Flowstate: A Language for Secure Replicated Computation

Priyanka Mondal  Owen Arden
pmondal@ucsc.edu  owen@soe.ucsc.edu
University of California, Santa Cruz

Challenges in a distributed environment
- Interactions in a distributed system span over multiple trusted domains.
- Interactions within a trusted domain are secure.
- There is mutual distrust among independent domains.

Information flow control in distributed systems
- Information flow and trust ordering can be used to prevent data leaks.

Lack of integrity and availability
- **Problem 1**: What if none of the workers is trusted enough to execute $e$?
- **Problem 2**: What if the intended worker is unavailable?

Replication of computation
- Replicate the computation at multiple workers.
- It increases trust, distribute the authority and increases availability.

Computation in Flowstate language
- **Problem**: Can not replicate the expression $e$ directly, as the workers are having different integrity labels.
  - **Solution**: Expression $e$ is attenuated to the integrity level of the workers with a translation function $\lceil \cdot \rceil$. If integrity of $e$ is $I$ then integrity of $\lceil e \rceil_w$ at worker $w$ is $(I \lor \text{w})$. Translation function is defined over expressions, types, memory locations, and security labels.
- **Replication Scheme**: $W$ is called ‘replication scheme’ which represents the integrity (and availability) required for the computation to be successful.
- **Trade-off between integrity and availability**
  - $W = (A \land B \land C)$ will not tolerate failure of $B$ but it gives more computational integrity, while $W = (A \lor B \lor C)$ will tolerate failure of two workers among $A,B$ and $C$ but provides lesser computational integrity.
  - Programmers need to choose $W$ wisely based on the requirements of the application they want to run.

Flowstate operational semantics
- Two kinds of operational semantics:
  1. Global semantics: $[c; e; l; c] \rightarrow [c'; e'; l'; c']$
  2. Local semantics: $[c; i; [w]] \rightarrow [c; i; [w; w; \text{w}]]$
- Sync evaluation rules:
  1. $\text{sync} \in \mathcal{W} \land [c; e; l; c] \rightarrow \text{sync} \in \mathcal{W} \land [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]$
  2. $\text{if sync} \in \mathcal{W} \land [c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]$
  3. $\text{sync} \in \mathcal{W} \land [c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]$
  4. $\text{sync} \in \mathcal{W} \land [c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]; [c; i; w; c; e; i; w; c; \text{w}; e; \text{w}; c; \text{w}]$

Future goals
- Instantiating Flowstate with consensus based systems and protocols like Blockchain, State Channels, BFT etc.
- Beyond reads and writes: what can we do with higher-level abstractions for distributed operations?

References