Transactions in Interactive Applications

Alan Geller

Microsoft

Scenario: Web Spreadsheet

- Imagine a spreadsheet built as a Web app, with the recalc engine running in the cloud
  - Allows huge spreadsheets
  - Fast recalc using parallelism across many servers
- Problem: this means large latencies on recalc, even of small spreadsheets
- Solution: asynch recalc – start recalc and give control back to the user
- New problem: multiple parallel recalcs need to be isolated
  - Simple isolation isn't enough: the user has started these recalcs in a specific order, and expects them to be processed in that order

Concurrency and Ordering

- “Standard” serializability is enough to guarantee isolation, but doesn’t guarantee that the user will see what they expect
  - “Equivalent to any ordering” vs. “equivalent to the order the user initiated the actions”
  - If the user types “c” “a” “t”, they don’t want to see “act” in the text buffer
- In this scenario, transactions (recalcs) have to be committed in the same order they were started
  - “Start-order serializable”, not just “serializable”
  - Interestingly, this is the same as instruction retirement in an out-of-order CPU

Other Uses

- Standard client applications that have complex processing in response to user actions
  - Without start-ordered isolation, you can't process multiple user actions in parallel
- Other time-ordered transaction sequences
  - Web-based source code control
  - ???

Summary

- Sometimes simple serializability isn't sufficient
- Isolation coupled with ordering requirements is not supported by much current software
  - Nor is it taught in schools, or written about in journals…
- There is both theoretical and practical work to be done here
  - In the local, in-memory case, there's a fairly simple scheduling algorithm using multiple versions
  - The distributed case is open…