Chapter 4

Applications for Clouds

The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency.

– Bill Gates

In This Chapter

Cloud applications can take many forms and can be created using many development systems. In this chapter we'll learn about the tools available to build cloud applications and the variety of applications, from personal productivity to enterprise back-end support, that are available in the cloud. In this chapter we'll learn:

- The development environment's path to clouds—Just how are coding systems changing to take advantage of the new environment? What kinds of changes are happening that will help programmers "bootstrap" themselves into the clouds?
- The role of Software Development Kits (SDKs) and Applications Programming Interfaces (APIs) in rapid development—As development systems change, just how much help are cloud providers giving us in making changes?

- How abstraction is starting to leave the browser behind—Software as a Service (SaaS) started us on the road to clouds, but the browser by itself couldn't break us of the "thick application" habit until "rich Internet applications" burst onto the scene, bringing us the world the best of both worlds.
- How far we have come with higher and higher-level languages and the beginnings of abstraction—We now have an unparalleled choice of development systems, which has become a dual-edged sword and led to some unintended consequences.
- Commercial off-the-shelf, government off-the-shelf versus stovepipes—Users are much less willing to pay for ultracustomized applications that perform only a single function. Just how are the new collections of systems meeting our needs?
- Storage clouds—How are cloud storage systems luring users? Thinly veiled storage clouds are being used to connect mobile users to cloud providers, and storage might very well be the way many corporations dip their toes into the world of clouds.
- Is Google getting closer to a "true cloud"?—Google and its amazingly close tie to the Android phone system leads us to believe that we might start seeing clouds that will become even less specific and might be paving the way for cloud computing to become more of a commodity.

Introduction

Just what does a cloud application look like, and what makes it different from the applications running on your desktop PC? With grids and HPC clusters in their lineage, cloud applications also need to have their functions split in several pieces; in the case of clouds, however, the user interface is typically the portion closest to the user, and some sort of back-end process makes up the bulk of the heavy-lifting component. A common misconception is that cloud applications must be very general in function and closely resemble traditional websites in form. How did these ideas arise? They came from the early days of Web-based applications, when active user interface tools were limited and most database applications were simple screen-scraping versions of green-screen applications (basic monochrome terminal applications ported to the Web with minimal enhancements to the user interface).

As an extension of the opinions about green-screen computing, some feeling about early cloud apps was based on response to the thin-versus-thick application debates. However, as new development environments such as Adobe AIR started appearing, the traditional line between thick and thin applications became blurred. The new trend is that Web-based applications neither have to be limited to the boundaries of a Web browser nor do they necessarily mean a compromise in the user interface. Much of the sense of what is possible in cloud applications is based on newer developments in the tools used to build those applications.

Browser Versus Desktop (aka Thick Versus Thin)

> The browser has become the preferred way for delivering many applications because it allows easy deployment across operating systems and simplified application maintenance. Plus, the modern programming languages used in the browser enable rapid application design and development.
>
> The Adobe® AIR™ runtime complements the browser by providing the same application development and deployment benefits while adding desktop integration, local data access, and enhanced branding opportunities. An emerging design pattern for Rich Internet Applications (RIAs) is to deliver a browser-based version of an RIA in the browser for all users and an RIA on the desktop for more active users.
>
> (*Source:* www.adobe.com/products/air/comparison.)

Not long ago there was a great debate in the Java programming world about whether it was possible to have contextual information brought up simply by having a cursor hover over a spot on a Web page. Although we're all very familiar with this feature now, it wasn't long ago that Java didn't directly support this feature. Frustratingly, it took several years to get Java to the point where it could rival traditional programming environments for functionality. Now, however, with the browser so tightly integrated into the operating system (be it PC or Mac or Linux), the language capability to extend through to the base hardware makes even applications that demand services from a number of different hardware-

based systems possible. Applications such as Web-based video conferencing are now commonly used, and Web-based application sharing like that from Adobe (Enterprise Connect), WebEx, or Microsoft's Live Meeting is now regularly accepted as a key component in collaboration in distributed organizations.

Access to services both complex and simple is gained through the architectural structure of browser plug-ins. These sub-applications connect to and extend the functionality of the browser in known, well-defined ways. This regular architecture carries a number of benefits and a couple of significant risks. Among the benefits are simple installation, small memory and CPU footprint, and rapid function extension. The drawbacks tend to be security-related, since users can often add a browser helper or plug-in without understanding the full ramifications of the act. Just look at how many users have Yahoo, Google, and MSN toolbars in their browser and can't explain how they got there!

Plug-ins and Code Generators

The behavior and impact of plug-ins is complicated by the fact that some plug-in code will act without user intervention to make rather profound actions. For example, choosing certain toolbar plug-ins has been known to change home pages, choice of video players, choice of music/MP3 players, etc. Too many users don't really read what the plug-in is for and in some rare cases have unknowingly shared some very private information. Simpler plug-ins are often installed on demand as an alternative to the huge overhead of a full installer session for traditional applications. Moreover, some complex plug-ins are much better behaved, such as the NetExtender SSL-VPN application from SonicWall, which will remove themselves and their history upon log-out. This makes the plug-in a way to deliver functionality with very close to zero footprint.

Add to this the revolution represented by systems such as Ruby on Rails, Flex, and Ajax, which all serve as programming abstraction layers, and you have a dramatic shift in the essential nature of the chunk of code we call an application. Anything these systems can do can also be done in a lower-level language (i.e., PERL or PHP instead of Ruby on Rails), but you also also spend considerably more time to develop and debug such a program. Another hidden advantage of programming abstraction

layers is that these systems tend to force some standardization, which also increases the possibility of reusing code. You lose a bit of control, but the payback is increased development speed and increased standardization.

What we do lose is operating speed. In order to accommodate any potential situation, systems like this must have libraries and functions to cover most situations. This extra baggage is a big contributor to "code creep" or "code bloating." So, while putting up a message onto a screen could be done in a few lines of a high-level language like Python, those few lines of code can potentially expand to several hundred or even thousands of lines of code as various libraries are brought into memory to handle housekeeping. All of this extra overhead happens because the higher-level language will insert extra code just to handle any type of eventuality, regardless of whether it will be used or not. Our view is that although abstraction layers do provide quite a few benefits, the code bloat (larger and larger applications) is one of the major reasons why Moore's law exists. Every time computing capabilities take a jump, applications tend to fill the empty space. From a philosophical code-development point of view, an example from the other end of the spectrum are small-code purists such as Drew Majors, author of Novell Netware.

The Advantages of Low-Level Languages

It was the four gentlemen called the "SuperSet," with Drew Majors at its head, who developed the kernel of Novell Netware. Netware got its amazing speed in part because the "SuperSet" wrote the kernel in Assembler instead of a much higher-level language. In fact, some of the fastest code on the planet is still written in low-level languages, simply because a low-level language doesn't need to accommodate any possible eventuality. It's much easier to tune a program into a speed demon when there is less "stuff" to sift through.

Assembler and C are still among the most popular computer languages for writing machine control systems and device drivers, because of their extremely concise nature. The downside is that such concise programs also tend to be *very* difficult to write and in some cases are considered an art form. These time- and talent-intensive systems are almost always reserved for systems that are timing-sensitive, such as those in video encoding/decoding systems, flight control systems, or any application that has

ramifications if things get out of sync. Programmers capable of writing such low-level code are rare and extremely expensive. It's no wonder that abstraction layers such as Ruby on Rails, Python, SPSS, SAS, etc., have been developed.

Also due to the smaller and more concise nature of the code, low-level languages like C are also popular for embedded computing. This type of concise code is also useful for extremely small processors such as the class of devices called PICs (peripheral interface computer), which are small enough and inexpensive enough that they're found in devices as small as watches and remote controls, as well as all the way up to automobiles. These stripped-down computers are also unique in that you can "burn" the program onto the PIC so that it can't possibly be erased. More advanced versions also have flash memory (just like the flash memory cards used in digital cameras), so that new, updated versions of the program can be swapped in.

So what those high-level abstraction layers give us is that the extra code makes building blocks easier to fit together. Similar in concept to the Lego™ child's toy, the blocks below have an expected pattern that's designed to fit into the block above. Many of the examples we've given thus far deal with individual systems and their applications, but the principles are identical when applied to cloud computing, and many of the application platforms we've discussed are used for both local applications and cloud-delivered apps.

With the massive surge in processor power available in a modern computer system, it's now possible to trade off speed of development and ease of maintainability versus tight, concise programming code. Maybe that's what "junk DNA" (http://en.wikipedia.org/wiki/Junk_dna) is all about: It may be Nature's programming code leftovers that are there to accommodate other situations.

We previously mentioned the Adobe AIR (Adobe Integrated Runtime) environment as an example of how far abstraction layers have progressed in the world of application development environments. AIR allows a developer to write for a single environment that is abstracted from the underlying operating system and hardware. What Adobe promises is a wrapper environment that allows both Java and Flash programmers to ignore whether they're writing for Microsoft Windows or the Apple Macintosh. AIR also gives the developer options to develop in HTML/AJAX, Adobe Flash, and Flex. While these languages were all intended to be Web-

based, AIR has certainly been applied to all sorts of unique applications. We regularly use Klok (free software from mcgraphix) to keep track of time spent on various projects. Klok runs on several different platforms, limited only to which platform AIR is currently available for. Whether the Adobe team will fulfill AIR's destiny and extend it further in the Linux world and perhaps even the mobile world is yet to be seen.

Adobe's AIR isn't the only commercial abstraction layer in the market, but it is the first heavily supported platform that we've seen that works equally on the two major commercial operating systems. While the Microsoft .NET environment has done an amazing amount of reducing the work necessary to produce amazingly complex systems, it is limited to the Microsoft operating systems family, completely ignoring the fact that the bulk of the public Web servers in the world are Apache, with the bulk of them installed on Linux or a Unix derivative. The Microsoft Silverlight environment, while not as encompassing as AIR, has potential in how you can manipulate Web media and is capable of handling the complexities of digital rights management for video-on-demand systems such as NetFlix.

A Brief History of High-Level Languages

To understand better what these systems are providing and why their popularity has exploded, we need to go back in history a bit to some of the very first programming languages. The original assembler for the IBM 360/30 was very straightforward but tedious, requiring an intimate knowledge of the computing hardware and how instructions worked on data. Writing utilities in Assembler provided access to the most primitive instructions and capabilities of the tape and disk systems; at the same time, programmers had the ability to display messages and accept input from the console. However, every single piece of code had to be in each punched card deck, and code reuse meant literally lifting sections of cards out of one deck and placing them into another.

In the late 1970s, IBM released a new version of BAL (Basic Assembly Language), a macro assembler that allowed programmers to take advantage of a library of prewritten snippets of code (macros) to do certain repetitive tasks. Examples of the repetitive tasks that could be automated included rewinding a 9-track tape reel, skipping to the third dataset on

the tape, etc. The ability to use tested and standardized code in programs immediately freed programmers from huge amounts of writing and debugging. Even though a program lost portability (unless the other system had the same Macro Assembler and version), what was gained was the ability to develop much more complex programs in a fraction of the time required by the earlier assembler.

It also meant that others on a programming team didn't have to adapt their code to reuse these functions. That alone had the effect of moving teams toward much more standardized coding. The macro assembler gained widespread use just in time to prepare programmers for a new programming language called COBOL (Common Business-Oriented Language), which in one "print format" line of code did what used to take hundreds of lines of assembler and days of debugging. At this point, you'll notice that we haven't discussed hardware abstraction. When everyone was using systems from a single large vendor, hardware abstraction was much less an issue, surfacing primarily when shifting code from one storage subsystem to another.

One of the authors (Brian Chee) heard a lecture in the mid-1970s by U.S. Navy Commander Grace Hopper (author of COBOL), in which she talked about things like cost analysis and how COBOL could potentially save the Navy hundreds of thousands of dollars in data processing costs and make data processing available to dramatically more people due to the reduced costs. Little did she know just how a big a leap we would make before she finally retired from the Navy as Rear Admiral Grace Hopper. As the computing world developed, new programming languages sprang up: FORTRAN (formula translator) for scientists, and a veritable Tower of Babel (BASIC, PL1, LISP, APL, SNOBOL and C were just a few). Each language reached out to a larger and larger and more specific audience by making it easier and easier to create more and more complex programs with less effort. SPSS, for instance, stands for "Statistics Package for the Social Sciences" and was initially targeted at the need for certain types of statistics in the social sciences; it eventually grew into one of the decade's most popular statistics system on mainframes and minicomputers, eventually reaching the PC. With IBM's recent purchase of SPSS, we predict that the SPSS suite of statistics modules will find their way into cloud modules in coming years.

SPSS had a huge impact on the scientific community because it was a very early example of a system that was nearly completely divorced

from traditional programming languages. Instead of needing to write an extremely complex program to do standard deviation calculations, now all you had to do was feed it data and ask for it in a single instruction line. Nearly a 3000:1 reduction in coding effort was commonplace, and it allowed nonprogrammers to do complex statistical analysis without needing a degree in computer science. Philosophically, this was an important predecessor to the later cloud applications, because it allowed users to develop complex data analysis routines without having to learn the intricacies of a "real" programming language. At a certain level, this extended the concept of "abstraction layer" up the stack, to the point of allowing abstraction for the user, rather than simply for the computing system.

Database Abstraction and Putting the Database on the Web

Another huge step along the way to Cloud City happened quietly in the mid-1990s at a small Honolulu computer distributorship called Aspect Computing, where James Laurel and Richard Chan faced a dilemma: They wanted to have a home life, but their livelihood was linked to computer retailers that often needed to obtain information on equipment stock at some very odd hours. What they really wanted was a way to leverage this new thing called the World Wide Web so that these computer dealers could query the Aspect Computing inventory system even when the shop was closed. Jim also wanted to create an abstraction layer that would allow the system to have security but would be flexible enough that the system could be reused for new applications as yet unimagined.

The product they developed, WebDB, eventually became a commercial product that for the first time allowed Web hosts to provide a peek into databases from a Web client. To put this achievement into proper perspective, keep in mind that at this time it was very rare for any database application to be able to handle queries over a network, and those that could required that the network link appear as a mounted disk drive (i.e., through something like Novell Netware, the disk had to appear as a drive letter.) Networked database applications of this time all had to have access to files stored on this drive mount. No peer-to-peer or client-to-server database apps were available outside of development labs. Apparently, this

technology was enough of a paradigm shift that Microsoft Corporation bought Aspect Computing in 1996.

Moving beyond drive letters was a critical piece of the overall puzzle in allowing virtual and cloud applications to be created. In a way, this was yet another abstraction layer, in that network database access became part of the "plumbing" of the Internet, allowing applications to perform queries and correlations across dozens or even hundreds of databases—far more than could be accommodated with the old "drive-mapped" methods—and leading eventually to true client server computing.

A standard way of programming database access was another vital link, as abstraction layers such as ODBC (Open DataBase Connectivity) became standardized across multiple operating systems. Another effort to develop the standard access method was begun in 1993, early in the Web's history, on a mailing list used to discuss Web programming projects. The result was the Common Gateway Interface (CGI), which became the basis for a great number of other programming efforts designed to create Web applications. It laid out a methodology to link programs outside the normal purview of the Web server, so that complex application results could be linked to the Web. This sideways step also allowed developers to extend the capabilities of Web applications in previously unforeseen ways.

Different Clouds for Different Applications

The definition that we're working with is that clouds are abstraction layers, hiding system details from the end users. The evolving goal that most cloud providers seem to be heading toward is an OS agnostic system, where users can choose applications from a vast library. A key element to be resolved is billing for these library items, but it's not hard to imagine a scenario in which billing would be handled like a mobile phone account, but with items like:

- Application use charges (since users don't buy apps anymore; perhaps this is what IBM had in mind when it bought SPSS?)
- Temporary and long-term storage charges
- Throughput (input/output) charges
- CPU or compute time
- Idle time (application in memory but suspended)

We all have to keep in mind that most users really don't care about where or even how their processing is done, just that they can do their task with the least amount of hassle. We've been around for a while and witnessed first-hand the transition from Hollerith cards to terminals, PCs, and now the Internet. Other than initial user resistance, each major technology change has swept through the business landscape and then has become part of the environment.

The direction of the evolution is all about computing turning from a world of customized solutions to the ubiquitous environment of a utility. The U.S. Department of Defense mandated that DoD systems move away from one-of-a-kind "stovepipe" systems to commercial off-the-shelf systems that can take advantage of the economies of scale. A new Navy submarine launched in 2009 is a good example of a "boat" that uses many commercial off-the-shelf components rather than custom-built systems (unlike the legendary $1000 toilet seat in the Air Force B1 bomber).

Processing Clouds

Jim Staten of Forrester Research provided an example of how the *New York Times* leverages the cloud. The *Times* wanted to makes its historic archives available for online access. They needed to process 11 million articles and turn them into .pdf files. Initial estimates outlined that hundreds of servers and about 4 Tb of storage would be necessary. The IT organization at the *Times* estimated a months-long delay before beginning, the need for a significant budget and highlighted the difficulty of locating the computing resources. The project manager gave Amazon Web Services a try and kicked off 100 EC2 instances and 4 terabytes of S3 storage. The job was finished the next day with a total cost of $240.

Another hard example comes from the *Washington Post.* Peter Harkins, a Senior Engineer at the *Washington Post,* used the Amazon Elastic Compute Cloud (Amazon EC2) to launch 200 server instances to process 17,481 pages of non-searchable PDF images into a searchable online library. With a processing speed of approximately 60 seconds per page, job was completed within nine hours and provided web portal access to the public 26 hours later. Harkins ruminates, "EC2 made it possible for this project to happen at the speed of breaking news. I used 1,407 hours of

> virtual machine time for a final expense of $144.62. The database of Hillary Clinton's 1993-2001 Schedule is publicly available at: http://projects.washingtonpost.com/2008/clinton-schedule/.
>
> Examples like this show how cloud computing techniques can be used to revolutionize PED processes. By increasing the use of automation and focusing our analyst on higher level exploitation tasks, near-real time exploitation and dissemination of critical intelligence products may be enabled in the very near term with cloud computing.
>
> (*Source:* http://kevinljackson.blogspot.com/2008/10/why-cloud-processing-exploitation-and.html.)

Amazon Web services (http://aws.amazon.com/what-is-aws) is just the tip of the proverbial iceberg when it comes to a cloud specifically purposed to bring on-demand computing cycles to organizations and users who need them. The trend we're starting to see is for companies to use cloud computing and storage to smooth out usage spikes and avoid upgrading data centers to size capacity for spikes rather than "normal" usage.

In the above-mentioned examples, those EC2 applications were still virtual machines that forced you to choose one operating system over another during the EC2 configuration phase. Some of the conditions for being a true cloud implementation were fulfilled, but not all: There was no sense of a seamless movement of processing from one platform to another, and no escape from a deep awareness of where the processing platform was located (in a virtual, if not physical, sense.)

To give a flavor of the variety of operating systems offered by Amazon and just how fast this list is growing; we thought we'd take a snapshot of what's offered, but also list where you can find the current listing.

> Amazon Machine Images (AMIs) are preconfigured with an ever-growing list of operating systems. We work with our partners and community to provide you with the most choice possible. You are also empowered to use our bundling tools to upload your own operating systems. The operating systems currently available to use with your Amazon EC2 instances include:
>
> Operating Systems
>
> Red Hat Enterprise Linux Windows Server 2003 Oracle

Windows Server 2008	Enterprise Linux
OpenSolaris	CentOS Linux
openSUSE Linux	Ubuntu Linux
Fedora Linux	Gentoo Linux
Debian Linux	

(*Source:* http://aws.amazon.com/ec2/#instance.)

Note: While it is rumored that the Apple Mac OSx will run under VMWare, there is still considerable debate whether such an action would put you in violation of the end-user license agreement.

A great amount of work still needs to be done on a job description language of some sort before cloud computing reaches the sort of state that the Web began to enter with the development of the Common Gateway Interface (CGI) in 1993. Reuven Cohen, co-founder and CTO of Enomily, Inc., is one of the people looking at the question of how to develop standards for cloud computing. He has approached everything from cloud resource description to cloud identity federation in his blog, "Elastic Vapor" (www.elasticvapor.com/2008/08/standardized-cloud.html).

We've said before that it's all about abstraction layers and whether can you see through the floor into the inner workings of the environment. Offerings are popping up everywhere in all the shades of gray. Some, such as Amazon's EC2, are close to the foundation hardware; some, such as AppNexus, only partially obscure the foundation; and a few, such as Google, fully obscure the foundation. What we really have today is a market in transition, with vendors feeling around in a speculatory arena trying to figure out what consumers really want.

Storage Clouds

Data storage space in any organization is like physical space in that nature abhors a vacuum. Anytime we've been involved with adding data storage space to an organization, we've been amazed at just how quickly it disappears. So therein lies the rub: Where do you find enough temporary storage to do huge projects? In the case of the *New York Times* PDF indexing project, they estimated that they needed 4 terabytes of storage; so, instead of trying to temporarily expand their data center, they turned to Amazon Web Services. This temporary boost in storage capacity is one of

the leading applications for storage clouds for enterprise use. For personal use, cloud storage for off-site backup and remote access to critical files have led to acceptance of the idea of cloud-based storage.

For many individual users, the first experience with a storage cloud will come through an encounter with one of the remote storage or backup clouds. Commonly used storage clouds include Boxee, DropBox, Microsoft's Mesh, Apple's MobileMe, and Amazon S3. A frequent encounter with these might include using Amazon's S3 (Simple Storage Service) to back up traveling laptops.

While there is a single Amazon S3 service, and a single programmatic interface to the service, to say that there is a bit of variety in S3 backup tools is an understatement. With names like Jungle Disk, S3 Backup, Brackup, Duplicity, S3Sync, and others, Amazon S3-based backup tools are available for just about every desktop operating system available today.

However, backup is just scratching the surface:

> Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.
>
> (*Source:* http://aws.amazon.com/s3/#functionality)

It's all about developer support, and Amazon has poured a huge amount of money into creating a collection of developer support tools that we've not seen since the days of the IBM programmers' library collection. With examples, docs, best-practice guides, a knowledge base, and tools all freely downloadable, Amazon seems determined to make friends with the developer community instead of taxing it with fees as other systems do.

Another good move by Amazon has been its eclectic approach to programming library support. Instead of just going for the Microsoft "low-hanging fruit" and sticking with C# and the .NET environment, the Amazon SDK (Software Development Kit) collection is a smorgasbørd of languages and developer systems. Interestingly enough, Amazon also provides support for the OpenMPI interface in batch processing mode

to attract Beowulf users. More information on Amazon's system and its programming can be found at http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=47.

On the flip side of this coin is Google's Web-minded approach. With a much more simplistic approach, Google has neatly sidestepped the huge support requirements that Amazon had to build. By concentrating primarily on Python for the development system, Google's approach gives unparalleled integration into the world of Google Services while also leveraging the huge number of Python programmers in the world. Instead of offering everything under the sun, as Amazon does, Google has been building its library of apps over the years as part of an all-encompassing Google environment. Instead of providing a simple storage facility, Google is concentrating on providing storage through the apps in the system. The world of Google is already tied together, already tightly integrated, and already well understood. Google seems to be saying to the market that not only have we built it, we're also making it inexpensive to play by pushing you into a single development environment while at the same time opening the entire Google world to you. What we're expecting to see is a collection of personal productivity tools to round out the office automation applications that are already part of the Google desktop.

In an effort to make their cloud solutions ubiquitous, all the major players are making inroads into blurring the line between mobile and desktop. In the past it was clearly computing power that separated the CPU-light mobile world from the bigger, faster computing capabilities of the desktop. Clouds place additional computing capability anywhere, allowing for CPU-hungry apps to run even on CPU-light mobile platforms by separating computing from the user interface. This "client-server" model has been used for years for network applications, where user interfaces on client machines communicate back to larger applications running on back-end servers to handle the heavy lifting. The key to this approach will be how fast and how far 3G and then 4G wireless networks provide Internet connectivity so that these new mobile platforms can keep the mobile platform connected to the back-end cloud computing environment.

An approach similar to Google's foray into the mobile world with Android has been used by Microsoft and Apple with their My.Phone and MobileMe services for mobile devices. The key differences are that the offerings from Microsoft and Apple are much more tightly tied to the operating system, making only minimal user interaction required

after the initial relationships between desktop and cloud-based files are established. The downside of this tighter integration is that cross-platform performance is either not available or available only on a minimally functional basis. One of the great unknowns about Google's foray into a cloud-based operating system for mobile platforms is whether it will lead ultimately to wider availability on a variety of platforms or to tighter integration with (and therefore more exclusive ties to) Google's own products and services.

Users can hope that, as the market develops and more open definitions of cloud processes and procedures gain acceptance, it will be easier to find application, processor, and storage cloud services that are tightly integrated into operating environments and available on a greater number of platforms. There have been promising signs of this direction, but the market is, as of this writing, still too immature for users to know for sure which direction will predominate.

With a market in all the shades of gray, only time will tell which approach best represents the consumer.

Email Protection Clouds

It's funny how sometimes things happen so slowly over time that they slip by your notice. The world of anti-spam has become so cumbersome that almost no one handles his or her own "black list" maintenance anymore. Even if you're using a small firewall that has a check mark for anti-spam, you're almost certainly already using a cloud service. The number and variety of blacklisting services in the anti-spam world is varied, but the most successful anti-spam systems seem to use a combination of several blacklisting services and in some cases multiple technologies that filter for spam, fraud, phishing, and other email-based malware. We saw a product from CheckPoint around 2004 that provided this type of service in a cloudlike arrangement, but it wasn't until late 2009 that it reappeared in the firewall product line from vendors such as Cisco.

Strategies for Getting People into Clouds

In reality, many of your people are already using applications that have leanings into the cloud; perhaps a few well-placed memos and services

could get your staff thinking about clouds and their potential benefits to the enterprise.

Let's start with a little of what's happening under the hood in the Apple MobileMe and the Microsoft My Phone services. The big selling point of both is the constant and convenient backup of your smart phone. With many people having upwards of 1000 contacts in their address book, the loss of the use of a smart phone could be devastating. As illustrated by an application from PocketMac Corporation, MobileMe can also be used as a DMZ for programs to transfer data back and forth in a secure way. In this example the PocketMac folks rely on the BlackBerry or Nokia phone to synchronize with MacMail and then upload to MobileMe. Now, with a database storing the address book information, they can harvest that data to synchronize with applications such as Sales Force, Meeting Maker, Lotus Notes, Entourage 2004/2008, and others. All in all, an interesting way to solve an address book synchronization problem, with the additional benefit of forcing the backup of the mobile device to a cloud storage service.

So, while backup of the Windows Mobile device is the primary selling point of Microsoft's My Phone service, this cloud storage solution will also more than likely morph into a similar service, especially considering how Microsoft has already added in several connectors to social networking services such as Facebook, Flickr, and MySpace. This type of service also gives us a hint as to how various platforms will leverage each other. In the case of the My Phone service, it's considerably easier to use a full keyboard to modify address book entries, or groom a music or picture collection. Use the cloud to do large modifications, while the mobile platform becomes the ubiquitous extension into the cloud. Not to mention it's a pretty handy way of moving that huge address book to your new phone.

We previously mentioned that the Amazon S3 service had a stealthy beginning, since some of the very first apps for it were automated backup systems for road warriors. Jungle Disk, Brackup, and Duplicity are a few that stand out, but backing up to the cloud has become a necessary task now being offered by ISPs all over the United States. The result is that being able to back up regardless of your location (as long as you have an Internet connection) has removed some of the pain of the task and seems to be getting more and more users to actually back up their systems. It's no wonder that traditional backup applications such as those from Paragon Software have shifted direction to embrace the cloud.

We've already mentioned SalesForce a couple of times, and while these folks certainly started in the customer relations management (CRM) game, they have recently tossed their hat into the world of clouds. So instead of just providing CRM, SalesForce is now providing the ability to host custom applications for its customers. The same applications can now take advantage of the direct (and secure) connections into their legacy CRM data store already in place.

Throwaway Clouds

Another strategy that was used by the people at the *New York Times* is to leverage the cloud for short-term or one-time projects. A good analogy is renting a car rather than buying one if you're only going to need it for a couple of weeks. Clouds can be very similar to a rental car agency, in that you can rent cloud service for a short period for specific projects. You could also use it to do a longer "test drive" of a model you're interested in. The analogy also works for variable-duration rentals in that longer duration normally means a lower cost per day. For instance, you negotiate to take a portion of SAP out for a test drive and you drop it into the cloud for the 90-day test drive. No fussing around losing several days while IT spins up a test machine for you, and if you don't like it, just let the cloud vendor blow it away when you're done. If you already have other modules in the cloud, connections become quite a bit easier, even on a temporary basis.

So why not take this concept a whole lot further? The VMWare folks have a repository on their site that has a truly staggering number of VMware appliances available for you to test drive. Think of a new-car lot open 24/7 with thousands of different models ready for you to take home and try out for a period. The big selling point is that you don't have to struggle setting up the environment just to find yourself with only a couple days left in the trial period. It's all ready to go: Just drop it into a cloud or a VMware system and turn it on. Everything is preconfigured and ready for you to explore the appliance.

Traveling Clouds

A fabulous example of "traveling clouds" arose when the Microsoft Unified Communications folks came over to show off their latest wares for

the InfoWorld editors during the summer of 2007. Considering that the entire constellation of servers for this demo required five Windows servers, with one requiring a 64-bit OS, this was a pretty tall order to spin up on short notice. In this case the product manager hopped on a plane with a big USB hard drive and quickly spun up a preconfigured Microsoft UC constellation consisting of:

- Active Directory server with a certificate authority setup
- SQL server for storage
- Exchange Server for email
- Share Point Server
- File Server
- SIP Gateway (an appliance, so not a VM in this case)

Since the whole smash was set up to talk over the virtual network (i.e., isolated), we really only had to change a single IP address on the Exchange server for external connectivity. So what would have taken quite a few days to set up before we could even see the functionality instead became an afternoon install and a full demo the next day. It was especially useful when the Interop iLabs folks were able to use the "tweaked" virtual machines for a live demonstration at the Interop Las Vegas trade show. Keep firmly in mind that this trick only works if the external USB disk is formatted NTFS to get past the 4-gigabyte file-size limitations that come with the default FAT formatting typical of these drives. (Since many virtual machines are several gigabytes in size, Amazon's EC2 system [and other cloud vendors] allow for shipping of large USB drives to them for local mounting over their internal networks. This local mounting tends to have special pricing, making the setting up of custom virtual machines much more palatable.)

Occasional-Use Clouds

Virtual machine images also become a way to handle special projects that only see the light of day a couple of times a year. In the case of the InteropNET, those virtual machines are spun up twice a year (once for Las Vegas and another for New York), saving a massive amount of time during hot-stage setup that used to be taken up doing a fresh sysgen for each show. In this case the InteropNET team were also able to synchronize

versions with the Global Data Vault's Cloud Hosting Service so that we could swap our VMs onto our blade servers during the show, while maintaining access between shows for data mining.

You also need to keep firmly in mind that you can download a free VMWare conversion tool that will allow you to prototype on a workstation version and then migrate to a full production system when appropriate. We regularly see engineers prototype servers under VMWare Fusion (Mac workstation), convert, and then SFTP up to the VMWare ESX server in the lab. You also need to be sure to spool off the images onto a disk first, to avoid the "oops" factor. And remember that on a Windows machine, the external disk needs to be formatted NTFS and on the Mac "MacOS extended file system" if you want to get those huge virtual disk files onto the external drive. Sorry, but FAT/DOS isn't going to cut it for those huge files.

We expect similar virtual appliance collections to start appearing as Microsoft kick-starts its Hyper-V community efforts. With the Advanced Network Computing Laboratory being InfoWorld's biggest testing facility, they're now spinning up both a VMware and Windows Hyper-V mini-cloud on a set of blade servers so that editors can drop in the VM of their choice for review infrastructure.

When you start talking about cloud storage heading out to the very edge, nothing gets closer than the tiny device called a PogoPlug. The University of Hawaii research community has been playing with the PogoPlug now for a while, and being able to mount a fairly eclectic collection of USB drives onto a NAS-like device without worries about format has been, to say the least, liberating. While traveling, one researcher had a 1.5-TB Lacie Mac OSextended drive, a Seagate 250-GB NTFS drive, and a couple of DOS thumb drives all mounted and available across the WAN with no firewall rules necessary. Since the entire authentication process is done in the cloud, the PogoPlug doesn't need that much CPU. Once the PogoPlug data center has finished providing users with a "dating service"-like approach, it gets out of the way, letting the conversations take place on a peer-to-peer basis. Yet all of this is still secure, because of the rigorous authentication over SSL that PogoPlug requires. Key to the success of this tiny device is how the creators have turned the network attached storage model on its head. Instead of expecting all the network conversations to start from the outside world and head inward, the PogoPlug keeps a heartbeat-style conversation going with the PogoPlug data center.

Authenticated users then ride back on the already-existing conversation. Since the conversation started from the inside going out, normal firewall rules don't apply, because of the assumption that conversations going outward are trusted. This device could be viewed in the same way that Skype has become an unwanted bug for IT. It's hard to control in that it starts as an outbound service, "tricking" the corporate firewall into trusting it. However, it should also be viewed as a superfast and easy way to replace the need for a departmental file server just to provide remote file access. It could also be used as a quick-and-dirty traveling project team server that would work even on some of those funky hotel networks. It's all about how you spin it, and knowing about it so you can work it to your benefit instead of letting it creep up on you.

Company in a Box

Some InfoWorld editors been toying with the concept of "a company in a box" ever since one of them mentioned a project that did some quick deploy networks for the Marine Expeditionary Force out of the back of a Humvee. Could this concept be used in the civilian world, and are there enough resources now that we can quickly spin up a company in a warehouse (or a tent) after a disaster? The gist is that there is a concept in the military called "shoot and scoot," where an entire artillery battalion regularly practice picking up and moving their mobile headquarters in a matter of minutes rather than hours. Quick-disconnect network trunks, gear in travel cases, and lots of documentation to handle the setup and tear-down all make for a system designed to move. This ability is not for everyone (not to mention that it can get pretty expensive), but it wouldn't hurt to consider at least some of the better ideas as part of your business continuity planning.

The answer to trying out this project has been a resounding yes they could, and yes they will. With the huge number of virtual appliances available now, we could easily see combining off-the-shelf VMs with a few roll-your-own VMs to bring us back from disaster quickly. Virtualization can be a massive boon to business continuity, far beyond the old concepts of hot sites, warm sites, and cold sites. The issue is that readiness costs lots of money, and the "hotter" the site ("hot site" means drive across town and you're running, warm means a bit of synchronization from storage,

and cold means a full restore), the more it costs to keep everything running and up to date. Some banks have gone as far as completely duplicating their data processing facilities somewhere else, right down to empty cubicles, file cabinets, and office support equipment—simply a breath-taking cost item for business continuity insurance. What clouds provide is a middle ground, where someone else keeps all your virtual machines warm *and* duplicated in multiple locations, all without the massive expense of a physically duplicated data center. You might not even need the computing side of the cloud during normal operations; just use the storage side to keep the VM images in sync. Then, if disaster does strike, spinning up those sync'd images is as simple as flipping a switch.

Some of the key factors to consider include:

- How much of your operation can live fully in the cloud, and how much has to physically be on premise?
- Set up preprovisioning agreements with SIP trunking vendors to swing your incoming lines over to the new trunk. Since quite a few companies are moving to SIP trunking anyway, this could be as simple as making sure the right people have the authority to do the move.
- Set up key services either under a VM now or perhaps use something like the Paragon Software system to periodically spin off a virtual disk image as a "warm" image. Then it will be just a matter of laying in the incremental data restores over the latest "warm" image. If you use something like Global Data Vault, then it's a matter of sync'ing the image from their data store.
- Use something like Asterisk or TrixBox to duplicate as much of your dialplan on your PBX as is reasonable. Since it's all network-based, creating and testing your portable PBX isn't a huge resource hog.
- Since most cellular/3G/4G providers are located pretty high in buildings, using them for your WAN connectivity isn't that big a stretch, especially considering how many have generator capability now.
- Confirm that your backup locations have enough power and have enough reception for your cellular/3G/4G WAN connection.
- Put your bare-minimum system into a surplus road rack (aka Hardig or Anvil road case). Using something like an HP "Shortie" blade server, which provides both storage and computing capability, could go a long way toward bringing up essential services quickly.

- The School of Ocean and Earth Sciences and Technology (SOEST) at the University of Hawaii is an old hand at putting complete science labs and computing facilities into shipping containers. They're looking at using the portable NOC product from American Power Conversion that's pre-set up to your specifications. Power generation, UPS, cooling, control are all preconfigured in a rolling data center.

What we're really getting at is that virtualization and clouds pay dividends on many levels. What hasn't occurred to people is that having a portable computing facility can also pay huge dividends in terms of business continuity during disasters. It also means that moving your company for other reasons become a whole bunch cheaper too. The point we're making is that clouds free you of the data center anchor, giving your organization a level of portability never achieved in the past. If you're already in the cloud, then you only have to move the stuff that isn't already cloudy. We think this sounds like a good idea even if your apps are left in-house, just to remove hardware dependencies and provide for portability. Just in case.

Clouds Flight Path for Chapter 4

- *Development languages and environments keep changing to take advantage of new layers of abstraction.* We're moving toward finding programming tools that are appealing to a wider and wider audience. Each new programmatic abstraction layer means that business can home in on key topics faster and with less costly human resources.
- *Software development kits (SDKs) and applications programming interfaces (APIs) are really just the way we plug together various applications.* SDKs and APIs are the foundation stones for some amazing programs today. Imagine having a programming language that truly allows you to concentrate on the business task rather than the tedium of the language. Emerging systems are making it even easier to link rich internet applications to back-end cloud applications that are increasingly platform-independent, while giving Web-connected users capabilities previously found only on hugely powerful desktop workstations.
- *Thank you Admiral Hopper, who led the way to high-level languages that make clouds possible.* Admiral Hopper (who also gave us the term

bug) was truly a visionary technologist, whose COBOL was the first abstraction layer of the new rich internet application platforms. The future is extremely bright for amazingly feature-full applications.

- *Database abstraction methods and how a Hawaii company led the way.* Some folks truly got rich as the world started building abstraction layers, and this tiny Honolulu company was one of the leaders. Like other abstraction systems, the modern database management systems have evolved into some incredibly complex systems that remove a huge amount of care and feeding complexity for your precious data. With new database-light systems, even simple cloud applications will be able to take advantage of the speed and reliability of modern database systems.
- *Using storage clouds only for backup just scratches the surface.* At every turn, the cloud industry is finding new ways to utilize cloud storage. We took a look at a few and tried to imagine how cloud storage can continue to revolutionize business computing. It's already evident that Amazon is using its storage cloud as the glue at the center of its constellation of services. While Microsoft looks like it's playing catch-up, we wonder whether it may just leap-frog the competition.
- *Is Google jumping ahead toward true cloud computing by moving us farther away from the hardware?* We got pretty far out in our view of where cloud computing can go, and Google's view seems to match our view pretty well. Now the question is how far Google will take this, and whether the market actually wants it. We figure they must be on to something, with Microsoft's cloud offerings feeling very familiar and the amazing amount of buzz about cloud-enabled apps on the Android mobile phones emerging on the market.