Divergence visualization in Lasp program execution

Creupelandt Grégory 59191300

Advisor: Peter Van Roy

What is Lasp?

Lasp is a group of **Erlang libraries** providing a service for the development of large **distributed applications**, focusing on **CRDT**s. The main idea is to avoid having a centralized server running a database but using **peer-to-peer** instead.

What are CRDTs ?

CRDTs are for Conflictfree Distributed Data Types. They are an inovative way to handle distributed variable on multiple nodes using only peer-to-peer. The well known CAP theorem does not allow strong consistency, strong availability and partition tolerance. But **strong availability**, **partition tolerance** and **eventually strong consistency** is achievable with CRDTs which is a very good compromise. Better than that, CRDTs allow to simplify a lot the system complexity by avoiding some bottlenecks such as consensus.

How do CRDTs work ?

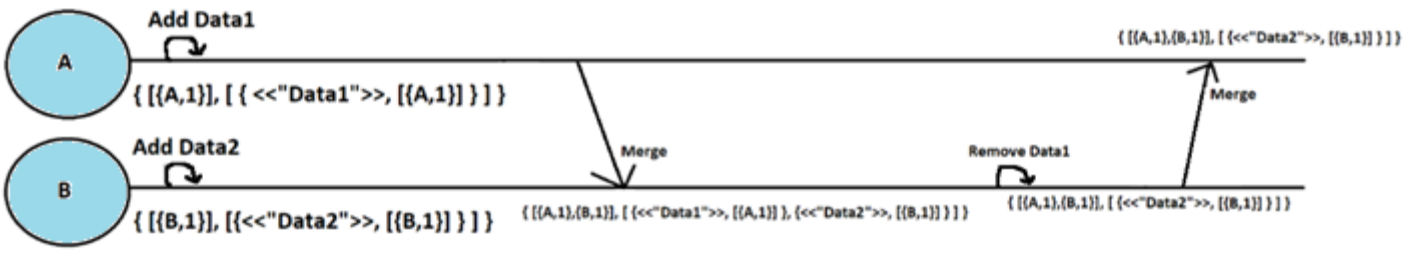
The idea is to use a **datastructure containing values and metadatas**. Each node will have a local version of this datastructure, let's call it **local state**. Frequently, it will send its local state to its peers. When receiving this state from peers, it will merge it with its local state in a very clever way allowing the result to represent the most recent modifications. In other words, by simply sending their local states, the nodes will converge to a global state that will be consistant on every node. This datastructure allowing to carry values and metadatas and developped in a clever way allowing **deterministic convergent merges** is called CRDT.

Conceptual example:

Let's imagine we have 2 nodes and we use an awset, it is a CRDT that acts like a bag,
Both nodes can add and remove elements and will eventually end up with the same local state after a certain time.

Amazing CRDT features:

- Nodes eventually end up with the same state
- No scheduler required (Order of messages has no impact)
- No consensus required (causality is directly handled via metadas)
- End user is not concerned about scheduler, causality, state marges… He can simply use a clear API to update or query the CRDT.



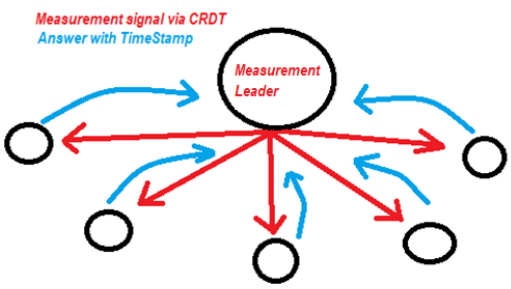
My Goals ?

Develop an API to allow the user to access information about the system convergence:

The system will eventually converge but this may take a relative long time (few seconds for example). This « convergence time » information may be useful for the user in many cases.
This is why I implemented a background measurement tool (resiliant to nodes joining the cluster or crashing) that measure the convergence time in background:

1) A leader is elected by leader election on nodes.
2) This leader puts an element in a special CRDT,
3) When other nodes detect this element in their local state, they answer with a timeStamp.
4) When the leader gathered all the answers, it knows the time it required for the system to converge.
5) Informations about the last measure such as convergence time and roundtrip time are accessible from every node with a clear API.

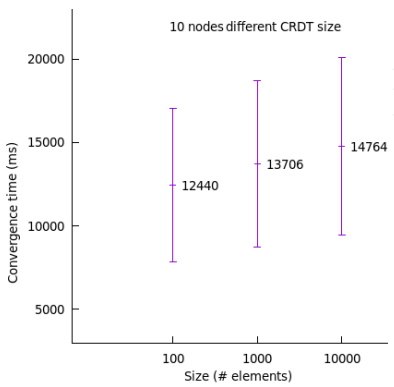
Schematic below represents the general easy-to-understand idea but is not a perfect representation since the exact communication and messages are not centralized like this but peer-to-peer.

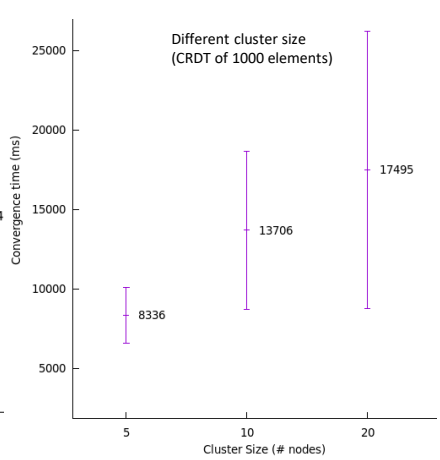


Measure how various parameters can impact the system convergence:

Parameters like the number of nodes in the cluster, physical distance between them, network speed and traffic, partitions, CRDT size, update frequency, etc… May impact the convergence time of the system but also the network trafic and the nodes memory and thus globally its performance.

Here are the cases currently under measure for convergence time, memory usage and network traffic:
Cluster size: 5, 10, 20 nodes
CRDT size: 50, 100, 1000, 10000, 20000 elements
Update frequency: 5, 10, 50, 100, 500 operations/sec
The measure are not really determining because of huge standart deviation. I am working on improving measurement precision but here are already some few examples I measured:





On going ?

- **Improve the API to allow user to modify convergence time.**
- **Test more complicated or extreme cases.**
- **Find the limits of the system.**
- **Improve measurement precision.**