Big Storage: Make or Buy?

Josh Coates, Scale8 Inc.

We hear it all the time. The cost of disk space is plummeting. Your local CompUSA is happy to sell you a 200-gigabyte ATA drive for \$300, which comes to about \$1,500 per terabyte. Go online and save even more—\$1,281 for 1 terabyte of drive space (using, say, 7X Maxtor EIDE 153-GB ATA/113 5400-RPM drives).

So why would anyone pay \$360,000 to XYZ Storage System Corp. for a 16-terabyte system? I mean, what's so hard about storage? Good question.

What you're facing is the typical make-or-buy decision: Is it worth building it yourself to appreciate the savings? Let's analyze what the "make" process entails.

You head to CompUSA hoping to find a low-end PC into which you can cram a whole bunch of hard drives. The choice isn't a huge one because most cases have room for only two, but, eventually, you spot a super tower that's up to the task. You drag it back to your office and, with your trusty screwdriver, manage to jam six 200-GB drives in there.

But the ATA controller on the motherboard has only two ports—one for the system disk and one for the CD-R—and they're both taken. You disconnect the cable from the CD-R and reconnect it as a secondary device to the primary ATA controller cable. After a few reboots, you feel comfortable that you still have a working system and that all your data drives are receiving power.

So far, you've managed not to injure yourself too severely, although the third time you had to open the case to reset the master/slave pins you did cut your hand on one of the sharp edges. No problem. You've spilled blood for a good cause.

Now it's time to figure out how to hook up six drives to the motherboard. You've got an empty secondary ATA controller that will accommodate two, but that still leaves four. Scouring the Web, you find several vendors that sell ATA controller cards for about \$100. You buy one, along with some extra ATA ribbon cables. A few days later you've managed to attach all your drives. You boot up the system and, voilà, you see drives E, F, G, H, I, and J or /dev/hdc, hde, hdf, hdg, hdh, hdi, and hdj (depending on your operating system flavor).

Now you need to use some kind of RAID protection.

**Why would
anyone SHELL OUT
\$360,000 FOR
A SYSTEM THEY
COULD BUILD
FOR A LOT LESS?**

RAID, which stands for redundant array of independent disks, is a method of combining several hard drives into one logical unit.

Unfortunately, hardware RAID is still pretty sketchy for ATA drives. You then look at the software options: Linux has MD and Windows has its RAID protection (on the server edition). So you install your software RAID. We'll pretend that you don't have any real problems with this except for a few reboots and reinstallations.

Now you've got yourself 1.2 terabytes of storage, and it only took about a day of really tedious work. Excellent! Just 14.8 terabytes to go.

After another week or so, you manage to cobble together 14 of these storage systems, which gives you just over 16 terabytes of raw storage. You plug them all into your network—after buying a cheap 100-megabit switch to string them together—and then you go about configuring and securing all 14 of them.

Are you now ready to start using your storage? Not quite.

The last hurdle is getting your storage-hungry applications to access the storage. Applications access data in two ways: directly from the local host and indirectly via a remote host. The local host access is typically done via a storage area network (SAN) or direct-attached architecture. With your new 14-node storage system, you won't be able to use this direct-access method. Your application will reside on a single node and will be able to access only 1.2 terabytes of data at a time. The other 13 nodes will be inaccessible. Even if your application is distributed, each of the 14 nodes will somehow have to have implicit knowledge about what data resides on which node. This is something most applications are just not built for.

Your other choice is indirect access. Exporting Network File System (NFS) or Common Internet File System (CIFS) is a common method for remotely hosting storage. The application host mounts a file system and gets to work. With your 14-node system, however, your application server will have to mount 14 separate file systems. Similar to the direct method, your application

will somehow have to have implicit knowledge about what data resides on which file system. You also have to administer permissions, security, and performance for 14 file systems. That's a whole lot of Perl scripts.

The cost of your do-it-yourself storage system is looking less attractive. Factoring in the cost of inexpensive servers, the disks, and an inexpensive switch, we're looking at \$50,000 for your custom, management-challenged, 16-terabyte system. Still, this is a far cry from \$360,000.

So, what about the original question: "If your local CompUSA will sell you a 200-gigabyte ATA drive for \$300, which comes to about \$1,500 per terabyte, why would anyone pay \$360,000 to XYZ Storage System Corp. for a 16-terabyte storage system?"

The answer, of course, is that time really *is* money. If you don't have the time, if you don't want the headaches, purchasing a preconfigured, specialized storage system could make some sense. Also, note that if high performance is a requirement, you are in a bind—purchasing a system is the only option. The do-it-yourself method saves an incredible amount of money, but the performance hit is very real.

However, if you've got the time, if you're on a serious budget, if you can live with the hassles, the "make" alternative is one to consider—especially if you are a capable system administrator or software engineer who isn't afraid of creating some custom plumbing (and you have a touch of the masochist in you). In that case, this is an excellent, albeit risky, choice. Q

JOSH COATES, founder and chief technology officer of Scale8, leads the development of that company's innovative storage and file system technologies. He has earned a reputation as a visionary in scalable clustering technology through his work in research labs and at leading enterprise-computing companies. At the University of California at Berkeley he began working with distributed systems and was part of the Network of Workstations (NOW) Group and Millennium Project. He was also a member of the three-man team that broke the world record in parallel sorting on a 16-node Intel-based cluster. Prior to Scale8, Coates worked at Inktomi, developing network-caching software applications.

© 2003 ACM 1542-7730/03/0600 \$5.00.

acm
queue
tomorrow's computing today

Coming in July/August Open Source Not Just for Geeks Anymore

- Making a Business out of Open Source
- Open Source in a Proprietary Environment
- Getting Serious: Open Source in Finance and Government