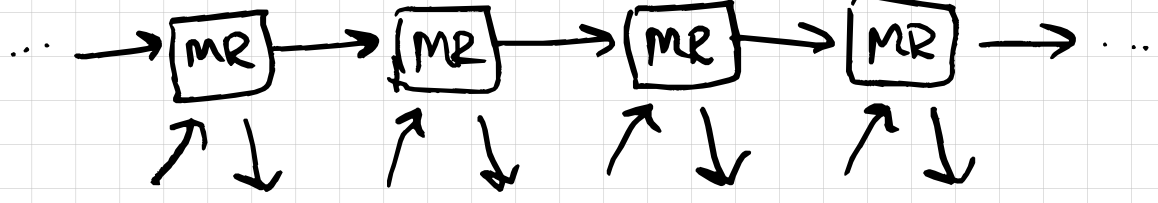
CSE138 Lecture 18

this time:

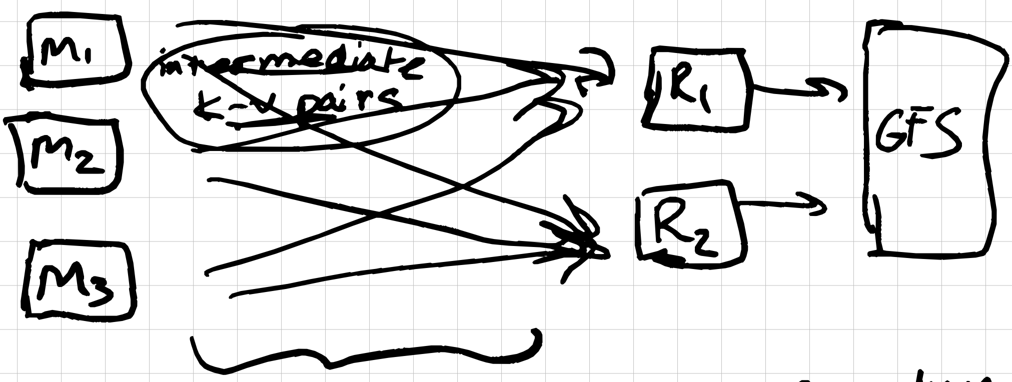
- MapReduce wrap-up
- the math behind replica conflict resolution



it would be slow to read/write data to disk for each MR job!

a system for orchestrating MR jobs: FlumeJava (2010)

master



shuffle

hash(key) mod N ← # of reduce workers

with MapReduce, this is not expected to change.

Another MapReduce example:

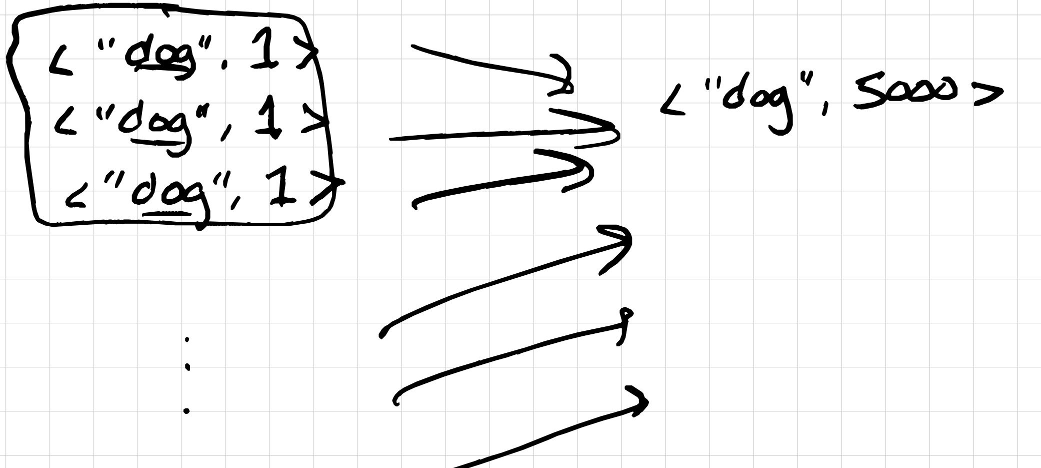
word count

doc ID	contents
doc1	"my", "dog", "spot"
doc2	
doc3	

<"dog", 1>   map function

```
for document in documents:
    for word w in document:
        emit<word, 1>
```

intermediate k-v pairs



could have combined these intermediate k-v pairs locally on each map worker

<"dog", 3>

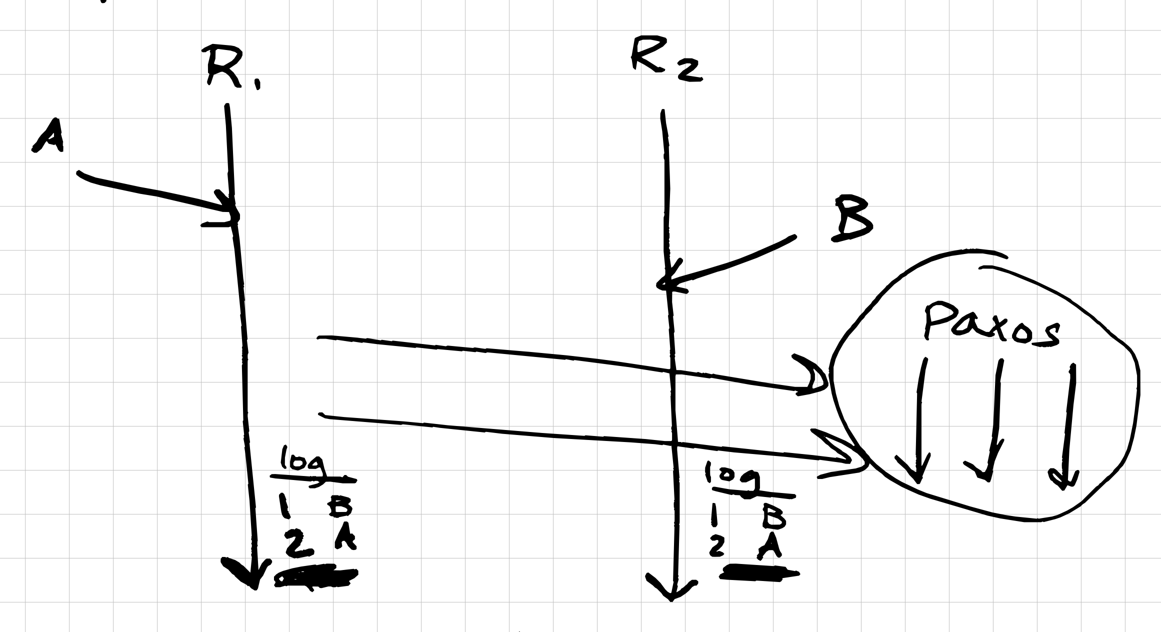
This is using a combine function.

$$\boxed{(a+b)+c = a+(b+c)}$$

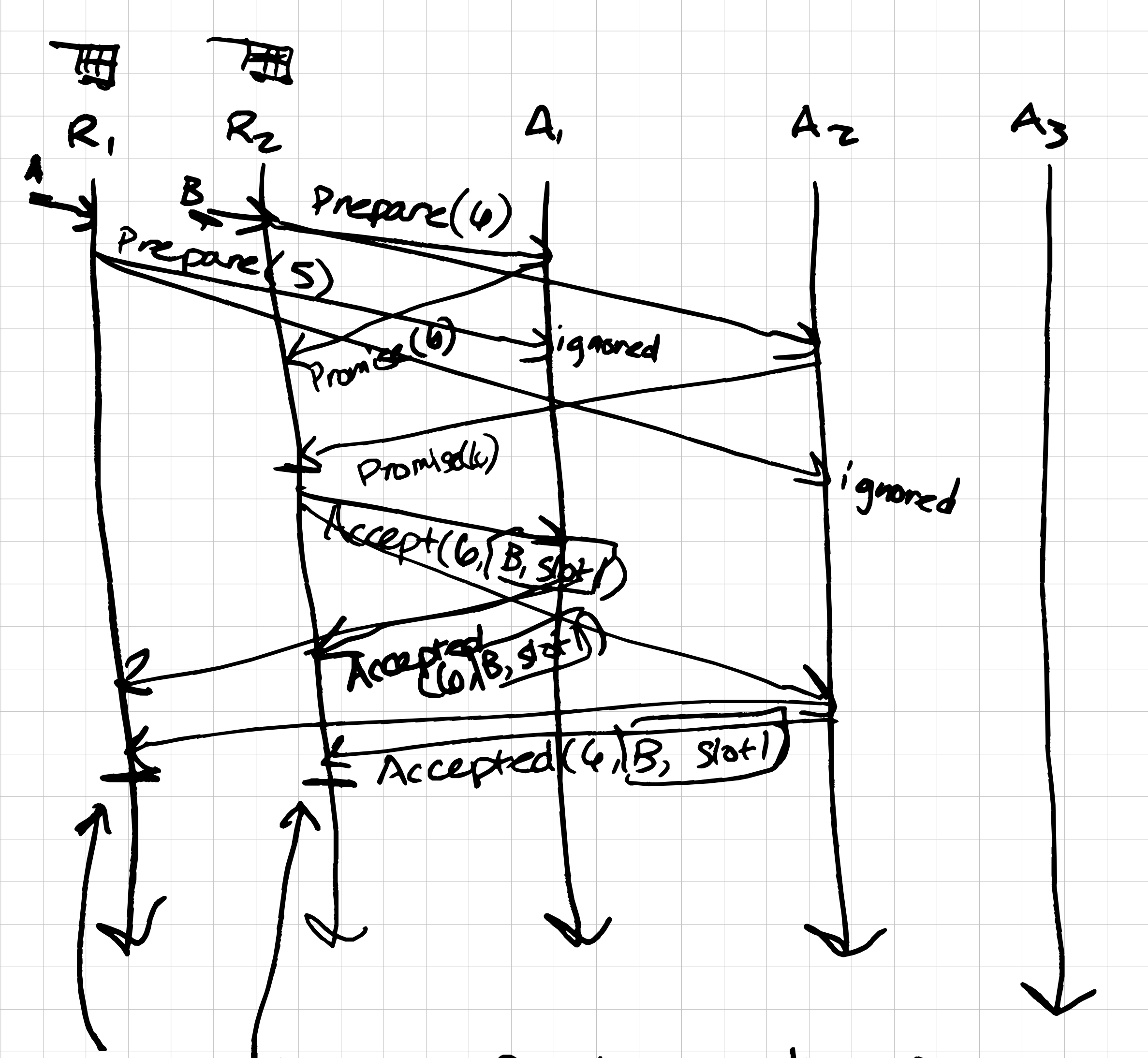
← associativity of addition

As long as the operation that reduce workers would do is associative, it's OK to do some of it in advance on the map workers, using a combine function.

The math behind
replica conflict resolution



if we have no leader, and everyone can receive updates,
and we want strong consistency,



now R_1 and R_2 both know that B goes in slot 1.

Consensus protocols are expensive!

often, you just need

Strong convergence:

replicas that have delivered the same set of updates have equivalent state.

$\{\square, \square\}$ $\{\square, \square\}$

Remember partially ordered sets?

A partially ordered set (poset) is
a set S,
together with a binary relation $\sqsubseteq$
relating elements of S.

S = subsets of a set $\{\triangle, \circ, \square\}$
with $\subseteq$ (set inclusion) as the relation

$\{\triangle\} \subseteq \{\triangle, \circ\}$

some elements of S are comparable

$\{\triangle\} \not\subseteq \{\circ\}$

$\{\circ\} \not\subseteq \{\triangle\}$

some elements of S aren't comparable.

Also, some axioms have to hold for our relation $\sqsubseteq$:

- reflexivity: $\forall a \in S, \quad a \sqsubseteq a$.
- antisymmetry: $\forall a, b \in S$,
  if $a \sqsubseteq b$
  and $b \sqsubseteq a$,
  then $a = b$.
- transitivity: $\forall a, b, c \in S$,
  if $a \sqsubseteq b$ and $b \sqsubseteq c$,
  then $a \sqsubseteq c$.

$$S = \{\emptyset, \{\circ\}, \{\triangle\}, \{\square\}, \{\circ, \triangle\}, \{\triangle, \square\}, \{\circ, \square\}, \{\circ, \triangle, \square\}\}.$$

All that was review.
But, something new: upper bounds.
pick 2 elements of S:
e.g. $\{\circ\}, \{\triangle\}$
and ask: which elements of S are at least as "big" as both of those, according to the ordering relation?

in this case: it's $\{\circ, \triangle\}$, and $\{\circ, \triangle, \square\}$.
those are the upper bounds of $\{\circ\}$ and $\{\triangle\}$.

$\emptyset$ and $\{\triangle\}$ have
$\{\{\triangle\}, \{\circ, \triangle\}, \{\square, \triangle\}, \{\triangle, \circ, \square\}\}$
as upper bounds.

Given a poset $(S, \sqsubseteq)$,
an upper bound of $a, b \in S$
is an element $u \in S$
such that $a \sqsubseteq u$ and $b \sqsubseteq u$.

An upper bound u of $a, b \in S$ is the least upper bound (lub), or join,
if $u \sqsubseteq v$ for each upper bound v of a and b.
$(u, v \in S)$

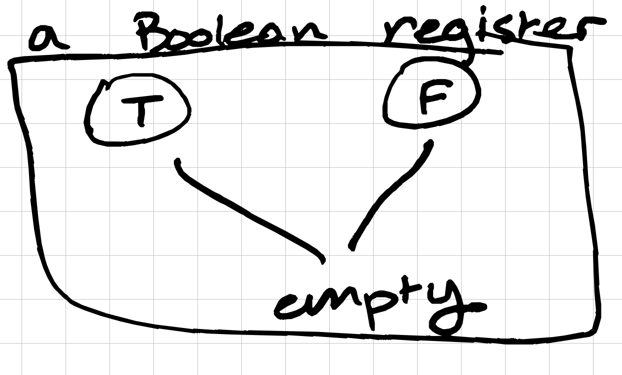
The lub of $\emptyset, \{\triangle\}$ is $\{\triangle\}$.

The lub of $\{\square\}$ and $\{\square\}$ is $\{\square, \square\}$.

A poset where every two elements have a lub is called a join-semilattice.

Here's one that isn't a join-semilattice

a Boolean register

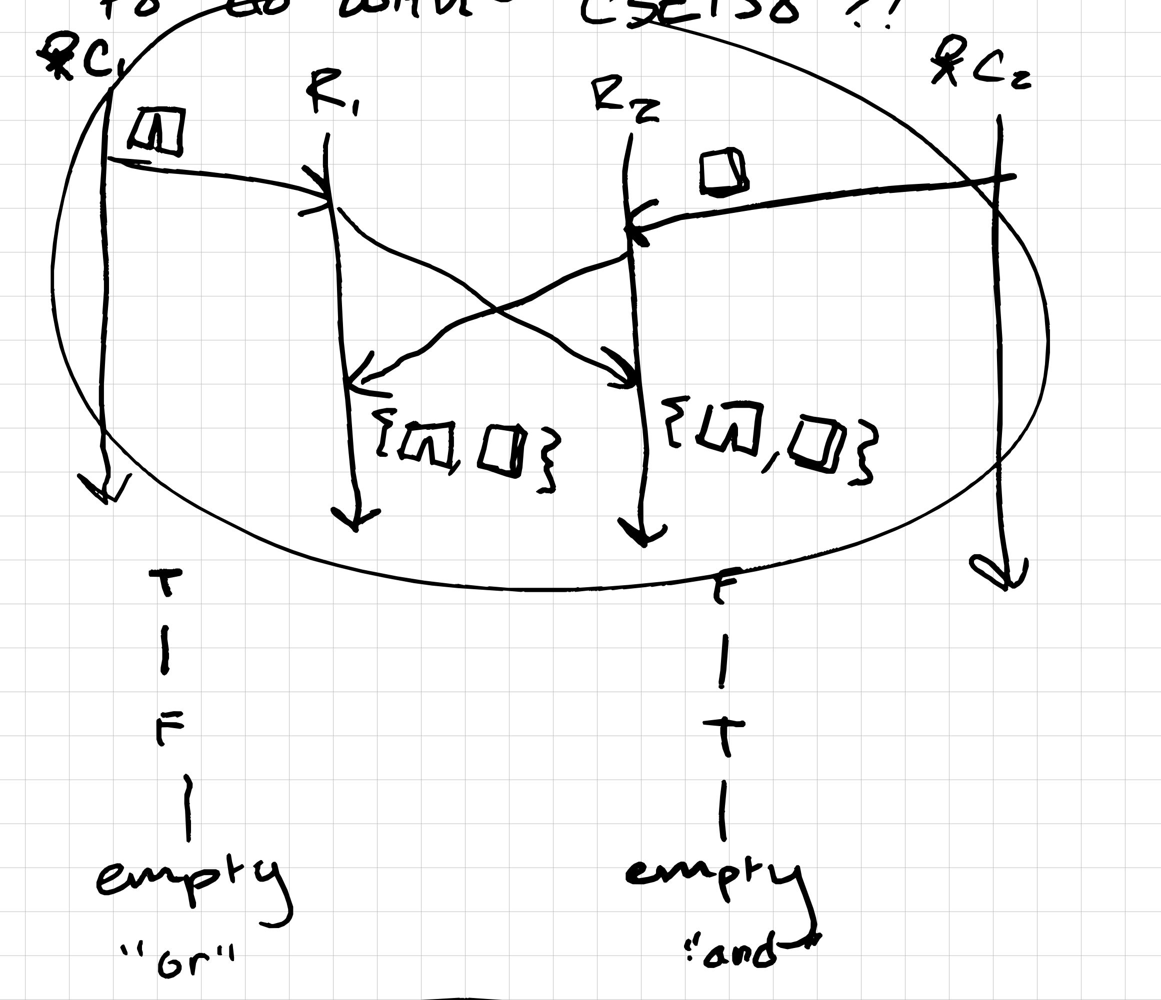


$S = \{\text{empty}, T, F\}$

empty $\sqsubseteq$ T,
empty $\sqsubseteq$ F,
T $\sqsubseteq$ T
F $\sqsubseteq$ F
empty $\sqsubseteq$ empty.

But there's no element here that is at least as "big" as both true and false!
So this is not a join-semilattice.

What does any of this have to ~~do with~~ CSE138 ?!



informal claim:

if the states that a replica can take on are elements of a join-semilattice, then there is a natural way of resolving conflicts between replicas, without consensus. Just use the lub!

conflict-free replicated data types (CRDTs) are distributed data structures that exploit this idea.

* Marc Shapiro, et al. (2011)

applications for CRDTs include collaborative text editing.

Check out Automerge!

Martin Kleppmann's (and others) work.

local-first software

---

* "merge what you can, fork what you can't"