ghaff@
illuminata.com

Open Source in the Next Computing Wave

Insight

Gordon Haff
9 January 2009

Open source software has been both benefactor and beneficiary of the "Internet wave" of computing during which large-scale, network-connected computing architectures built from relatively standardized hardware and software components have came into their own. The Nineties may have been the years when small workgroup systems running operating systems such as NetWare and Windows NT arrived throughout the business world in a big way. But the Noughts have made distributed systems a core part of almost every datacenter.

Open source has fitted this evolution well. Linux itself came to be, in a sense, the unified Unix that had failed to be birthed through more conventional commercial partnerships.[1] And Unix-style operating systems, historically closely tied to the rise of computer networks (including standards like TCP/IP that underpin the Internet), were a great technical match for an increasingly network-centric style of computing. At the same time, those computer networks provided the widespread connectivity that collaborative open source development needed to flourish. For users, these networks provided easy access to the open source software and a means to connect to, and engage with, a community of other users. Nor did it hurt that the large scale of many of these new computing infrastructures made cost a bigger issue than ever before—which helped to drive the proliferation of x86-based servers and open source software.

Today, a fresh set of trends and technologies is changing the way that we build computing systems and operate them. Two of the biggest are virtualization and cloud computing. Virtualization effectively decouples operating systems and their applications from server hardware, and thereby makes it easier to physically move them from one machine to another. Cloud computing is changing where applications run—from on-premise to out-in-the-network.

Business dynamics are also changing. Even if it's often just a self-interested concern about their power bill, we are starting to see a greater awareness of environmental issues among those responsible for operating datacenters. The current economic climate is also forcing more systematic thinking about costs in general, including those associated with overall complexity and the security and resiliency of large distributed infrastructures.

These trends intersect in powerful ways; a new wave of computing is gathering momentum as a result. And open source is once again playing a major role.

1 We use "Unix" here in the sense of a family of modular operating systems that generally share programmatic interfaces and other conventions and approaches.

Cloud Computing's Coming of Age

We consider cloud computing first. There's certainly plenty of buzz about it. For our purposes here, we define cloud computing as accessing computing resources over a network—whether those resources take the form of a complete application (Software as a Service—SaaS); a developer platform such as Google Apps or Microsoft Azure; or something that's more akin to a barebones operating system, storage, or a database (Amazon Web Services).[2]

As recounted by, among others, Nick Carr in his The Big Switch, cloud computing metaphorically mirrors the evolution of power generation and distribution. Industrial Revolution factories—such as those that once occupied many of the riverside brick buildings I overlook from my Nashua, New Hampshire office—built largely customized systems to run looms and other automated tools, powered by water and other sources. These power generation and distribution systems were a competitive differentiator; the more power you had, the more machines you could run, and the more you could produce for sale. Today, by contrast, power (in the form of electricity) is just a commodity for most companies—something that they pull off the grid and pay for based on how much they use.

The economic argument underpinning cloud computing has two basic parts. The first relates to how firms should generally focus their resources on those things that differentiate them and give them advantage over competitors. Computer systems—especially those devoted to mundane tasks such as email—aren't one of those differentiators for many companies.[3] The second part relates to size and scope of computing facilities. Efficient IT operations involve a high degree of standardization, up-front design, and automated operation; applying this degree of "industrialization" to datacenters and their operation isn't really viable at small scale.[4]

Open Source in the Cloud

Open source is very much part of cloud computing. The benefit of open source to the cloud providers is clear at several levels.

First, there's the matter of cost. Open source isn't necessarily "free as in beer" (to use the popular expression)—that is, zero cost; companies often want subscription offerings and support contracts even if the bits are nominally available for free. But it does tend to be less expensive than proprietary alternatives even when some production-scale features are extra-cost options (as in the case of the monitoring tools in MySQL Enterprise). And this is no small consideration when you look at the size of providers like Amazon and Google which often seem to add datacenters at a rate that many companies once added computers.

Open source software is also just a good match for this style of computing. For one thing, cloud providers—almost by definition—are technically savvy and sophisticated. Although they don't want to reinvent every wheel, they're generally ready, able, and willing to tweak software and even hardware in the interests of optimization. Open source software and, more broadly, open source communities with which they can engage, are therefore a good fit given that they can modify source code and otherwise participate in evolving software in a way that meets their requirements.

There are some areas of friction between open source and cloud computing. We see this in the ongoing social and community pressure on large cloud vendors such as Goggle to make their "fair share" of contributions to open source projects.[5]

2 See our To Cloud or Not to Cloud for more discussion of the different forms that cloud computing takes.

3 One of the earliest examples of widespread outsourcing of a computing task was payroll. This function is certainly important but having "better payroll" (whatever that would mean) isn't something that advantages a company.

4 There's an ongoing debate over how big "big" needs to be. See our Bigness in the Cloud. But there's general agreement that the entry point is somewhere around large datacenter scale.

5 Most "copyleft" open source licenses, such as the GPL, don't require that code enhancements be contributed back to the community when the

Proprietary Web-based applications and services—such as those from Google, 37signals, Salesforce.com, and even some traditional software vendors—also tend to mirror certain open source strengths such as easy acquisition.

However, in the main, it's a largely healthy and mutually beneficial relationship. Open source is widely embraced by all manner of technology companies because they've found that, for many purposes, open source is a great way to engage with developer and user communities—and even with competitors. In other words, they've found that it's in their own interests to participate in the ongoing evolution of relevant projects rather than simply taking a version of a project private and then working on it in isolation.

Virtualization

Virtualization is the other buzziest IT topic today. Truth be told, when it comes to enterprise computing, it's actually of more immediate interest than cloud computing given that it's a more developed set of technologies and its use cases are better understood.[6]

To better understand how server virtualization[7] plays with both cloud computing and open source, it helps to think about what virtualization really is and how it is evolving. The core component of server virtualization is a hypervisor, a layer of software that sits between a server's hardware and the operating system or systems that run on top in their isolated virtual machines (VM). Essentially, the hypervisor presents an idealized abstraction of the server to the software above. It can also make it appear as if there are multiple such independent servers (all of which, in reality, cooperatively share the physical server's hardware under the control of the hypervisor).

This ability to share a single (often underutilized) physical server is certainly a salient trait of virtualization. In fact, it's the main reason that most companies first adopt virtualization—to reduce the number of physical servers they have to purchase to run a given number of workloads. However, looking forward, the abstraction layer that virtualization inserts between hardware and application software is at least as important whether it's used to run multiple operating system images on a single server or not.

Historically, once an application was installed on a system, it was pretty much stuck there for life. That's because the act of installing the application—together with its associated operating system and other components—effectively bound it to the specifics of the physical hardware. Moving the application meant dealing with all manner of dependencies and, in short, breaking "stuff." Placing a hypervisor in the middle means that the software is now dealing with a relatively standardized abstraction of the hardware rather than actual hardware. The result is greatly increased portability.

Portability, in turn, enables lots of interesting practical uses. For example, administrators can take a snapshot of an entire running system for archive purposes or to rollback to if there's a problem with a system upgrade. VMs can be transferred from one system to another, without interrupting users, to balance loads or to perform scheduled maintenance on a server. Ultimately, virtualization enables what is often called a virtual infrastructure or a dynamic infrastructure—by whatever name, an infrastructure in which workloads to move to wherever they are most appropriately run, rather than where they happened to be installed once upon a time.

Open Source in Virtualization

VMware both brought proprietary server virtualization to the mainstream and has been the

---

software is delivered only in the form of a service, as is typical with cloud computing.

6 Although various antecedents to, and subsets of, cloud computing go back some time—think hosting providers or even timesharing.

7 At its most conceptual, virtualization is an approach to system design and management that happens in many places and at many layers in a system. For our purposes here, virtualization refers specifically to the particular approach to server virtualization described.

vendor who has most benefited from it to date. However, a variety of virtualization options are now available, enabled in part by enhancements to processor hardware from AMD and Intel that simplify some of the more difficult aspects of simplifying x86 hardware.

Among these options are Xen and KVM. Both open source projects are part of standard Linux distributions.[8] They are also available in the form of a "standalone hypervisor"—essentially a small piece of code (often embedded in flash memory) that lets a server directly boot up into a virtualized state without first installing an operating system. "Guest" operating systems can then be installed on top in the usual manner. Xen is the more widely-used and mature of the two today. But Red Hat bought Qumranet—the startup behind KVM—in early September 2008 and is focusing on KVM as its strategic virtualization technology going forward; KVM has also been incorporated into the mainline Linux kernel since version 2.6.20.[9]

Virtualization has a close relationship to cloud computing, especially cloud computing implementations that provide users with an execution environment in the form of a virtual machine.[10] Virtualization brings a lot of the properties you'd want a cloud computing environment to have. You want to be able to store snapshots of your environment to use in the future. Check. You want to be able to spin up applications dynamically and shut them down when they're no longer needed. Check. You want to insulate users from details of the physical infrastructure so that you can make transparent upgrades and other changes. Check. Virtualization isn't a universal requirement for all types of cloud computing. High performance computing in particular often uses an alternative form of virtualization that's more about distributing a single large job to a large number of servers using certain standard protocols—sometimes called a grid. However, server virtualization is certainly an ideal complement to many cloud computing implementations.

Cloud computing providers have adopted open source virtualization approaches (especially Xen) for many of the same reasons that they've widely adopted open source in general. Amazon Elastic Compute Cloud (EC2) is an illustrative and well-known example of virtualization, paired with Linux, in the cloud.[11] With EC2, you rent VMs by the hour. Each VM comes with a specific quantity of CPU, memory, and storage; currently there are five different size combinations available. Users can then build their own complete VM from scratch. More commonly, they'll start from a standard Amazon Machine Image (AMI)—an archived VM pre-loaded with an operating system, middleware, and other software.

Initially, these AMIs consisted almost entirely of community-supported Linux distributions. However, one of the things that we now see happening as cloud computing evolves from a developer-centric, kick-the-tires stage to something that supports production applications and even entire businesses,[12] is that some of the same concerns that are relevant to software running in an enterprise datacenter are finding their way into software running at cloud providers.

An example of this trend is AMIs with Red Hat Enterprise Linux (RHEL) and the JBoss Enterprise Application Platform (currently in a supported public beta phase). This allows enterprises running RHEL inside their firewall to run the same operating system on Amazon Web Services (AWS). They might do this as part of migrating to, or running, just new applications in the cloud—or for using the cloud to handle temporary workload spikes. Precise support policies can vary by software

8 Xen is also the basis for the server virtualization in Sun's OpenSolaris and xVM. See our Virtualization Strategies: Sun Microsystems.

9 Red Hat is doing this for both business and technical reasons. See our Red Hat Makes Buy for KVM—But VDI Too.

10 Providers of other types of cloud computing, such as SaaS, may also use virtualization—but, such details of their technology infrastructure are hidden from users of the service. In fact, that's sort of the point.

11 At the end of 2008, Amazon also added support for Microsoft Windows and SQL Server to EC2.

12 See our SmugMug and Amazon S3.

vendor, but—in general—running the exact same operating system and middleware stack in a remote environment as locally should allow applications that are certified and supported in one place to also be certified and supported in the other.

The Green in Green

It's popular to talk about "green," environmentally-sensitive computing these days. Some green computing is, indeed, green for green's sake. It may be driven by government regulation[13] or it may be part of a high-level corporate initiative undertaken for brand image or other reasons. However, much of the time, companies undertake energy efficiency and conservation projects for the most pragmatic of reasons: they can help the bottom line. Especially when bringing new IT capacity on-line, it's profitable to factor power and cooling costs into any financial analysis.

Optimizing power use dovetails with the other two trends that we've discussed—virtualization and cloud computing—in important ways.

Perhaps the most obvious intersection is with virtualization in its basic guise as a way to consolidate applications onto fewer physical servers. Reducing the number of servers cuts acquisition costs, certainly. However, it's also the case that the server with the lowest environmental impact—in all dimensions, not just power draw—is one that doesn't exist.

More broadly, virtualization brings dynamism to an IT environment through features such as the live migration of virtual machines from one server to another and the dynamic allocation of hardware resources in response to workload changes. These in turn enable what's often called "automation," essentially distributed workload management in response to user-specified policies. In the open source space, Red Hat Enterprise MRG (messaging, realtime, and grid) is an example of technologies organized around an automation theme.

[13] Such as the RoHS (Regulation of Hazardous Substances) directive in the European Union.

Among the benefits of automation is that, as the demands on a datacenter's infrastructure change over the course of a day, the course of a quarter, or in response to traffic spikes, physical resources that aren't immediately required can be turned off until they are needed. If fewer Web servers or application servers are needed at night, they needn't be powered-up all the time. There are also analogous examples in the storage arena where data that doesn't have to be instantly available can be moved to lower-power near-line storage such as tape or MAID (massive array of idle disks).

Automating and Integrating

In practice, most datacenters are still in quite early days when it comes to broadly applying automation. There are a variety of reasons for this. One is simply that it's a largely new way of thinking about operating servers. After all, it took years before (just about) everyone got comfortable with letting an operating system schedule jobs on multi-processors within a single server. The adoption pattern of automation in a distributed environment will be no different. What's more, virtualization—to say nothing of the tools that automate its use—is still quite new on the time scale of IT evolution. Short version: These things take time.

There's also a prosaic but very real issue associated with changing IT operations for the benefit of the power bill; the budget for IT gear, software, and the people needed to run it efficiently is often (indeed, usually) separate from the budget that pays for utilities. And, however much many managers want to—in principle—make the right decisions for their company take as a whole, in practice actions tend to be driven by individual and departmental budgets and other incentives. Which brings us back to cloud computing.

Many of the things we think of as Green IT fundamentally relate to efficiency. So is, fundamentally, cloud computing. It's based on the premise that specialized service providers can deliver computing less expensively and with a better level of service than small operations that don't have IT as a core competency.

Efficiency partly relates to the idea that larger scale helps to smooth out demand. If different customers consume computing at different times and in different patterns, the aggregate demand is smoothed out—and requires less infrastructure (at consequently reduced cost and power consumption)—than would if individual customers were to provision themselves for their own peak loads.

It also suggests an "industrialization" of the datacenter, to use a term applied by Irving Wladawsky-Berger of IBM. By this he means professional, disciplined, and efficient IT management. This suggests things like reducing complexity (in terms of number of applications and platform types) and the use of the sort of automation tools such as we discussed earlier. Ultimately, one of the big things that cloud providers sell is the quality of their IT, whether manifested as high service levels, low cost, ability to scale, or some other attribute. And, even for those applications that enterprises decide to continue running internally, cloud providers provide a benchmark of what is possible.

If we talk specifically about software services delivered by an external provider, however, one final aspect of cloud computing is especially interesting in a Green IT context. When we buy software in the form of a service rather than installing it locally—say SugarCRM On-Demand or Zoho Business—we no longer need to buy, operate, or power a local server. The software is still running someplace, of course: on a service provider's hardware. The difference is that we are now implicitly paying for all those costs as part of our software subscription. They're no longer hidden in someone else's budget.

Thus, cloud computing—as it develops—should help lead to more efficient IT operations and, as a sort of side effect, will make the full costs of running an application more visible.

Conclusion

A new wave of IT is now forming at the intersection of cloud computing, virtualization, and a generalized drive towards simplicity and efficiency. Cloud computing's aim is to provide software, platforms, and infrastructure in the form of a service from providers optimized to do so at scale. The collective goal of technologies and concepts such as automation, orchestration, and virtualization is to break tight bonds between applications and the underlying physical infrastructure in order to use that infrastructure more efficiently and to change the resources available to those applications on-the-fly.

Each of these individual trends is important. But some of the biggest wins for IT departments will come when they consider the ways that these trends play off and cross-support each other. And their technology choices should reflect this—including those that involve open source software.

Open source is clearly a significant part of this new wave. Part of its role is essentially more of the same as open source projects proliferate and mature throughout software ecosystems. Open source benefits such as easy acquisition and trial, ability to modify, and general adherence to standards that have been important to end-users are also equally (or even more) important to the cloud providers delivering a new generation of software services.

However, "free and open source software" (FOSS) has never been just about viewing and modifying bits. It's introduced new ways of thinking about how we collectively develop, consume, and pay for software. And continuing that thinking about what protections and rights the users of software have and should have may be as important a continuing role for FOSS as the actual operating systems, middleware, and applications built on the open source model.