by daniel chudnov
Information Technology Specialist
Office of Strategic Initiatives
Library of Congress

A View From the Clouds

CLOUD COMPUTING IS DEFINITELY A THING NOW, BUT IT'S NOT NEW AND IT'S NOT EVEN NOVEL.

It's strange to watch something you've taken for granted for years suddenly become "a thing." Cloud computing is definitely a thing now, but it's not new and it's not even novel. Back when we were first learning about the internet in the 1990s, every diagram you saw showing how the internet worked had a big cloud in the middle. That cloud represented the diverse links, routers, gateways, and protocols that passed traffic around in common, low-cost ways between our local networks and other local networks. If you hooked your local network up to the cloud, everything else seemed like it was right next door. Ten years passed, and most of us didn't spend much of the intervening time thinking deeply about that cloud anymore. We've all been leaning on the cloud to connect us to each other for all these years. So why the sudden interest in "cloud computing" now?

The simple explanation is that you can do a lot more than just talk to each other in the cloud now. For a very low marginal cost, you can host and edit documents, manage group projects, run surveys, track usage statistics, and apologize to your former high school mates for various rudenesses you committed way back when. But that's not exactly news either, is it? Everyone writing about Web 2.0 a few years ago did plenty to hype these kinds of new applications. So what changed?

Nothing Has Changed … Except Everything

On the one hand, nothing changed. More and more of these kinds of applications appeared, providing web-based ways to do things we've done for years on our desktops and, before that, on the reliable old "pen and paper" platform. A few key differences matter the most: Better usability makes it easier for new apps to gain popularity; social features like simultaneous editing of documents and users as access points offer efficiencies over old approaches (and inefficiencies due to distraction!); free or low prices can make offloading service hosting into the cloud a worthwhile move. This last point is the obvious one right now, as budgets stretch and IT skills stay scarce, but the option of switching to cloud-hosted services wouldn't be compelling at all if those first two components—greater usability and efficiencies from social features—weren't part of the deal.

On the other hand, everything's changed. If going from pen and paper to a computer was the first step, and going from software on your computer to software in the network cloud was the next step, having computers themselves in the network cloud feels like the next big shift. Many of us have been using "virtual machines" (VMs) for years. Virtualization is a decades-old concept in computer science: splitting up the hardware resources

on a single machine to give one or more users the experience of using two or more machines. This has become much more powerful as RAM and multicore CPUs have become cheaper. Almost any desktop computer you buy from a big hardware maker these days and most laptops (though probably not most netbooks) can run software like VMWare or Parallels or VirtualBox to provide a Windows user a virtual Linux machine or a Mac user a Windows VM. To the user, it feels just like you're using one computer inside of another, and if configured properly, it can feel like you're using more than one, but seamlessly so.

MANY OF US HAVE BEEN USING 'VIRTUAL MACHINES' FOR YEARS.

Even this moved into the cloud several years ago, as Virtual Private Servers (VPSs) became a common hosted service offering of many ISPs. A VPS is like a VM hosted in the cloud: Instead of firing it up on your desktop, you log in to it over the network, and it could be anywhere. These days, for as low as $20 a month, you can rent a reliable VPS from dozens of different providers. With a VPS like this, you can have full superuser rights over that hosted VM and run whatever software you want. Signing up for a VPS takes just a few minutes, and within a few hours of your request, most ISPs will email you the keys to the virtual car, which you can fire up and play with to your heart's content.

With a typical VPS contract, you pay your monthly fee—maybe a little more or a little less than standard charges depending on how much virtual RAM and disk storage you request—and hang onto your VM for as long as you want it. This is great for stable services such as websites, blogs, and small business applications. But what if your application requires access to a ton of data? Or what if your application gets popular suddenly and you need to be able to run it on a much more powerful host, or on many hosts, to keep up with traffic?

Services such as Amazon.com's Amazon Elastic Compute Cloud (EC2) take the VPS concept and push it to the extreme. Today, for pennies an hour, with an EC2 account you can log in to the web-based EC2 management console and fire up one, two, 10, or 200 virtual hosts with just a few clicks. Each one costs pennies per hour—and just a few pennies more or less depending on how powerful each is—and when you're done with them, you can turn them all off with just a few more clicks. It's awfully cool. I've used it myself, and after a fairly brief learning curve (this isn't really for casual users without any system administration experience), I was firing up servers left and right, for 10 minutes here, for 2 hours there, and shutting them down immediately thereafter. I accomplished a few important tasks, then stopped, and the whole thing cost me a whopping 50 cents or so. That's right—50 cents.

What 50 Cents Bought Me

A few weeks ago I was gearing up to relaunch my old link-sharing application "unalog," which I've written about before. It's back up now, and I have EC2 to thank in part for having it back online. When you're about to launch a new application or even an update of an old application, it's always best to test out the whole process first to make sure it will go smoothly and to take notes along the way so you'll have good instructions for yourself or others during the final launch to production. I was trying to do this on my desktop using VMWare, with a VM running Ubuntu GNU/Linux, just like my VPS where I wanted to host this application. My desktop VM was too slow, though. The unalog application has a sizeable database; loading this into a virtual instance of the application bogged down my machine so much that it crawled to a standstill. I wasn't about to relaunch without a successful test run, though, so after stewing for a while over my testing failure I remembered EC2 and Amazon Simple Storage Service (S3).

S3, also from Amazon.com, is one of another kind of cloud service that many of us have been using for years: cloud storage. Services such as Allmydata, Carbonite, Dropbox, and MobileMe all let you "attach" to remote storage like it's a folder on your desktop. You can move files into and out of that folder just like any other, but when you update that folder, its new content gets replicated into a storage cloud. This works well for many people, often at a cost of pennies per gigabyte, especially because you can attach that folder to multiple computers, such as a desktop and a laptop or a work machine and a home machine. This supports other practical-use cases like having an offsite backup too.

I use Amazon.com's S3, through a desktop folderlike tool called Jungle Disk, for keeping my personal files handy. These columns I write, the talks I give, and my music collection are all backed up onto S3 using Jungle Disk. I've been doing this for years, and for less than the cost of a handful of new drives to shuttle around every year, it's worked perfectly. It's particularly good for me because I also use S3 to back up my server's data—my blog, archives of old websites, and unalog data. This was especially good for me in this case because EC2 can talk very efficiently to data in S3, and when I remembered that, I knew my problem of where to test my application was solved.

I logged into the EC2 management console for the first time (you'll need to set up your account first if you haven't before) and fired up an Ubuntu VM just

like the one running on my VPS account. I quickly installed the software dependencies my app needed and copied over the data (closer to 1GB than 100 MB) from S3 in a matter of seconds. EC2 and S3 run near each other, so having that copy finish in a flash was a major bonus. Within a few more minutes I had my application running and loading data, and when that finished, I ran it all through the indexer.

That's when it broke. I wasn't indexing it correctly!

BEING ABLE TO 'RENT' VMS FOR PENNIES PER HOUR WORKED GREAT FOR ME.

Good thing I was running a test instance. I had checked my code using Bazaar, a distributed version control tool, so I made some changes to the deployment notes based on some configuration issues that came up and pushed these changes out from the EC2 host. Looking at my indexing mistake, I merged the documentation changes into my local desktop checkout, made some indexing changes there, and pushed those changes back to the EC2 host. Then I reran the indexer and it worked. I turned the site on at the temporary URL EC2 provided for me, and there was my site, up and running, with all its old data intact. Even search worked like a charm! Then, I shut it all down and went to bed. It was late.

The next day I had some free time available, so I went through the whole process again, but with one change. To avoid spending a lot of money, I had used a low-end EC2 VM, without too much RAM or CPU power. That cost me nearly 20 cents. Seeing as this fit within my allotted budget, this time I created a high-end virtual server and repeated all the deployment steps I had documented. It went a lot faster because of my indexing fixes and because of the faster virtual host. It worked. I turned it off. This second run cost me about 30 cents.

With the confidence I'd gained in running these test deployments, fixing problems, and documenting all the relevant installation notes, I then went back to my regular VPS and turned the application on. It worked.

What's the Opposite of a Silver Lining?

Being able to "rent" VMs like this for pennies per hour worked great for me, and I'll do it again in a heartbeat. It led me to think of many other ways to take advantage of what EC2 offers. That right there, to me, is why some of the current hype about cloud computing is justified—it opens doors to new possibilities. But I'll leave you with one caveat. Using EC2 or similar services for short-term problem solving can be a great way to liberate your tech staff from the limits imposed by local hardware and budgets. There's a key twist, though, and that's in how you pay for it. It's pay-per-drink; the more you use, the more you pay. This is wonderful, but it might also make it hard for an institution to pay for. It can be easier for an organization to write an annual check for $15,000 than to pay a variable monthly bill. If you want to do more than just try out cloud computing, do the math first, estimate your costs carefully, and plan accordingly. ■

Daniel Chudnov is a librarian working as an information technology specialist in the Office of Strategic Initiatives at the Library of Congress and a frequent speaker, writer, and consultant in the area of software and service innovation in libraries. Previously, he worked on the DSpace project at MIT Libraries and the jake metadata service at the Yale Medical Library. His email address is daniel.chudnov@gmail.com, and his blog is at http://onebiglibrary.net.