Vector Clocks


Sruthi Sree Kumar Follow 3 min read · May 17, 2020

Like Lamport's Clock, Vector Clock is also a logical clock, which is used to assign timestamps for events in a distributed system. Vector clock also gives a partial ordering of the events. One of the shortcomings of Lamport's clock is that it cannot identify concurrent events that are causally related. Here, instead of using integer values for the timestamp, we use a vector of integer values to represent the timestamp. If we have N processes in the group, then each process will have a vector with N elements.

The rule for incrementing the timestamp for the vector clock is also similar to Lamport's clock with a small addition to it. The events update the logical timestamp according to the below rules:

1. Before executing an event, the process i increments the i-th element of its vector clock by 1

```
V[i] = V[i] + 1
```


2. During a send event, the process i increments the i-th element of its vector clock by 1 and sends the time(Vmsg[1...N]) along with the message.

```
V[i] = V[i] + 1
```

3. During a receive event, the process i increments the i-th element of its vector clock by 1. For all other processes, it takes the maximum of the corresponding element in the incoming message and its own local vector and sets it as the corresponding element in the local clock itself.

```
V[i] = V[i] + 1
V[j] = max(Vmsg[j], V[j]) for j ≠ i
```

We can use the vector timestamps to verify if two events are either causally related or concurrent. Before that, we will see how two vector timestamps are compared.

For two given timestamps V1 and V2,

```
V1 = V2 , iff V1[i] = V2[i], for all i = 1 to N (i.e, V1 and V2 are
equal if and only if all the corresponding values in their vector
matches)

V1 ≤ V2 , iff V1[i] ≤ V2[i], for all i = 1 to N
```

Two events with vector timestamps V1 and V2 are causally related, if

```
V1 < V2 , iff  V1 ≤ V2 & there exists a j such that 1 ≤ j ≤ N & V1[j]
< V2[j]
```

Two events with vector timestamps V1 and V2 are concurrent, iff

```
NOT (V1 ≤ V2 ) AND NOT (V2 ≤ V1 )
```

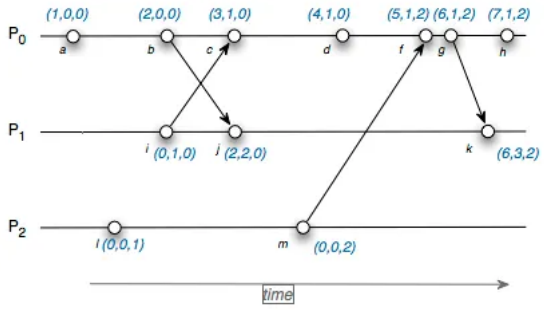


Fig 1: Vector Clock

The figure depicts how the vector timestamp values are incremented for different types of events. Let's see a few examples of concurrent events and causally related events.

Examples for Causally related events:

a -> j as (1,0,0) < (2,2,0)

a -> b as (1,0,0) < (2,0,0)

m -> k as (0,0,2) < (6,3,2)

Examples for Concurrent events:

For the events b and l we have vector timestamps (2,0,0) & (0,0,1)

respectively. Here neither b ≤ l nor l ≤ b. Hence, they are concurrent events. Similarly, d and j are concurrent events.

Vector clock is also a logical clock that obeys causality and ensures partial order of events. Compared to Lamport's clock, it can identify concurrent events efficiently as Vector Clock is represented using more space.

References:

Image: https://www.cs.rutgers.edu/~pxk/rutgers/notes/clocks/index.html

https://www.coursera.org/learn/cloud-computing/lecture/dy8wf/2-5-vector-clocks

Responses (5)

Ishu

What are your thoughts?

Young Bryan Yu
Jun 4, 2024

> Before executing an event, the process i increments the i-th element of its vector clock by 1

A local event within a single node (process).

1 Reply

Khushal Sahni
Oct 9, 2024

Super clear explanation, better than most articles online, thanks!

Reply

Young Bryan Yu
Jun 4, 2024

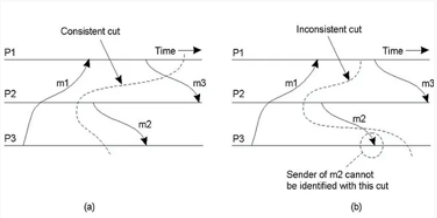
> 2. During a send event, the process i increments the i-th element of its vector clock by 1 and sends the time(Vmsg[1...N]) along with the message.

This is synonymous to step 3. The sender node's vector clock remains unchanged, but the receiver node's vector clock is changed.

Reply

See all responses

More from Sruthi Sree Kumar and Big Data Processing

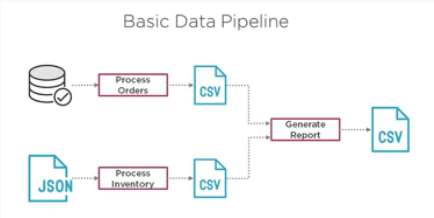


In Big Data Processing by Sruthi Sree Kumar

Global Snapshot, Chandy Lamport Algorithm & Consistent Cut

A Global Snapshot or a global state consists of local states of each process in the...

May 17, 2020 119



In Big Data Processing by M Haseeb Asif

Getting Started with Luigi—What, Why & How

This is the first of a two-part series about getting started with Luigi. The second part i...

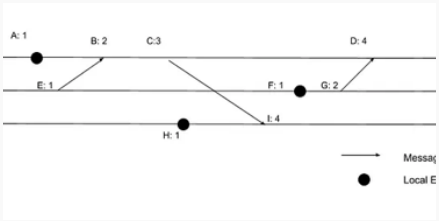
Oct 2, 2022 25

In Big Data Processing by M Haseeb Asif

Data warehouse vs Data Lake vs Data Lakehouse

With the ever-increasing amount of data produced, data analytics systems are...

Jan 2, 2022 131 3



In Big Data Processing by Sruthi Sree Kumar

Lamport Timestamps

The Clock is an important building block in cloud computing systems and distributed...

May 13, 2020 89

See all from Sruthi Sree Kumar | See all from Big Data Processing

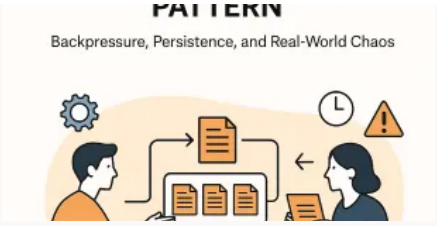
Recommended from Medium

Ben Meehan

Understanding Change Data Capture (CDC): The Backbone of...

Modern software systems generate and consume massive volumes of data at every...

5d ago 7

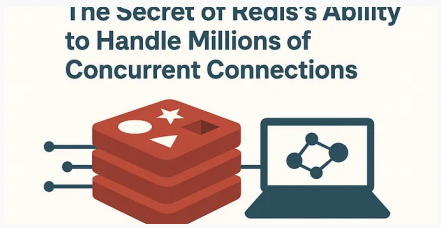


Alpesh Dhamelia

Mastering the Producer-Consumer Pattern: Backpressure,...

In today's era of microservices and distributed systems, data flow management...

Jul 30



In Stackademic by Umesh Kumar Yadav

The Secret of Redis's Ability to Handle Millions of Concurrent...

Today, I want to dive deep into why Redis effortlessly handles millions of concurrent...

Sep 24

Jirapong P

Simplest HA KV ever? Redis replacement Dragonfly

A blazing-fast Redis-compatible datastore with zero cluster config and effortless high...

May 14

Tech Guide

How I Built a Lightning-Fast Redis Cache—And What Broke at Scale

It started out perfect.

Aug 4 10

dematrix

How I created an Order Matching Engine

Exchanges need an order matching engine to execute the orders. There are two types of...

May 31 5 1

See more recommendations