Resource Management in Minimalist Systems

By
Brian A. Francosky
April, 2019
Resource Management in Minimalist Systems

By
Brian A. Francosky

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

at
Grand Valley State University
April, 2019

Yonglei Tao
April 20, 2019
# Table of Contents

Abstract ........................................................................................................................................... 4
Introduction ....................................................................................................................................... 4
Background and Related Work ............................................................................................................. 5
Program Requirements ....................................................................................................................... 5

**Implementation** ............................................................................................................................... 6
  Test Server Specifications .................................................................................................................... 6
  Microsoft Server Standard 2019 ........................................................................................................ 6
  Red Hat Enterprise Linux 7 ............................................................................................................... 7
  OpenBSD 6.4 ..................................................................................................................................... 9
  VMWare ESXi Hypervisor 6.7 .......................................................................................................... 10

Results, Evaluation, and Reflection .................................................................................................... 11
Conclusions and Future Work .......................................................................................................... 12
Bibliography ..................................................................................................................................... 13
Appendices ..................................................................................................................................... 13
Abstract

In order to keep a server running efficiently and safe from vulnerabilities, it’s often best to use an approach that keeps the attack area of a server low by using only the amount of software needed to accomplish a given task. In creating a small server for personal or small business use, I wanted to understand how to build a simple system that could be affordable while having a small energy footprint. In order to minimize the cost of each respective piece of hardware, I wanted to measure the requirements for the base operating systems from various vendors. Looking into Windows Server 2019, VMWare ESXi, Red Hat Enterprise Linux and OpenBSD, I measured their system requirements on a low-power server and attempted to analyze each system for simplicity and functionality. The ideal system should have minimal OS-level processes and the smallest CPU and storage footprint. Remote management tools and package management options were also key to an ideal operating system.

Introduction

Computers have gotten increasingly complicated since the sixties and seventies. Much of the advances in machine learning and big data have come because computers are excellent at handling many tasks at once thus also handling large magnitudes of data. Multi-threaded processors and processes have assisted in making the advancement accelerate at a furious pace, but with all those benefits come some very strong disadvantages. Another year continuously comes and goes and new touted hardware and software are becoming the buzz words of the industry.

Complex situations sometimes require complex hardware and software. This is not a point of contention. Clustering for increased power and connecting ultra-high speed networking allows computers to run massive enterprise systems with complex data analytics. This adds, however, the need for additional software and Operating Systems, as well as interfaces to all the parts of the whole. These features are extremely important and power the businesses of today. Unfortunately, it seems that often the need to increase power and store and compute more information, has lead us to take shortcuts and not truly optimize our hardware and software in a cost-effective manner.

The paradigm that has come out of this is the idea of cloud computing and SaaS and other “as a service” options. Virtualization is used everywhere to allow one host to run many different servers which can be quickly reprovisioned and adjusted on-the-fly. All the different software that is controlling these features is continuously being controlled through licensing and due to that licensing by core is now the norm. As newer processors have an abundance of cores I questioned how I would approach a small business if I wanted to sell a low-cost server that didn’t need to, but could, run without any cloud options. Ideally on a small network, most applications can be developed with a simple set of tools, mainly a compiler for local software and a web server to host up web based applications.

I created a test server using a variety of different operating systems to get a better understanding how this would be set up, putting remote administration and development options inside to see what type of resources would be used. The extensible nature of software allows you to customize in any way possible and that is the same in this case. Any cloud based options could be added to these systems with added cost. An assumption must be made that if instead of a basic web application using Javascript, CSS and HTML
additional resources would be consumed if you were instead to use JQuery and Angular, or add a cloud based database on Heroku. Further, dependencies and costs that are associated with large databases such as Oracle would have to be examined. That would be outside of the scope of this research, but could be used to further promote this type of reliance on expensive software beyond the needs for a small business with a closed network.

**Background and Related Work**

I have been working as a system administrator for many years, and a computer technician before that. I’ve seen wise decisions when it comes to technology and poor decisions. I’ve seen wonderful things built with little to no cost, and I’ve seen millions being thrown at an application that never truly came into fruition.

There are many journals and projects related to cloud-computing but few that discuss options for small businesses, or detail the hidden costs they may entail, to help those that may not want to have their data in the cloud. I have found that many of the latest technologies and buzz words tend to grow in huge favor before coming circularly back around to older concepts. Having a local server may seem like an older concept but that concept doesn’t have to assume that more modern solutions, such as backing up to cloud storage, won’t be used.

Much of the intrigue of this topic comes from virtualization itself, as well as small low-powered devices such as the Arduino and Raspberry Pi. Low powered machines are seemingly capable of running multiple virtual machines, so I often try to push down to the lowest common denominator when building servers. Pushing down to bare metal just naturally feels the most simple way to go, and takes stacking software out of the equation.

Accidents that rely on complicated software, such as the Boeing 737 Max crashes or the accidents that were not avoided in the testing of driverless cars, will happen more and more as we push software on top of things that previously were controlled through mechanical and electrical means. Some of these thoughts and ideas added to the question of managing the resources of these systems to try to keep them as simple as possible—albeit for applications far less demanding as the two mentioned above.

**Program Requirements**

The solution to this problem is a system that has lower power requirements and low resource utilization. By meeting those requirements a system can be efficiently backed up or recovered, easily transferred and stored, and more attention and money can be spent on smaller, but faster disks and processors. In most cases lower powered hardware is completely sufficient.
Implementation

Test Server Specifications

<table>
<thead>
<tr>
<th>Motherboard</th>
<th>MSI AMD FM2+ A68H DDR3 SATA 6Gb/s USB 3.0 HDMI Micro ATX Motherboard (A68HM-E33 V2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>AMD AMD A6-7400K Dual-Core 3.5 GHz Socket FM2</td>
</tr>
<tr>
<td>RAM</td>
<td>Patriot Memory Viper 3 4GB DDR3 1600MHz (PC3-12800) x2</td>
</tr>
<tr>
<td>HDD</td>
<td>Seagate 500GB SSHD SATA 8GB NAND SATA 6Gb/s</td>
</tr>
<tr>
<td>NIC</td>
<td>ULANSeN Broadcom NetXtreme 10/100/1000Mbps Gigabit Desktop PCI-E</td>
</tr>
<tr>
<td>PSU</td>
<td>APEVIA ATX-VS450W Venus 450W ATX</td>
</tr>
</tbody>
</table>

Microsoft Server Standard 2019

Microsoft entered into the server market with Windows NT in the early nineties, but my exposure to Windows as a server began with Windows 2000 in the early 2000s. By keeping a bare-bones feel to the common desktop environment allowed administration to be simple yet understood by anyone who has used a Windows desktop. Configuring complicated functions such as Active Directory, a new type of directory management or network services was simple though the use of wizards and the Microsoft Management console. Many of the administrative features would continue to later versions of the software.

Microsoft Server 2003 was the first of the rebranded server operating systems and the first place I became familiar with IIS and Group Policies in Active Directory (previously released in NT and Windows 2000, respectively). Remote Desktop Protocol, which was introduced with Windows 2000, was a common tool I was using to administer servers remotely and allow for multi-user connections. The common use of service pack updates and resource kits made upgrading simple and easy. This version also continued the trend to offer multiple versions of the operating system (including a range of cost) for different markets.

Between the release of Windows Server 2008 and Windows Server 2012 my professional focus switched mostly to *NIX systems, but I still often ran VMWare virtual machines (the only option at the time) and labs with Windows Server installed and in places where the software was only available for Windows. For this project, I downloaded the latest version of Windows Server 2019 and many things had changed. Most notably, a very bare bones installation option which launches with no start bar and only a command prompt. Windows Admin Center can be used to connect to these servers, or remote administration with PowerShell as recommended by Microsoft.

The headless installation was the most impressive in this regard. If there is no need for the resource hungry GUI, it can be avoided. Luckily, rendering of GUI applications is still available. There is still a full GUI install, which finally removes the tile-based view of Windows Server 2012 with a much simpler version akin to Windows 10. A base install running IIS consumes under 800MB of RAM, and around 7GB of disk storage. Around 60-70 process run with the majority being svchost.exe processes.
which handle system processes. The particulars are outside of the scope of this project, and not visible to the closed nature of Microsoft’s licensing. Security features such as the Microsoft anti-malware and firewall, as well as connection managers and other system processes are started by default.

A headless installation isn’t the most attractive option for all servers and applications. While package management can be handled through MSIs and the Install-WindowsFeature Powershell cmdlet, sometimes having a traditional desktop view on a server can be helpful. Installing the desktop experience increases the resource usages to around 1.5GB of RAM and around 20GB of local storage.

<table>
<thead>
<tr>
<th>Windows Server 2019 Standard</th>
<th>For physical or minimally virtualized environments</th>
<th>Core based</th>
<th>WS CAL</th>
<th>$972</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Server 2019 Essentials</td>
<td>For small businesses with up to 25 users and 50 devices</td>
<td>Specialty server</td>
<td>No CAL required</td>
<td>$501</td>
</tr>
</tbody>
</table>

Managing a server with Powershell is convenient and additional support for C# and .NET development is easy to add through Server Manager or Powershell. These are optionally added with a 200MB download from Microsoft (or from the installation media). Both the headless installs and the desktop experience version run well and the official Microsoft documentation states that hosts can run on as little as 512MB of RAM. .NET and C# applications are perfect for custom development especially when complimented with Microsoft SQL Server. The additional C, C++ and C# compiler is an additional 200MB package.

Of course, like most Microsoft software, Windows Server comes at a price. Even a single-core server with 6 cores requires a standard license of $972 at the time of this writing. That being said, Microsoft has cleaned up the services and applications to fit the modern computing world, it also no doubt learned to trim up their products to best accommodate large enterprise hosting and HyperV. It’s easy to see how Microsoft has many offerings for data centers in order to complete with other lean alternatives. My testing focused on Windows Server Standard but alternatively it would be possible to use Windows Server 2019 Essentials, and with no CAL required that could be an attractive option at $501, not including SQL Server.

**Red Hat Enterprise Linux 7**

Red Hat has Enterprise Linux has been making up the operating systems in enterprise servers for a long time. Since the first release in 2000, many large enterprises have used it to run everything from business analytic stacks to web servers. While based on a free Linux kernel, Red Hat packages can include support contract and updates via Satellite repositories for a subscription price.

Red Hat is one of the oldest distributions that was adopted for a solution in an enterprise setting. Other distributions that didn’t require support contracts were used in my labs and personal servers, but companies gained comfort in knowing that a full support staff could be contacted with any issues. The support from Red Hat was valuable, as was their updates and errata towards packages and application.
Version 7 will be the last version before the effects start showing which stems from their acquisition by IBM.

Much like Windows Server, there are many different installation options that can be used to fit whatever type of business needs are trying to be addressed. I began with a minimal install, consisting only of a headless server with development tools and the Apache web server. Administration and hardware monitoring tools were added in an attempt to meet the functionality that Microsoft Server provides by default. One of the benefits that Red Hat offers is the ability to customize the installation during the setup, as opposed to a limited selection that can be altered after-the-fact.

The minimal server install takes up just under 2GB of hard disk space, while consuming just under 700MB of memory. As Linux has a much more mature set of command line tools, this is how I have setup most production servers. Adding the default Gnome package will give a full fledged desktop but increases disk usage to 4.4GB and increases memory usage to 1.7GB.

Packages can be complied from source or acquired through the official or third-party repositories and RPMs that are usually available from either Red Hat channels or Fedora and CentOS repositories. Git and Ansible make configuration and package management easy and overall support for the most common tool is very good. Driver support is also strong from the majority of vendors. Using SSH (ironically a product developed by OpenBSD) is available by default, but VNC and a host of other remote management software is available.

Monitoring of system activities and system resources can be accomplished in a variety of ways. I tend to lean toward custom monitoring scripts that rely on classic *NIX tools such a ps, free and df, but all of the Gnome tools are available. SNMP tools can be used to get reports from a large range of services,
hardware and software. Third party applications such as Nagios can be added and the open-source community is very active in development in this space.

One of the nice benefits of using open-source software is I have confidence that what I see using up system resources can be viewed or modified. It’s easy to tell that dbus and kernel processes are running, but if I ran into real trouble and had the need, I would be able to look at the source code and trace processes through the stack to try to remedy the situation. For proprietary hardware I would be leaving my faith that my problem was important enough—and that the support staff was competent enough—to resolve my issue. That support will come at a cost, starting at $349 with self-support and increasing as more assistance is provided.

**OpenBSD 6.4**

Although the idea of running Linux or a proprietary UNIX on a server is less commonplace these days (we have completely removed our AIX and HPUX stacks in favor of Linux), there seems to be more confidence in using open-source version of the traditional BSD forks for applications that truly want to be lean and transparent. Many companies such as Netflix, who use a stripped down version of BSD to run their Open Connect CDN, rely on the stability of the operating system to host their mission critical infrastructure. Being an outlier overall may make it more difficult to find specific libraries for OpenBSD, though not using a wide variety of third-party products is part of the reason that I included it in my list.
OpenBSD offers a simple, headless install by default—not even X support is included. X installation is done through the built-in pkg_add interface, or can be done by compiling from source. Installation of drivers or kernel support for hardware tends to be more difficult to come by than is expected these days, and support for OpenBSD related issues is much harder to come by.

Httpd is included as a web server and the LLVM Clang compiler is added for development of C/C++ applications. OpenJDK (or Oracle Java) can be added in a similar fashion. This falls in line with the Red Hat options, but keeping the system processes to a minimum helps with the simplicity it can provide. It’s not often that I’m able to research system issues without having to parse through many different logs.

System management relies on the same UNIX tools that the Linux distributions support. Ansible, Puppet, and a variety of other configuration management software can be built on BSD or have packages readily available if you’re willing to compile from source. Scripting checks and alerts are a little more difficult as the common proc filesystem from Linux is missing, but sysctl is available for more in-depth monitoring and scripting.

While the installer for OpenBSD is completely open-source, third-party support options are available for purchase. Whether this is a net positive or negative depends on how complicated the stack will be, but the option is there and there are many different providers. The end result is that the extremely low-cost (ie. free) is a great benefit to small business who simply cannot afford to pay for licensing costs at the onset of a business.

**VMWare ESXi Hypervisor 6.7**

The decision to test VMWare was twofold. First, using a hypervisor on a server makes some processes less complicated. For example, a second host can be added to the cluster and the resources can be combined and more easily provisioned. Second, and more important in the perspective of this research, is that I continue to see decisions made to use additional complicated solutions that, without reason, add overall complexity to the base system. That’s not always a bad thing, but sometimes a simple bare metal server is a better purchase.

The base of the VMWare hypervisor used to be hosted a Linux kernel. That has since changed and now VMWare uses its in-house developed PhotoOS kernel. But for this research I wanted to understand the benefit of using virtual machines over bare metal, so VMWare’s hypervisor and system monitoring stack was what was measured. Regardless of the benefit, the overhead of the hypervisor will always be there and that is apart from the added licensing cost to purchase such an option.

Without running a virtual machine, the system was consuming around 1.2GB of RAM and 1.4GB of storage. Of course, much of this is similar to a minimal Linux or Windows installation, which makes sense since this software is managing them. The system ends up running around 450 processes at idle, though, and there are far fewer customization options available to trim down the hypervisor. Many of these are networking and storage systems which are virtualized when running one or more virtual machines, and services can be turned off through the interface.
The monitoring tools can be both console or web based. The console can be monitored through PowerCLI or ESXCLI commands and handle remote administration very well. Using a web browser to make a connection to the management interface is available, and is a great way to monitor the resources of a single machine or the entire cluster. There is the slight issue that VMWare seems torn between web access and a full thick client for management. This can be seen with he back-and-forth between using HTML5 or Flash interfaces as they seem to continuously change their mind on the preferred method.

Licensing for a single ESXi host is very close to the other contracts, with a wide variety of different products and support options. To keep a close comparison to the other systems in this research a base essentials package is around $500 with an additional $308 per year for per-incident support. This comes close to the cloud based options you can find using dedicated AWS instances which suggest around $12 a month for a comparable online system. The issue with this is necessary bandwidth and support would bring the scope of its benefits outside of my target audience.

Results, Evaluation, and Reflection

Whether there are more or less processes may not seem to be an issue as long as the system is functioning properly. When trouble arises, though, the fewer processes that are running means the easier it is to isolate issues. Further, more processes mean more complexity and often adds the need for more updates, security or otherwise. In my testing I didn't find any of these types of issues, but the system wasn’t
under load and I would expect any hardened operating systems to function for years without issues, let alone months. Still, the fact that all of these solutions have lean options is a relief and should bring some confidence that the focus can remain the software being hosted and not the operating system itself.

Measuring memory consumption shows the large increase needed to run desktop environment. For each of the systems just running a GUI increases memory usage by up to 300%. Storage, something that should never be taken for granted, also shows an increase. This effects not only time to do things such as scans but make backups and restores take longer as well. As non-volatile storage get closer to memory level speeds, acquiring a 32GB drive is far less expensive than a 512GB drive. Therefore it would be in the best interest to keep storage utilization as low as possible.

Each of the researched systems could be adapted for a minimalist server used for a wide variety of different small business applications. As previously mentioned, there would always be a way to increase redundancy and performance or even reduce the cost with alternative hardware. One of the more difficult things to measure without special equipment is overall system energy consumption. Also, having a specific type of application to test would have better measured the performance of such applications. The general comparison to the now common solution of cloud computing is difficult, as there are hidden costs to the information that is provided by vendors.

Conclusions and Future Work

With a better understanding of the cost and technical resource consumption I plan to offer these systems in conjunction with services provided by myself as well as a few other colleagues that can provide a professional level of development for them. I hope to target small businesses with a total package cost of ownership as well as add options that allow customers to install, recover and upgrade these systems without any assistance from more technical administrators. Measuring the cost and development requirements for custom systems can be difficult but I’m confident that almost any scenario could be built with far less time and money than commercial OTS solutions.

More importantly, a proper solution for a server would have to have additional redundancies build in. The price on my test server was only around 200 dollars, but an HP micro-server is as low as $315 with the additional redundant power supply servers costing around $700. The requirements would have to be adjusted based on the criticality of the system. This cost, in addition to the licensing cost of the software aforementioned, is what will be compared to cloud-based solutions. These solutions, however, would not required a majority of the data to be transferred over a high-speed network. Further, with new hacks and vulnerabilities being found every day, that may be an overall positive.

Finally the major focus of my future work will be to entice small-business and hobbyist owners to rely on a custom application to manage their businesses and applications. Although software development is not my interest, I know many web developers and application developers that would sign on to a project such as this. Measuring the increase in resources of these applications is where my focus will remain, hoping to go for a minimalist system that is low on resources for all and any applications. Hopefully this
research is a starting point in the discussion to show how it can be approached in a simple way yet still have great results.

**Bibliography**

Microsoft
Red Hat Enterprise Linux
OpenBSD
VMWare
Virtualbox
Apache Software Foundation
ECMA International

**Appendices**

No appendices at this time.