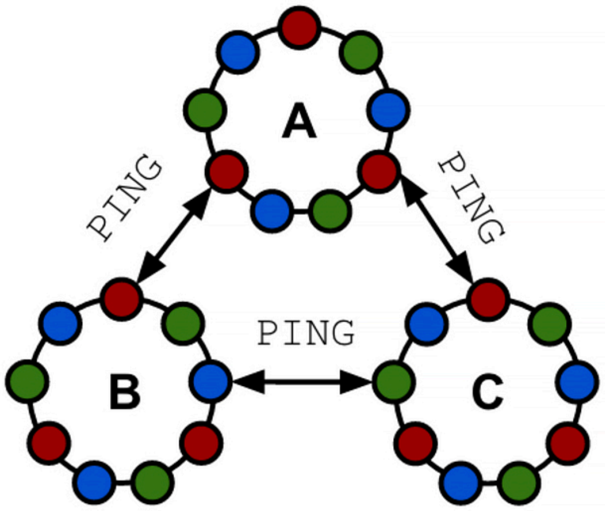
How Ringpop from Uber Engineering Helps Distribute Your Application

Lucie Lozinski

February 4, 2016



Remember the candy jewel rings kids used to wear? Slide the one-size-fits-all ring on any finger and you could immediately become a fashion icon. Ringpop, an open source library developed at Uber to make our applications cooperative and scalable, is just as sweet as its lollipop counterpart. It fits any old-school application with ease and transforms it into a set of cooperating nodes. Best of all, Uber's Ringpop is built to handle the heat of Uber Engineering's traffic and growth without a meltdown.

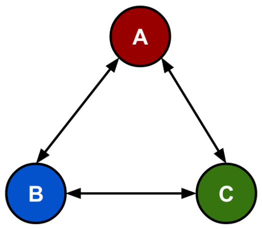
An Intro to Ringpop

Uber's hypergrowth challenges our engineers to think creatively about scalability. With many millions of trips per day occurring across six different continents, we can only handle traffic if we adopt systems that detect and resolve their own failures, adapt to our ever-increasing amount of data, and distribute load to maintain high availability.

If you've read the docs or seen the code, you know that Ringpop is a Node.js library that brings cooperation and coordination to distributed applications that would otherwise run as sets of independent worker instances. Ringpop has three parts:

1. A membership protocol that allows independent workers to discover each other and detect failures (SWIM).

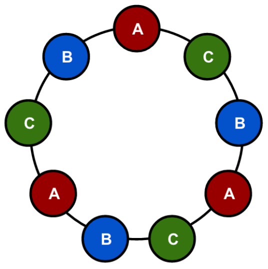
Ringpop as a Membership Protocol



Ringpop 1) implements a **SWIM gossip protocol** variation, 2) gossips over **TCP**, 3) computes membership and ring **checksums**, and 4) retains members that are **down** in its member list.

2. Consistent hashing to assign work across the workers.

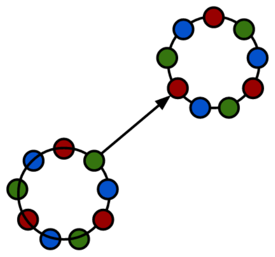
Ringpop as a Consistent Hash Ring



Ringpop 1) uses **FarmHash** as its hashing function, 2) uses a **red-black tree** for its ring, and 3) adds a uniform number of **replica points** per node.

3. Forwarding capabilities.

Ringpop Forwarding



Ringpop 1) has codified a **handle or forward** pattern, uses **TChannel** as its transport and forwarding channel, has forwarding **transparent** to the developer, 4) packs forwarded requests as **HTTP** over TChannel.

Ringpop detects new capacity when it's added to the cluster, removes capacity from the cluster in the event of failure, and distributes the load evenly over whatever capacity there is at any given time. In typical uses, Ringpop acts as routing middleware, directing requests to its owner before it ever reaches its handler.

To implement Ringpop, embed the library into your application, and it will organize all instances into a hash ring. Each instance announces itself, becomes a node in the ring, and discovers the others through a membership protocol. Eventually, every node that makes up your application will know about every other node. Since nodes cooperate without you, there's no need to keep track of all of them and their respective loads.

Availability and scalability are not new aspirations; Amazon's Dynamo paper (the one that inspired projects like Riak and Apache Cassandra) sprouted from the same never-go-down, always-keep-growing demands for large-scale organizations. But what Ringpop brings to the table is a way to get that same partitioning and cooperation at the application level.

Ringpop's Benefits for Geospatial

Ringpop's benefits become much clearer if we explain its first use case: Uber's Geospatial service. Geospatial, the first layer of Uber's matching system, keeps track of the real-time location of every active online driver partner on our platform to gather potential matches for ride requests. Pinged continuously with driver location updates, it then searches those locations for riders with the app open. Here, database storage would be useless because of how fleeting the location data is. Instead, Geospatial workers distribute the load and carry that ephemeral state. Ringpop creates an ordered, cooperative, intelligent space that makes this system possible.

Before Geospatial, we used a system that held all active vehicles in memory. This slowpoke system had to search through every single car to find the ones near the requested pick-up location, even if they were nowhere near the vicinity from the get-go. This worked for a while but, with our growth, became unsustainable. Ringpop enables Geospatial's load distribution, worker coordination, and ultimately horizontal scalability. Here's how:

Gossiping Members

First, the SWIM gossip membership protocol lets us divvy up work automatically. When we add servers the code doesn't change. New processes discover each other, disseminate information in an infectious manner, and repeatedly ping each other so that each node knows about every other node's existence and status. After you embed Ringpop into your application's code, your instances will keep track of membership.

Consistent Hashing

Next, the consistent hashing function of Ringpop eliminates the need for manual reassignments when the application cluster is resized. The hash ring algorithm lets you assign objects to intervals rather than to specific workers.

Ringpop bootstraps itself with a bootstrap file or host list. No multi-cast, yet. The first node has no one to join. It exists as a 1-node cluster.

The second node starts up and queries the same bootstrap list. This time it finds **A**.

A and **B** exchange info. They form a 2-node cluster and hash their addresses along the ring.

Imagine that your manager quits. It's easier to assign you and your teammates to the closest manager than to loosen all of the office's managerial assignments and then redistribute all employees to new managers in the name of even distribution. Likewise, when node 1 in the hash ring goes down, work proceeds as usual. All of those objects between nodes 1 and 2 get picked up by node 2. Thus, when you add or lose a worker to your application, you don't reassign and rebalance object data to new spots, significantly reducing latency (because math).

Forwarding

Requests for data can hit any random node, and Ringpop will forward the request to the appropriate destination within the ring. This handle-or-forward pattern means your business logic doesn't concern itself with any of the details around routing, membership, or forwarding.

```
module.exports = function handleOrForward(ringpop, key, req, res, arg2, arg3, handle) {
var dest = ringpop.lookup(key);
if (dest === ringpop.whoami()) {
// Handle request on local node
handle(ringpop, key, res, arg2, arg3);
return;
  }

// Forward request
var channel = ringpop.channel;
var peer = channel.peers.add(dest);
var outreq = new RelayRequest(channel, peer, req, function buildRes(options) {
res.headers = options.headers;
res.code = options.code;
res.ok = options.ok;
return res;
  });
outreq.createOutRequest();
};
```

Ringpop's handle-or-forward pattern makes sure that every request reaches the correct node.

Typically, all applications that talk to a sharded application have to know how it's partitioned in order to send a particular request to the correct node. Ringpop eliminates the need for any coupling between client and server. Your application will route requests within itself, and clients will communicate with your application without knowledge of the sharding scheme.

C decides to join and now we have a 3-node cluster. Each of them ping one another periodically.

A receives a request and hashes the shard key. It uses the **handle or forward** pattern to determine where the request should be processed.

A decides to forward the request to **B**.

Ringpop in the World

Ringpop allows developers to tune their use of consistent hashing and the membership protocol for a specific application's needs. Ringpop developers take ownership of the scalability and availability of their services rather than relying on external infrastructure-level solutions or systems specialists.

At Uber, engineering teams are already embedding Ringpop into new projects—for sharding, leader election, ordering or batching writes into a distributed database, request coalescing, work delegation, data aggregation, and caching. Now, we're eager to see the inventive ways that engineers around the world adopt our open sourced Ringpop to build faster, smarter, more cooperative, self-healing applications.

Comments

Lucie Lozinski