A better way to store data

September 6 2012, by Douglas Gantenbein

These days, nearly everyone stores things in the "cloud"—business-critical documents, personal photos, e-mail accounts … everything.

Microsoft introduced Windows Azure Storage in 2008. Since then, that cloud offering has gained widespread use, not only within Microsoft, but also by thousands of external customers, and it currently stores more than 4 trillion objects.

Storing massive amounts of information in the cloud comes with costs, however—primarily, the cost of storing all that digital data. Now a Microsoft Research team, working with members of the Windows Azure Storage group, has developed a powerful mathematical tool that significantly reduces the amount of space stored data requires. That, in turn, slashes the cost of storing that data, saving Windows Azure Storage millions of dollars.

The work focuses on a particular challenge within the cloud: managing data to help keep it safe and secure while minimizing the amount of storage space it requires. That's a big challenge. For enterprises and people to trust their valuable data to the cloud, they need a high degree of confidence that their data will be safe. Anyone storing that data, meanwhile, wants to keep costs low.

The easiest way to provide integrity for data is to duplicate it. Three full copies typically are enough to keep the data safe and durable in the event of server failures—and servers will, in time, fail. But obviously, duplication uses a lot of storage. In the case of keeping three full copies,

the storage cost, or "overhead," is simply the whole number: Three.

Microsoft Research, combined with the Windows Azure team, took a new approach to reducing storage demands. The team included Cheng Huang and Jin Li from Microsoft Research Redmond, Parikshit Gopalan and Sergey Yekhanin from Microsoft Research Silicon Valley, and Huseyin Simitci, Yikang Xu, Aaron Ogus, and Brad Calder from Windows Azure Storage.

The team built on a common approach to keeping data accessible and durable while requiring less space. That approach is to "code" the data—in effect, create a shortened description of the data, so that it can be reassembled and delivered to a user.

Windows Azure Storage condenses stored data with a technique called "lazy erasure coding." The name comes from the way coding works in background, not in the critical write path. When a data chunk—called an "extent"—is opened and filled, it is duplicated with three full copies. When it is sealed, erasure coding is launched in the background, when the data-center level is low. The extent is split into equal-sized data fragments, coded to generate a number of parity fragments, with each of the data fragments and parity fragments being stored in a different physical unit strategically placed so that the failure of any single module in a data center—be it a power unit, a switch, a computer, or a disk—will affect only one data or parity fragment. Once the data is erasure-coded and all data and parity fragments are distributed, all three original copies can be deleted. Since the entire-erasure coding operation is performed in the background, it leads to minimum impact in performance.

A well-understood way to perform the erasure-coding operation is called Reed-Solomon coding, which was devised in 1960 and was used in the U.S. space program to reduce communications errors. It also helped

made compact discs possible by catching errors in the discs' digital coding. For example, by using 6+3 Reed Solomon code, which converts three copies of data to nine fragments—six data and three parity, each ⅙ the size of the original data—it cuts the data footprint in half, to an overhead cost of 1.5, resulting in not only half the necessary servers, but also half the power usage and half the physical server space. Lazy erasure coding leads to big cost savings.

Coding data has a cost: It slows performance for servers to reassemble data from code, much as it might take a person longer to read a sentence in which every other letter is missing. Data retrieval also can be slowed if a data fragment is stored on a hard disk that has failed or is on a server that is temporarily offline while being upgraded. This is why the goal of this new approach is to reduce the time and cost of performing data retrieval, especially during hardware failures and common data-center operations such as software upgrades. In addition to reducing data-retrieval time, the goal of the new approach is to perform lazy erasure coding that enables even greater data compression—reducing the data-storage overhead to 1.33 or lower.

The team sought to achieve this with minimal performance losses. Just to achieve storage overhead of 1.33, it is possible to use a 12+4 Reed-Solomon coding, which splits the extent into 12 fragments, deriving four parity fragments that protect the 12 original ones, each 1/12 size of the original data.

"But there is an undesirable effect," Li says. "If a piece of data fails, you will need to read 12 fragments to reconstruct the data. That leads to 12 disk I/O actions and 12 network transfers, and that's expensive, double the disk I/O actions and network transfers needed in 6+3 Reed-Solomon coding."

Huang explains further.

"Encoding and decoding is just one of many operations done in cloud-storage systems," he says. "If you spend a lot of computational resources on that," he says, "then it eats into other operations—data compression, encryption, de-duplication, cleaning up redundant data, and other things."

Reed-Solomon coding is designed for deep space communication, in which error occurs commonly and equally to both data and parity symbols and the design tenet is to tolerate as many errors as possible given a certain overhead. The error pattern in the data center behaves differently from that of deep space communication. First, a well-designed and -monitored data center has a low hard-failure rate. That means that most of the extents in the data center are healthy, with no failed data fragment. Only a small number of extents have one failed data or parity fragment. The extents with two or more failed data or parity fragments are rare and will only appear for a short duration, as the data center will repair those extents quickly and bring them to healthy state. Second, if a data fragment cannot be accessed, the predominant reason is a temporary error caused by a system upgrade, load balancing across servers, or other routine operations.

Built upon the rich mathematical theory of locally decodable codes and probabilistically checkable proofs, this new approach is called Local Reconstruction Codes (LRCs) and enables data to be reconstructed more quickly than with Reed-Solomon codes, because fewer data fragments must be read to re-create the original data in the majority of the failure patterns. In fact, only half the fragments are required—six, rather than 12. In addition, LRCs are much simpler mathematically than prior techniques, resulting in a smaller Galois field—a mathematical construct that reflects the complexity of the operations used to combine the data pieces.

The "local" in the coding technique's name refers to the concept that, in

the event of a fragment being offline, as for a server failure or an upgrade, the code needed to reconstruct data is not spread across the entire span of a data center's servers.

"The data needs to be available quickly, without reading too much data," Yekhanin says. "That's where the notion of locality comes from."

The new coding approach also meets the two main criteria of data storage: Data needs to be reliably stored so it is durable, and it needs to be readily available. Data durability is excellent with LRCs—a data chunk can have three failures and still be rebuilt with 100 percent accuracy. In the unlikely event of four failures, the success rate to rebuild drops to 86 percent. LRC code has better durability than triple-replica and 6+3 Reed-Solomon code.

Best of all, the new coding approach results in a data overhead of 1.29—a 14 percent reduction over Reed-Solomon code's overhead of 1.5. That's perhaps not a huge gain in itself, but spread over the enormous amounts of data contained in Windows Azure Storage, it's significant.

"The new Local Reconstruction Codes allow us to achieve our target storage overhead to keep our storage prices low," says Calder, a Microsoft distinguished engineer. "LRCs provide faster reconstruction times over prior known codes, while still providing the durability we need with low storage overhead."

LRCs represent a significant achievement in information theory and storage design, and the work was awarded the Best Paper award during the 2012 USENIX Annual Technical Conference, for Erasure Coding in Windows Azure Storage.

LRCs could find wide application in computing. Li says one possible use

for them may be in "flash appliances"—devices made by combining several flash-memory drives. These memory devices are fast but require a special process called "garbage collection" to clean up old or unused data. LRCs could help improve this process, because their design enables the flash memory to operate efficiently even during garbage collection.

This work shows the strength of Microsoft's diversity, from theoretical computer scientists such as Gopalan and Yekhanin, to communications and multimedia experts such as Huang and Li in Microsoft Research, to distributed storage and systems experts as Simitci, Xu, Ogus, and Calder in Windows Azure Storage.

"It shows how broad the organization is," Yekhanin says. "We have lots of people working in lots of different areas, and they can connect to create some amazing technology."

Provided by Microsoft Research

Citation: A better way to store data (2012, September 6) retrieved 17 July 2024 from https://phys.org/news/2012-09-a-better-way-to-store.html

This document is subject to copyright. Apart from any fair dealing for the purpose of private study or research, no part may be reproduced without the written permission. The content is provided for information purposes only.