Rules-based Programming for Distributed, Concurrent, Fault-Tolerant Code

Collin Lee, John Ousterhout, and Ryan Stutsman

Is there a better way to write DCFT Code? Why do we care?

Rise of Distributed, Concurrent, Fault-Tolerant Code

- Needed in Large-Scale Application Infrastructure:
  Bigtable, Hadoop, RAMCloud, Chubby, Zookeeper, etc.
- Typically manages collection of distributed resources.

DCFT Code is Hard to Implement

- Nondeterminism due to concurrency and faults.
- Previous action does not provide enough information to determine next action.
- No crisp algorithmic solutions.

Example DCFT Modules from RAMCloud

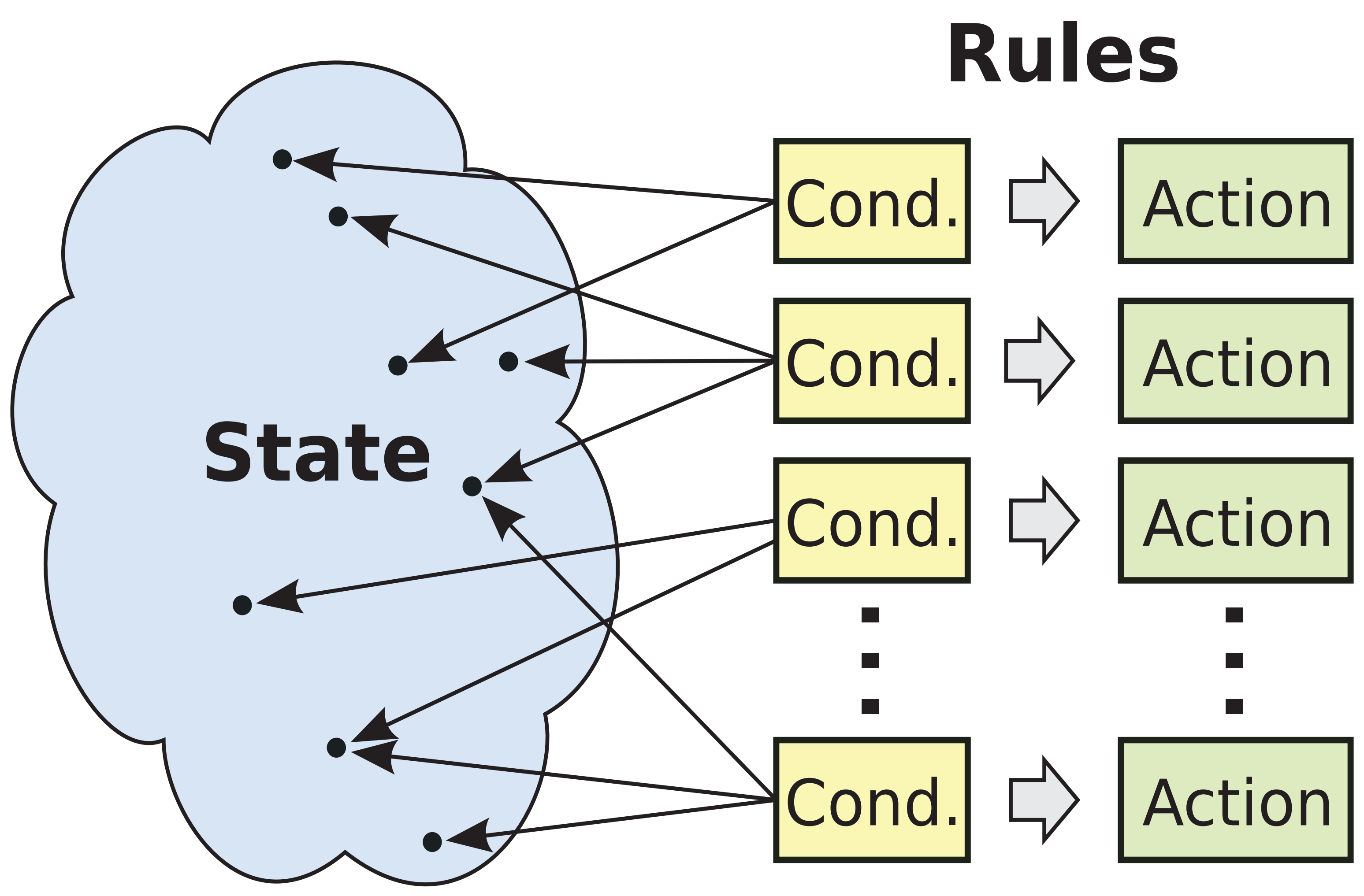
Log Replication and Recovery

- Manages replication across thousands of machines.
- Failures arise at arbitrary times.
- Reaction to failures depends on system state.
- Enforces ordering constraints to ensure safe operation.

Cluster Membership Updater

- Manages multiple asynchronous RPCs in one thread.
- Dynamically adjusts the number of outstanding RPCs.

Rules-based Code



Action : small, nonblocking piece of code executed when a given condition is satisfied

Cond. : condition predicated on state variables associated with the module

- Progress is made by repeatedly evaluating the rules until the goal is reached.
- Execution adapts automatically to concurrency and faults.
- Changes in control flow occur between rules.

Structuring rules for efficiency: Tasks and Pools

Tasks

- A task groups related rules, state, and a goal.
- Once a task has reached its goal, its rules no longer need to be evaluated (see Pools).
- *In RAMCloud, tasks are instances of classes that contain rules, state and goal information.*

Pools

- Divide tasks into two groups: ***active*** and ***inactive***.
- Tasks in ***active*** pools will evaluate their rules.
- Completed ***inactive*** tasks are removed from the pool and thus will not evaluate their rules.

Orthogonal but Related Work

The “Threads vs Events” debate is orthogonal to Rules-based programing in that it deals with the issue of concurrency but not fault-tolerance. Both threads and events are necessary but not sufficient for DCFT modules.

Threads

- Serial programing model does not work for DCFT modules.
- Need for additional synchronization increases complexity and chance of deadlock.
- Tasks could be threaded but it does not benefit RPC or I/O heavy actions.

Events

- Similar to rules in its management of concurrency and asynchrony, but, unlike rules, event execution is still logically serial.
- Traditionally uses call-backs to explicitly define what code will execute next (serial execution).

Discovering rules

The rules-based programing style was discovered in retrospect following the completion of many DFCT modules in RAMCloud.