Hype oils the wheels of IT, and cloud computing is currently among its most-hyped topics. Consultants, industry reports, business magazines, blogs, and books are trying to tell developers what cloud computing is. At the same time, cloud-infrastructure providers are eager to extol the virtues of a new computing and programming model and the benefits you could accrue by getting on board.

In The Cloudspotter’s Guide (Sceptre, 2006), author Gavin Pretor-Pinney writes:

Leonardo da Vinci once described clouds as ‘bodies without surface’, and you can see what he meant. They are ghostlike, ephemeral, nebulous: you can see their shapes, yet it’s hard to say where their forms begin and end.

The same nebulous quality dogs discussions about computing clouds. It makes sense to clarify the basics.

**What Cloud?**

The “cloud” in cloud computing originated in network diagrams, where it indicated the boundary of network interconnections. The cloud-computing paradigm is characterized by

- **transactional resource acquisition.** Upon request, users receive immediate access to computational and storage resources according to an agreement with a service provider (involving payment, if the provider is a commercial entity). This contrasts with resource-sharing schemes, such as grid computing, in which users submit a computation job that goes into a queue until the required resources become available.

- **nonfederated resource provisioning.** Even if available resources are physically distributed, they’re offered by a single provider. They’re not the sum total of a federation of independent providers—again, in contrast to grid computing.

- **a metered resource.** The provider meters resource usage—whether computational, storage, or network—and bills users, if it’s a commercial entity, or manages fair use and sharing, if it’s a public operation.

Transactional resource acquisition gives users an intuitive interface to a one-stop service shop. Nonfederated provisioning frees providers from...
negotiations and service amalgamation with different resource providers.

These characteristics are requirements for all cloud infrastructures. However, as is often the case with requirements, there’s more than one way to meet them.

For example, if we’re interested in who operates the cloud infrastructure, the options are

- **private clouds.** An organization installs its own farm of computers and disks and uses cloud technologies to provision resources. In effect, the organization centralizes all IT cloud operations at a single point.
- **public clouds.** A special cloud-infrastructure provider offers computing services to public users.
- **hybrid clouds.** An organization operating private clouds might offload part of its workload to a public cloud—for example, when the workload exceeds private capacities. This is called cloud bursting, and the combined infrastructure is a hybrid cloud.

If we’re interested in what kind of stuff we can deploy on a cloud infrastructure, cloud providers offer three service levels:

- **Software as a Service (SaaS).** Users access software applications, typically paying per use. SaaS providers include Salesforce.com, Google Apps, Microsoft Cloud Services, and Rackspace.
- **Platform as a Service (PaaS).** Software developers access a development platform on which to write and deploy their own applications. The platform typically comprises APIs for one or more supported languages. Example platforms include Google App Engine, Microsoft Azure, and Force.com.
- **Infrastructure as a Service (IaaS).** Software developers can access bare infrastructure for computing, storage, and networking. They carve out a small data center from the provider’s resources, using it as their own and deploy applications in it. The resources are usually, but not necessarily, virtualized. Amazon EC2 RightScale is an example service; private clouds built with virtual machine (VM) management software are another.

Figure 1 shows these service options and some example products, tiered according to the distance from the physical infrastructure and the abstraction level provided to the user.

**Choosing the Right Cloud and Service Model**

The choices on how to write an application for the cloud (or how to port an existing one to the cloud) mirror the choices among the different kinds of cloud types and cloud deployment options.

**Private Clouds**

If you build a private cloud, you’re free to build it any way you want. You have to buy and install the infrastructure, of course, but then you can use it as a farm for VMs, implementing the IaaS model, or you might choose a platform that’s suitable for implementing the PaaS model. Finally, you might build applications for other people to access and use over your private cloud, implementing the SaaS model.

Building a private cloud is not a minor matter, however. First, it requires a suitable physical infrastructure, including computers (the so-called compute fabric), storage (the storage fabric), and the interconnect fabric to bring them together. Typically, the storage connections to the computers must be efficient to avoid I/O bottlenecks—for example, fat-tree network topologies using Fibre Channel or Infiniband.

Second, a private cloud must have a suitable software infrastructure for provisioning resources to its users. Such an infrastructure is more than a portal for dispensing VMs. For instance, the software infrastructure must let users request a number of VMs and some amount of storage, specify the VM contents and upload any required software, and manage the VMs (start up, monitor, pause, shut down, and so on). Users must be able to transfer data to the cloud and download processed data from it. Other functions might include moving VMs around the physical infrastructure to respond to load patterns.

Several cloud infrastructure software solutions are available. Eucalyptus (http://open.eucalyptus.com) is probably the best known because it focuses on compatibility with Amazon Web Services (AWS) and offers the same commands and APIs. Other solutions include Enomaly (www.enomaly.com) and OpenNebula (www.opennebula.org).

**Public and Hybrid Clouds**

If you choose a hybrid approach, your choices are limited by the options available from the external cloud provider that you’re cooperating with. The limitations include the way cloud-to-cloud interoperability will actually take place. In general, you need to consult with the cloud provider.

In both public and hybrid clouds, you need to consider the provider’s quality-of-service commitments and service-level agreements (SLAs). If your application
must be available 99.99 percent of the time, your SLA with the provider should cover that. Cloud providers usually have their SLAs available on their Web sites, so it’s easy to check what they offer and, more importantly, what they leave out.

SaaS
When a cloud provider offers a ready-made SaaS application, developers usually don’t have many options to customize it. However, some applications offer limited options, and some require tailoring. You won’t be developing a new application but customizing the SaaS application to your company’s needs. Such customizations aren’t fundamentally different from those that consultants perform when a company is introducing new corporate software, such as Enterprise Resource Planning suites. These customizations are often carried out in cooperation with the cloud provider (or even by the cloud provider).

The situation changes when the application offered by SaaS is a software development suite, as with Force.com. Such SaaS environments let users develop fully fledged applications, leveraging the provider’s software stack. Hence, Force.com users will be using the software infrastructure that powers Salesforce.com applications (see the sidebar “Build an Application Using Force.com”).

IaaS
The IaaS model offers more flexibility. As long as you can pack the application in a VM supported by the cloud provider and include all the required libraries and dependencies with it, you can do pretty much whatever you want. The main restrictions will be on the VM images supported. For example, if the application requires MS-Windows, the provider must offer MS-Windows VM images. Such restrictions aren’t fundamentally different from what system administrators place on developers in corporate environments.

The details of porting an application to an IaaS infrastructure depend on the infrastructure. Different infrastructures provide different abstractions and mechanisms for describing and accessing VMs and storage. At this point, compatibility between cloud providers is nonexistent, although the industry has realized that lock-in is a major concern for prospective users.

Even though vendor details differ, the IaaS model itself brings some common issues that developers should be careful about.

Support for parallelism. An ever-larger share of applications are exploiting parallelism. Even a typical desktop or laptop machine can run more than a couple of threads concurrently, so developers should make sure that a cloud provider supports an application’s concurrency model. If you’re using Message Passing Interface (MPI), the cloud provider needs to support it; if you’re using OpenMP, the required libraries must run on the target cloud platform, and so on. Of particular interest are data-intensive applications that require parallel I/O. For instance, is MPI-2 I/O offered?

Running on the cloud isn’t the same as running efficiently on the cloud. Even more than in desktop applications, developers using IaaS should verify that the performance gained from parallelism is what they expected.

I/O details. In IaaS, data collocated with the VM isn’t persistent. It’s lost when the VM stops and might even be lost even if the VM crashes. Developers need to store permanent data separately and bring it to the VM only when it’s required. Moreover, if your application has special I/O requirements, make sure that the IaaS provider can actually deliver the performance required.

PaaS
PaaS is probably what most developers intuitively understand by the phrase “programming for the cloud.” A PaaS infrastructure gives programmers the developer tools they need to program in that specific infrastructure. The tools are usually sets of libraries and APIs that give access to the computational and storage resources.

Programming language. Developers must use the programming language supported by the PaaS provider. Google App Engine supports Java and Python. Microsoft Azure supports any .NET-supported language via suitable plug-ins to Microsoft Visual Studio as well as other languages.
Frameworks. Support for a programming language doesn’t get you very far without support for libraries and frameworks. The Google App Engine supports the Java 6 runtime environment, so runtime environment libraries are in place as well as Java Web technology standards such as servers, java.net, and JCache. The Python interface includes the Python (version 2.5) standard library, and it’s possible to use the popular Django Web development framework in Google App Engine (although you might have to change the code).

However, support for a particular framework doesn’t necessarily mean that the framework will perform in the cloud as it does outside the cloud. The framework might operate differently in the cloud environment from the way it operates in a server environment (for which it was probably designed). Developers wishing to use a familiar framework in the cloud should experiment and benchmark it to make sure it fits their needs.

Programming abstractions. PaaS frameworks provide specific abstractions for system elements, such as data storage and access, data querying, and message queues. The concepts might be similar to what you’re used to (for instance, in PaaS, permanent data is normally stored under a term like “blob” or “bucket” rather than the usual file-system files). However, the details are different, and abstractions aren’t portable across PaaS providers.

At a higher level, PaaS frameworks can support programming paradigms that stand apart from more established approaches. MapReduce, for instance, is a programming paradigm based on well-known functional programming idioms. It can transform data by assigning computational nodes first to map input values to processed values and then to combine (reduce) the results. It can be remarkably effective, but it’s been touted as the next big thing and preferable to traditional database processing—which isn’t what it was designed to do, by the way.

Databases. PaaS providers don’t always support relational databases. The dominant model is key-value pairs, which isn’t new in itself. What is new, however, is its proposed use as the dominant model for data-based (as opposed to database) programming for the cloud, and even beyond. The cognoscenti call this model NoSQL, and it could affect the way we program in general.

The Law
Cloud computing can involve legal as well as technical considerations. Over the past several years, governments and international bodies have passed several laws to regulate the use of data by companies and organizations and to require compliance with security standards for their computer systems.

For example, in 1996, the US passed the Health Insurance Portability and Accountability Act (HIPAA), which regulates the use of protected health information (PHI)—that is, information regarding an individual’s health, healthcare provisions, costs, and so on. Organizations that handle such information must ensure, for instance, that no PHI is transmitted on open networks without encryption and that they have implemented data access and audit controls.

In 2002, the Federal Information Security Management Act (FISMA) passed, requiring US federal agencies to develop information-security programs for their...
infrastructures and to monitor compliance with standards developed by the National Institute of Standards and Technology.

In the same year, the Sarbanes-Oxley Act (SOX), passed in response to the Enron scandal, regulated financial reporting data. To ensure the data’s accuracy, enterprise systems must employ SOX-compliant IT security controls.

The Payment Card Industry Data Security Standard (PCI DSS) is a worldwide industry standard to protect cardholder information. A company hosting an application on the cloud—not the cloud provider—is responsible for ensuring PCI

Table 1: Key features of cloud provider offerings

<table>
<thead>
<tr>
<th>Feature</th>
<th>Amazon Web Services</th>
<th>Google App Engine</th>
<th>Microsoft Azure</th>
<th>Force.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing architecture</td>
<td>Elastic Compute Cloud (EC2) supports virtual-machine image uploads and gives client APIs to instantiate and manage them</td>
<td>Google’s distributed architecture</td>
<td>Hosted in Microsoft data centers, providing an OS and developer services that can be used individually or together</td>
<td>Multitenant architecture</td>
</tr>
<tr>
<td>Virtualization management</td>
<td>OS-level running on Xen hypervisor</td>
<td>Applications run on instances of Java VM or Python runtime</td>
<td>Hypervisor (based on Hyper-V)</td>
<td>Handled by Force.com infrastructure; built-in governors allocate resources among customers</td>
</tr>
<tr>
<td>Service</td>
<td>IaaS</td>
<td>PaaS</td>
<td>PaaS</td>
<td>SaaS through the Force.com development environment (can be seen as PaaS)</td>
</tr>
<tr>
<td>Load balancing</td>
<td>Elastic load balancing</td>
<td>Automatic scaling and load balancing</td>
<td>Built-in load balancing</td>
<td>Built-in among tenants</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>Placing EC2 instances in multiple “availability zones”</td>
<td>Automatically pushed to a number of fault-tolerant servers, App Engine’s Cron service</td>
<td>Application instances are replicated in independent fault domains; all data is replicated three times</td>
<td>Data storage is optimized for multi-tenancy; caches and data distribution support performance and fault tolerance</td>
</tr>
<tr>
<td>Interoperability</td>
<td>Published API for interacting with instances; hosted applications can be written in standard programming languages running in the virtual machines (VMs) and accessed through the interfaces they provide</td>
<td>Hosted applications written in Java and Python and accessed by the interfaces they provide; access to Internet resources from code and App Engine’s Fetch service</td>
<td>Hosted applications run in MS-Windows and accessed by the interfaces they provide; applications can also use a simple API to access the Azure fabric</td>
<td>Force.com Web Services API provides access to data stored in Force.com from the outside; application Web logic can be exposed as standard Web service; HTTP API is provided for REST services</td>
</tr>
<tr>
<td>Storage</td>
<td>Simple Storage Service (S3), Amazon Elastic Block Storage (EBS), Amazon SimpleDB, Amazon Relational Database Service (RDS)</td>
<td>App Engine datastore (not relational, built on Bigtable); objects with properties are stored without a schema, supporting transactions</td>
<td>Persistent data stored in non-relational blobs, tables, and queues; SQL storage offered by SQL Azure</td>
<td>Persistent data stored in objects; object instances are analogous to relational database table</td>
</tr>
<tr>
<td>Security</td>
<td>SAS 70 Type II Certification, firewall, X.509 certificate, SSL-protected API, access control list</td>
<td>SAS 70 Type II Certification, secure access to intranet via Google’s Secure Data Connector</td>
<td>SAS 70 Type II Certification, applications run on 64-bit MS-Windows Server 2008</td>
<td>SAS 70 Type II Certification, access control on data based on user identity and organizational roles</td>
</tr>
<tr>
<td>Programming framework</td>
<td>Amazon Machine Image (AMI), MapReduce</td>
<td>Java and Python, scheduled tasks and queues, access to services such as URL fetch, mail, Memcache, image manipulation</td>
<td>.NET and unmanaged code as long as it runs on MS-Windows</td>
<td>Metadata-driven development model, Visualforce framework for user interface development, Apex programming language</td>
</tr>
</tbody>
</table>
DSS compliance for any credit card information that it collects, processes, or stores on the cloud.

The US favors self-regulation in data protection. By contrast, in 1995, the EU passed the Data Protection Directive, which prohibits—among other things—transferring EU residents’ personal information outside the EU unless specific conditions are met. To streamline compliance for US companies, the US Department of Commerce consulted with the EU to develop the Safe-Harbor Privacy Principles. US companies that adopt these principles are in compliance with the Data Protection Directive.

In general, cloud infrastructure providers should show that their offerings meet the obligations set down by law. For instance, Google has applied for FISMA certification for Google Apps (http://googleenterprise.blogspot.com/2009/09/google-apps-and-government.html), while Amazon.com offers a white paper on how to create HIPAA-compliant applications in AWS (http://awsmedia.s3.amazonaws.com/AWS_HIPAA_Whitepaper_Final.pdf).

In standards such as PCI DSS, the onus for compliance is on the cloud user. In privacy matters like the EU Data Protection Directive, the cloud user must investigate exactly what kinds of data are moved where and whether the movement falls foul of the directive.

Many cloud providers achieve SAS 70 (Statement on Auditing Standards No. 70: Service Organizations) Type II certification. This assures service users that an independent third party (an auditor) has examined the provider’s organizational controls over the processing of sensitive information.

Finally, in a global economy, developers must also consider how data protection and privacy laws differ among nations and cultures. For example, Europe and the US have different legal and social histories that are reflected in different perspectives on the relative merits of policy and regulation. The US generally prefers a self-regulatory approach, where industries decide on the norms they will adopt, Europe opts for a policy-driven approach, where laws regulate data handling and privacy (see http://ec.europa.eu/justice_home/fsj/privacy/index_en.htm for more details on the European approach). EU member states pass laws on the national level to enforce the Data Protection Directive. Stories of data protection enforcement can be illuminating (see http://www.ico.gov.uk/what_we_cover/data_protection/enforcement.aspx for representative, even entertaining in a morbid way, instances from the UK).

Cloud providers roll out new offerings all the time, so it’s futile, in a sense, to attempt to give a comprehensive picture of the field. Table 1 lists four important cloud providers and some key features of their services. The providers cover the IaaS–PaaS–SaaS spectrum.

Cloud computing isn’t an entirely novel paradigm. Rather, it expresses technologies that are reaching maturity after many years of progress, aided by specific market forces. There is no single cloud and no single way either to employ cloud computing or to develop applications of it. We’ve been through many hype cycles, but cloud computing is likely here to stay. A lot of software runs on clouds now, and more will run on them in the future. Developers are already spoilt for choice between cloud implementations, deployment options, and different programming approaches.

As always, a careful look at your application’s requirements at the beginning of a project can narrow the cone of uncertainty and bring the solution in the clouds down to earth.

Panos Louridas is a consultant with the Greek Research and Technology Network and a researcher at the Athens University of Economics and Business. Contact him at louridas@grnet.gr and louridas@aueb.gr.