TRANSACT'13 Town Hall

Language-Level Standards for TM

Justin Gottschlich (moderator)

Best Application Track Paper

Transactionalizing Legacy Code: An Experience Report Using GCC and Memcached. Trilok Vyas, Yujie Liu and Michael Spear (Lehigh University).

Outline

- C++ specific work
  - Open issues
- Other languages?

C++ and SG5

- Standard C++ Study Group
- Formed in May 2012
  - 74 members, ~15 highly active members
- Initial direction from “Draft Specification of Transactional Language Constructs for C++”

- **Goal**
  - Standardize C++ transactional language constructs

C++ TM Specification

- Transaction blocks and expressions
  - **`__transaction_relaxed`** and **`__transaction_atomic`**
  - Can wrap a block of code (in '**`{}`**') or an expression (e.g., **`if (__transaction_atomic(x++))`**)

- Annotations
  - **`[[transaction_callable]]`** and **`[[transaction_safe]]`**
  - Instruct compiler to create instrumented clone of function
  - Identify code that can/can't be called from atomic transactions

- Exception-related stuff
  - E.g., to abort *and not retry* a transaction that encounters an exception

SG5 Open Issues

- Atomic transactions
  - safe-by-default
- Cancel and exceptions
  - flat vs. closed nesting
- Relaxed transactions

C++ SG5 Questions

- What's the most important thing to get right?
- Are we missing anything critical?
  - tm_waiver
- Are we going in the right direction?
- Are there high level ideas we should consider?
- Are there usability issues with the spec?

Other Languages

- What's after C++? Can we work on these in parallel?
- Should we aim for same TM constructs across many languages?
  - For Haskell, maybe not
- What other key questions are we not asking?