**Branch Consistency - a declarative consistency model with branching as a first class primitive**

- **Treat branches as the first-class primitive**
  - Explicitly reasons about branches (world views), not independent objects
  - Guarantees isolation between branches

- **Application-centric**
  - "Consistent". No meaning outside of an application
  - Declarative:
    - Users specify what a conflict is
    - Users specify when/how to merge

**Conflict Definition**

- **What is a conflict?**
  - Defined by the user. 
  \[ f: \text{txn,world view} \rightarrow \{0,1\} \]
  - Determines whether can execute a transaction on this world view.
  - Conflict definitions are associated with transactions

**Conflict Handling**

- **How should we handle conflicts?**
  - **Computational Time Logic**
    - Determine when/how to branch
    - Constructs World View DAG
  - **Linear Time Logic**
    - Express properties of individual branches.

**Conflict Resolution**

- **How/when/if do we resolve the conflict?**
  - Via a user-defined resolution function
  - Optional and asynchronous
  - Explicitly merge branches, not objects

**Prototype: Transactional storage with parallel snapshots**

- Transactional
- Supports multiversion concurrency control and branches
- Supports arbitrary conflict definitions
- Never forces merging

- Handles conflict through branching
- Non-blocking (including merging and replication)
- Efficiently models the World View DAG

**What branch consistency enables**

- No more distinction between local vs remote storage
- No more reliance on properties of data/operations
- Composition of consistency levels through varying conflict definition
- Flexibility: emulates existing consistency models
- Performance. Branching can be made cheap