

Computing Commoditization

Copyright © 2004 Smoot Carl-Mitchell

Introduction

This paper outlines trends toward the commoditization of hardware, communications, and software, and the substantial impact these trends are having on the computing industry. This paper also provides a framework from which to view the changes.

Change is rampant in the computer industry today. Never before have so many computing cycles been available at such low cost, with such high promise, and so little help to make it work effectively. At one end of the computer spectrum, we have large systems that are more capable than ever, while on the other end we see very capable smaller systems with the processing power to meet all but the largest data processing problems. At the same time, vendor consolidation has reduced our supplier base from scores to a comparative handful. The financial pressure on IT continues as well: budgets are smaller, yet we are asked to do more. And the more we architect on the leading edge, the more the responsibility for integrating new solutions with existing ones falls to us. Where is this all leading? What trends will cause us to change our plans yet again, and how soon?

Computing Trends

We are in the midst of a revolution in enterprise computing. While this revolution will not happen overnight, the end state seems fairly clear. We will see the commoditization of computer hardware, communications and software. This commoditization will help drive the following trends:

1. Most new system deployments will be on commodity hardware platforms using Intel or Intel clone processors.
2. Two operating systems will dominate new server deployments - GNU/Linux and Windows.
3. GNU/Linux deployments will be on a steep upward curve.
4. Windows deployments will be on an upward curve with declining increases for the next few years.
5. Almost all Internet-centric services will run on large clusters of GNU/Linux servers. GNU/Linux will become the OS (operating system) of the Internet replacing Sun Solaris.
6. Proprietary UNIX systems will steadily migrate to GNU/Linux based systems. More UNIX/Hardware companies will employ services to retain their base in the enterprise market.
7. Proprietary UNIX vendors will Open-Source their proprietary add-ons and port them to GNU/Linux. The add-ons will become part of the standard GNU/Linux software base.
8. Multimedia, voice and data communications will converge using the Internet as their communications backbone.
9. Internet bandwidth will continue to increase by several orders of magnitude and cost per bit transferred will decline dramatically. This will further enable very inexpensive remote data communications allowing very high-speed computer-to-computer communications regardless of the physical location of

the servers. This will also enable new classes of applications and new styles of computing.

10. The traditional separation between SAN, WAN and LANs will disappear as costs decline and technologies converge.
11. The cost of software will decline as more and more capable Open Source software becomes available.
12. Users will demand interoperability among software systems and drive further standardization of network protocols, data interchange formats and software APIs.
13. Responsibility for the integration of solution components will shift even more from the super-vendor back to the user. This will happen in both subtle and obvious ways and have extensive financial consequences. Just as the user community thought things would get less expensive, the hidden costs that vendors can not or will not continue to bear will begin to fall on users as well.

These trends will enable the creation of true computing utilities where services are offered which are independent of the underlying infrastructure. Much like Internet standards, users will clamor for computing utility standards, so they can find the most cost-effective service provider among a set of competing providers. These standards will continue to drive up the services stack. As a result, traditional proprietary software vendors will transform into service providers, retreat to a niche market, or go out of business, since commonly used software will rapidly become commodities dominated by Open Source software.

Sound familiar? It should. For the most part the components are already falling into place to build such a computing utility. Consider Google's Internet search engine infrastructure. It is built on a foundation of many thousands of commodity machines (ix86), running a standard OS (GNU/Linux), connected to a standard network infrastructure (the Internet), and offering a standard user interface (HTTP/HTML). Google's infrastructure is the wave of the future. The rationale for any business to operate and maintain its own complex datacenter is becoming less compelling except in certain specialized cases.

A prerequisite for a utility computing model is large-scale commoditization and standardization of computer hardware, communications and software. We will discuss these areas at length later in this paper. First we need to look at the relationship between commoditization and utility computing.

Commoditization and Utility Computing

Already we are well on the way towards the commoditization of hardware and network communications. OS and Application software are lagging behind, but are rapidly catching up principally because of the availability of Free and Open Source software. Eventually, a utility grid of computers, linked by very high-speed networks, using a standard operating system interface, will provide most of the computing resources used in daily life. Getting to computer resources will be as simple as turning on a lightbulb. Users will gain access to this grid with standard Web-enabled interfaces running on everything from cell phones to notebooks to desktop systems. Eventually, given the ubiquity of the grid and the ease of access, most end-user devices will be much simpler systems than the complex desktops and notebooks today.

In part this trend is driven by the demand for open standards and the need for interoperability in an increasingly dynamic and interdependent business environment. Today's computer users must rapidly adapt their computing

environment to changing requirements. Hewlett Packard for example has embraced the trend under the term "Adaptive Enterprise." Therefore, to make a computing enterprise truly adaptive, the underlying software and hardware systems must be based on well-known and widely used standards. Otherwise, the system is too complex to adapt and too difficult to manage.

The trend toward the commoditization of computer hardware, communications and software is a natural evolution of the success of the computer technology. This statement may come as a shock to those who work on computer systems and see their complexity and often their fragility, and who do not understand how something as complex as a large hardware/communications/software system can ever become as simple to buy, use or operate as, say, a lightbulb.

The Lightbulb and the Electric Grid

The lightbulb and the electric utility infrastructure behind it are good examples of the eventual state of computing. Today, we take the lightbulb and the electricity that powers it for granted. But the lightbulb's widespread use is the result of decades of evolution and experimentation in an industry that moved from one-off designs to commodities used by millions of people every day. In less than a century, the use of electricity has moved from a luxury to a virtual necessity. Computing is on the same trajectory.

Let us examine the lowly lightbulb and why it is today a commodity. By definition a commodity is a widely available product based on a set of well-defined and commonly accepted standards. You may not think very much about the standards behind a lightbulb. However, there are a whole plethora of them. They range from the size and shape of the lightbulb's socket to the voltage maintained by the electric utility grid which powers the lightbulb. And do not forget the standards around the electric plug, which allow a lightbulb to tap the energy in the grid. There is an entire infrastructure with clearly defined standards that allow a lightbulb manufacturer like GE or Westinghouse to produce bulbs of consistent quality on a massive scale.

Imagine the situation if there were several electric grids providing different voltages. Moreover, what if each grid only let you use electric plugs of a specific type? In such a scenario it is doubtful whether lightbulbs would be as cheap or as widely used. Imagine buying a lightbulb where you needed to know who provided your electricity and what their voltage and interface specifications were. This was the state of computing 30 years ago and the vestiges of that legacy still exist today.

Computer System Evolution

Thirty years ago computers were large and expensive devices with their own set of specialized components, which you could only buy from the original manufacturer. One vendor's computer could not communicate with another vendor's computer. Each computer had differing word sizes and even byte sizes. Remember the DEC-10 with a word size of 36 bits? Or the CDC 6600 with a word size of 60 bits? Peripheral devices were even more non-standard. Disk drives could only work with a single vendor's system. It was hardly a standardized world back then unless you stuck with a single vendor. And being locked into a single vendor reduced your ability to negotiate better pricing from your vendor. To change vendors meant incurring massive migration costs.

Fortunately, several developments started computing on the path towards standardization and eventual commoditization:

- The microprocessor and related standard computer hardware
- The Internet and the dominance of the TCP/IP protocol suite

- Free and Open Source software

These three developments are related and interconnected and in some very fundamental ways one could not happen without the others. Let's discuss them in order.

Hardware Commoditization - the Microprocessor and Related Hardware

In large part the commoditization of hardware is driven by the success of the microprocessor and related hardware technology. The microprocessor started out as a low-cost computing solution used initially by hobbyists and embedded system developers. Today the microprocessor is used in almost all computer systems from PCs to super computers. This is due principally to the relatively low cost of manufacturing a microprocessor as compared to designing and assembling discrete components. But unit manufacturing costs for a microprocessor are only low when you can produce chips in large volumes, since the initial capital and R&D costs are quite high. So a particular microprocessor family can only be competitive when it is produced in large quantities. Typically, this means it must be deployed on all kinds of computing systems from PCs to servers.

The ix86 Processor Family

The most successful microprocessor is the ix86 family of microprocessors. While the ix86 was widely considered to be an inferior architecture when first introduced, it succeeded in the marketplace because at the time it was "good enough" for the task of powering desktop PCs and its price was right for the emerging PC marketplace. That marketplace was ignored by the much more capable RISC microprocessor developers, who were more interested in selling their chips to workstation and later server vendors at a higher margin. Over time the ix86's capabilities have evolved to the point where it is able to disruptively enter the workstation and server market, previously dominated by proprietary RISC microprocessors.

Today the adoption of the ix86 is accelerating due to the wide adoption of Windows on the ix86 architecture, and now the availability of GNU/Linux on the same architecture. As a result, many users are replacing RISC-based computers (e.g. PA-RISC, SPARC, Alpha, etc. running proprietary UNIX OS) by ix86 boxes running GNU/Linux or Windows. This is not to say other microprocessor architectures will not still be in use as well. But for the other players in this market (really just IBM with their Power family of microprocessors) will require them to commoditize their processor family and deploy it on a wide variety of computing devices.

Related Hardware Standards

As the microprocessor gained in acceptance, related standardization also occurred in memory and I/O subsystems. Standardized memory chips resulted in a dramatic decrease in memory costs and a dramatic increase in the memory installed on everything from portable electronic devices to clustered high availability servers. PCI bus technology also became an interface standard. By adopting common bus architecture, interface card manufacturers only needed to develop one form factor and bus interface specification for their interface cards. This reduced duplication of effort and greatly shortened the development time for new interface cards. More importantly, a bus interface standard drove down the unit cost. Consider PCI Ethernet cards. When Ethernet was initially introduced, interface cards built for proprietary buses cost upward of \$2,000. Today a cheap 10/100 Ethernet card costs less than \$20. Server cards are somewhat more expensive, but that has more to do with pricing and support policies than the actual cost of producing the cards. With a common and widely

used bus standard, the unit cost of interface cards is reduced by virtue of its wide adoption, which in turn accelerates the rate of adoption.

Network Communications Commoditization - The Internet and the TCP/IP Protocol Stack

It is hard to believe that less than 20 years ago, the Internet was a relatively unknown research internetwork. Today it spans the globe and is the dominant worldwide data communications infrastructure. Today it is poised to become the vehicle for voice, multimedia and data communications. Eventually, the Internet will become the sole network for all forms of network communications.

The Internet and its TCP/IP protocol technology commoditized data communications. The Internet today is an indispensable vehicle for both communication and commerce. There is scarcely any business that does not have an Internet presence. And the integration of Internet technology in our daily lives is nothing short of phenomenal.

The Internet Development Model

The Internet is the communications foundation on which the future utility computing infrastructure will ride. The Internet and its model for development literally changed everything. The key to the Internet's success lay in its cooperative and engineering driven development model. The engineering philosophy behind the Internet was to come up with new ideas and deploy them quickly and see how they work in practice. Many protocol refinements were done on the fly in the real world test lab that was the early Internet. This was in contrast to the competing ISO/OSI model of protocol development, which was bureaucratic and slow. TCP/IP won the protocol war in the 1980s by being able to implement and adaptable to a wide variety of computing environments.

The TCP/IP protocol suite is built on a remarkably simple design. The core idea was to make IP a simple mediation layer between the hardware components of the system and the higher layer software components. Put simply, the complexity of reliable data communication was pushed to the higher layers of the protocol stack. IP simply routes packets across whatever hardware is available without regard for what the contents of the packets are. Moreover the hardware did not have to be uniform. Hardware links could be replaced with links of higher speed or reliability without impacting the higher layers of the protocol stack. And hardware links of differing speeds and framing characteristics could easily interoperate.

The Killer App

The design of the transport layer (TCP and UDP) also facilitated the implementation of new application protocols. The "killer app" of the Internet was the Web. The Web consisted of an application transport protocol (HTTP) and a presentation protocol (HTML and today XHTML). Both ride nicely on top of the TCP/IP infrastructure of the Internet with its robust name resolution protocol (DNS). The key to the Web's success is its independence from the underlying transport and packet switching infrastructure. The Web is also completely distributed and required no centralized control mechanisms. Anyone can publish a Web page if they have Internet connectivity.

Another key to the success of the Web is that it fosters a cooperative model of protocol development. This even extended into the development of the Web browser which as a client talking HTTP to web servers and also translating the HTML into a user accessible format. Again, in keeping with the TCP/IP philosophy, most of the hard work is handled at the top of the protocol stack. The underlying Web transport layer (HTTP) was kept as simple as possible. These

standards continue to evolve. For the most part Web browser developers have adhered to these standards, since the goal is to allow any end user to access any web page with any web browser. The Web browser itself became a commodity as a result. But as a commodity it must adhere to a specific set of standards. And the standards are generally adhered to because of the demand of the user community for a seamless and not a fragmented World Wide Web of information.

Network Layering

The growth of the Web and other higher layer services also reveals the strength of the network layering model used by the TCP/IP developers. TCP/IP was originally a novelty and quite a sophisticated piece of engineering. Today implementations of the protocol stack, while divergent in their details, adhere to the reference standards. The core of the stack (IP, TCP and UDP) consists of commodity components of the network infrastructure. TCP/IP is found on almost every computing device available today. The same commoditization is moving up the protocol stack into the application layer. We see this with the various implementations of HTTP and HTML. They are also becoming commodity components of the global network infrastructure. Much the same thing will happen with other protocol components as they begin to become widely deployed.

OS and Application Software Commoditization - Free and Open Source Software

Much like hardware and communications protocols, general-purpose software is moving in the same commodity direction. Free and Open Source Software (FOSS) have been a major influence in the commoditization of general purpose software. To understand this trend, we need to understand what FOSS is.

Free Software

Free software is any software where there is granted freedom to the end user to modify the software and to use the modified software free of any restriction. "Free" in this case means freedom. Such software by definition must have its source code freely available so end-users may modify it at their discretion. The only restriction is that any modifications to the software made by the redistributors must be distributed under the same terms as the original software. This restriction prevents someone from taking Free Software and turning it into proprietary software. Free software is promoted by the Free Software Foundation, which is also the author of the General Public License (GPL). GPL is the most common license for Free software.

Open Source Software

Open Source software includes free software as a subset. Open Source, however, encompasses a much broader set of software and includes in its looser definition software that may be redistributed or modified under a different set of conditions. Both FOSS licensing requirements require source code to be made available to an end user. This is a key discriminator from the more traditional proprietary software distribution model where software is only distributed in binary format usually under restrictive licensing terms.

Motivations for Using FOSS

A question always raised by hard-nosed business types is: What is the motivation for software developers to embrace the Free and Open Source development model? After all, an examination of the typical licensing terms for FOSS suggests the price charged for the software gravitates towards zero. Is it just altruism, or is it possible to build a sound business model around FOSS? I believe there are several motivations that, taken together, permit the use of FOSS as part of a viable business model.

Social and Political

The first is the philosophy espoused by Richard Stallman of the Free Software Foundation. Richard believes software should be free for the end users to use and modify if they like to meet their own needs. He finds the closed source model of software stifling, since the end user cannot modify the code. To put their money where their mouth is the FSF has developed a whole set of Free Software tools and applications, as well as the licensing framework for Free Software. Richard's motivations appear to be social and political and are geared towards promoting and protecting individual liberties, which are good things.

Software as a Service

A second motivation comes from the insight that software itself has no intrinsic value. If you do not know how to use a piece of software then it has no value. So how you use software is far more important than the software itself. This leads immediately to the insight that software is an enabler of services. In fact good software is a service multiplier. A business can make far more money showing people how to effectively use software or, better yet, it can provide a service enabled by software to users and charge a fee for the service. People will pay for good service.

This is a radical departure from the proprietary licensing model imposed by most software vendors. In a typical licensing model, you must pay a toll before you even use the software. In the "software as a service enabler" model, the value of the software lies in how it is used. In this model making the software widely available at a low price or for free is a plus; the more widely used the software, the bigger the service business. Software becomes a loss leader for services delivery or it is used to build a computer system infrastructure to deliver a service.

Commodity Hardware Vendors

A third motivation comes from vendors of commodity computer systems. Vendors like HP, IBM and Dell have a strong motivation not to be tied down to a single OS vendor. Up to this point in time, vendors selling ix86-based computers have had few alternatives but Microsoft Windows, because Windows is a virtual monopoly in at least the ix86 desktop space. But the ideal position for a vendor is to have a standard and reliable OS that is not tied to a third-party vendor. The only alternative for computer vendors was to develop their own OS. The high development and maintenance costs of doing this were only practical when hardware was not a commodity. With commodity hardware, it makes good business sense to deploy an OS that is not tied to any single vendor. The Free and Open Source model fits a commodity hardware vendor's needs extremely well. It is also an especially good fit for a utility computing environment.

Challenges to Commoditization

Businesses or individuals with a vested interest in the status quo are going to put up barriers to commoditization. Their efforts may delay the inevitable outcome, but will not change it substantially. Challenges will come from several directions:

- Fear, Uncertainty and Doubt (FUD)
- Patents
- Anti-circumvention laws

FUD

FUD is designed to delay the inevitable. Most of the FUD around commoditization is directed at FOSS. We will continue to see efforts by proprietary software vendors to portray FOSS as unsupportable or more expensive. For the most part these attacks completely miss the point that all software will become less expensive in a commoditized computing future. The inevitable economics of commoditization will force high margin software vendors to cut their prices or adapt to the new computing reality.

Patents

One way for proprietary software vendors to attack commoditization is to use the patent system as a brake on commoditization. Patents are supposedly for unique and non-obvious inventions. Today there are a lot of software patents on things that are obvious and for which there is prior art. Ideas should not be patentable, but ever since the US Patent Office allowed software patents, it is unclear whether ideas are indeed being granted patents.

The problem here is not that patents are bad things, but that patents are difficult and expensive to challenge. By patenting some obvious software practices, software vendors can create a barrier to smaller and more nimble competitors to enter the software market. This strategy is bad for the end users and bad for competition.

Anti-Circumvention Laws

A particularly disturbing trend is the push for anti-circumvention laws. Such laws are designed to prevent the piracy of copyrighted works such as electronic books or movies. While noble on the surface, such laws could have a chilling effect on the ability to reverse engineer existing proprietary software interfaces, if not narrowly written.

A good example of the abuse of an anti-circumvention statute was the recent arrest of a person who reverse engineered the DVD copy protection software so he could look at DVDs on his GNU/Linux computer. It turns out the software to view DVDs only exists for Windows. Again while the intent of the law is good, in practice it can be used to freeze out competitors which is bad and runs counter to the eventual commoditization of computing.

The Future

As usual, the future does not happen all at once or at the same pace for everyone. It is also true that people or businesses with a vested interest in the status quo will fight commoditization of computing. While the trend towards a utility computing model seems clear, it will not happen overnight. Moreover, there will be computing sectors that transform more slowly than other sectors. One sector that is rapidly moving towards a utility computing model is the Internet-enabled services sector. We have already seen how Google's server farm is powering Web searches with commodity hardware and software. Other Internet services are powered in much the same way.

This trend towards commodity hardware, communications and software has not gone unnoticed by traditional hardware and software vendors. Some software vendors such as Novell and Red Hat have embraced the trend. Hardware vendors like HP and IBM have put considerable resources into the trend towards commodity software by supporting a number of Open Source initiatives. Other companies such as Microsoft see the commoditization of software as a threat to their currently highly profitable software business.

Some of the outcomes are uncertain. For example, will a new service vendor emerge to deploy a true computing utility? Will the utility model be embraced

by one of the incumbent hardware vendors or software vendors? Will a new and as yet unforeseen way of doing computing emerge?

We will see challenges to commoditization and barriers put in its way. However, much like the massive industrialization that transformed the US from a rural to an urban society, the trend is inevitable. It will come with some dislocation and pain. We will also see a reshuffling of the major players in the computing world and perhaps a new major player will emerge in the next few years. Regardless of the details of the outcome, computing will never be the same.

Feedback

If you have any questions or comments, please contact me at smoot@tic.com.

Acknowledgment

I would like to thank Rich Kittle (stratpro@comcast.net) for his review of and contributions to this paper. His suggestions and additions were invaluable. I would also like to thank Kristi Browder and Jim Becker at Encompass US (an HP User Group) for their help with review and distribution.

Copyright

Copyright © 2004 FOR Encompass USE ONLY - NOT FOR COMMERCIAL PURPOSES. All rights reserved.

Published by the Enterprise Computing Association (Encompass US), a non-profit Massachusetts membership corporation. Permission has been granted by the author, Smoot Carl-Mitchell, solely to Encompass for reproduction of this white paper. The ideas and concepts set forth in this publication are solely those of the respective author, and not of Encompass, and Encompass does not endorse, guarantee, or otherwise certify any such ideas or concepts in an application or usage. Printed in the United States of America.

Encompass and the Encompass logo are registered trademarks and service marks of Encompass.

All products or name brands are trademarks of their respective holders.