

# **Transactional Memory Support in the GCC Open-Source Compiler Framework**

<<names to be added>>

# Motivation

---

## ❑ Transactional Memory (TM)

- A promising technology to simplify parallel programming
- User specifies atomic blocks; system manages concurrency
- Performance of fine-grain locks; simplicity of coarse-grain locks

## ❑ Current state of TM research

- TM programming using library-based APIs
  - Error prone; difficult to port and maintain applications
  - Reduces effectiveness of compiler optimizations
- Compiler optimizations for TM
  - Multiple research groups reimplementing the same functionality
  - No sharing of infrastructure
- Obstacles for feature research and widespread adoption
  - TM not integrated with common languages
  - TM not integrated with popular programming tools
  - No common applications to motivate comparisons
  - Difficult to use TM for applications beyond concurrency

# This Proposal

---

## □ Goal: introduce a basic TM support in GCC

- Provide basic TM features for C and C++
- Provide basic TM optimizations within the compiler
- Provide interoperability between applications and TM systems

## □ Not goals

- Do research on TM implementations (HW, SW, or hybrid)
- Do research on high level programming models
- This is an infrastructure proposal that will enable such research...

## □ Why GCC

- Widely used programming framework
- Open source

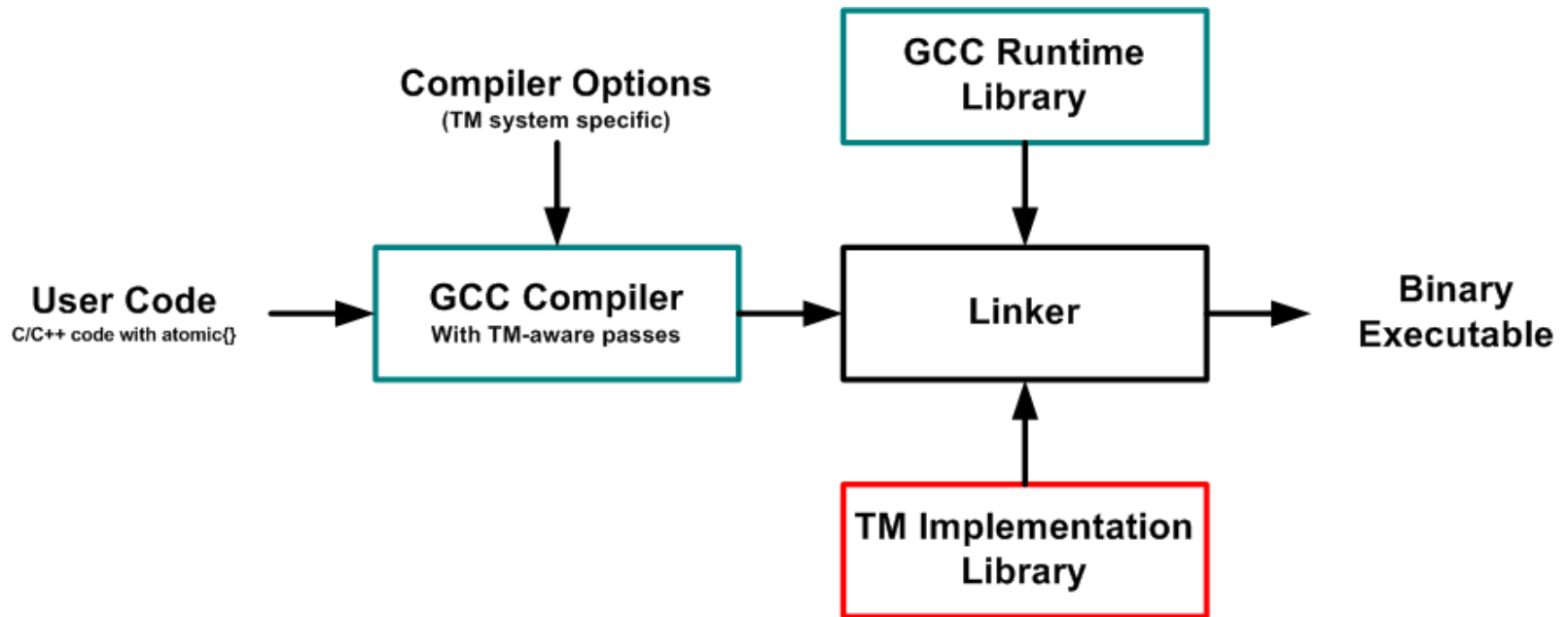
# Statement of Work

---

- ❑ Define simple language and IR constructs for TM
  - `atomic{}` construct in GCC front-end for C/C++
  - A set of new IRs to express the attributes of atomic blocks
- ❑ Define a simple memory model for transactions
  - E.g., weak consistency model of OpenMP
- ❑ Define general interface to TM implementations
  - Ensure GCC-generated code can be linked to wide range of TMs
  - Software, hardware, or hybrid TM
- ❑ Compiler pass for TM code generation
  - Inserts TM barriers, clones TM functions (`#tm_function`)
- ❑ Compiler passes for optimizations
  - Redundant barrier elimination, partial inlining of TM barriers, etc.
- ❑ Code release to the community

# TM Support in GCC

---



## □ This effort:

- GCC front-end changes and TM-aware optimizations
- Test with some of the available TM implementations

## □ Users provide

- Application code
- TM implementation (if they want to use their own)

# Requirements

---

## ❑ Manpower

- 2 development engineers for one year

## ❑ Collaborations

- TM research community (academia & industry)
- GCC community
- Parallel applications community

## ❑ Importance of collaboration

- Interfaces must be flexible enough to support future uses

# Potential Impact

---

- ❑ Widespread use of TM for parallel application development
  - Portable, high-level code that can be easily shared
  - Practical evaluation of TM's programmability benefits
  
- ❑ Practical evaluation of various TM implementations
  - Using same code and robust compilation framework
  - Will spark further research on the topic
  
- ❑ Uses of TM beyond concurrency control
  - GCC will provide access to TM as a speculation mechanism
  - Will spark further uses for security, debugging, fault-tolerance, thread-level speculation, etc.