Document Number: N3422=12-0112
Date: 2012-09-14
Authors: Michael Wong, Justin E. Gottschlich
Project: Programming Language C++, EWG, SG5 Transactional Memory
Reply to: Michael Wong <michaelw@ca.ibm.com>

SG5: Software Transactional Memory (TM) Status Report

1. Introduction

Herb Sutter officially formed “SG5: Transactional Memory” in May 2012, at the Advanced Parallelism Summit hosted at Microsoft, given the growing interest in transactional memory based on talks given by the C++ TM committee in Kona [1] and Bellevue [2] in February and May of 2012, respectively. Michael Wong was nominated and accepted the position of SG5 Chair. Justin Gottschlich was nominated and accepted the position of SG5 Vice Chair. SG5 currently has over 40 subscribed members from industry and academia on its reflector. A formal ISO Google discussion group was formed to support SG5 [4] know as the ISOCPP TM group. Michael Wong, Justin Gottschlich, and Tatiana Spheisman will act as co-editors of the draft document to be submitted to WG21.

The purpose of SG5 is for experts in both TM and C++ to collaborate to identify a set of constructs to be added as a Technical Specification (TS) for 2017. Although there are many possible designs, the group chose as a starting point to bridge on the work of a consortium of TM and C++ interested experts from various companies that have been meeting for the last four years and had produced the “Draft Specification of Transactional Language Constructs for C++” [3] (referred to as the *C++ TM Specification* for the remainder of this document) to build a set of language constructs that is acceptable to the C++ Programming Language and can eventually be added as part of the TS , for eventual ratification. It will also be the intention of SG5 to work closely with the WG14 to ensure a reasonable subset of TM is introduced to the C language as well.

On August 21, SG5 had its first tele-conference call. Since that time, the group has been convening every two weeks for roughly one hour and plan to continue this rigorous schedule, with roughly 8-10 people per call. Our conference calls are open to everyone and we have been encouraging feedback from new members.

For the October 2012 C++ Meeting in Portland, we will begin review of C++ TM Specification v.1.1 with focus on the immediate concerns through the ISOCPP TM group. We plan to work through each construct and discuss its appropriateness to Standard C++, as well as any additions others would like to introduce.

2. Progress Report

We have had three SG5 conference calls prior to the creation of this document. Our primary focus during these calls has been on discussion of the most basic concepts within the C++ TM Specification to challenge their value. Thus far, our conversations have centered on discussion about the two most basic types of transactions, *atomic transactions* and *relaxed transactions*.

The following bulleted list covers the core points of our discussions.

1. Is there a semantics difference between atomic and relaxed transactions?
2. Aside from cancel, what else makes atomic transactions different from relaxed transactions?
3. If cancel is removed, can atomic transactions be allowed to execute unsafe operations?

The answer to question 1 seems to be yes. In a data-race-free program, atomic transactions are strongly isolated, while relaxed transactions are not. That is, in certain circumstances it is possible for a partial state of an uncommitted relaxed transaction can be viewed by other threads.

The answer to question 2 is atomic transactions, by definition, may only include safe operations, while relaxed transactions can contain both safe and unsafe operations. This means that atomic transactions cannot deadlock given the safety constraints, whereas relaxed transactions can.

The answer to question 3 seems to be no, as it would violate the current definition of atomic transactions.

At the moment, the group is also working on several proposals to offer degrees of safety.

2.1 Non-Transactional Synchronization

Another difference between atomic and relaxed transactions, as pointed out by SG5, is the way in which they would handle non-transactional synchronization embedded within them. Although this behavior is not yet specified for atomic transactions, if it were, it would require that the atomic transaction remain strongly isolated in a data-race-free program. Therefore, this behavior would be different than that of a relaxed transaction, which does not require strong isolation in a data-race-free program.

2.2 Alterative proposals

In an effort to understand the need for atomics transactions, the discussion has revolved around their differences with relaxed, their costs or overhead, and any performance differences. The costs could be described as Language costs, Library costs or Runtime costs. Language cost is a difficult to quantify deleterious effect of adding excessive constructs to C++. Library cost is the cost of specifying safety attributes on all functions within Atomic Transactions. These safety attributes can be viral in the sense that they require all caller/callee to be also attributed. In return, they offer early compile time messages. Runtime cost is related to the Performance aspects.

A few member have offered proposals to weaken the requirement of requiring safety attributes on everything inside an atomic transactions. In the case where a transaction call unsafe functions but still in a safe way but only perform unsafe actions, say along an error path, then some members have proposed a more dynamic checking proposal that reduces the onerous tagging of safety attributes everywhere.

We hope the Portland meeting will allow this discussion to take us closer to closure, as many of the TM researchers plan to be at the meeting to discuss this with the C++ experts. We hope C++ experts can be at SG5 to render their opinion on which direction to go with respect to the kind of transactions.

3. References

Past C++ Presentations
[1] Kona, HA, February 2012 (http://justingottschlich.com/wp-content/uploads/2012/02/tm_csm_final.pptx)
[2] Bellevue, WA, May 2012 (http://justingottschlich.com/wp-content/uploads/2012/05/2012.SG1_.tm_final.pdf)

[3] Draft Specification of Transactional Language Constructs for C++ v.1.1

[4] SG5 Google Group (https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!forum/tm)