Future Network and MobileSummit 2013 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2013
ISBN: 978-1-905824-36-6

Poster Paper

Round-trip time mitigation through speculative display updating for applications rendered in the cloud

Bert VANKEIRSBILCK[1], Kevin DE WOLF, Pieter SIMOENS[1,2],
Filip DE TURCK[1], Bart DHOEDT[1]
[1]*Ghent University, Department of Information Technology (INTEC), Internet Based Communication Networks and Services (IBCN) – iMinds*
Gaston Crommenlaan 8 bus 201, 9050 Ghent, Belgium
Tel.: +32 9 331 49 38, Fax: +32 9 331 48 99, Email: bert.vankeirsbilck@intec.ugent.be
[2]*Ghent University College, Dept. INWE, Valentyn Vaerwyckweg 1, 9000 Ghent, Belgium*

Abstract: The advantages of cloud computing have revitalized interest in thin client computing. In this thin client computing approach, an application is executed on the thin client server, which in the cloud computing paradigm is part of a cloud environment. The user interacts with a viewer, that acts as a service-hatch: forwarding the user events over the network to the server and accepting the returned graphical updates. The major downside to this approach is that at least one network round trip time (RTT) is required to present the application output that results from the user's actions. In this paper a novel speculative display mechanism is proposed to mitigate the RTT requirement. The mechanism relies on online server side profiling of the graphical output that follows user events, that after synchronization with the viewer is used to speculatively update the viewer's screen content upon receipt of user events. This way, the impression is created that the network delay is decreased.

Keywords: Cloud computing, Speculative, Remote application execution, Latency, Round-trip time

1. Introduction

The advent of cloud computing has introduced new possibilities to employ the thin client computing paradigm, that is based on networked protocols to forward user input from the viewer to the server, and to return the graphical application output for presentation to the user on his device. This traditional thin client approach implies that at least one network round trip time (RTT) is required to present the application output that results from the user's actions. Wide Area Networks (WAN) and mobile networks typically exhibit larger latencies, making the RTT the major influence on the quality experienced by the user.

In this paper a novel speculative display mechanism is proposed to mitigate the network RTT. The mechanism relies on online server side profiling of the graphical output that follows user events to dynamically construct and maintain a connected state graph. In this state graph, a node represents a state that is linked to other states via user events and their related graphical updates. This state graph is synchronized with the viewer, that knowing the current state, can simply look up incoming user events to speculatively update the screen with the graphical updates to present the expected next state. This way, from a user perspective, a RTT can be avoided, at the possible expense of mispredictions leading to invalid screen content. The user events must be sent to the server to generate the correct application output and to compare it to the speculatively displayed content. This allows to update the state graph and its links, to verify the viewer displayed content to assess if it is necessary to apply corrections, and to correct the state of the viewer if he would have diverged from the actual state the application is in.

Using the proposed speculative display mechanism, the impression is created that the network delay is decreased, leading to a higher Quality of Experience (QoE) for the user. However, since the assessment of the QoE constitutes a broad study on its own, in this paper the focus is on objective metrics of the speculative display mechanism such as the measured RTT or reaction speed and processing overhead.

2. Related work

We alleviate on prior work [1], where we explored the potential of using a static cache in thin client computing, and assessed the repetitiveness of screen updates.

In [2], the authors propose speculative thin client operation, focusing on viewer side changes to maintain compatibility with existing server implementations and thin client protocols. They have shown predictability of screen updates for both Virtual Network Computing (VNC) [3] and Remote Display Protocol (RDP) [4] and apply a simple Markov system for the prediction at viewer side, that relates series of user and screen events to following screen events.

In [5], a server-based adaptive display pre-fetching mechanism is proposed that consists of pre-executing the possible subsequent user events on the server and sending the related graphics to the viewer. On reception of the user input, the matching graphical update is presented to the user. This way, using spare server computation resources, spare bandwidth on the network and spare memory on the client, round trip times are effectively avoided.

3. Algorithm

The logic for the speculative display system has been designed to be asymmetric such that the server has control over the cache contents at the viewer side. The main advantage of this approach is that the viewer remains computationally simple.

3.1 Viewer side algorithm

At the viewer side, the algorithm is very simple. Specifically, on receipt of a mouse event or a key stroke, the frame buffer updates that correlate to the hotspot that corresponds to the user event in the current state are drawn on the screen.

3.2 Server side algorithm

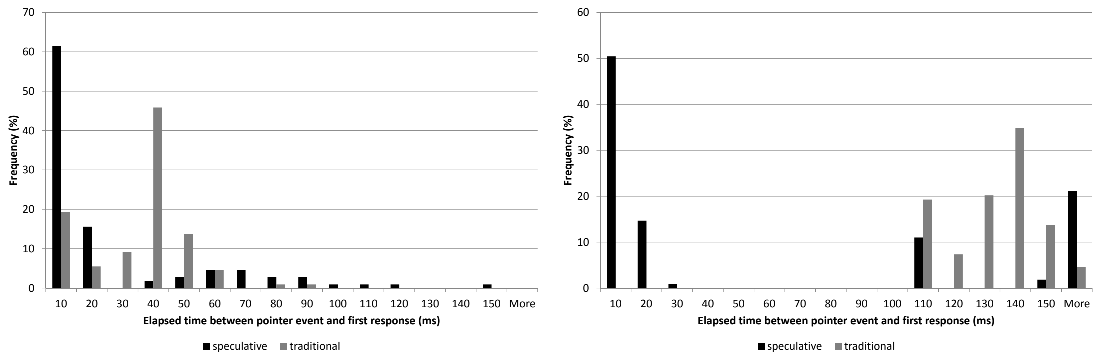
At the server side, the viewer's decisions are mimicked and compared to the actual graphical output of the application. Several cases can occur:
The viewer has approximately drawn the correct content on the screen:
The potential small errors are corrected on screen as well as in the cache. The viewer is also informed of these content corrections to keep the caches synchronized. *The frame buffer updates related to the corresponding hotspot found in the current state differ largely from the actual application output:*
Other states are checked for hotspots corresponding to the user event for which the related frame buffer updates resemble the application output. If a matching hotspot is

found in another state, it is assumed that the previous state should have been the state in which the matching hotspot was found. Consequently, the previous state is corrected by altering the target of the hotspot that had taken the system to the incorrect state to the found matching state. Also, the state the viewer believes to be in, is corrected to the state known by the server.

Otherwise, no matching hotspot is found in any state, triggering the creation of a new hotspot for the current state. For key strokes, this hotspot is defined by the key stroke signature itself. For mouse events, this implies creating an area containing the mouse pointer location, that is composed using the information available in the server.

4. Experimental results

The speculative display mechanism has been tested using *gedit*, for which sixteen actions were recorded. These actions consisted of opening the different menus in the application by clicking on the menu and closing it by clicking the menu again. This yields seven actions, for the menus 'File', 'Edit', 'View', 'Search', 'Tools', 'Documents' and 'Help'. For all of these actions, an alternative action was defined to open the menu the same way, but closing it by clicking in an unrelated area in the application. The final two actions were obtained by opening the 'Help' menu, selecting the 'About' item, and closing the corresponding dialog box using the 'Close' button and closing with the 'X' button in the window decorator. This set of actions was specifically defined as series of mouse events that cause a transition from a given state to itself over other states. For example, the filemenu-open-close action exists of a mouseclick on the file menu, leading to the expansion of the menu as a second state, followed by a second mouseclick on the file menu causing the collapse of the menu returning to the initial state. This approach allows randomization of the actions. For our experiments, we have created scenarios by uniformly distributed random drawing of actions from this set.



(a) no network latency (b) 50 ms network latency

Fig. 1: Comparison of first responses for the traditional thin client protocol and for the speculative display mechanism

Figure 1 shows the speedup in reactivity acquired by the speculative system in comparison to the traditional thin client system. Figure 1(a) shows the impact when no network latency is configured. Where the traditional system typically spends 30 to 50 ms to reply to a user event, the speculative display system provides an answer within 10 ms. However, when no prediction can be made, 60 to 70 ms elapse, indicating some overhead at the server side to compare frame buffer updates with the ones in the cache. The contrast between the speculative display system and the traditional thin client system is more apparent in Fig. 1(b), for which the results were obtained over a network configured to exhibit a latency of 50 ms, resulting in a minimum RTT of 100 ms for the traditional system. Here, independent of the network latency, the speculative display mechanism responds to the majority of the user events within 10 ms.

5. Conclusions

In this paper a server-centric network-latency mitigating mechanism is proposed, that augments remote application rendering in the cloud by speculatively showing estimated application output to the user, in spite of creating an impression for the user that the underlying network latency does not influence the reactivity of the system. Besides the presentation of the algorithms, necessary alterations to the traditional thin client computing architecture are detailed. The experimental results show that the reactivity of the system is spectacularly improved irrespective of the network latency present. The algorithm at the viewer side is fairly simple, leading to speculative responses within 10 ms. The results also show that little overhead is induced by the system, both concerning overhead synchronization messaging and server and viewer CPU load amounting to 2% and 1% CPU load respectively.

As part of future research, we see opportunities to evaluate the impact of prediction accuracy, which is not yet investigated in this work. Although the user might have the impression that the application responds well, he might also be confused by incorrect content on screen.

References

[1] B. Vankeirsbilck, P. Simoens, J. De Wachter, L. Deboosere, F. De Turck, B. Dhoedt, and P. Demeester, "Bandwidth optimization for mobile thin client computing through graphical update caching," in *Australasian Telecommunication Networks and Applications Conference (ATNAC)*, pp. 385 – 390, December 2008.

[2] J. R. Lange, P. A. Dinda, and S. Rossoff, "Experiences with client-based speculative remote display," in *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ATC'08, (Berkeley, CA, USA), pp. 419–432, USENIX Association, 2008.

[3] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual network computing," *IEEE Internet Computing*, vol. 02, no. 1, pp. 33–38, 1998.

[4] Microsoft Corporation, "Windows Remote Desktop Protocol (RDP)." http://msdn2.microsoft.com/en-us/library/aa383015.aspx.

[5] M. Sumalatha, S. Sridhar, and G. Satish, "A novel thin client architecture with hybrid push-pull model, adaptive display pre-fetching and graph colouring," *International Journal of Ad hoc, Sensor and Ubiquitous Computing (IJASUC)*, vol. 3, pp. 67 – 77, june 2012.