A Meta Rete Interface For Perpetually Online Distributed Rule-Based Systems

Maarten Hubrechts, Kennedy Kambona, Thierry Renaux, Simon Van de Water, Mathijs Saey, Coen De Roover, Wolfgang De Meuter

Software Languages Lab, Vrije Universiteit Brussel, Belgium

Problem: building a distributed CEP system

- Rule-based Systems (i.e., Rete algorithm)
- Cluster Computer + elasticity

= ?

- How to render Rete distributed?
- How to adapt to changing cluster configurations?
- How to assure quality attributes? (e.g., load balancing / resilience)

An actor based distributed Rete

Represent each Rete node of the Rete graph as an actor:

+ Possible to statically partition Rete graph across cluster nodes
+ Location transparency
+ Fine-grained concurrency

- Does not support varying cluster configuration
- Does not support declarative implementation of non-functional concerns

Mete: cluster configuration / quality attributes through declarative meta-programming

A Meta program:
- Elasticity of cluster computer covered by meta rules
- Declaratively implement non-functional concerns by providing user-defined meta rules

Example user-defined meta rule: Respawn crashed Rete node

Experimental setup:
- 5000 facts are sent at a rate of 1 event / millisecond
- Base Rete graph is compiled from one rule that gets fired for each incoming fact
- Simulated failure of entire base Rete graph after 5 seconds

Observation:
- Rule activations resume after 50 seconds of downtime
- Expected 5000 rule activations happen

Meta interface:

Left-hand-side:
- Node (nodeID)
- NodeCrashed (nodeID)
- test (nodeID != root && nodeID != agenda)

⇒
- retract [Location (nodeID, machineID)]
- retract [NodeCrashed (nodeID)]
- assert [RespawnNode (nodeID)]

Right-hand-side:
- RequestSnapshot (nodeID)
- RequestTimer (timerID, milliseconds)
- RequestLoadCheck (nodeID)

Implemented in Elixir: Actor model based language (based on Erlang)
+ Built-in persistent distributed database called Mnesia

A Mete program:
1) Meta facts that reify the distributed base Rete graph
2) Meta rules that reason about distributed base Rete graph
3) Meta interface used to write new user-defined meta rules

+ Elasticity of cluster computer covered by meta rules
+ Declaratively implement non-functional concerns by providing user-defined meta rules

Meta interface:

Meta facts that reify the distributed base Rete graph:

Left-hand-side:
- Node (nodeID)
- NodeProcessID (nodeID, nodePID)
- AlphaNode (nodeID)
- BetaNode (nodeID)
- Edge (fromNodeID, toNodeID, side)
- Location (nodeID, managerID)
- ManagerProcessID (managerID, managerPID)
- Snapshot (nodeID, timestamp, sides, outputLog)
- TimerElapsed (timerID)

Right-hand-side:
- RequestSnapshot (nodeID)
- RequestTimer (timerID, milliseconds)
- RequestLoadCheck (nodeID)

...