White paper on Thin Client

The Scenario

In a Government / Banks / university / corporation / Education with many employees etc. It is very likely that most of users have somewhat similar needs in terms of applications and files. All users would most likely want access to Word, Excel, Browser, organizations' line of business application(s), accounting software in case of finance department and so on. Further, files are need to be accessed by multiple individuals, perhaps simultaneously, perhaps from many different physical locations.

How does an organization deal with these requirements?

There are numerous variations and exceptions with more innovations coming all the time. That being said, there are currently two predominant approaches that can be taken to meeting these needs.

1) Desktop PC or a Fat Client Model - only data resides on servers

How it works:

In this approach, one simply provides each user with their own computer, install OS, Word, Excel, mail client, Browser etc. on each one and let everyone work independently.

One starts his computer and arrive at desktop. Then commence running applications (Excel, Word etc.) which are using computer's resources (processor, memory, storage etc.). If no network connection, one can keep on working with all the files that are saved on the local hard drive.
Things get more interesting when applications need to use data off a server. When data is being accessed off the server, the following occurs:

1. Workstation makes a request to server - "Please send me the data in this large excel file that has been saved on the X drive"
2. Server pushes all the data (and if it is a big spreadsheet it might be a LOT of data) across the network cable to your workstation.
3. User commence working on the spreadsheet; add rows, insert sub-totals, recalculate etc. This work is being done on your local machine but all saving or requests for more data are all going across your network cable. Again, when dealing with a big file, this can be very large quantities of data and therefore slow down the performance of the application.

Pros & Cons:

On the upside:

1. Other users' activities on their computers have minimal effect on your performance.
2. Minimal reliance on complex, expensive servers in this model. Typical use of servers in this environment is for file storage, checking & maintaining passwords and perhaps hosting a database.
3. Users can be "self-sufficient" - as long as they have a laptop and the working data they need - when they go home on the weekend or if they travel to conference.

The downside:

1. **Every computer needs to be maintained**. They must be updated for security and bug fixes as well as the updates of each and every software program
2. Each computer's hardware must be maintained at levels acceptable for the software applications they are going to be using. As the software programs require more resources, you must upgrade each and every computer that will use that program.
3. If any data is stored on the local computer, it then needs to be backed up to protect the organization's data. **Data security is a Major Concern**.
4. If a new application is needed, it is likely that it has to be installed on multiple computers.
5. Because each workstation brings all the data across the network cable to be worked on locally for each user, there is often a tremendous amount of network traffic. In a modern network this may not be an issue but if there are very large quantities of data or there are multiple physical locations that must communicate, the **network bandwidth capacity** may not allow for quick transmission of all the data required.
6. Because of points 1 & 2 above the organization is committed to continuous investment in each individual workstation to ensure the hardware is capable of running the current version of the software.

This is where the second approach comes to the rescue.

2) Thin Client (Virtual) Model - data & applications on servers

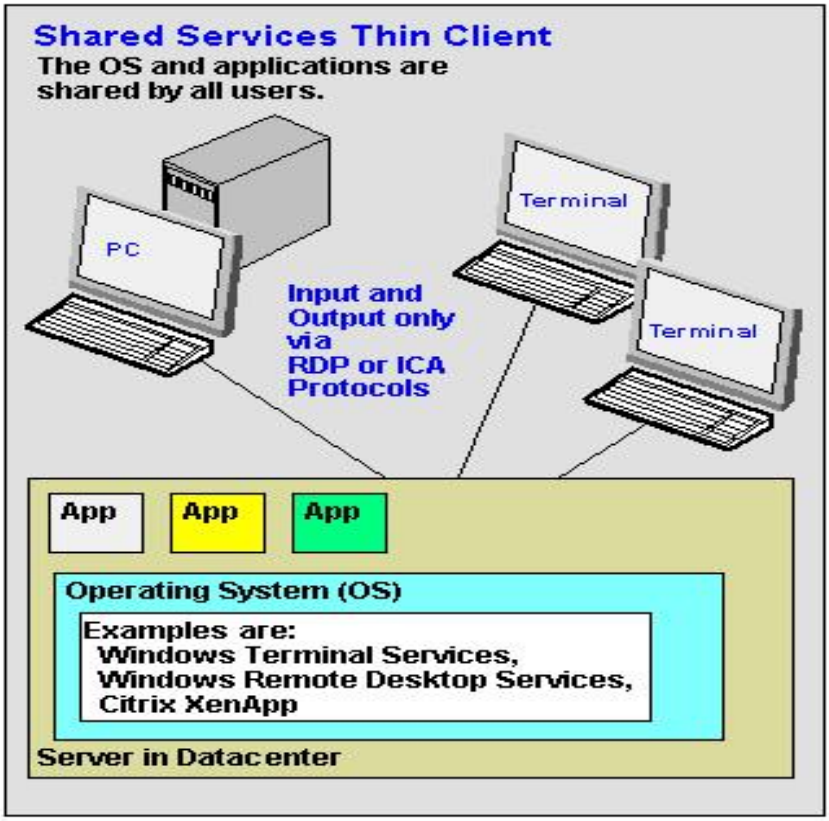
How it works:

In this model, instead of using multiple powerful workstations & laptops to run the application, all that work is done by one or more servers. Instead, end users are given terminals, sometimes called "dumb-terminals". Essentially all they do is present the user with a picture of a computer interface on their monitor and accept the users' input (mouse clicks and keyboard typing), sending it along to the server for processing. All data, programs, and processing remain on the server. Some ways to do this-

1.Shared Services (UI Processing)

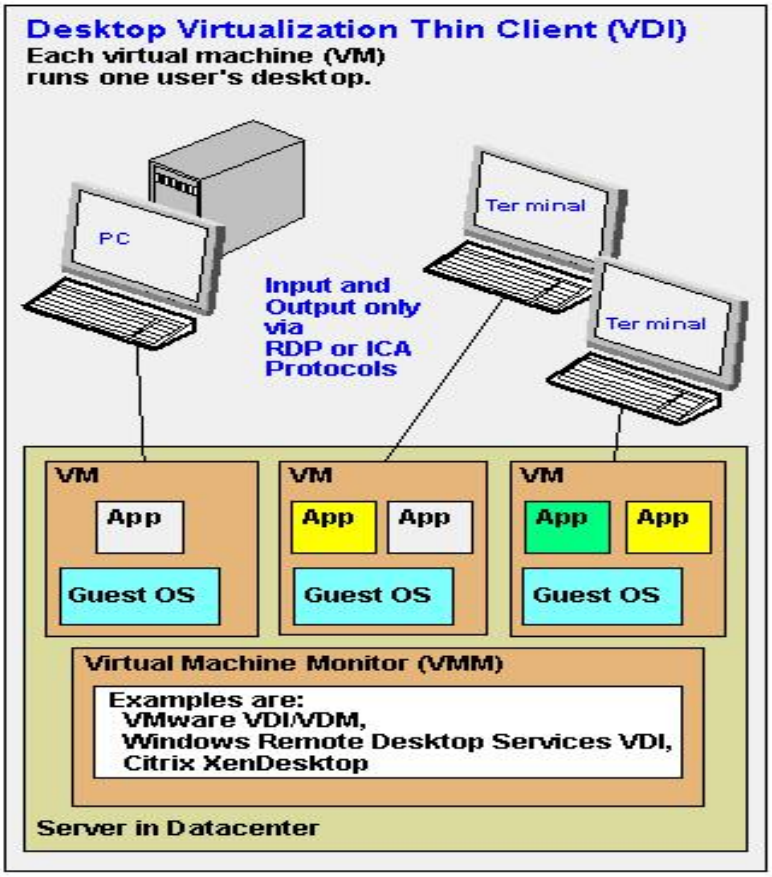
Using shared terminal services software such as Windows Terminal Services, Windows Remote Desktop Services or Citrix XenApp, users share the operating system and applications in the server with all other users at thin client stations. Although presented with their own desktop, users do not have the same flexibility as they do with their own PC and are limited to running prescribed applications and simple tasks such as creating folders and shortcuts. See Terminal Services, Remote Desktop Services and Citrix XenApp.

In the following illustrations, the lines show the conceptual flow of data between the clients and servers. In reality, all clients and servers are wired to a local network switch.



2.Desktop Virtualization (UI Processing)

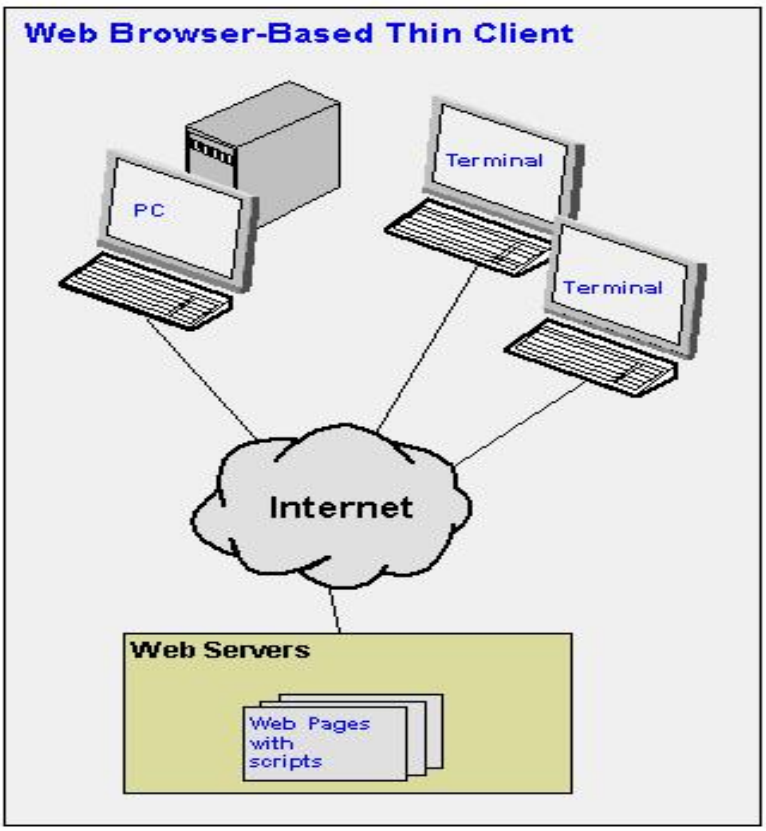
Using products such as VMware Desktop Manager (VDM), the VDI component in Remote Desktop Services and Citrix XenDesktop, each user's desktop (OS and applications) resides in a separate partition in the server called a "virtual machine" (VM). Users are essentially presented with their own PC, except that it physically resides in a remote server in the datacenter. They can modify the desktop and add applications like they could with their own PC ("fat client"). For details on the virtual machine architecture, see virtual machine. See Remote Desktop Services, Citrix XenDesktop, VMware and desktop virtualization.



3.Browser Based (Data Processing)

This approach uses ordinary PCs connected to the Internet, and applications are executed in the Web browser. Although the user's machine does the data processing, it is thin client computing, because the software and data are retrieved from the network. Very little, if anything, is stored locally. If users spend most of their time running Web apps, they are doing thin client computing whether they have a fully loaded PC or not.

Web-based email is the most ubiquitous example of browser-based processing, and Web-based productivity applications are also extremely popular. In some cases, copies of the data can be stored locally, but the software scripts that download into the user's browser last for only the current session. Years ago, this was the approach of the "network computer," which failed due to ever diminishing prices of PCs.



Pros & Cons:

The benefits of Thin Client computing are many:

1. "Thin Clients" need little to no maintenance. Only the servers need to be upgraded & maintained.
2. Only the server's hardware must be maintained at levels acceptable for the software applications and the number of users (more users = more horsepower required for servers).
3. No data can be stored on the local computer. Thus data is better controlled and backed-up.
4. If a new application is needed, it must only be installed once - on the server.
5. Thin Client software is long established and well know in the IT community.
6. Network traffic is made more predictable / even. Potentially depending on the data being presented to the remote user, network traffic may even be reduced. Thus even with minimal network bandwidth capacity end users may not experience any slowness. Especially when remote access across the Internet / VPN is required, this may be the biggest benefit of all.
7. Because of points 1, 2, 3, & 4 the overall result is often much lower total cost of ownership for the organization versus the traditional model.