The People's Cloud

Peer-to-peer cloud computing could free us from the tyranny of data centers

NOT LONG AGO, any start-up hoping to create the next big thing on the Internet had to invest sizable amounts of money in computing hardware, network connectivity, real estate to house the equipment, and technical personnel to keep everything working 24/7. The inevitable delays in getting all this funded, designed, and set up could easily erase any competitive edge the company might have had at the outset. • Today, the same start-up could have its product up and running in the cloud in a matter of days, if not hours, with zero up-front investment in servers and similar gear. And the company wouldn't have to pay for any more computing oomph than it needs at any given time, because most cloud-service providers allot computing resources dynamically according to actual demand.

By **OZALP BABAOGLU** & **MORENO MARZOLLA**

Illustrations by **ROB WILSON**

Scattering the Clouds

With the right software, geographically distributed hardware can provide a unified cloud-computing resource

With the computing infrastructure out of sight and out of mind, a start-up can concentrate its attention on launching and improving its product. This lowers the barriers to entry, letting anyone with an Internet connection and a credit card tap the same world-class computing resources as a major corporation. Many of the most popular and successful Internet services today, including Netflix, Instagram, Vine, Foursquare, and Dropbox, make use of commercial clouds.

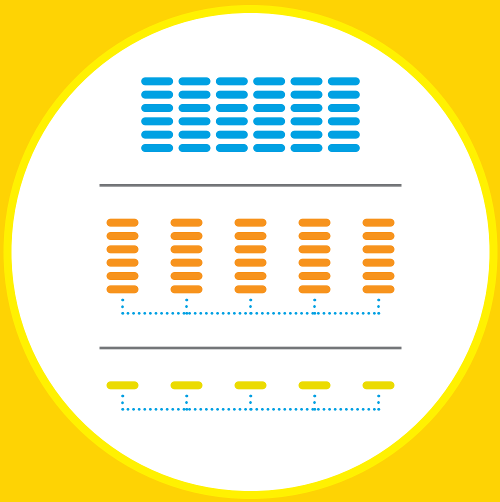
These clouds might seem a bit ethereal to their end users, but they in fact require some very down-to-earth facilities. Their stadium-size data centers are immensely costly to construct, and, not surprisingly, most are run by giant corporations like Amazon, Google, and Microsoft. Each offers a variety of service models, depending on exactly how the customer interacts with its cloud-computing environment.

The lowest-level model is known as infrastructure as a service, or IaaS, which outfits each customer with one or more virtual machines running on the cloud provider's physical equipment. One actual computer might, for example, simulate five different virtual computers, each leased to a different customer. In addition to leasing such virtual machines, an IaaS provider may include a choice of operating systems to run on them. Notable examples of such IaaS clouds include Google's Compute Engine and Amazon's Elastic Compute Cloud.

At a higher level of abstraction are platform-as-a-service, or PaaS, clouds. These include an environment for developing the online applications that are to run on the provider's equipment. Customers don't have to manage virtual machines. They just create their applications using various software libraries, application-programming interfaces, and other software tools such as databases and middleware, and then one or more virtual computers are spun up automatically as needed to run all of this. Examples of PaaS clouds are Amazon's Elastic Beanstalk, Google's AppEngine, Microsoft's Azure, and SalesForce's Force.com.

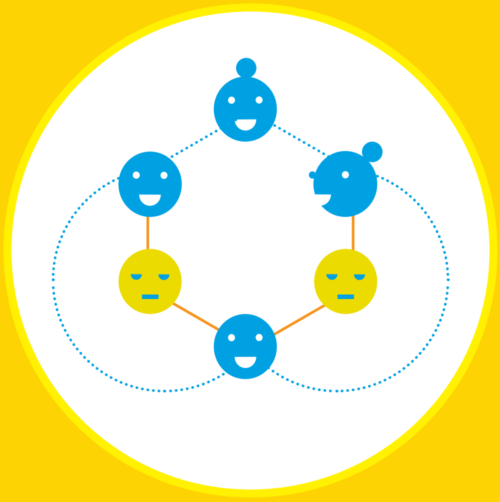
At a still-higher level are software-as-a-service, or SaaS, clouds. Their customers know nothing of the underlying infrastructure or computing platform: They simply use some Web-based application or suite of applications to handle the task at hand. This is probably the model of cloud computing that most people are familiar with. It includes services like Apple iWork, Gmail, Google Docs, and Microsoft Office 365.

But is this the only way cloud computing can work? At the University of Bologna, in Italy, we've been investigating a very different strategy to do cloud computing without those giant centralized facilities at all—using peer-to-peer technologies of the kind sometimes associated with shady file-sharing operations. Their use here, though, could help democratize cloud computing. Our prototype software is still at a very early stage, but its development and similar successes by researchers elsewhere show real promise.



SOME CLOUD-COMPUTING PROVIDERS put all their hardware eggs in one data-center basket [blue]. Others employ multiple data centers networked together [orange]. The logical extension is a peer-to-peer cloud made of individual computers [yellow].

PLACING THE PHYSICAL INFRASTRUCTURE for a cloud-computing operation where it's usually found, in a single massive data center, has definite advantages. Construction, equipment procurement, installation, and maintenance are all simplified, and economies of scale reduce costs. On the other hand, a single large data center consumes an enormous amount of electrical power, often comparable to what you'd need to



THE COMPUTERS in some P2P networks resemble gossiping office workers. Gossip-based protocols allow information to flow reliably, even if some computers leave the system and break previously established links [orange lines].

GOSSIP-BASED PROTOCOLS are used to maintain an unstructured peer-to-peer network of individual computers, some of which do the work of one customer [orange] while other combinations serve different customers [blue and yellow].

run a small town, and dissipating the waste heat it generates is usually a big headache.

Perhaps the most serious shortcoming, though, is that a centralized cloud-computing facility can end up being a single point of failure, no matter how cleverly it is designed. Redundant power supplies, backup power generators, and replicated network connections help, but they can't protect absolutely against catastrophic events such as fires, hurricanes, earthquakes, and floods.

Another drawback of centralized clouds is that their geographic location, which may be best for the owners, may not be best for the customers. This is the case, for example, when governments place restrictions on sensitive data crossing national borders. A data center located in one country may then be off-limits to customers in some other countries.

Cloud-service providers have increasingly addressed these concerns by using not just one but several far-flung data centers connected through fast private networks. Doing so not only protects against local catastrophes, it also provides customers with more options for locating their data.

What would happen if you took this trend in geographically distributing cloud infrastructure to its logical conclusion? You'd end up with clouds made up of millions of individual computers distributed across the globe and connected through the Internet. We would call this a peer-to-peer (P2P) cloud because it shares many of the characteristics of various P2P systems developed for file sharing, content distribution, and the payment networks of virtual cryptocurrency schemes such as Bitcoin.

In principle, a P2P cloud could be built using the ordinary computing, storage, and communication equipment found now in people's homes, with essentially zero initial investment. Broadband modems, routers, set-top boxes, game consoles, and laptop and desktop PCs could all contribute. The challenge is to turn this motley collection into a coherent and usable cloud infrastructure and offer its services to customers. You also have to ensure that the salient features of clouds–on-demand resource provisioning and the metering of service–are maintained.

This would surely be tough to do, but think of the advantages. First, there would be no single entity that owns or controls it. As with most other P2P applications, a P2P cloud could be created and operated as a grassroots effort, without requiring the permission or consent of any authority. People would choose to participate in a P2P cloud by installing the appropriate client software on their local machines, and the value of the resulting cloud infrastructure would be commensurate with the number of individuals who are contributing to it.

A second advantage comes from the fact that a P2P cloud's components are small, individually consume little power, and well distributed. This drastically reduces concerns about local catastrophes. It also removes the usual worries about heat dissipation. Although such P2P clouds couldn't provide the quality-of-service guarantees of a Google or an Amazon, for many applications that wouldn't much matter.

THE IDEA OF CREATING a huge computing resource from a large number of loosely coupled machines is not new. This has long been done, for example, with volunteer com-

puting, where people execute applications on their personal computers on behalf of others. Volunteer-computing systems usually require you to install certain software, which then runs when your computer has no higher-priority tasks to do. That application then uses your spare computing cycles to fetch and process data from some central server and upload the results back to the same server when it's done.

This strategy works well for many scientific problems, for which a central controller can farm out pieces of the desired computation to workers that operate independently and in parallel. If one fails to return a result within some reasonable period, no problem: The same task is simply handed out to some other volunteer worker.

The Berkeley Open Infrastructure for Network Computing (BOINC) is a popular volunteer-computing system that can load and run different client programs. Examples of projects found on the BOINC platform include SETI@home (to analyze radio signals from space in the search for extraterrestrial transmissions), Rosetta@home (to calculate how proteins fold), and Einstein@home (to detect gravitational waves).

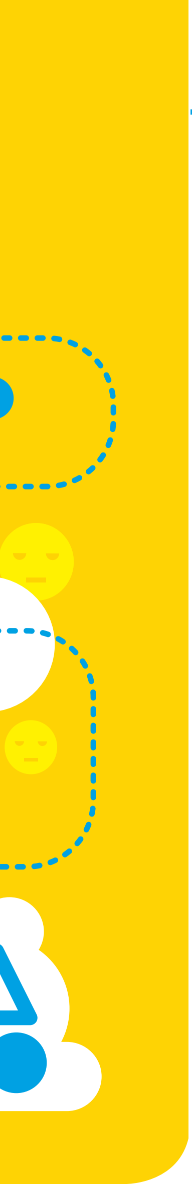
Another type of volunteer computing is known as a desktop grid. In conventional grid-computing projects, multiple high-performance computers in different locations are harnessed to work together on a single problem. Desktop grids allow people to contribute the processing power of their personal computers to such efforts. BOINC supports desktop grids, as does EDGeS (Enabling Desktop Grids for e-Science), a project of several European institutions that is based on BOINC, and also XtremWeb, a project of the French National Center for Scientific Research, the French Institute for Research in Computer Science and Automation (INRIA), and the University of Paris XI.

THE SUCCESS OF THE MANY volunteer-computing projects demonstrates the extreme scale that a P2P cloud could in principle attain, both in terms of the number of different computers involved and their geographic distribution. Using such a collection would, of course, mean that equipment failures will be common. And besides, the people who contribute their computers to these clouds could turn them on and off at any time, something the people who run P2P networks refer to as "churn."

So the first task for any P2P cloud is to keep track of all functioning and online devices enrolled in the system and to dynamically partition these resources among customers. And you have to do that in a completely decentralized manner and despite churn.

To deal with such challenges, many P2P systems make use of what are called gossip-based protocols. Gossiping, in this context, is when the computers linked together in a large, unstructured network exchange information with only a small number of neighbors. Gossip-based protocols have been extensively studied and have been used to model, for example, the spread of malware in a computer network, the diffusion of information in a social network, even the synchronization of light pulses in a swarm of fireflies. Gossip-based protocols are appealing for P2P clouds because they are simple to implement and yet enable complex global computations to take place efficiently even in the face of churn.

So when we built our prototype system at the University of Bologna, which we call the Peer-to-Peer Cloud System (P2PCS), we included several decentralized gossip-based protocols. They are used for figuring out what equipment is up and connected, monitoring the overall state of the cloud,



partitioning the resources available into multiple subclouds, dynamically allocating resources, and supporting complex queries over the set of connected computers (for example, to identify the most reliable ones). Creating those capabilities was an important first step. But there are still many other requirements for a practical system, only some of which we have attempted to tackle.

If all the equipment is owned by a single organization, building a P2P cloud with it should be straightforward, even if the bits and pieces are located in different people's homes, as might be the case with broadband modems or routers operated by an Internet service provider or set-top boxes operated by a cable-television company. The computing devices will be all pretty similar, if not identical, making it easier to configure them into a single computing environment. And because the equipment's one owner presumably installs the P2P-cloud software, you can be reasonably confident that the data and computations will be handled properly and according to the organization's security policies.

This is not true, however, if the P2P cloud is made up of a diverse collection of different people's computers or game consoles or whatever. The people using such a cloud must trust that none of the many strangers operating it will do something malicious. And the providers of equipment must trust that the users won't hog computer time.

These are formidable problems, which so far do not have general solutions. If you just want to store data in a P2P cloud, though, things get easier: The system merely has to break up the data, encrypt it, and store it in many places.

Unfortunately, there is as yet no efficient way to make every computation running on untrusted hardware tamperproof. For some specific problems (such as mining bitcoins), verifying the results is significantly faster than computing them, which allows the client to check and discard faked results. For those problems that do not have an efficient verification procedure, the best way to detect tampering is to compare results for the same calculation coming from independent machines.

Another issue, common to all P2P systems, is that there must be appropriate incentives to get enough people to cooperate and to discourage free riding. Otherwise, the system is bound to degenerate completely. Coming up with incentives would be easy enough for a company that uses its own devices to create a cloud. That company might have a monetary incentive for creating such a cloud, and the people housing the equipment might have an incentive to keep connected to it because they get better service that way.

Volunteer-computing systems don't enjoy the luxury of having such incentives in place. But they typically have such laudable objectives that getting people to contribute their free CPU cycles is not a problem. Who would not want to help make history when SETI@home, which has been around since 1999, detects the first extraterrestrial radio transmission? For volunteer P2P systems of other kinds, though, the incentives have to be carefully worked out.

DEVELOPMENTS ARE ADMITTEDLY at an early stage, but several research projects and a few commercial systems that have hit the market suggest that P2P clouds can indeed be built and used productively, at least for certain purposes.

Our work on the P2PCS, for example, demonstrated that it is possible to use gossip-based protocols to handle the dynamic allocation of resources and the basic monitoring of the system. Other researchers—at the University of Messina, in Italy (Cloud@Home), at INRIA (Clouds@Home), and associated with the European Union's Nanodatacenters project—have been exploring similar concepts.

The Nanodatacenters project is particularly interesting. The researchers involved worked out how to form a managed P2P network from a far-flung constellation of special home gateways controlled by Internet service providers. Because these "nanocenters" are near end users, the network can deliver data much faster than a few large data centers could.

Some commercial distributed-storage solutions are also based on P2P computing principles. An early version of Wuala's cloud backup, for example, allowed users to trade space on their hard disks. Sher.ly offers a similar service but is oriented toward the business sector: It allows companies to use their own machines and infrastructure to create a secure, always-on private cloud to share files internally. There are also a number of open-source P2P systems for distributed file storage (such as OceanStore, developed at the University of California, Berkeley) or computations (such as OurGrid, developed mostly at the Federal University of Campina Grande, in Brazil).

These pioneering experiments are still few and far between compared with traditional cloud environments. But if they succeed, and if researchers can find ways to deal with the hurdles we've described here, you could easily find yourself making use of a P2P cloud in your daily routine. You might not even know you're doing it. ■

POST YOUR COMMENTS at http://spectrum.ieee.org/cloud1014