Cirata Data Platform

The distributed coordination engine

Technical white paper

Table of contents

Cirata Inc. follows a policy of continuous development and reserves the right to alter, without prior notice, the specifications and descriptions outlined in this document. No part of this document shall be deemed to be part of any contract or warranty.

The distributed coordination engine

by Yeturu Aahlad, Chief Scientist, Co-founder, Cirata

Introduction

Other solutions that deliver active-active replication either rely on a central transaction coordinator, or use the group communication approach. The challenge with using a central transaction coordinator is that it becomes a single point of failure and can easily become a performance and scalability bottleneck. Active-active replication solutions that use the group communication approach only work over a LAN.

Cirata's patented Distributed Coordination Engine (DConE)[1] enables activeactive replication without any of these drawbacks. With DConE, there are no single points of failure or scalability bottlenecks. The servers that make up a distributed system can be on the same LAN, or connected over a WAN. At the same time, DConE is transparent from two perspectives: (1) DConE doesn't alter the behavior of the underlying application, and (2) DConE maintains the illusion of a single server system performing at LAN-speed in a distributed environment in which the servers are actually thousands of miles apart. This has significant implications in terms of maximizing productivity and reducing costs by enabling consistent real-time data access and collaboration across a globally distributed organization.

DConE's ability to deliver LAN-speed performance over a WAN while maintaining one-copy equivalence of the data across a distributed system comes from three key elements of its design that work in concert: (1) a highly efficient, fault tolerant and completely decentralized active-active replication algorithm; (2) a configurable quorum-based approach that only requires a quorum of servers referred to as nodes, to agree on the ordering of transactions rather than all nodes; and (3) a design that makes the most efficient possible use of server and network resources.

In addition, DConE was designed to be independent of the underlying application. It can ultimately be used as the foundation for the distribution of any application or database.

This white paper will explain how DConE is able to deliver the reliability, availability, scalability and performance large enterprises require for their mission-critical distributed systems.

Distributed transaction processing with DConE

In order to understand DConE, it is useful to first gain an understanding of the Paxos algorithm. The Paxos algorithm was designed by mathematician Leslie Lamport, to provide a framework for enabling active-active replication in a fault tolerant manner, without the use of a central transaction coordinator.

In the development of DConE, Cirata has taken the Paxos algorithm as a baseline and added significant innovations that have made it practical for mission-critical high, transaction volume distributed environments.

The Paxos algorithm[2]

"...I got tired of everyone saying how difficult it was to understand the Paxos algorithm... The current version is 13 pages long, and contains no formula more complicated than n1 > n2." Leslie Lamport.

Under the Paxos algorithm, a replicated state machine is installed with each node in a distributed system. The replicated state machines then function as peers to deliver a cooperative approach to transaction management that ensures the same transaction order at every node.

The replicated state machines that implement the Paxos algorithm can fill one of any three roles: (1) proposers; (2) acceptors; and (3) learners. There are three phases in the Paxos algorithm, which may be repeated during the process of reaching consensus: (1) election of a node to be the coordinator or proposer; (2) broadcast of the

[1] Aahlad, Y. et al. U.S. Patent and Trademark Office (USPTO) patent number 8,364,633, entitled "Distributed computing systems and system components thereof," issued January 29, 2013
[2] Lamport, L. Paxos Made Simple (2001).

transaction proposal to its peers that then assume the role of learners, who either accept or reject the proposal; and (3) acceptance, once a majority of the nodes acknowledge the proposer and accept its proposal, allowing consensus to be reached. The replicated state machine that assumed the role of coordinator then broadcasts a commit message to notify all of its peers to proceed with the transaction.

To avoid scenarios where multiple nodes try to act as coordinators for the same proposal, Paxos assigns an ordering to the successive coordinator nodes and restricts each coordinator's choice in selecting a value to be agreed upon for the proposal number. To support this, each node keeps track of the most recent agreed proposal sequence number that it has seen. When a node issues a proposal, it generates a sequence number for the proposal with a value higher than the last one it is aware of and broadcasts it to the other nodes.

If a majority of the other nodes reply indicating they have not seen a higher sequence number, the node is then allowed to act as coordinator, or leader for the proposal. At this point, the other coordinators cannot proceed until consensus is reached on the current proposal.

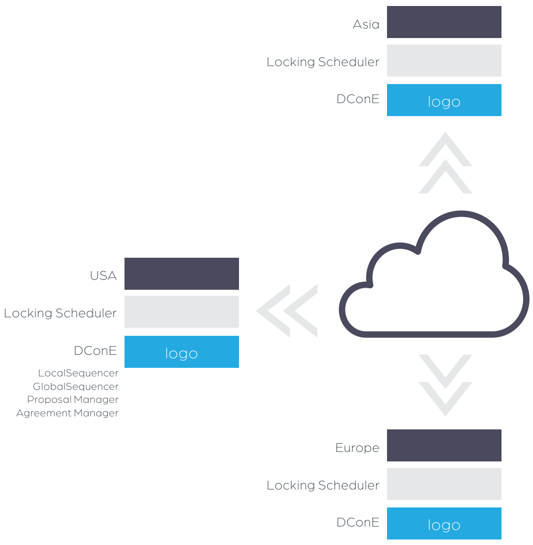
The proposer's sequence number cannot be used by other nodes attempting to be coordinator at the same time, and all future proposals must use a higher sequence number to achieve consensus for future transactions.

Next we'll examine how DConE builds on these fundamental principles.

Achieving consensus with DConE

In order to understand DConE's approach to distributed transaction processing, we will focus on the following core components of each DConE instance that support its active-active replication capability: the proposal manager, the local sequencer, the agreement manager, and the global sequencer.

When a write transaction is submitted for processing by an application client at any node, the proposal manager component of the local instance of DConE generates a proposal for the transaction, which includes the transaction data. It then assigns a local sequence number (LSN) to it.



The LSN reflects the order in which the transaction was submitted relative to all other transactions at that location. The local sequencer then saves the proposal with the assigned local sequence number to its proposal log. If a network or server outage occurs before the local instance of DConE is able to submit the proposal to its peers during the agreement process described below, it will resubmit that proposal after it recovers.

Next, DConE's agreement manager determines an agreement number, which represents a proposed global sequence number (GSN) for the proposal that the local DConE instance will submit to its peers at other nodes. In accordance with Paxos, the agreement number is simply an increment of the GSN of the last proposal accepted by all of the nodes.

This agreement number is then used to obtain consensus about the ordering of the transaction across all of the nodes, so that one-copy equivalence is maintained. The proposal with the agreement number is then written to the agreement log. Each DConE instance's agreement log contains all completed agreements, regardless of where they originated. In the event of a network outage, the agreement log will indicate where the node left off before it lost its connection to the other nodes in the distributed system, making it useful during DConE's automated recovery process. The roles played in the recovery process by both the agreement log and the proposal log will be described in the upcoming section, "Automated Backup and Recovery."

An agreement protocol is then initiated by the local DConE instance's agreement manager, and the proposal is submitted to its peers. Once a quorum of the DConE instance's peers reach agreement on the proposal, the agreement number is used as the GSN across all of the nodes and we have our global transaction ordering. The quorum concept enables DConE to reach agreement without requiring that all nodes are available or agree. The quorum concept is a key element of DConE's performance as well as its fault tolerance. The quorum concept will be discussed in detail in the upcoming section, "Achieving Performance and Scalability."

If agreement is pre-empted by a competing proposal, the agreement manager repeatedly attempts to reach agreement with a new agreement number. Each time agreement is reattempted, an entry with the new agreement number is created in the agreement log. Once agreement is reached by a quorum, the local application node enqueues the agreed upon proposal in its global sequence. At this point the local DConE instance passes the transaction off to its respective locking scheduler for processing, in the agreed global sequence number order. It is important to note that the DConE instance where the proposal originated does not wait for any of the other nodes to complete execution of the transaction; it only waits for agreement to be reached, allowing users to experience LAN-speed performance.

Preserving local sequence[3]

Because DConE supports concurrent agreements for performance reasons, it is possible for the quorum to reach agreement on one proposal that was submitted after another at one of the nodes. However, if the underlying application requires local sequence to be preserved when determining global sequence, DConE can be configured to do so.

An example of an application that would require this configuration is a billing system. If payments can only be made for open accounts, and a customer makes a payment and then immediately closes their account, the system would want to ensure that the order in which the transactions were submitted was preserved, so that the payment would be accepted rather than rejected.

In order to understand how DConE determines global transaction ordering in a way that preserves the local sequence, consider the following example. Assume that a node submits its first two proposals to DConE and the proposal manager assigns LSN 1 and LSN 2 to the respective proposals. Assume further that a total of 25 proposals with GSNs 1 through 25 have been agreed, and no intervening proposals have been submitted by the other nodes. Assume further that the quorum reached agreement on LSN 2 before reaching agreement on LSN 1. If local sequence didn't matter to the application, then LSN 2 would have agreement number and GSN 26, and LSN 1 would have agreement number and GSN 27.

The proposals would then be written in that order at all of the nodes. If the requirement is to ensure that local sequence is preserved at all of the nodes regardless of where the proposals originate, a combination of the LSN, the agreement number, which in this case may or may not end up being the GSN, and the proposer id, which represents a globally unique identifier for the DConE instance where the proposal originated, are used to construct a global sequence that preserves the local sequence order. In effect, the global sequence is sorted in local sequence order within proposer id and passed to the locking scheduler at each node.

The locking scheduler

Everything discussed up to this point has been independent of the underlying application. However, the locking scheduler at each node that DConE passes transactions to once they have been agreed is specific to the underlying application. The locking scheduler behaves like a database scheduler, not a distributed lock manager. The term "locking scheduler" comes from the fact that it relies on the locks specified by the underlying application for concurrency control, so that large numbers of non-conflicting transactions can be processed in parallel.

The locking scheduler is agnostic with respect to the global order. The order in which the locking scheduler submits transactions to the underlying application at each site is driven by a local queue of globally sequenced events (the GSN queue) passed to it from its respective DConE instance. This architecture accomplishes two things: (1) it allows completely local locking schedulers at each node to achieve one-copy equivalence without any knowledge of global state; and (2) it allows DConE to be implemented with any application without modification. It is the locking scheduler which interfaces with the underlying application, not DConE.

[3] Aahlad, Y. et al. U.S. Patent and Trademark Office (USPTO) patent number 8,364,633, entitled "Distributed computing systems and system components thereof," issued January 29, 2013

Achieving performance and scalability

DConE provides seven key innovations beyond the Paxos algorithm that allow it to perform and scale. These include quorum, concurrent agreement handling, backoff and collision avoidance, dynamic group evolution, distributed garbage collection, distinguished and fair round numbers for proposals and weak reservations. Each of these innovations will be described in this section.

Quorum

The quorum concept used by DConE allows performance to be optimized and the impact of network and server outages to be minimized based upon the distribution of users and activity across the sites. The quorum configuration options that are available include majority, singleton and unanimous. With majority quorum, only a majority of the nodes are required to respond to any proposal. Once the majority agrees to a proposal, all nodes in the replication group are required to accept it. DConE also supports the concept of a distinguished node that can act as a tiebreaker in the event that there is an even number of nodes in the distributed system.

With a singleton quorum, only one node has to respond to proposals. The node selected to be the singleton quorum under this configuration would be the node with the greatest number of users and level of activity. The benefit is that no wide area network traffic is generated during the agreement process at the node with the highest transaction volume. Agreement is handled entirely by the local DConE instance at the quorum node. The other nodes submit their proposals to obtain agreement from the singleton quorum node, but typically experience fast performance because they only require the designated singleton node to agree to their proposals, not complete execution of them, before they hand them off to their respective local locking schedulers.

Unanimous quorum requires all nodes to respond, and is inherently the least efficient configuration.

DConE also supports rotation of the quorum from one region to another based on a follow-the-sun model. This allows performance to be optimized on the basis of normal working hours at each site in a globally distributed system

In addition, the quorum approach works in combination with DConE's automated recovery features to minimize the impact of network outages and server crashes on a distributed system.

Concurrent agreement

The Paxos algorithm only allows agreement to be reached on one proposal at a time. This has the obvious effect of slowing down performance in a high transaction volume environment. DConE allows multiple proposals from multiple proposers to progress simultaneously, rather than waiting for agreement to be reached by all of the nodes on a proposal by proposal basis.

Back-off and collision avoidance

DConE provides a backoff mechanism for avoiding repeated pre-emption of proposers by their peers. Conventional replicated state machines allow the preempted proposer to immediately initiate a new round with an agreement number higher than that of the pre-emptor. This approach can lead an agreement protocol to thrash for an extended period of time and severely degrade performance.

With DConE, when a round is pre-empted, the DConE instance which initiated the proposal computes the duration of backoff delay. The proposer then waits for this duration before initiating the next round. DConE uses an approach similar to Carrier Sense Multiple Access/Collision Detection (CSMA/CD) protocols for nonswitched ethernet.

Dynamic group evolution

DConE supports the concept of dynamic group evolution, allowing a distributed system to scale to support new sites and users. New nodes can be added to a distributed system, or existing nodes can be removed without interrupting the operation of the remaining nodes.

Distributed garbage collection – reclaiming persistent storage

Any system that deals with distributed state should be able to safely discard state information on disk and in memory for efficient resource utilization. The point at which it is safe to do so is the point at which the state information is no longer required to assist in the recovery of a node at any site. Each DConE instance sends messages to its peers at other nodes at pre-defined intervals to determine the highest contiguously populated agreement common to all of them. It then deletes all agreements from the agreement log, and all agreed proposals from the proposal log that are no longer needed for recovery.

Use of distinguished and fair round numbers

DConE's use of distinguished and fair round numbers in the process of achieving consensus avoids the contention that would otherwise arise when multiple proposals are submitted simultaneously by different nodes using the same round number. If this option is used, the round number will consist of three components: (1) a monotonically increasing component which is simply the increment of the last monotonic component; (2) a distinguished component which is a component specific to each proposer and (3) a random component. If two proposers clash on the first component, then the random component is evaluated, and the proposer whose number has the larger random number component wins. If there is still no winner, then the distinguished component is compared, and the winner is the one with the largest distinguished component. Without this approach the competing nodes could end up simply incrementing the last attempted round number and resubmitting their proposals. This could lead to thrashing that would negatively impact the performance of the distributed system. This approach also ensures fairness in the sense that it prevents any node from always winning.

Weak reservations

DConE provides an optional weak reservation mechanism to eliminate preemption of proposers under high transaction volume scenarios. For example, if there are three proposers – one, two and three – the proposer's number determines which range of agreement numbers that proposer will drive. This avoids any possibility of collisions among the multiple proposals from each proposer that are proceeding in parallel across the distributed system.

Automated backup and recovery

DConE's active-active replication capability delivers continuous hot backup by default by turning every node into a mirror of every other. This is leveraged to provide automated recovery over a WAN, or a LAN when a node falls behind due to network or server failures. No manual intervention is required.

If a node loses contact with its peers, but is still available to users at its location, those users will still have read access. Writes are not allowed at the node during the outage, because it cannot participate in the agreement process. This prevents a split brain scenario from arising that would result in the node going out of sync with its peers, thus violating the requirement for one copy equivalence across all of the nodes. However, writes can still continue at the remaining nodes as long as a quorum is still available. This minimizes the impact of network outages and server failures on the distributed system.

As soon as the failed node comes back online, its DConE instance catches up automatically with all of the write transactions agreed by its peers while it was offline. This is accomplished by using the agreement log described earlier in the section entitled "Achieving Consensus with DConE." The agreement log contains the last transaction completed at the node before the outage occurred. When the recovery process begins, the node's DConE instance requests all transactions after the last transaction recorded in its agreement log from its peers. In addition, any proposals left in the proposal log that did not complete the agreement process are automatically resubmitted by the local DConE instance, once the catch-up is complete. This means that regardless of whether an outage occurs before or after agreement is reached on any transaction across the nodes in a distributed system, no data will be lost.

In addition, DConE's automated recovery capabilities eliminate the need for disk mirroring solutions that only work over a LAN, not a WAN, and require administrator intervention to achieve recovery. As a result, these solutions can introduce the risk of extended downtime and data loss due to human error.

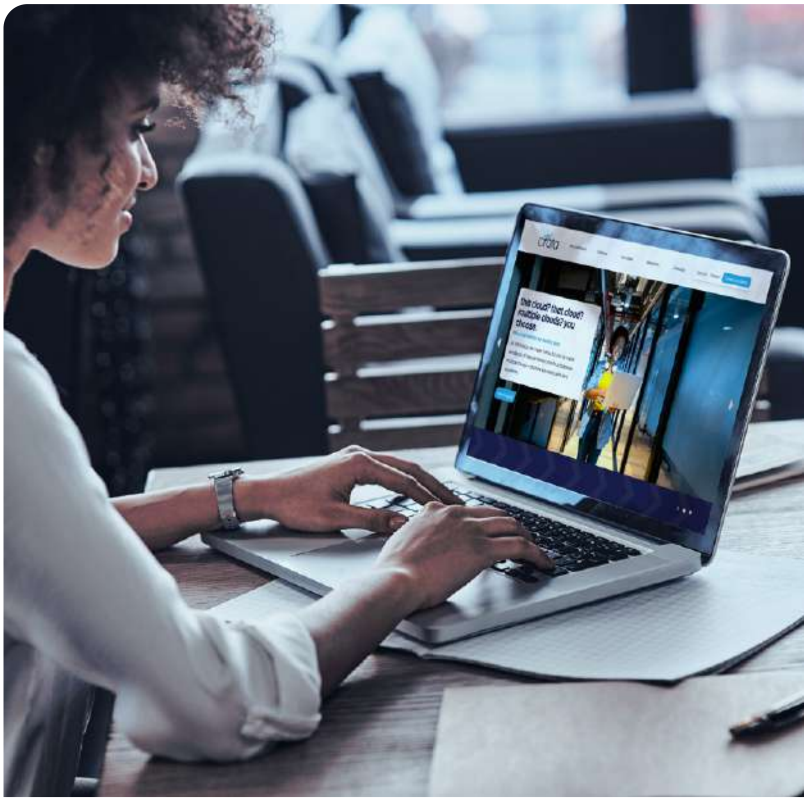
Finally, DConE's automated recovery features also make it possible to take servers offline for maintenance without disrupting user access, since users can be redirected to a server at another site while theirs is offline. This makes full 24-by-7 operation possible in a globally distributed environment.

About Cirata

Welcome to Cirata – a new company with over 45 patents and 15+ years of data science expertise in rapidly integrating high value datasets to leading cloud platforms for game changing AI activation and analytics insights.

We accelerate data-driven revenue growth by automating data transfer and integration to modern cloud analytics and AI platforms without downtime or disruption.

For more information on Cirata, visit www.cirata.com.



5000 Executive Parkway, Suite 270, San Ramon, CA 94583
US +1 877 (926-3472)   EMEA +44 (0) 114 3039985
APAC +61 2 8211 0620   All other +1 925 380 1728

www.cirata.com