Artificial Intelligence & Machine Learning: The Smarts of the Swarm

From a more abstract point of view, swarm behavior is the collective motion of a large number of self-propelled entities. From the perspective of the mathematical modeler, it is an emergent behavior arising from simple rules that are followed by individuals and does not involve any central coordination.

Wikipedia

BIOMIMICRY AS INSPIRATION

Biomimicry or biomimetics is the imitation of the models, systems, and elements of nature for the purpose of solving complex human problems. (Wikipedia)

DataCore's patented method of synchronizing shared cluster parameters is built on an AI/ML algorithm that has been in production use for over 1.2 billion years—no joke. However, maybe our most fundamental application of biomimicry is the design of our massively parallel cluster technology: the smart machine learning behavior of a swarm composed of nearly independent agents—nodes—that only share a minimal set of sensors, parameters and objectives.

Machine learning is a field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. (Wikipedia)

While geometrically inspired cluster configurations may have a nice ring to it theoretically, the truth is that a cluster in real time often is a harsh and less than deterministic environment. Only very simple rules and concepts survive at those defining moments, while others collapse under the weight of their structure. With the DataCore Swarm architecture, even in a split-brain network scenario, if the data can be reached, it can be read: no transient effects of cluster reconfiguration ever affect that simple fact. Because of their independent nature, nodes cannot drag the structure down—or each other. A flock minus a couple unfortunate birds remains a flock, while a ring that loses a couple of segments is no longer a ring. This doesn't imply that DataCore Swarm is entirely impervious to hardware failures. It simply deals with emergency situations in a sensible, autonomous fashion, where—just like in nature—essential functions are prioritized locally while others take a temporary back seat. The split-brain scenario mentioned above will not interfere with the basic functions of the swarm, i.e., the reading and writing of objects, but may result in temporary over replication. When all connectivity is properly restored, the secondary functions return in full force and quickly trim the excess replicas; if anything, the data was even better protected during this "time of crisis."

Now, for a swarm architecture to really work well, its nodes mainly need to be simple and nimble. In DataCore Swarm, those two requirements play well together.

SIMPLICITY FIRST

First of all, to define our nodes, we started out with a standard hardware and OS environment. We threw out everything that wasn't strictly necessary, especially on the software side, and added just the bare necessities. We cast the remaining software stack, OS and all, into a single-image deliverable that is loaded in RAM at boot time. This bare-metal deployment happens either by booting from PXE net or even from a simple USB stick. All software is kept in RAM; there are no disk installs, patches, or upgrades ever; disk space is only used for actual payload. After boot-up, the generic x86 chassis turns into a dedicated, locked-down appliance that only understands our RESTful HTTP subset and SNMP for management. There are no accounts on a production node, so no SSH or other remote logins are ever able to compromise node security.

Disk drives are used raw—no file systems to corrupt, limit performance, or compromise storage efficiency. No inodes or b-trees to waste disk space: frankly, we at DataCore tend to dislike persisted state other than pure content. Hundreds of millions of small objects on a single drive? No problem! All the information in the cluster can be located in microseconds using strictly zero IOPS by means of a lightning-fast distributed RAM-based index that is populated on the fly at boot time from an efficient linear journal. Objects are stored with their metadata, physically encapsulated together for maximum robustness.

Nodes are simply tied together with Gbit or 10 Gbit/s Ethernet to form a cluster. No configuration is required thanks to auto-discovery, auto-provisioning, auto-load and intelligent capacity balancing of even wildly dissimilar multi-vendor node configurations. Coordinated cluster management is enabled through SNMP or a web console accessible from any node. Since all nodes are running the same code and fulfilling the same tasks, any node can be asked to perform read, write, delete, or info operations on any object in the cluster, as determined by its UUID (universally unique identifier). Both 128- bit numbers generated by DataCore Swarm and S3-compatible named object identifiers generated by the application are natively and simultaneously supported. Just ask node A for object X and it will find out for you that node B is holding the most accessible instance of X at that point in time; redirect your HTTP request to B and get out of the way—no proxy stuck in the middle. Write operations happen in a similar way:talk to A and it finds the most optimal node B to store your object X based on a machine learning data paradigm, then HTTP-redirects you to it. For any standard HTTP stack or web browser, these are lightweight, standard operations.

ELASTIC CONTENT PROTECTION STRATEGIES

Objects are protected against data loss by means of replication or erasure coding. Simply stated, the latter delivers a similar value to object storage as RAID does to block storage. It protects data from being lost as a consequence of drive failure.

Replication protects by maintaining two or more replicas of every object on different nodes in the cluster. For instance, with three replicas being the policy, any two drives or nodes can fail at any time without loss of data or even availability. Replication policies (1, 2, 3, 4...) in DataCore Swarm can be freely selected at runtime, can be set on a per object basis, can automatically change over time—multiple times if desired—as a consequence of a policy specified in a Lifepoint. When required, at the cost of a small amount of overhead, the replicate-on-write option on a storage call can assert the successful persisting of an object on two separate nodes before returning a success status in its protocol.

Erasure Coding (EC) extends the replication-based implementation to create and maintain a selectable number of data segments and parity segments per object, spread over separate nodes. This allows for both better durability (data loss protection) as well as a smaller footprint on disk than replication schemes, at the cost of higher CPU requirements at create and recovery time. Unlike the competition, DataCore Swarm stands for free and independent choice of EC scheme for any object in the cluster. A “5 out of 7” scheme, for instance, distributes 5 data segments and 2 parity segments across 7 nodes. Any 5 segments will suffice to rebuild the original object, so two simultaneous disk or node failures can be survived without any data loss.

The protection provided by a redundancy scheme on a storage infrastructure is called durability and expressed in a number of “nines”—simply a convenient notation to express the probability of conservation of a given object during a full year. For example, “11 nines” indicates 99.999999999 percent probability. This, incidentally, is the durability specified by Amazon for its AWS S3 service in its standard redundancy format, which simply maintains three replicas across different data centers. Using DataCore Swarm Erasure Coding (EC), it is possible to double that number of nines while using only half the disk footprint, including data center space, power, cooling, and other related expenses. Typically, EC schemes are set on a per-class-of-object basis, which allows for the management of very different storage and performance Service Level Agreements (SLAs) within the same DataCore Swarm cluster. This way, it can cater to an extremely wide spectrum of use cases and requirements. On suitably dimensioned network and CPU hardware, DataCore Swarm’s parallel striping of multiple EC segments across different nodes for I/O produces a much higher single connection throughput than would be possible using a single disk drive.

Unlike any of our competitors, DataCore Swarm is truly flexible about the simultaneous and synergistic use of both protection mechanisms, hence the Elastic Content Protection (ECP) moniker (Patent 9,916,198). For instance, since EC is more suitable for large rather than small object sizes, DataCore Swarm has a threshold parameter for object size, underneath which small objects will automatically be replicated rather than erasure coded. This allows for maintaining a suitable degree of durability and space utilization regardless of object size.

DISK DRIVE FAILURE IS A DAILY FACT OF LIFE—SO DON'T PANIC

When you have an infrastructure with thousands of disk drives, drive failure simply is a daily fact of life. At DataCore, we have always felt that the infrastructure itself should take care of this, and treat it as a routine event to be managed, not an exception or an emergency requiring immediate human attention, let alone expensive intervention. The cluster will take care of the hard part automatically. Replacement of hardware such as nodes or disk drives can take place batchwise at any convenient time.

All nodes participate in the disk failure detection process, which makes this nearly instant. As long as there is a bit of spare capacity available, the data whose redundancy was affected (i.e., lost replicas) will immediately be re-protected in a massive, parallel, cluster-wide recovery process: all nodes holding an "orphaned" replica will trigger the creation of an additional replica on another node. Since the former are spread across the cluster, all nodes contribute by doing a small portion of the work. In this way, a disk drive that might take 24 hours to regenerate in a regular RAID configuration may be re-protected in less than 15 minutes. The bigger the cluster, the faster this Fast Volume Recovery process runs; hence, the lower the opportunity for multiple "overlapping" drive failures that might cause data loss. For erasure-coded data, a very similar active regeneration process is triggered (with other vendor's products, this typically happens slowly as a background process or only at object read time). This is why DataCore Swarm is capable of delivering many more nines of durability with an identical EC scheme in an otherwise comparable hardware environment.

When it is desirable to protect content across different data center zones or locations, several options are available. The Local Area Replication scheme leverages high-bandwidth, low-latency connections to split a cluster into several subclusters; sets of replicas or EC segments are then split across those subclusters, providing DR protection and continuous availability even in the face of catastrophic data center failure.

Longer-distance, lower-bandwidth and/or interruptible connections still allow the use of asynchronous Wide Area Replication, where a built-in, massively parallel mechanism of feeds leverages all nodes to interconnect clusters.

COOL AND COMPLIANT

As disk is displacing tape as the medium of choice in many long-term archives, power consumption and associated cooling loads have become a major battleground. They do indeed play a major part in the total cost of ownership (TCO) of such an archive. DataCore's patented Darkive technology takes a smart approach to maintain the random access superiority of disk while approaching the overall power costs of tape. When enabled, Darkive will spin down drives and step-down node power on a large majority of the cluster resources, keeping awake the strict minimum to guarantee service and cycling power on a regular basis for background maintenance. Up to 80 percent or more of power and cooling can be saved without sacrificing any meaningful functionality or performance. This can translate into power savings of up to 25 percent of TCO or in excess of $1,000 per petabyte per month.

Depending on the application, guaranteeing content authenticity and integrity over the long haul may be crucially important. That is where DataCore Swarm's roots in Content Addressed Storage (CAS) come in handy. Indeed, where CAS used to be a separate storage category, DataCore Swarm now offers a superset of CAS' compliance characteristics next to all the goodness of its general-purpose software-defined storage. Mandatory retention periods, programmed object deletion, and content sealing using field-upgradeable hash algorithms, as well as a true legal hold capability, are all standard fare.

SCALING ON THE FLY AND OTHER BALANCING ACTS

As applications evolve, so do their content storage requirements. At that time, you reap the benefits of its massively parallel swarm architecture. No configuration, no provisioning, no installs. Just add x86 nodes—even different models or different brands—and they will be automatically net booted and adopted by the cluster. Intelligent rebalancing in function of available RAM and disk resources happens organically and continually as part of the regular operation of the cluster. Other similar background processes in the Health Processor assert object integrity, cardinality (proper level of replication), and handle time-programmed lifecycle events driven by Lifepoint metadata. The number of replicas can change over time to reflect the evolving value of data; even the redundancy mechanism can auto-change from replication to erasure coding, for instance, to optimize disk footprint when expected access patterns are less intense.

A RIVER OF HARDWARE

Retiring a single disk or a whole node from the cluster happens with a single click on the management web console. As a consequence, the cluster will create replacement replicas or

EC segments of all contained objects elsewhere, and indicate when it is safe to remove that piece of hardware. It is interesting to note that the latter typically doesn't even get accessed in the exercise, considering that its forced retirement may have been decided based on its bad physical condition. Since adding and subtracting nodes in the cluster can happen in flight, migration can take place in an organic fashion, without the massive disruption and extra costs associated with forklift upgrades. Think of your content as a cloud of data that safely stays in place with a slowly flowing river of hardware underneath.

OBJECT, ENCAPSULATED

DataCore Swarm's unit of content, unsurprisingly, is the object. It isn't a file, as no traditional file systems are used inside of Swarm. They simply aren't robust or fast enough to scale into the hundreds of billions of objects. Physically, an object is stored in Swarm as a simple, contiguous sequence or stream of bytes on raw disk. Extremely granular address resolution goes down to 512 or 1024-byte boundaries, so that even when storing huge numbers of small objects, almost no space is lost in gaps between them and storage efficiency remains very high. The contiguous stream consists of two parts, data and metadata, that are written strictly once and always remain physically encapsulated together. The metadata part contains two classes of headers: those that drive Swarm behavior, such as Lifepoints, and custom ones that are inserted by the application, e.g., a Customer ID. Objects are written to the disk on a single "writing front" of free space. Deleted objects are simply marked as such—space is reclaimed asynchronously afterwards by the Health Processor. The segments of erasure coded streams are stored in pretty much the same way as regular streams, while the full set is represented by a manifest listing all segments; both content reading and completeness checking for the set happens based on that manifest. DataCore Swarm even leverages that structure to allow simultaneous cluster-wide, massively parallel uploads of a single large stream by a multitude of parallel clients.

SPEED AT SCALE, THE SWARM SURPRISE

When objects are written, a record is also added to a small circular journal in the front section of the disk. At boot-up time, that journal serves to repopulate the extremely fast RAM index that holds pointers to the disk and sector where a given stream is located. This explains one of the most surprising and appealing characteristics of the DataCore Swarm architecture: the capability of knowing, within a matter of just a couple of milliseconds, on which node, disk and sector a given object is located among 100 billion others in the cluster. Then, DataCore Swarm can position the disk drive arm at the front of the object and read it off the disk, often all in one single disk I/O. Executing a comparable operation on a traditional file system typically takes tens of IOPS. This way, the response time to access an object is entirely independent of the number of nodes or the object count in the cluster.

DYNAMICALLY GENERATED INDEX (GOLDEN PAGES)

The RAM index in Swarm was supercharged by the introduction of a patent-pending scheme called Golden Pages. Before that the reading of an object in the cluster typically required a multicast transmission to all cluster nodes to locate the object. Upon receipt, all nodes would quickly inspect their RAM index for local presence of the required object, returning a response only in the positive case. In itself, this is a simple, reliable, and very robust mechanism that remains available as a fallback for edge cases. The drawback is that this simultaneous index look-up in all nodes starts claiming substantial global amounts of CPU once you get to very high object read rates (greater than 10K objects/sec), regardless of cluster size. Instead, Golden Pages leverages a moderate amount of additional RAM to maintain an additional distributed index that allows for the use of unicast, asking one node rather than all nodes, thereby freeing up large aggregated amounts of CPU and pushing out scaling boundaries by more than an order of magnitude.

WHAT'S IN A NAME?

Superficially, there may seem to be a contradiction between the fact that a DataCore Swarm object is written out just once in pretty much a single operation and the requirement of being able to update it. Looking under the hood, it's easily explained why that isn't the case in reality. DataCore Swarm offers three types of objects or streams; they all leverage the same mechanics underneath. The basic type is the immutable stream: using an HTTP POST operation, it is sent to the cluster, stored on disk in replicated or erasure-coded format, and a 128-bit UUID is sent back to the client as a sign of success. Its existence and location is recorded in the index journal, as well as in a slot of the RAM index itself: about 50 bytes of RAM, including Golden Pages.

The second stream type, anchor stream or alias stream, differs from the basic type in that it can be updated using an HTTP PUT. Rather than updating or overwriting in place, a new stream is written and the existing UUID is "rerouted" to point to it; for this additional level of indirection maintained in the RAM index, an additional RAM slot per stream is required.

Last but not least, so-called named streams fully conform to the naming scheme introduced by Amazon AWS for their S3 storage service. Rather than offering a random UUID back to the client application, they allow the application to bring its own name in an HTTP POST-based create operation: anything that conforms to a standard URI can serve that purpose. There also is one level of container objects, called buckets, essentially named objects themselves, that can group named objects together by having their name recorded in the contained objects' metadata.

A DataCore Swarm cluster can have multiple Tenants and associated Domains that can be secured and administered separately as a kind of virtual cluster. Domains in turn contain buckets that contain named streams, making DataCore Swarm effectively a superset of AWS S3 in that regard. Multitenancy is essential to those use cases where different user populations need to be served in fundamentally independent ways.

DATACORE SWARM: TAKING CARE OF THE BEEHIVE

Can you remember what storage looked like just a dozen years ago? How its place at the bottom of the physical technology stack was faithfully reflected by the utter lack of interest in the strategic value of its content? But then, it started to steadily rise from the lowly ranks of cost centers to one of the organization's most valuable assets through continued insight, analysis and accessibility. Since the beginning, DataCore has been at the bleeding edge of that evolution. Just like the swarm that diligently works to fill and maintain the beehive, it has been building technology that focuses on content and cares for it, stores it, protects it, heals it, weeds it, finds it and provides easy access to it, so that organizations can get to their data when and where they need it.