Document Number: P0265R0
Date: 2016-02-15
Authors: Michael Wong, fraggamuffin@gmail.com
with other members of the transactional memory study group (SG5), including:
Hans Boehm, boehm@acm.org
Brett Hall, bretthall@fastmail.fm
Victor Luchangco, victor.luchangco@oracle.com
Jens Maurer, jens.maurer@gmx.net
Maged Michael, maged.michael@acm.org
Torvald Riegel, triegel@redhat.com
Michael Scott, scott@cs.rochester.edu
Tatiana Shpeisman, tatiana.shpeisman@intel.com
Michael Spear, spear@cse.lehigh.edu
Project: Programming Language C++, SG5 Transactional Memory,
Reply to: Michael Wong, fraggamuffin@gmail.com (Chair of SG5)

SG5 is NOT proposing Transactional Memory for C++17

1. Introduction

SG5 feels that the Transactional Memory Technical Specification should not be standardized yet because there needs to be more user experience. The principal purpose of a TS is to gather implementation experience and usage, which can happen only when the implementation has had enough time for users to use it and to give feedback. This has not happened yet.

Technical Specifications introduce experimental syntax and semantics that seem promising but whose value has not yet been demonstrated at a level that justifies inclusion in the Standard. Although we are optimistic about the utility and importance of transactional memory in C++, this optimism should be confirmed (or contradicted) by experience from the field before the features should be standardized.

While TM has been included in many languages, the only significant usage in the field (aside from Haskell, which is not performance-oriented) is from Wyatt Technology (see next section on Status). While that implementation uses C++, it is not specific to our TS.

We have had a few requests from the field expressing interest in trying to use transactional memory support in C++ for financial, low-latency high-performance computing, and games: in the wake of talks at CERN, Barcelona SuperComputing Centre, Bloomberg, and Activision/Blizzard Entertainment, all have expressed interest. But experimentation in the context of a TS is entirely appropriate for these groups, as long as the TS is implemented and available in standard tool chains.

An earlier draft of TM has been available since GCC 4.7 with essentially the same semantics but slightly different syntax. When GCC 6 is released with a software implementation of TM TS in the first half of 2016 (possibly to be followed soon by an implementation in Clang or other commercial compilers), we will have a valid implementation of the TS. This is of course as one would expect: a TS motivates implementation even before standardization.

When there are field reports based on experience with these compilers, then we can consider adding some of all of the TM TS into the C++ Standard—but not before.

2. Current Status of Transactional Memory

The current status of the TM TS is that it has been published.

- N4513 is the official working draft
  http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4513.pdf

- N4514 is the published PDTS:
  http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4514.pdf

- N4515 is the Editor's report:
  http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4514.html

- Github is where the latest repository resides (updated to reflect the PDTS draft published post-Lenexa):
  https://github.com/cplusplus/transactional-memory-ts

The work of SG5, and the TM TS in particular, build on an earlier draft design created by an informal industry consortium:
https://gcc.gnu.org/wiki/TransactionalMemory

The following blog indicates that substantial functionality, based on that earlier design and similar to the TM TS is already included in GCC 4.7:
http://developerblog.redhat.com/tag/transactional-memory/

An implementation of the TM TS will be available in GCC 6.0, expected to be out in a few months, except for support of atomic_cancel and a few minor details.

Clang support will also likely start soon.

One interesting usage experience is from Wyatt technologies:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4438.pdf

While the TM TS is agnostic with respect to hardware support, it is designed to enable the use of that support on machines where it is available. Currently, such machines include mainstream commercial offerings from IBM and Intel. And while we have no inside information from these vendors, public indications are that they intend to continue TM support in future hardware generations.