Utilizing Multiple Virtual Machines in Legacy Desktop Applications

Matt Piotrowski
UC Berkeley
CS 267
Parallel Computing

The Problem

- Personal computers will soon have many cores.

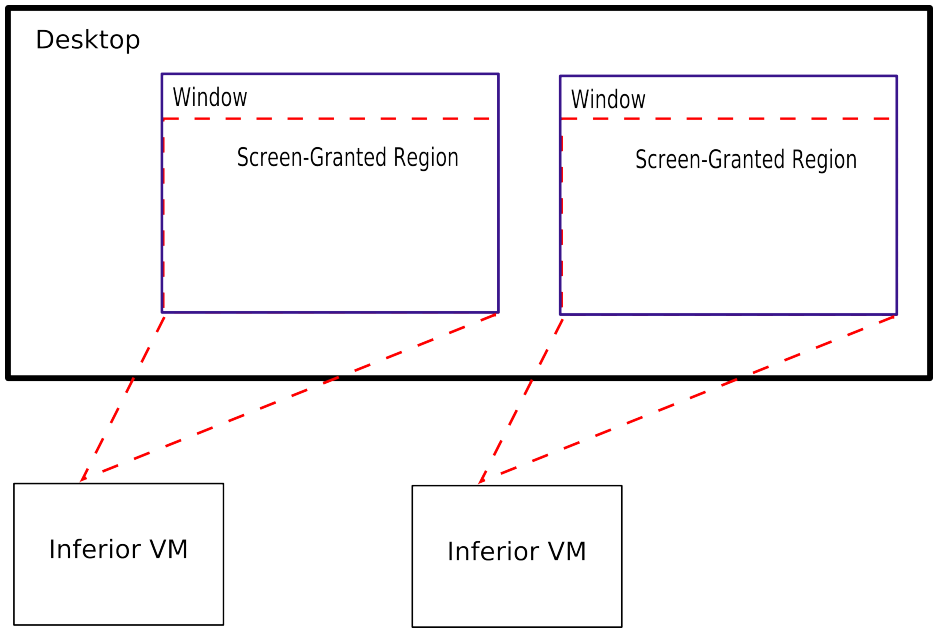
- One of the ways to utilize these many cores is to run many virtual machines on them, as discussed in [1], among others.

- However, legacy desktop applications haven't been written to take advantage of virtual machines.

[1] Rose Liu, Kevin Klues, Sarah Bird, Steven Hofmeyr, Krste Asanovic, John Kubiatowicz. Tessellation: Space-Time Partitioning in a Manycore Client OS. In Proceedings of HotPAR '09.

Our Solution

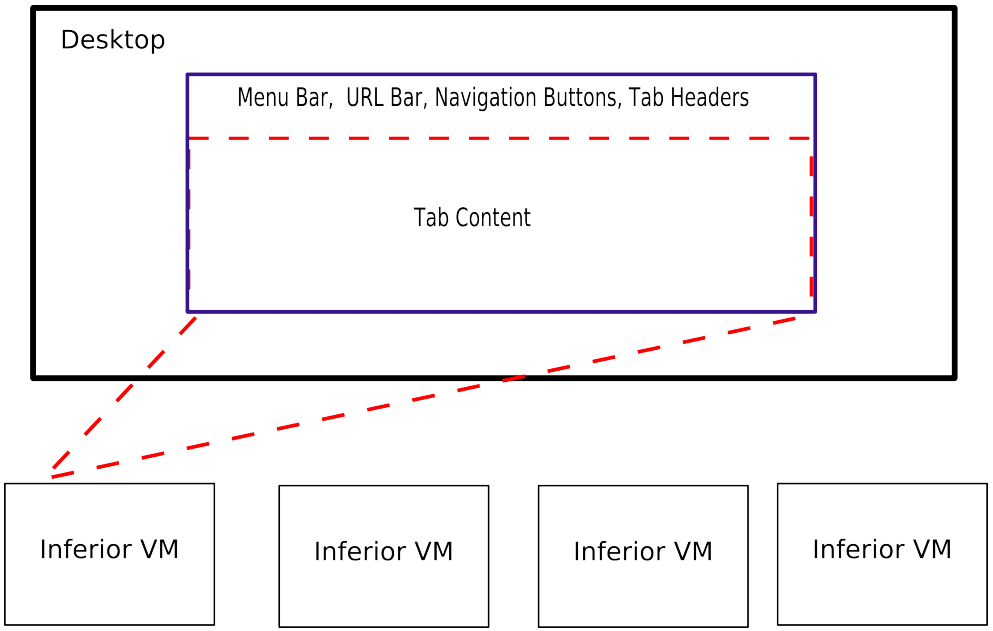
- We create a new operating system primitive that allows an application to run legacy desktop code in virtual machines.
- The graphical output from these virtual machines is transparently mapped into the windows of the application.



- In this example, an application has created two VM's (called inferior VM's) and screen-granted their output to the lower portion of its windows.

Web Browser

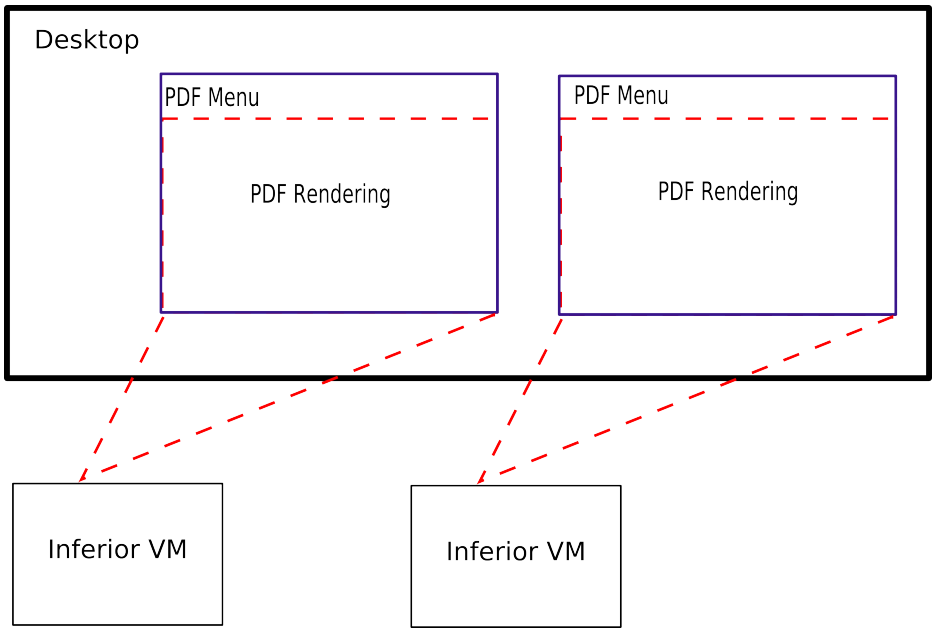
- We created a web browser where each tab is rendered in its own VM.
- Each VM is running WebKit, a large legacy codebase for rendering web pages.



- In this example, there are 4 tabs open. The current tab receives the current screen-grant.
- Additional benefits of per-tab VM rendering include increased reliability and security.

PDF Viewer

- We created a PDF viewer where each document is rendered in its own VM.
- Each VM is running evince, a large legacy codebase for rendering PDF documents.

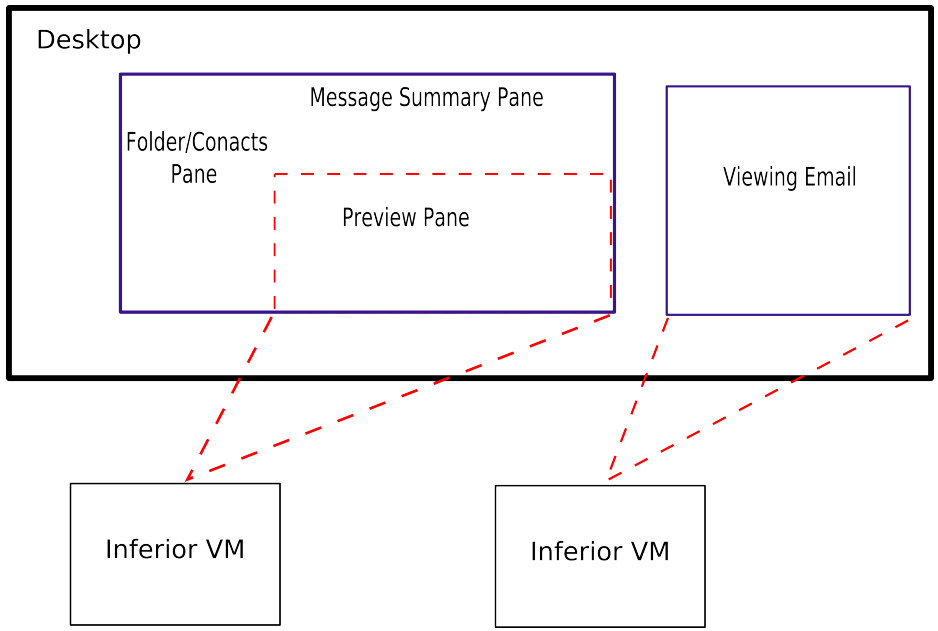


- In this example, two documents are being displayed and everything below the menu bar has been screen-granted to VM's.
- Enhanced reliability and security is also obtained.

Other Apps

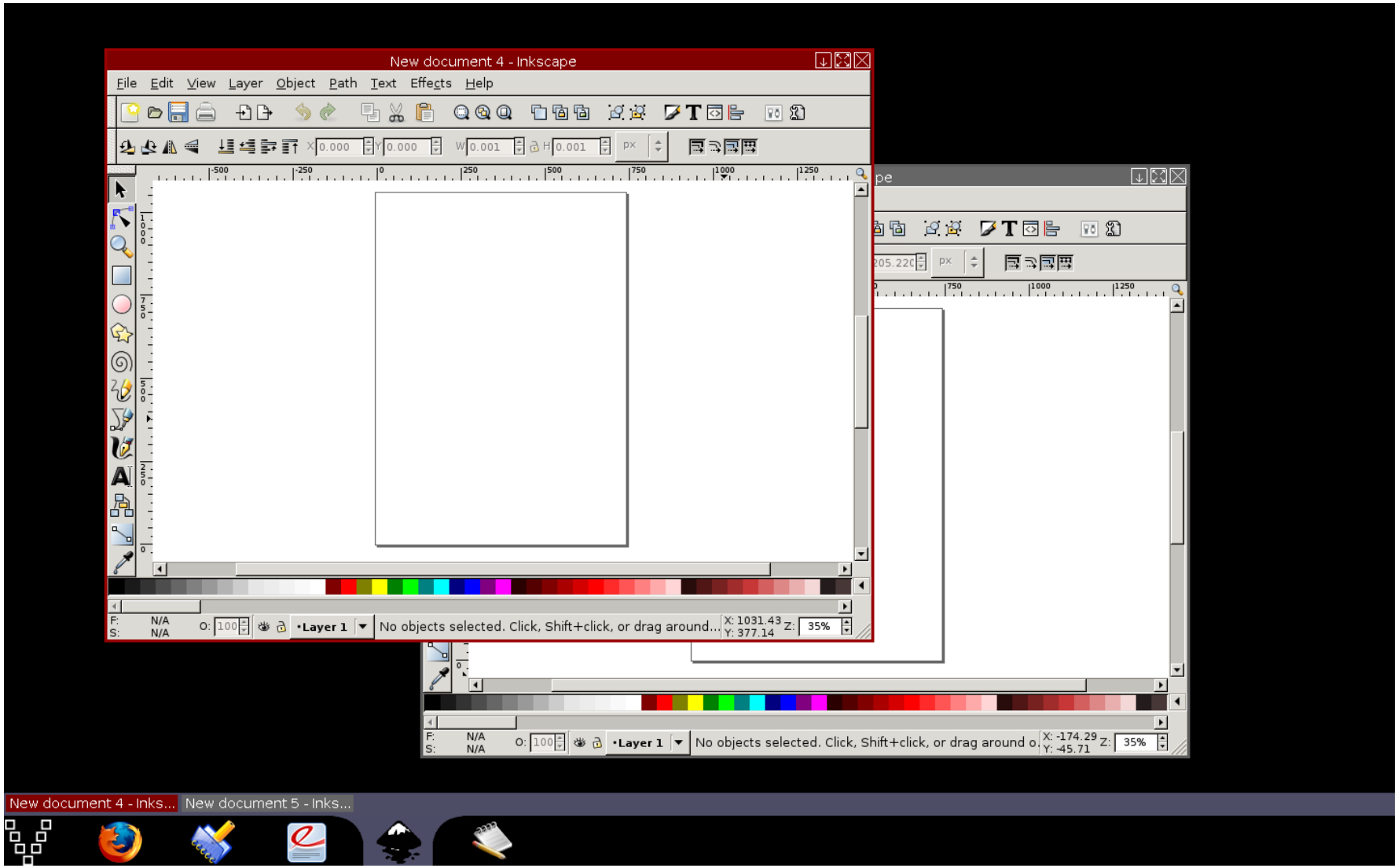
- The screen-grant mechanism lends itself well to many desktop applications. An email client, office suite, and movie player can easily be adapted to use screen-grants.

- Below is the design of an email client using inferior VM's and screen grants.



OS as Primitive User

- The OS itself can use our new primitive to run every application in its own VM and screen-grant parts of the desktop to them.



Overhead

- Ran 40 instances of WebKit displaying the Alexa Top 40 websites, with WebKit running in a VM and WebKit running in a process. Overhead of 48MB per instance for the VM case.
- Worth noting that we share the zero page within and across VM's but we could potentially greatly reduce the memory overhead with a copy-on-write VM fork, as discussed in [2]
- Startup time of VM: we pre-execute the VM to point of code launch; takes about 800 milliseconds.

[2] Carl Waldspurger. Memory Resource Management in VMWare ESX Server. In 5th Symposium on Operating Systems Design and Implementation.

Limitations/Future Work

- Applications that rely on 3D hardware acceleration won't work in a VM since access to this hardware is not provided. However, if 3D developers return to software-based rendering utilizing multicore, these applications would work

- Implement the designs of the email client, movie player, and office suite.

- Explore hints an application may be able to provide to indicate the nature of an inferior VM it is creating.