ghaff@
illuminata.com

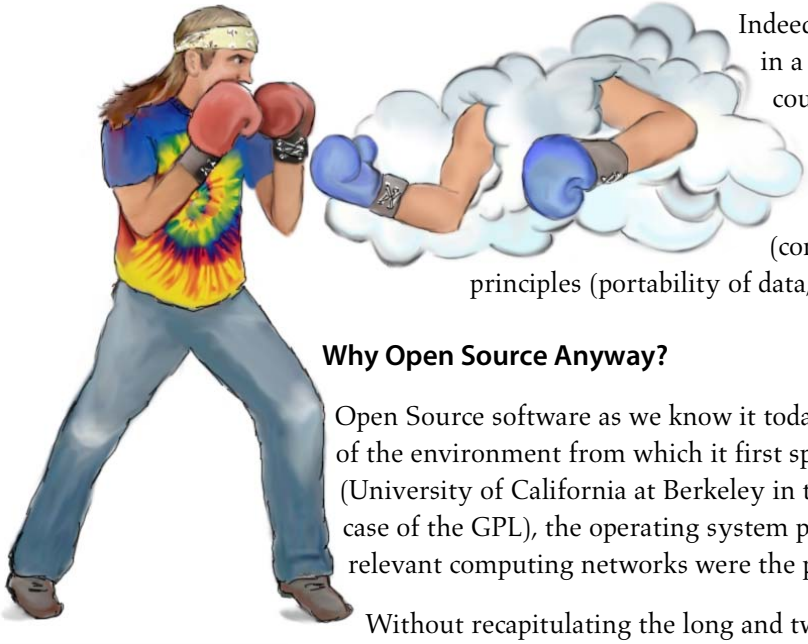
The Cloud vs. Open Source

Research Note

Gordon Haff
31 January 2008

Open Source was clearly the software story of the decade between about 1995 and 2005. Linux unified the Unix family of operating systems to a degree that none of the formal Unix standardization and cooperation efforts ever came close to achieving. Other Open Source software from programming tools to databases to application servers captured huge market share, especially among new, network-facing applications and services. Open Source is in many ways a child of the Internet. At the same time, the vast, economical panoply of Open Source infrastructure has played no small role in the Internet's build-out.

Open Source software remains immensely important. However, its impact won't continue to expand to nearly the same degree that it's done over the past ten years. There are a variety of reasons, but one predominates: the shift of software from the desktop and the datacenter out into the network cloud. In short, the new action in software isn't taking place where source code availability has the same applicability or import as it does when the software is being loaded up and run on local hardware.



Indeed, focusing too narrowly on Open Source in a Cloud Computing[1] world is counterproductive. Source code may matter, or it may not, depending upon the circumstances. But it's the many other aspects of Open Source development (community, extensibility) or Open Source principles (portability of data, open formats) that matter far more.

Why Open Source Anyway?

Open Source software as we know it today is, in important respects, the product of the environment from which it first sprang. The setting was largely academic (University of California at Berkeley in the case of BSD Unix, and MIT in the case of the GPL), the operating system platform was largely Unix, and the relevant computing networks were the precursors to the Internet.

Without recapitulating the long and twisted history of Unix' genesis at Bell Labs, its growth into a commercial operating system and the subsequent Unix

[1] See our *Defining Cloud Computing* for our view of what Cloud Computing is (and isn't). We consider Software as a Service (SaaS), Hardware as a Service (HaaS), Data as a Service (DaaS), and Web 2.0 to all be part of the cloud. The common thread is that the Network functions as a sort of abstraction layer and allows access mostly through "Web-y" protocols, languages, and standards like HTTP, RSS, XML, Javascript, and REST

wars, suffice it to say that a number of historical factors greatly influenced what we usually think of as Open Source today:

- Throughout the early history of Unix, source code was widely available and widely-shared. For example, large chunks of mid-Seventies vintage Sixth Edition Unix became widely available in samizdat fashion; and were eventually published in a well-known book Lion's Commentary on UNIX 6th Edition (though the book was published with the right to use the code for educational purposes only).
- During the 1980s, the mechanisms to exchange files and communicate through email and newsgroups was fairly commonplace in the computer science departments at places like Berkeley and MIT at a time when they were still relatively unknown in the broader world.[2]
- Unix was designed as a portable operating system that could run on a wide range of incompatible hardware platforms, but had to be modified to do so.

Thus, this culture had a history of sharing source code, mechanisms to share source code and work on it collaboratively, and—by no means least—access to source code was very useful because you needed it to port programs to all the varied OS flavors, processor architectures, and hardware designs out there.[3] "Software Freedom"[4] therefore focused on viewing, modifying, and redistributing source code —often with license terms that reflected specific technical aspects of a Unix environment—because the source code and the right to run it were what mattered most. The ideological underpinnings of the Software Freedom movement are not really about Open Source per se but, historically, Open Source was the practical mechanism to achieving the greatest degree of freedom.

As a result, today, most people tend to think of Software Freedom and Open Source as more or less the same thing—even though they really aren't.

The Rise of the Cloud

Until recently, the basics of operating systems and the way that software gets installed on them and used hadn't changed a whole lot from the state of affairs in the early days of Open Source.[5]

To be sure, we've seen a considerable reduction in the number of platform variants over the years with the result that a single set of program binaries can usually run across x86 systems running either most mainstream Linux distributions or Windows. So access to source code is arguably less important to typical users than it once was.

Nonetheless, Open Source as essentially an approach to development and community has been broadly successful. That end-users can look at and modify source code as well is almost incidental in most cases.

However, we're starting to see a major shift in the way that software gets delivered and run. I'm speaking, of course, about Cloud Computing. This is the concept that, rather than loading up software on your own computer, you use it in the form of a service over the Internet. Canonical examples include Google search and Salesforce.com, but examples are proliferating—albeit more quickly in the consumer space than in enterprise applications. Think of the whole "Web 2.0" industry from Facebook to Flickr.

Why? The most widely-used analogy is to the electric power utility. Originally, everyone generated their own power. Now it's usually just delivered over the grid with each user taking whatever is needed at a given point in time, and paying based on how much they use. The reason is primarily a matter of scale. It's far more cost-effective to have specialists operate a few big power

[2] The first version of the GPL license dates to February 1989; BSD got its start about a decade earlier.

[3] Although "freeware" (software that could be freely used without payment) was ubiquitous on PCs essentially from the beginning, Open Source didn't take off in that space until it was already flourishing elsewhere. I suspect one reason is that source code was far less useful on the relatively homogeneous "IBM PC."

[4] "Free software is a matter of liberty, not price. To understand the concept, you should think of free as in free speech, not as in free beer." tinyurl.com/4zrk3

[5] See our *The Future of the Operating System*.

plants than for thousands or millions of consumers to each run off their own generators.

There remain many questions about how the Cloud will evolve. What types of processes and applications will enterprises be willing to move into the Cloud? To what degree will users continue to run their own Web-enabled applications in the Cloud rather than third-party services?[6] Do the economic scale points actually favor a relative handful of mega-providers or should we expect to see many smaller ones? Will organizations insist on creating internal-use-only Clouds, just as they have with Grids?

However, whatever the particulars of the Cloud's evolution, it's clear that much more software is moving out into the network, even if all of it doesn't in the foreseeable future. And that has significant implications for Open Source.

Open Source Peaks

One implication is that we may not see Open Source continuing to extend its reach into new types of applications and uses to the degree that we've seen to date.

The genesis of Open Source was in the operating system and a variety of closely-related components such as compilers. The next big step was into middleware—application servers such as JBoss and databases such as MySQL. Over the past few years, we've also started to see more Open Source projects that are true applications: in other words, software that ordinary end-users—rather than developers and system administrators—interact with. Examples include SugarCRM (Customer Relationship Management), JasperSoft (Business Intelligence), and Alfresco (Content Management).

Open Source's march to date has been on several dimensions. It's moved "up the stack" from the operating system to the applications. It's progressed from network-facing servers back to the application tier, and even into back-end transactional applications in some cases. It's taken on more roles that businesses view as mission-important, even mission-critical. And many of the network-centric applications where Linux got its start have themselves become increasingly important over time. Open Source applications touch more and more parts of the business; they're increasingly judged suitable for even "high surface area" applications—to use Kevin Efrusy's term.[7]

Without other major changes to the software universe, it would be reasonable to assume that these trends would more or less continue. Perhaps Open Source wouldn't find its way into every niche area. Open Source tends to be better suited for software with a large community of potential developers and, especially, users who will monetize it at some level. But there are certainly large swaths of horizontal application types where Open Source still has a minimal presence.

However, the advent of Cloud Computing changes the picture. Whatever becomes of existing IT infrastructures over time, it's increasingly clear that new styles of applications are more likely to be accessed over the network than installed on a local server in the traditional manner. Furthermore, new companies that don't already have an IT infrastructure in place are far more likely to explore SaaS and HaaS alternatives of various types than they are to host everything in a homespun datacenter.

In short, the action in software is shifting. This makes it harder for Open Source to usurp additional roles in the enterprise datacenter because those roles aren't going to be in the datacenter to usurp; they're moving into the network.

Open Code in the Cloud

There is, of course, no reason that Open Source cannot be used in the Cloud just as it is in the enterprise datacenter. And, indeed, we see examples of this. You can install the Open Source SugarCRM on your own server; or you can also purchase a

[6] I.e. using Hardware as a Service (HaaS) like Amazon's EC2 and S3 rather than a Software as a Service (SaaS) approach. See our *Web Services Flow Down Amazon*.

[7] See our *Open Source: What Makes for Success?*

subscription to Sugar On-Demand, a hosted version. Similar options are available for other Open Source projects such as Trac and Wordpress. Furthermore, if you're using a HaaS provider, such as Amazon, you can run any software you like—Open Source or otherwise.

However, with respect to SaaS and Web 2.0 specifically, where neither the user nor anyone else on the user's side of the wire touches the software directly, the dynamics around Open Source are—at the least—different. Furthermore, as Tim O'Reilly has noted, the "freedoms that led Richard Stallman to the GPL" aren't especially protected by current Open Source software licenses in a Web 2.0 world.[8]

First, some basics. One big reason that SaaS changes things is that most Open Source licenses—and all the widely-used ones—don't consider software delivered in the form of a service as "distributing" that software. And it's distribution, not use, that triggers the "copyleft" aspect of licenses like the GPL and thereby requires that modifications and enhancements be contributed back to the community. Thus, so long as an application is only accessed as a service, the company offering that service can base it on Open Source code and add proprietary value-add enhancements without any requirement to make the code for those extensions available. As far as the GPL is concerned, the code is just being run internally by the vendor delivering the service.

Some see this as simply a loophole—albeit an enormous, and enormously important one. There was, indeed, pressure from some quarters to plug it during the drafting process of the latest GPL license, GPLv3. However, instead, the Free Software Foundation created a separate license, the Affero GPL, that adds a provision to make the copyleft terms of the GPLv3 also applicable to software delivered in the form of a service over the network.

Such a worldview implicitly assumes that copyleft is the only reason that Open Source users contribute back their enhancements. Copyleft may or may not have played a major role in the rise of Open Source. Certainly, the GPL has long been the most common Open Source license, used by Linux, GNU, and many others. However, the BSD license—which does not require that code changes be made available—is also widely used. It's an interesting historical debate whether the ultimate impact of Linux was far greater than the BSD operating system because of license differences, or because of other reasons—of which there were many. In any case, Open Source does not begin and end with the GPL and copyleft.

And that's just looking at history. Today, 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. Therefore, the concern that, left to their own devices, companies will wholesale strip-mine Open Source projects and "take it all private" seems anachronistic. That's not to say that everyone will always contribute as much code without copyleft as with it, but the suggestion that copyleft is all that's holding the whole Open Source process together just doesn't square with the facts.

Other Freedoms

At the same time, to focus on source code is to focus on a specific type of openness and freedom that was important historically—but may not be as important going forward. Indeed, in the case of Web services running on massive server farms and cooperating over a network with all manner of other code, services, and data, the value of code is questionable. After all, you can hardly just load it up on a server and do anything useful with it anyway. One needs all those servers and interlocking pieces. Also, the ability to view, modify, and redistribute source code is only one of many rights or protections to consider in a Cloud Computing world. For example, consider these other things that might matter more:

Ownership and portability of data. When you store information in the Cloud, can you get it back out? And can you get it back out in a way such that it's useful and portable? These questions become even

[8] tinyurl.com/ypav7g

more fraught when you consider that this is not a simple question of downloading files that you have stored on a disk somewhere on the network. Your "data" may also consist of your network and relationships to other data and people in the Cloud. What type of portability even makes sense in that context?

Open APIs. Open Source as we know it today evolved largely in the context of Unix-like operating systems and the programs that ran directly on top of them using "libc" and other system libraries. While we may run monolithic programs over the network, much of the action in Web 2.0 has been in services such as Facebook, Flickr, Google Maps, and Salesforce.com that expose application programming interfaces (API) at a higher level. This allows developers considerable freedom to extend these platforms. Thus, whether a platform or application is Open Source or not, given public APIs, it can be extended and consumed in ways that are very analogous to Open Source. At the same time, the predictability and transparency of the terms of service for APIs—especially in the case of consumer-oriented services—raise their own issues.[9]

Privacy and security. At the O'Reilly Open Source Conference (OSCON) last summer, Eben Moglen of the Software Freedom Law Center referred to Google and its ilk as a "private surveillance system that [the government] can subpoena at will." He went on to describe this as a "uniquely serious problem." It's hard to dispute that such services create an unprecedented centralization of data—both that explicitly placed into the Cloud and that generated with each search or purchase. This is anathema to those who saw Open Source and inexpensive computers as a great victory for the decentralization of computing and information.

Cribbing From Open Source

Open Source bears at most an incidental relationship to computing in the network. It's not irrelevant. But that's because Open Source-ish methodologies and approaches have become pervasive in software development.

Consider a company like Adobe Systems—long known for dominating large swaths of graphic arts workflows with expensive, proprietary software. Adobe has hardly become an Open Source paragon. However, it Open Sourced Flex—its cross-platform framework for building rich Internet applications. And PDF, long a de facto standard for fixed-layout document format, recently became an official ISO standard.[10] Furthermore, when Adobe released its new Adobe Lightroom software—a competitor to Apple's Aperture—last year, it did so through a long public beta that clearly influenced the program's final design. Thus, even though Lightroom is not Open Source, it's made use of much of the same sort of community involvement so often associated with Open Source projects. The latest version of the program also has an SDK that allows programmers to extend the program—another hallmark of Open Source projects like Firefox. And—it almost goes without saying—you can trial the software before buying it.

Approaches such as these are hardly unique to Adobe. They're not even particularly new. What is new is how ubiquitous they have become. Indeed, they're so ubiquitous that it's easy to forget how utterly alien the transparency and interactivity of the Lightroom development process would have seemed just a decade ago. Open Source has much to do with that change. Even closed or private source software borrows much from the community models that Open Source helped to foster and tune.

Conclusion

Open Source has caused a major sea change in the way that software is developed and consumed. Many of the characteristics that we associate with Web 2.0 and other computing in the Cloud—the importance of building community, rapid

[9] See our *Mashups Meet Commerce*.

[10] To be sure, Adobe's motivation may have been partly to counter a similar move by Microsoft as a way to promote its competing XPS approach. See our *The Problem with Openness*.

incremental updates, and easy entry points—have much in common with Open Source.

However, Cloud Computing tends not to expose source code to users—and that's a major departure from Open Source which, by definition, does. Some view this as a loophole to be closed. However, that is to assume that viewing and modifying code is an end to itself rather than a means to ensuring users more fundamental rights and protections around using software and accessing their data.

Open Source can and will coexist with Cloud Computing. But, with dramatic changes brought by Open Source already incorporated into many previously closed, proprietary development processes, it will no longer be so uniquely at the center of the big questions around how user freedoms are to be ensured, and how software is to be built and delivered.