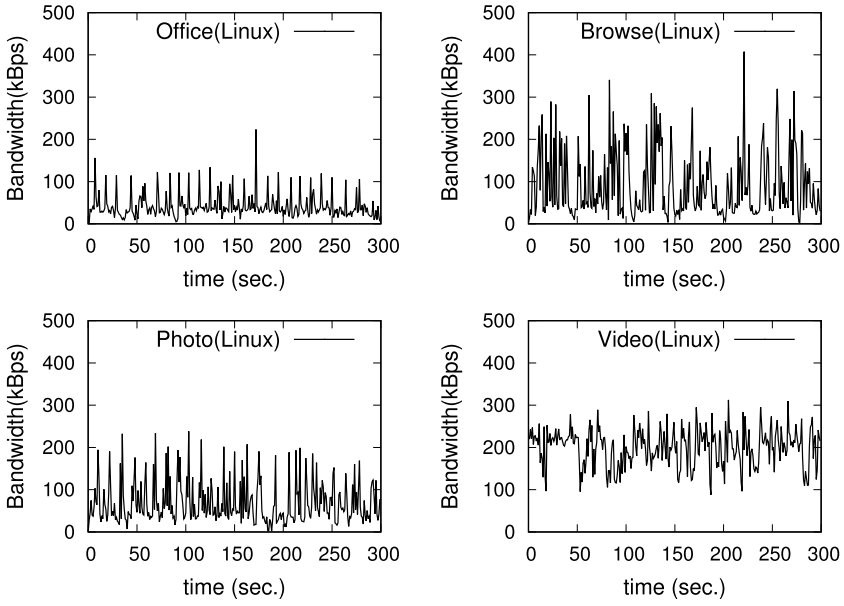


Fig. 7. CPU utilization in client.

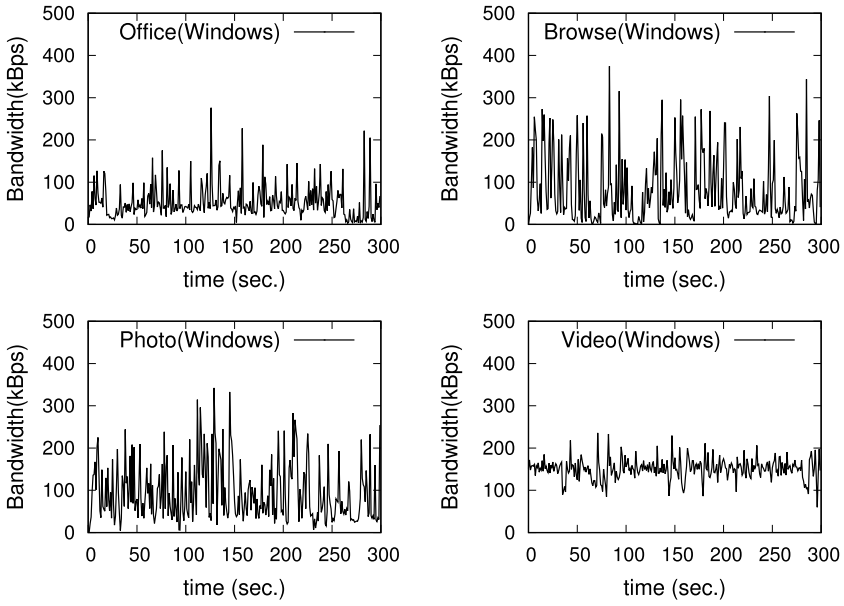
6.2. Overall performance

First we run one scenario at a time. Figs. 6 and 7 show the average CPU utilization for both client and server. On the server side, the Office scenario has the lowest CPU utilization with 12.86% on Linux against 12.57% on Windows, since it contains a large number of slow-motion display updates. The Video playback scenario as the representation of high-motion display has the highest CPU utilization among all the scenarios that is around 10% higher than slow-motion scenarios. Browsing Web and Photo Editing have some high CPU-intensive actions, but human interactions lowers it down. It is worth noting that the CPU usage of GStreamer[9] is quite higher than that of QEMU. This is due to the CPU intensive video encoding task. In the future we may use GPU acceleration

[7] http://nsnam.sourceforge.net/wiki/index.php/Main_Page.
[8] http://www.gnu.org/software/xnee.
[9] https://gstreamer.freedesktop.org.



(a) Scenarios in Linux Guest OS.



(b) Scenarios in Windows Guest OS.

Fig. 8. Bandwidth in four scenarios and two operating systems.

to perform the hardware encoding part in order to reduce the CPU use. It is also critical to keep the CPU utilization as low as possible for the client device. Fig. 7 shows that the CPU utilization on the client has no relevance to the scenarios and that the average CPU utilization of a client is around 14.5%. Leveraging hardware-accelerated video decoding, the CPU utilization of the Raspberry Pi device is lower than 50% in average; this means that a limited thin client can still be used on FastDesk architecture.

Fig. 8 shows the bandwidth of each 5 min scenario. The bandwidth costs of both web browsing scenario change suddenly, since web browsing contains a lot of clicks which makes the display vary frequently. Fig. 9 shows the average bandwidth cost for each scenario and demonstrates that FastDesk requires no more than 2 Mbps for control and image transmission. Even though the high-motion scenario needs more screen updates, the Video Playback scenario requires only 1.56 Mbps. Since Video Playback is the most bandwidth consumption scenario, a comparison of Video Playback bandwidth with other systems will be described in Section 6.4.

To test the scalability of FastDesk, we run 2, 4 and 6 scenarios together on a server. We select web browsing, photo editing and video playback scenario on both Windows and Linux system to conduct the test. Two Linux systems for photo editing and video playback are used in these two scenarios. The two Linux and Windows virtual machines perform Photo Editing and Video playback are used in the four scenarios. And all of them are used in the 6 scenarios. Fig. 10(a) and (b) show the CPU utilization and bandwidth cost for the server over 5 min. The CPU utilization of the server is 30.97%, 69.03% and 82.76% for 2, 4 and 6 scenarios, respectively. When running 4 scenarios, the CPU utilization is 2.23 times higher than with 2, and 1.21 with 6. This demonstrates the scalability of FastDesk. Fig. 10(a) demonstrates that FastDesk server affords at least 6 clients together in various scenarios without any performance degradation in our testbed, since the server CPU utilization

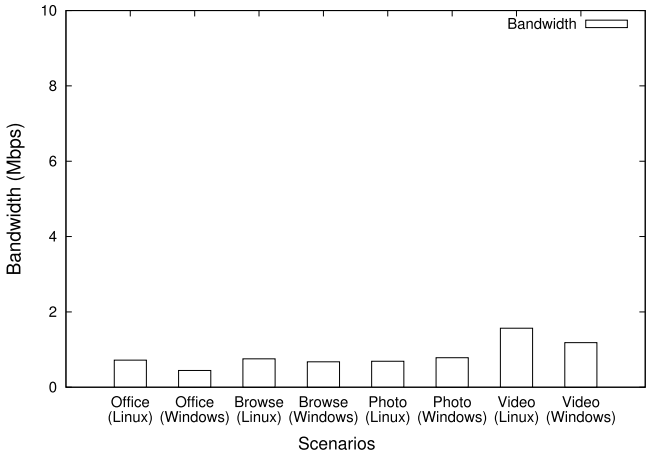
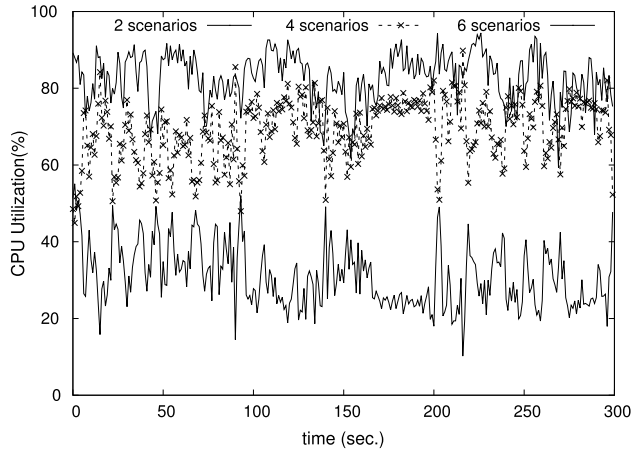
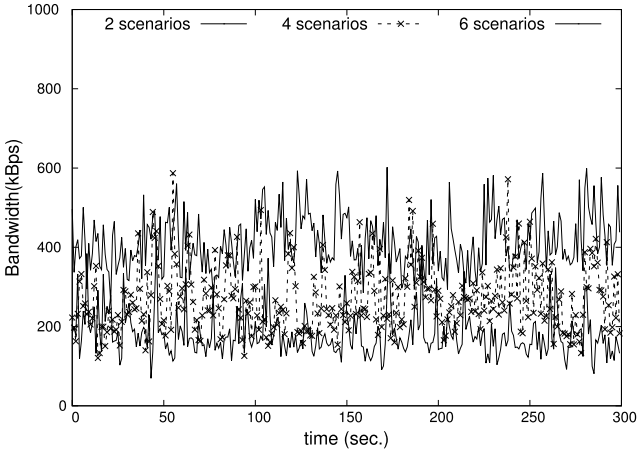


Fig. 9. Average bandwidth usage in four scenarios and two operating systems.



(a) CPU utilization in server with 2, 4 and 6 running scenarios.



(b) Bandwidth in server with 2, 4 and 6 running scenarios.

Fig. 10. CPU utilization and Bandwidth in server with 2, 4 and 6 running scenarios.

has never reached 100% at any time. Fig. 10(b) reveals that even with 6 connections on a single server, the bandwidth does not exceed 5 Mbps.

We also compare the CPU utilization of FastDesk against other platforms. The results are shown in Fig. 11. Fig. 11(a) shows the CPU utilization on servers. As mentioned before, the CPU utilization of FastDesk is quite higher than others because GStreamer consumes lots of CPU resources to encode video stream, while other protocols use their own less CPU-consuming encoding methods. However we can use GPU acceleration to perform hardware encoding in future to reduce the utilization of CPU. The detailed information has been shown in Fig. 6 before. Because the RDP protocol is a closed source protocol built in Windows and there is no official RDP support in Linux, the RDP data of the four scenarios are blank. Fig. 11(b) presents the CPU utilization on clients. In most cases, VNC performs the best since it transfers raw data so that the client does not need or only needs little CPU resources to decode network data. However, the problem is that VNC requires more bandwidth than others. TeamViewer performs the worst. Another interesting observation is that the CPU utilization of FastDesk in different scenarios are almost the same while other clients vary. Thanks to the hardware decoding ability, FastDesk uses almost the same CPU in different scenarios.

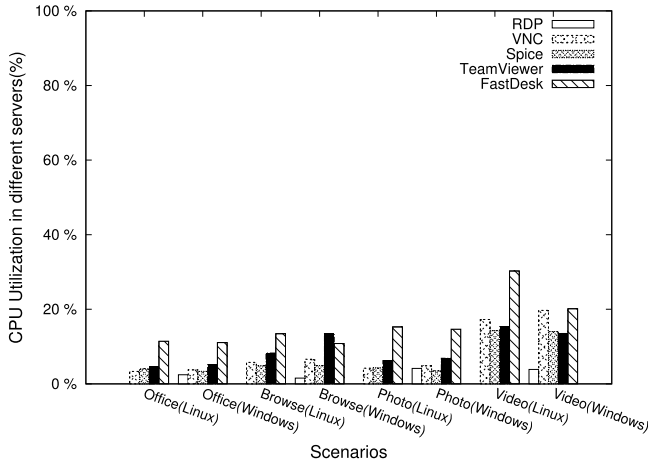
6.3. Interaction response analysis

Good response time is the key to overall user satisfaction, especially in highly interactive scenarios. Therefore, we mainly focus on the Photo Editing Scenario. We record the time between a mouse click or keystroke and a complete screen update for the corresponding signal. A full response time consists of the processing time of both client and server, the transmission time of the remote display protocol and the application execution time. Since some actions in photo editing last for a relative longer application execution time, we also test Photo Editing scenario on a local PC as a baseline in order to exclude the impact of the application execution time. Fig. 12 shows the average latency of a sequence of actions on a Photo Editing scenario over both LAN and WAN environments. VNC has the slowest response time for LAN at 635 ms, and X11 is the worst for WAN at 1935 ms. They are much slower than others due to client-pull mode or bandwidth limitation they face.

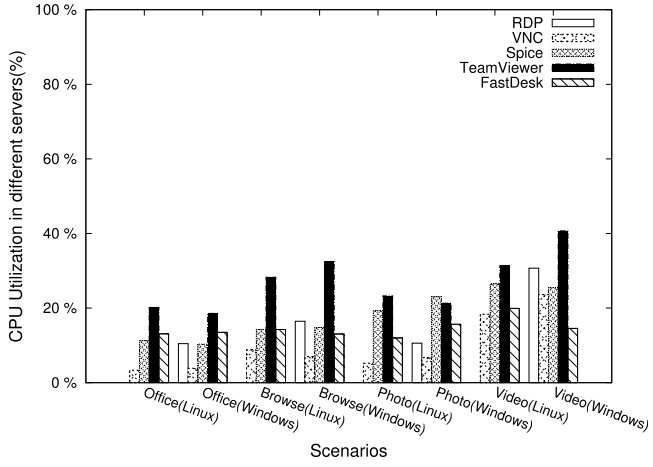
Fig. 12 shows that FastDesk has a 300 ms latency on a LAN environment and does not suffer much performance degradation in a WAN environment. Although RDP achieves nearly native performance in a LAN environment, it suffers a major degradation in a WAN environment and is 1.8 times slower than FastDesk, and TeamViewer in Windows is alike. FastDesk provides the fastest response time in WAN environment over all the platforms. THiNC achieves a similar result than FastDesk, as they both adopt the server-push mode.

6.4. Video quality analysis

Multimedia performance is measured using a benchmark based on video quality [12], which takes both playback delays and frame drops into consideration. Video quality $VQ(P)$ at a given playback rate P is calculated according to Formula (11). In actual situation, a video is watched at full playback rate $IdealFPS(P)$ which is equal to the specified playback rate P. Then, the data transfer rate can be calculated as the total transferred data $Data(P)$ divided by the total playback time $PlaybackTime(P)$. Slow motion scenario is the ideal situation where all the data $Data(slowmo)$, of which frames would not be dropped, are transferred to the client, and the playtime $PlaybackTime(slowmo)$ does not increase or decrease. The video quality is actually the ratio of the data transfer rate at the full playback rate and the data transfer rate in slow motion scenario. In the test, we play the video at 1 fps $IdealFPS(slowmo)$ as the ideal situation. 100% video quality is the optimal quality, which means all video frames are played at real-time speed. 50% video quality means that either half of the video data are dropped or the video



(a) CPU utilization in different servers.



(b) CPU utilization in different clients.

Fig. 11. CPU utilization on various platforms.

plays two times slower than expected.

$$VQ(P) = \frac{\frac{Data(P)/PlaybackTime(P)}{IdealFPS(P)}}{\frac{Data(slowmo)/PlaybackTime(slowmo)}{IdealFPS(slowmo)}}. \quad (11)$$

Fig. 13 shows the video quality results on both LAN and WAN environments. X11, VNC, RDP, Spice and TeamViewer deliver a very poor video quality. TeamViewer in Linux has the worst quality for LAN at only 7.84%, and X11 has the worst quality for WAN at no more than 2%. These systems suffer from their mechanisms and algorithms. These algorithms are unable to keep up with the speed of screen updates, leading to frame dropping or longer playback time. X11 will ensure that you will get all the needed data. If the network speed is not fast enough, the remote side will slow down the video. The remote side will not play the next frame until it gets all the needed data. This results in a longer playback time. Thus, the video quality of X11 drops. VNC drops one third of the video quality in the WAN environment, as it uses a client-pull mode which expects the client to send screen update requests to the server. In a higher latency WAN environment, the rate of update request will be slower than the video playback rate, such that some of the frames are dropped. However, THiNC and FastDesk achieve almost ideal video quality. THiNC reaches 97.62% and 71.42% video quality in LAN and WAN environment, respectively. FastDesk gets 95% video quality in a LAN environment, and 92.94%, the highest

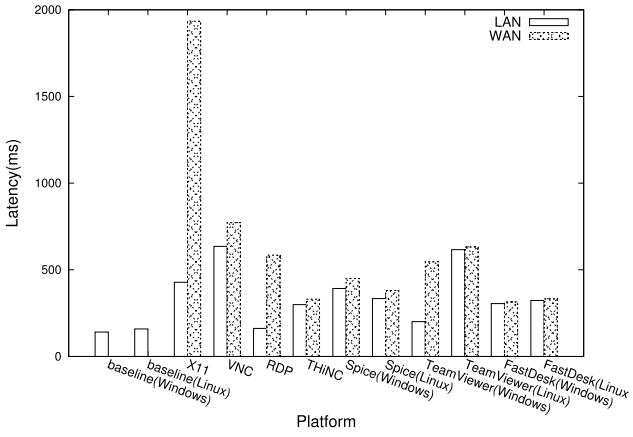


Fig. 12. Latency in photo editing scenario.

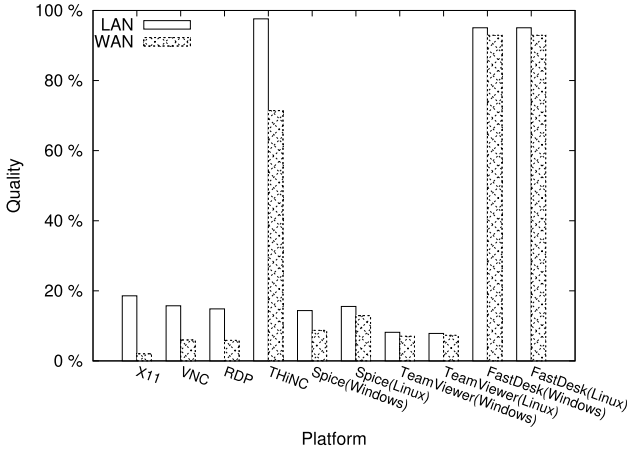


Fig. 13. Video quality in video playback scenario.

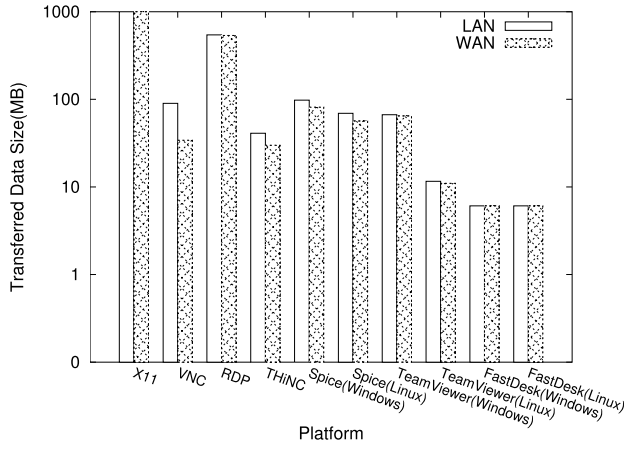


Fig. 14. Total data transferred during video playback.

one, in a WAN environment benefiting from using a server-push mode.

Fig. 14 shows the amount of data transferred during the video playback for each system. FastDesk is transparent to the Guest OS, so the video performance and data sizes are almost the same in these two operating systems. Among all the systems, FastDesk is the least bandwidth consuming for video playback. It sends

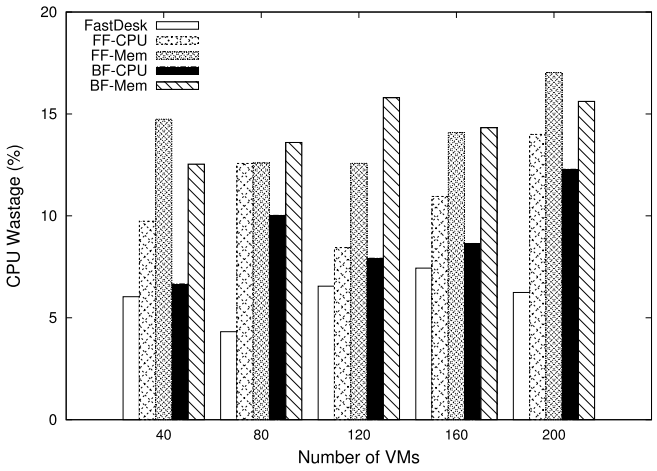


Fig. 15. CPU wastage of different algorithms.

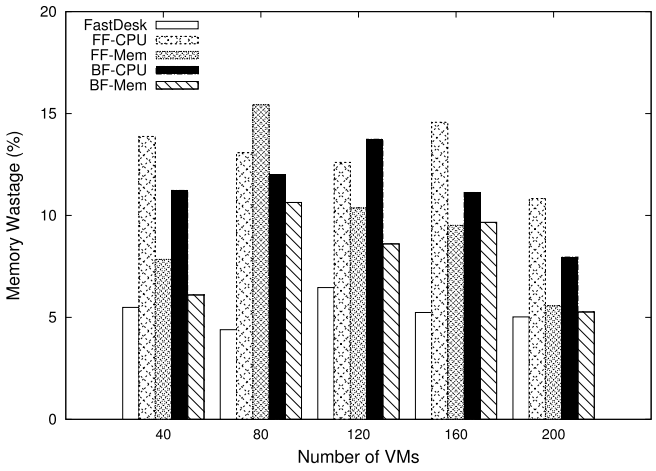


Fig. 16. Memory wastage of different algorithms.

6.09 MB and 6.08 MB of data in LAN and WAN environment, respectively, with 94% video quality on average. Although THiNC's video quality is slightly better than FastDesk in a LAN environment, it needs more bandwidth. Since THiNC intercepts drawing commands in the driver of Guest OS, it has to contain more semantic display information which increases the data size. Some protocols have their own compression methods, like RFB and RDP. The default option for the VNC client is to automatically negotiate the compression level. It will detect the network status to decide which one to use. We leave it to the client to choose the compression method. RDP uses disk cache to speed up and to save network bandwidth. We have turned on the option to save the network bandwidth. In spite of this, they still transfer too much data.

6.5. Server resources wastage analysis

In order to show the large-scale virtual desktops placement scenarios and evaluate the performance of the proposed algorithm more exactly, the experiment is conducted using simulation on a PC with a 3.20 GHz Intel Core i5-4570 processor and 8 GB of RAM. The VM data are collected using the results of the above experiments. The parameter settings of the algorithm are: iteration times $= 50$, $N_a = 4$, $\alpha = 0.4$, $\rho = 0.3$, $\tau_0 = 0.01$, $q_0 = 0.9$. Besides the proposed algorithm, we also implement FirstFitDecrease-CPU, FirstFitDecrease-Memory, BestFitDecrease-CPU, BestFitDecrease-Memory [45] to compare the server resource utilization.

The resource wastage is the key to evaluate the VM placement algorithms. There are five different scenarios in the experiment, whose VMs' numbers are separately 40, 80, 120, 160, 200.

We compare the CPU wastage and the Memory wastage of the proposed algorithm against other four heuristics, and Figs. 15 and 16 show the results under those five algorithms. As shown in the figures, the modified Ant Colony Optimization algorithm always get the minimal resource wastage against other four strategies. Meanwhile, the results show that the resource utilization of our algorithm varied little in five different scenarios.

6.6. SLA violation analysis

In order to measure the performance of our allocation strategy, we assume that an SLA violation occurs when the CPU is overloaded or there are no free memory or network bandwidth. We allocate as many VMs as possible using a resource reservation policy and randomly run 8 scenarios. The average of SLA violation ratio is depicted on Fig. 17. It shows that the SLA violation increases

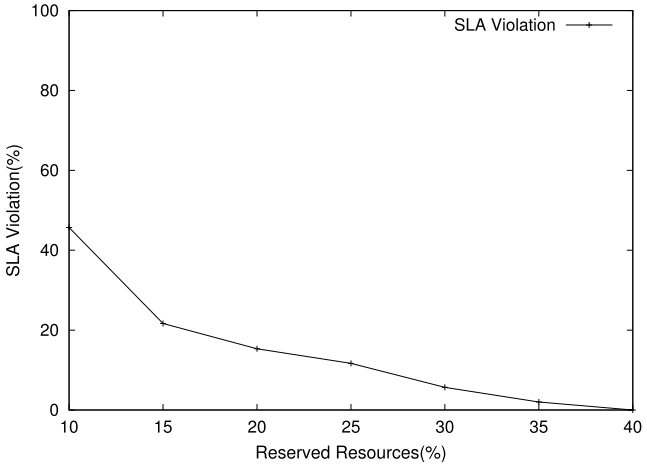


Fig. 17. Average SLA violation.

dramatically when the reserved resource is lower than 20% and reaches up to 52% when the reserved resource is 10%.

CPU and network resources are shared among all the VMs. If a user is doing some CPU-intensive work inside a VM such as compiling software, the available CPU resource for other VM are lower. Reserving 10% of CPU means that all the CPU utilization should not exceed 90%. This can be easily exceeded if multiple users are doing CPU-intensive work simultaneously.

When reserving more than 20% system resources, SLA violations happen less frequently, which means that 20% is the optimal ratio of reserved resource in our experiment.

However, searching the optimal ratio of reserved resource offline is not practical, since the ratio of SLA violation is changing in a large scale over different scenarios. In practice, the selection of reserved resources has to be done dynamically based on the fluctuation of the current workload. We will investigate this in our future work.

6.7. 3D Support

Another obstacle in desktop virtualization is gaming. One remarkable advantage of DaaS is that end users do not need to upgrade their hardware to use the cutting-edge software, which is usually done by service providers using virtualization. However, the existing virtualization technology like KVM and Xen has poor

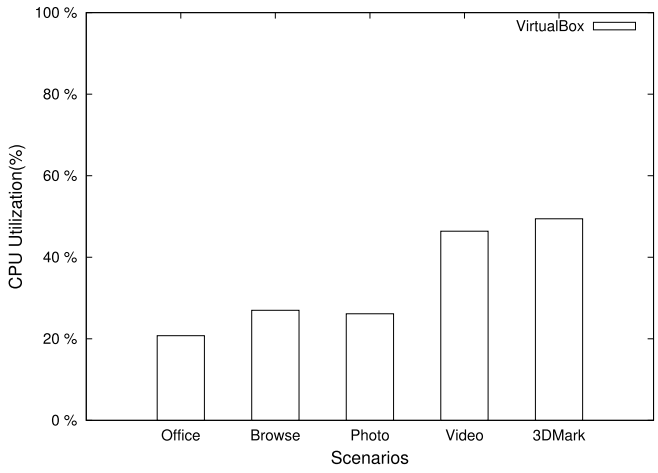


Fig. 18. CPU Utilization of VirtualBox.

GPU support. Although the host may have some powerful graphics cards, the guest machines cannot benefit from it. Consequently, the performance of some modern games becomes extremely low, some games which requires high version of DirectX cannot even run.

VMware and VirtualBox have their own guest 3D drivers to enable graphics card acceleration by passing the drawing calls to host to make use of the real hardware. Hence, we port our FastDesk to VirtualBox to enable 3D support. VirtualBox uses Chromium [46] to create a virtual graphics card, then passes the graphics API commands to the host. At the host side, a kernel module will receive the commands and process them to the result, then send back to the virtual machine.

We test the 3D support on a server with an i7-965 processor and a Nvidia 9800GT graphics card. With the help of VirtualBox, we can finally run some 3D program. Besides, we use 3DMark 03 to test the performance. It is worth noting that the 3D support is still an experimental feature of VirtualBox and some programs may not work properly.

Fig. 18 shows the CPU utilization of VirtualBox in the five different scenarios. In this figure we have not distinguished the CPU utilization of VirtualBox and GStreamer like before in Fig. 6. The CPU utilization is the whole usage of the VM, the VM itself and GStreamer module which processing the screen update in real time. We only want to check the performance of 3D processing in our system by comparing it with other scenarios. From the result we can find that although the CPU utilization of 3DMark is the highest with 49.43%, it is only slightly higher than the CPU utilization the video playback. So it is feasible to use FastDesk to deliver 3D program to end users.

7. Conclusion and future work

This paper presents FastDesk, a novel remote desktop virtualization system for multi-tenant. By intercepting the virtual display driver under Guest OS, FastDesk works seamlessly with common applications, desktops and operating systems without any modification. FastDesk takes advantage of a video-stream architecture and an efficient video codec to provide high quality display updates with low latency and bandwidth consumption. Meanwhile, it proposes a heuristic Ant Colony Optimization based algorithm, which could solve the VM placement problem and reduce the server resources wastage. With a server-side display scaling, FastDesk can also support small screen devices. Another interesting feature of FastDesk is its support for multi-tenant environment through a management component.

We measure the overall performance of FastDesk in various scenarios over a variety of network environments and compare FastDesk with widely used remote desktop systems. The results show that FastDesk delivers good user experience with low CPU utilization on both client and server. FastDesk has good scalability as the result demonstrates that a server affords at least 6 clients without degrading performance. Moreover FastDesk achieves good response time in both LAN and WAN environments and outperforms all the other systems in the WAN environment. FastDesk gives the same top video quality as THiNC but uses less bandwidth. Although video streaming has a higher CPU cost than virtual machine, we may use GPU acceleration to perform hardware encoding and hence reduce the CPU burden. Besides, our proposed VM placement algorithm always gets the minimal resource wastage against other classical heuristics.

We plan to use GPU to do encoding so that servers can have more CPU resources for VMs. Another future research direction is to investigate which VM placement strategy would perform better in each scenario and how to allocate resources according to users' past behaviors. In addition, we can expand our system to support distributed data centers.

Acknowledgments

This work was supported by National Key Research & Development Program of China (No. 2016YFB1000502), NSFC (No. 61672344, 61525204) and Shanghai Science and Technology Innovation Fund (No. 16511102502). Ruhui Ma is the corresponding author. We would like to thank the anonymous reviewers for their helpful comments.

References

[1] J. Shuja, A. Gani, K. Bilal, A.U.R. Khan, S.A. Madani, S.U. Khan, A.Y. Zomaya, A survey of mobile device virtualization: Taxonomy and state of the art, ACM Comput. Surv. (CSUR) 49 (1) (2016) 1.
[2] M. García-Valls, T. Cucinotta, C. Lu, Challenges in real-time virtualization and predictable cloud computing, J. Syst. Archit. 60 (9) (2014) 726–740.
[3] G. Lai, H. Song, X. Lin, A service based lightweight desktop virtualization system, in: 2010 International Conference on Service Sciences, (ICSS), IEEE, 2010, pp. 277–282.
[4] wikipedia.org, Remote Desktop Virtualization, 2016. URL https://en.wikipedia.org/wiki/Desktop_virtualization.
[5] J. Li, Y. Jia, L. Liu, T. Wo, Cyberliveapp: A secure sharing and migration approach for live virtual desktop applications in a cloud environment, Future Gener. Comput. Syst. 29 (1) (2013) 330–340.
[6] L. Deboosere, B. Vankeirsbilck, P. Simoens, F. De Turck, B. Dhoedt, P. Demeester, Cloud-based desktop services for thin clients, IEEE Internet Comput. 16 (6) (2012) 60–67.
[7] R.A. Baratto, L.N. Kim, J. Nieh, Thinc: a virtual display architecture for thin-client computing, in: Proceedings of the 20th ACM Symposium on Operating Systems Principles 2005, SOSP, 2005, pp. 277–290.
[8] W. Yu, J. Li, C. Hu, L. Zhong, Muse: a multimedia streaming enabled remote interactivity system for mobile devices, in: Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia, ACM, 2011, pp. 216–225.
[9] T. Richardson, Q. Stafford-Fraser, K. Wood, A. Hopper, Virtual network computing, IEEE Internet Comput. 2 (1) (1998) 33–38.
[10] B. Tritsch, Microsoft Windows Server 2003 Terminal Services, Microsoft Press, 2004.
[11] C. Border, The development and deployment of a multi-user, remote access virtualization system for networking, security, and system administration classes, ACM SIGCSE Bull. 39 (1) (2007) 576–580.
[12] J. Nieh, S. Yang, N. Novik, Measuring thin-client performance using slow-motion benchmarking, ACM Trans. Comput. Syst. (TOCS) 21 (1) (2003) 87–115.
[13] C. Taylor, J. Pasquale, Improving video performance in VNC under high latency conditions, in: 2010 International Symposium on Collaborative Technologies and Systems, (CTS), IEEE, 2010, pp. 26–35.
[14] L. Deboosere, J. De Wachter, P. Simoens, F. De Turck, B. Dhoedt, P. Demeester, Thin client computing solutions in low-and high-motion scenarios, in: Networking and Services, 2007, ICNS, IEEE, 2007 pp.230–235.
[15] D. De Winter, P. Simoens, L. Deboosere, F. De Turck, J. Moreau, B. Dhoedt, P. Demeester, A hybrid thin-client protocol for multimedia streaming and interactive gaming applications, in: Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video, ACM, 2006, p. 15.

[16] J. Zhang, L. Zhang, H. Huang, X. Wang, C. Gu, Z. He, A unified algorithm for virtual desktops placement in distributed cloud computing, Math. Probl. Eng. 2016 (2016).
[17] C.L.T. Man, M. Kayashima, Virtual machine placement algorithm for virtualized desktop infrastructure, in: 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, (CCIS), IEEE, 2011, pp. 333–337.
[18] C. LeThanhMan, M. Kayashima, Desktop workload characteristics and their utility in optimizing virtual machine placement in cloud, in: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligent Systems, Vol. 1, (CCIS), IEEE, 2012, pp. 333–337.
[19] M.R. Chowdhury, M.R. Mahmud, R.M. Rahman, Study and performance analysis of various VM placement strategies, in: 2015 16th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, (SNPD), IEEE, 2015, pp. 1–6.
[20] X. Li, A. Ventresque, J. Murphy, J. Thorburn, A fair comparison of vm placement heuristics and a more effective solution, in: 2014 IEEE 13th International Symposium on Parallel and Distributed Computing, (ISPDC), IEEE, 2014, pp. 35–42.
[21] N. Bobroff, A. Kochut, K. Beaty, Dynamic placement of virtual machines for managing sla violations, in: 10th IFIP/IEEE International Symposium on Integrated Network Management, 2007, IM'07, IEEE, 2007, pp. 119–128.
[22] A. Kochut, K. Beaty, On strategies for dynamic resource management in virtualized server environments, in: 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007., MASCOTS'07, IEEE, 2007, pp. 193–200.
[23] J. Wu, J. Wang, Z. Qi, H. Guan, Sridesk: a streaming based remote interactivity architecture for desktop virtualization system, in: 2013 IEEE Symposium on Computers and Communications, (ISCC), IEEE, 2013, pp. 000281–000286.
[24] A. Kivity, Y. Kamay, D. Laor, U. Lublin, A. Liguori, kvm: the Linux virtual machine monitor, in: Proceedings of the Linux Symposium, Vol. 1, 2007, pp. 225–230.
[25] S. Agarwal, A. Nath, Desktop virtualization and green computing solutions, in: Proceedings of the Second International Conference on Soft Computing for Problem Solving, (SocProS 2012), Springer, 2014, pp. 1439–1449 December 28–30, 2012.
[26] R. Scheifler, J. Gettys, The X window system, ACM Trans. Graph. 5 (2) (1986) 79–109.
[27] citrix.com, Citrix Metaframe, Virtual Desktops, 2016, URL http://www.citrix.com.
[28] teradici.com, Pc-over-ip technology explained — teradici pcoip solutions, 2016. URL http://www.teradici.com/pcoip-technology.
[29] teamviewer.com, Teamviewer - free remote control, remote access & online meetings, 2016. URL http://www.teamviewer.com/.
[30] vmware.com, Vmware horizon (with view) 2016. URL http://www.vmware.com/products/horizon-view.
[31] amazon.com, Amazon workspaces product details, 2016. URL https://aws.amazon.com/workspaces/details/.
[32] spice space.org, the simple protocol for independent computing environments protocol, 2016. URL http://www.spice-space.org/.
[33] P. Simoens, P. Praet, B. Vankeirsbilck, J. De Wachter, L. Deboosere, F. De Turck, B. Dhoedt, P. Demeester, Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices, in: Telecommunication Networks and Applications Conference, 2008., ATNAC 2008., IEEE, Australasian, 2008, pp. 391–396.
[34] openstack.org, Openstack scheduling, 2016. URL http://docs.openstack.org/mitaka/config-reference/compute/scheduler.
[35] cloudstack.apache.org, Granular global config parameters, 2016. URL https://cwiki.apache.org/confluence/display/CLOUDSTACK/Granular+Global+Config+Parameters.
[36] J. Xu, J.A. Fortes, Multi-objective virtual machine placement in virtualized data center environments, in: Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'L Conference on & Int'L Conference on Cyber, Physical and Social Computing, (CPSCom), IEEE, 2010, pp. 179–188.
[37] Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in cloud computing, J. Comput. System Sci. 79 (8) (2013) 1230–1242.
[38] J. Kimball, T. Wypych, F. Kuester, Low bandwidth desktop and video streaming for collaborative tiled display environments, Future Gener. Comput. Syst. 54 (2016) 336–343.
[39] D. Mukherjee, H. Su, J. Bankoski, A. Converse, J. Han, Z. Liu, Y. Xu, An overview of new video coding tools under consideration for VP10: the successor to VP9, in: SPIE Optical Engineering+ Applications, International Society for Optics and Photonics, 2015 95991E:1–95991E:12.
[40] A. Berryman, P. Calyam, M. Honigford, A.M. Lai, Vdbench: A benchmarking toolkit for thin-client based virtual desktop environments, in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science, (CloudCom), IEEE, 2010, pp. 480–487.
[41] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (5) (2012) 755–768.
[42] J. Békési, G. Galambos, H. Kellerer, A 5/4 linear time bin packing algorithm, J. Comput. System Sci. 60 (1) (2000) 145–160.
[43] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Vol. 16, John Wiley & Sons, 2001.
[44] K. Deb, K. Miettinen, Multiobjective Optimization: Interactive and Evolutionary Approaches, Vol. 5252, Springer Science & Business Media, 2008.
[45] E.G. Coffman Jr, M.R. Garey, D.S. Johnson, Approximation algorithms for bin packing: a survey, in: Approximation Algorithms for NP-Hard Problems, PWS Publishing Co., 1996, pp. 46–93.
[46] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P.D. Kirchner, J.T. Klosowski, Chromium: a stream-processing framework for interactive rendering on clusters, in: ACM Transactions on Graphics, Vol. 21, (TOG), ACM, 2002, pp. 693–702 3.

Tao Song received his M.Eng. degree on Software Engineering major from Shanghai Jiao Tong University and B.Eng. degree on Automation from China University of Mining and Technology. He is currently a Ph.D. candidate in Shanghai Jiao Tong University in China. His research area includes data center networking, cloud computing and swarm intelligence.

Jiajun Wang received the B.Eng. and M.Eng. degrees from Shanghai Jiao Tong University in 2013 and 2016. Currently, he is working as a software engineer at Microsoft. His research interests mainly include GPU virtualization and system security.

Jiewei Wu received his B.Eng. and M.Eng. degrees from Shanghai Jiao Tong University in 2011 and 2014. Currently, he is working as a software engineer at Baidu. His research interests include operating system and virtualization technology.

Ruhui Ma received his Ph.D. degree in computer science from Shanghai Jiao Tong University, China, in 2011. He has worked as a postdoc at Shanghai Jiao Tong University (2012 and 2013), at McGill University, Canada (2014), respectively. He is an Assistant Professor in the Department of Computer Science and Engineering at SJTU. His main research interests are in virtual machines, computer architecture and network virtualization.

Alei Liang received his B.Eng. and M.Eng. degrees from Hefei University of Technology in 1991 and 1997, and Ph.D. degree from Shanghai Jiao Tong University in 2002. He is currently an Associate Professor in the school of Electronic, Information and Electrical Engineering at Shanghai Jiao Tong University. His research interests include Computer Architecture, Networking and Swarm Intelligence.

Tao Gu received his Bachelor degree from Huazhong University of Science and Technology, and M.Sc. from Nanyang Technological University, Singapore, and Ph.D. in computer science from National University of Singapore. He is currently an Associate Professor in the School of Computer Science and Information Technology at RMIT University. His current research interests include mobile and pervasive computing, wireless sensor networks, distributed network systems, sensor data analytics, and online social networks. He is a Senior Member of the IEEE and a Member of the ACM.

Zhengwei Qi received his B.Eng. and M.Eng. degrees from Northwestern Polytechnical University, in 1999 and 2002, and Ph.D. degree from Shanghai Jiao Tong University in 2005. He is currently a Professor at the School of Software, Shanghai Jiao Tong University. His research interests include program analysis, model checking, virtual machines, and distributed systems.