Rotura Decentralized Dataset (RDD) white...

Abstract

We present Rotura Decentralized Dataset (RDDs), a distributed memory abstraction that lets programmers perform in-memory computations on large blockchain network in a fault-tolerant manner. RDDs are motivated by two types of challenges that current computing frameworks handle inefficiently: Blockchain scalability and smart contract runtime. In both cases, keeping data in memory can improve performance by an order of magnitude. To achieve fault tolerance efficiently, RDDs provide a restricted form of shared memory, based on coarsegrained transformations rather than fine-grained updates to shared state. We have implemented RDDs in a system called Tura Blockchain, which we will evaluate through a variety of user applications and benchmarks.

Introduction

RDD(Rotura Decentralized Dataset) are the primary data structure in Rotura Runtime. RDDs are reliable and memory-efficient when it comes to parallel processing. By storing and processing data in RDDs, Rotura Runtime(R2) speeds up data process.

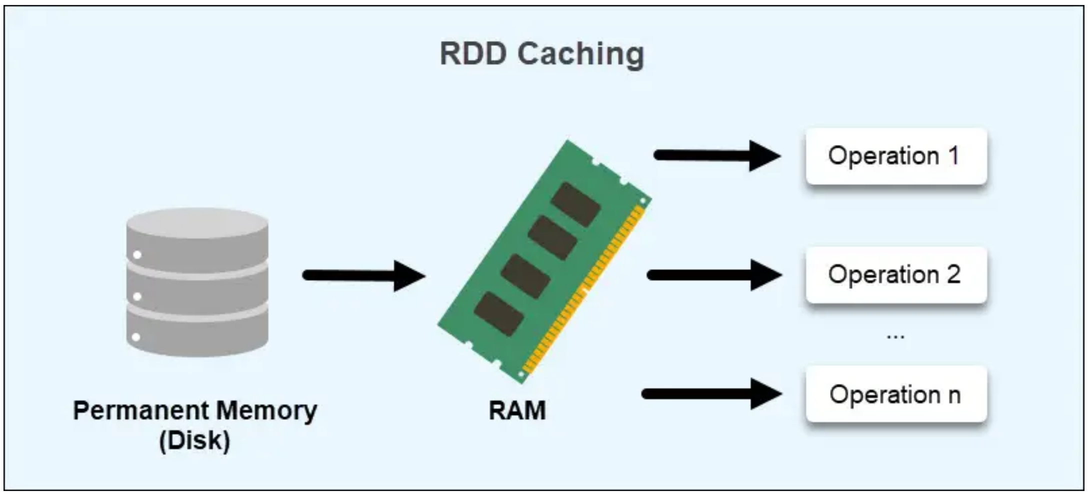
This whitepaper provides a comprehensive overview of RDDs.

What Is a Rotura Decentralized Dataset(RDD)?

A RDD is a low-level API and R2's underlying data abstraction. An RDD is a static set of items distributed across clusters to allow parallel processing. The data structure stores any Python, Java, Scala, or user-created object.

Why Do We Need RDDs?

RDDs address paralell computing in data sharing.



RDDs aim to reduce the usage of external storage systems by leveraging **in-memory compute operation storage.** This approach dramatically improves data exchange speeds.

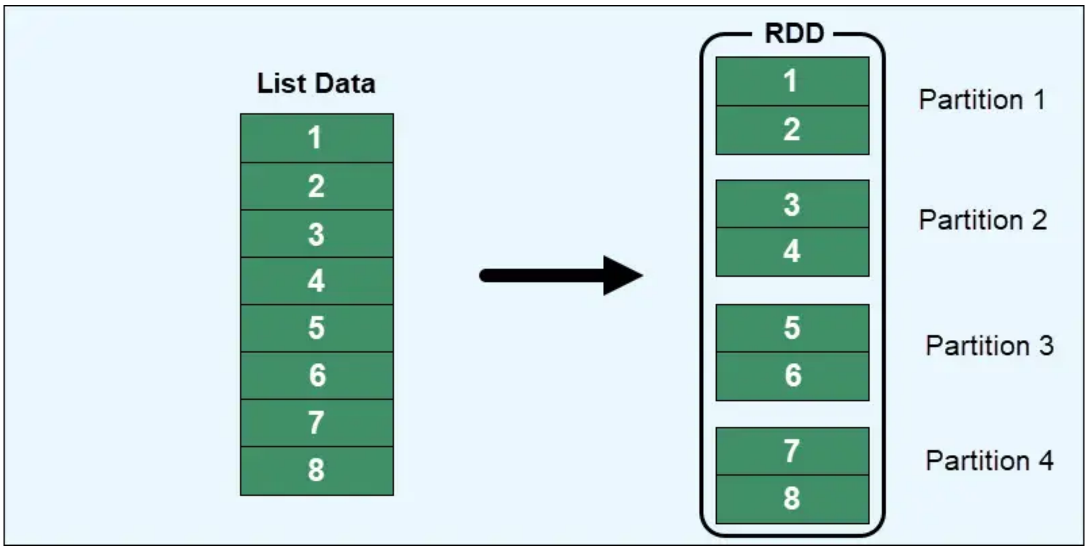
Speed is critical when working with large data volumes. R2 RDDs make it easier to train AI model and handle large amounts of data for dapps.

How Does RDD Store Data?

An RDD stores data in read-only mode, making it immutable. Performing operations on existing RDDs creates new objects without manipulating existing data.

RDDs reside in RAM through a caching process. Data that does not fit is either recalculated to reduce the size or stored on a permanent storage. Caching allows retrieving data without reading from disk, reducing disk overhead.

RDDs further distribute the data storage across multiple partitions. Partitioning allows data recovery in case a node fails and ensures the data is available at all times.



R2's RDD uses a **persistence optimization technique** to save computation results. Two methods help achieve RDD persistence:

- `cache()`

- `persist()`

These methods provide an interactive storage mechanism by choosing different storage levels. The cached memory is fault-tolerant, allowing the recreation of lost RDD partitions through the initial creation operations.

R2 RDD Features

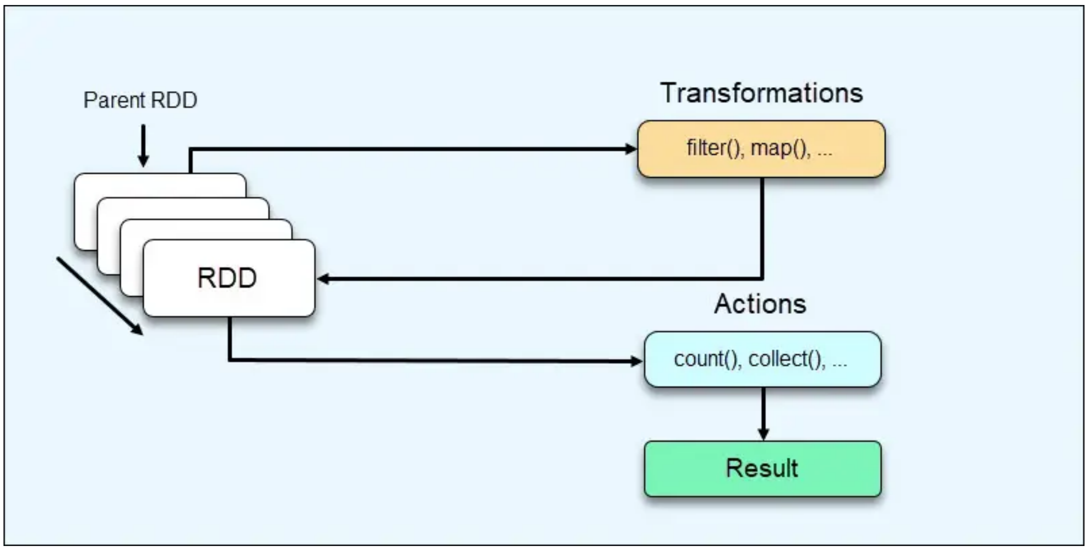
The main features of a R2 RDD are:

- **In-memory computation**. Data calculation resides in memory for faster access and fewer I/O operations.
- **Fault tolerance**. The tracking of data creation helps recover or recreate lost data after a node failure.
- **Immutability**. RDDs are read-only. The existing data cannot change, and transformations on existing data generate new RDDs.
- **Encryption**
- **Signature**
- **Lazy evaluation**. Data does not load immediately after definition – the data loads when applying an action to the data.

R2 RDD Operations

RDDs offer two operation types:

1. **Transformations** are operations on RDDs that result in RDD creation.
2. **Actions** are operations that do not result in RDD creation and provide some other value.



Transformations perform various operations and create new RDDs as a result. Actions come as a final step after completed modifications and return a non-RDD result (such as the total count) from the data stored in the R2 Driver.

Advantages of RDDs

The advantages of using RDDs are:

- **Data resilience**. The self-recovery mechanism ensures data is never lost, regardless of whether a machine fails or cyber attack.

- **Data consistency**. Since RDDs do not change over time and are only available for reading, data consistency maintains throughout various operations.
- **Performance speeds**. Storing data in RAM whenever possible instead of on disk. However, RDDs maintain the possibility of on–disk storage to provide a massive performance and flexibility boost.