
Transactions and existing code

DB: Should we worry about putting TM in existing languages?

MH: Yes, but we should also pursue new languages.

EM: Should we care about interoperating with existing code?

JM: Yes. Even in Fortress, where the idea has been to start from scratch, we need to interoperate with existing code.

Features of transactions

DB: Is it a good idea to try to extend transactions to thorny semantic areas, for example IO?

JM: There must be a line that we draw somewhere.

RH: Perhaps the line is at "memory," since it's "transactional memory."

MH: We do need to make sure that we don't lose the simplicity of TM in extending it via various features.

EM: Using "atomic" versus "open_atomic" seems to clearly delineate simple TM from programmer-managed TM.

JM: But perhaps people will use the complex mechanisms more than is good for them.

VS: It would be a mistake to try to make transactions into a silver bullet.

Short atomic blocks (e.g., ~10 statements) can be very useful.

DW: That seems backwards to me, because where transactions have been successful is in databases, where they are very long. Why not learn from that?

VS: This is a different situation with different requirements.

SC: It's worth pointing out that in databases, it is certainly not simple to reconcile atomicity/isolation and performance.

EM: An interesting thing to note is that software mechanisms have been acceptable for concurrency control in databases.

BK: Of course, databases are dealing with disk IO.

SJ: But if transactions were limited to 10 statements, what would I teach my students?

VS: I do not believe that transactions will be the silver bullet, so the reality is that there will be multiple primitives and we will have to think about their interaction.

Weak and strong atomicity

JM: Would everyone want strong atomicity if it were free?

MS: I do not see why it is necessary.

RH: What about moving data between transactional and private state (the

"privatization problem")?

MH: We are working on solutions. [NOTE: see the Rochester slides for elaboration.]

MM: Back to the original question, does anyone really want weak atomicity?

SD: Why are we expecting hardware to do everything in TM?

DB: We are not expecting it to do everything, but it needs to provide the right primitives.

MS: I would like to give the programmer a semantics in which races between transactional and non-transactional code are always caught.

Transactions and the expert programmer

SD: What does TM offer the expert programmer?

RH & DB: There are complex pieces of code that we have written that would have been easier with transactions.

SC: We do not have a critical mass of evidence showing that TM outperforms than hand-crafted solutions by domain experts.

Key

DB: David Bacon
DW: David Wood
EM: Eliot Moss
JM: Jan-Willem Maessen
MH: Maurice Herlihy
MM: Milo Martin
MS: Michael Scott
RH: Rick Hudson
SC: Sid Chatterjee
SD: Sandhya Dwarkadas
SJ: Suresh Jagannathan
VS: Vijay Saraswat