

Transactional Memory Support in the GCC

Martin Schindewolf¹, Albert Cohen², and Wolfgang Karl¹

¹Universität Karlsruhe (TH),
Institut für Technische Informatik
Lehrstuhl für Rechnerarchitektur

²INRIA Saclay

June 2nd, 2008



Our group

- Part of Institute of Computer Science & Engineering, Universität Karlsruhe (TH)
- Prof. Dr. Wolfgang Karl
- 1 PostDoc: Dr. Rainer Buchty, 4 PhDs

Research Interests

- *Adaptive Systems*: Reconfigurable Computing
- *Tools for Multi-Cores*: Performance Evaluation, Data Locality Optimization
- Digital On-demand Computing Organism for Real-time Systems (*DodOrg*): Self-X, Runtime Systems, Monitoring
- Transactional Memory

- Motivation
- Goals
- Roadmap
- Preliminary Work
 - Example for use of new pragma
 - GCC's IR of example program
- Compatibility/Portability
- STM issues



<http://www.hipeac.net/>

- TM promising technology
 - Simplifies parallel programming
 - Supports high level abstraction
 - Provides reasonable performance
 - STMs already available
(McRT, TL2, Rochester, tinySTM, ...)
 - HTMs and Hybrid systems emerging
- Few Compiler frameworks available
- No stable open source compiler

- Reach application developers
 - have **real** applications
- Basis for further research
(compiler optimisations, Thread Level Speculation, high-level parallel languages, ...)
- Robust, stable implementation
 - Orthogonal to OpenMP
 - Support exception handling
 - Basic features (closed nesting)
- Release implementation as a development branch of GCC

- 1 Atomic blocks with flat or closed nesting
- 2 Weak isolation (considering single-lock semantics)
- 3 Attributes: `tm_callable`, `tm_pure`
- 4 Irrevocable function calls
- 5 Rollback and commit handlers
- 6 Failure atomicity
- 7 C++: classes, inheritance, templates

Preliminary Work (1) Example

Extension of the C front-end with `_Pragma("tm atomic")`

```
int n = 0;
int main() {
    _Pragma("tm atomic")
    {
        int i = 2;
        n += i;
    }
    return n;
}
```

Preliminary Work (1) GCC's IR

```
;; Function main (main)
main ()
{
  int D.1209;

  _Pragma("tm atomic")
  {
    {
      int i;
      i = 2;
      n.0 = n;
      n.1 = n.0 + i;
      n = n.1;
    }
    GTM_RETURN
  }
  D.1209 = n;
  return D.1209;
}
```

Intel

- Support established attributes
 - `tm_callable`
 - `tm_pure`
 - `tm_unknown`
- `__tm_atomic`-keyword supported
 - Supply file that maps keyword to `_Pragma("tm atomic")`
 - Drawback: enforces C99 compliance of the code
- Ongoing efforts:
 - Have common syntax to call TM-functions
 - Expose `abort`-calls to the programmer

tinySTM

- Flat nesting

wanted STM features

- Closed nesting
- Irrevocable function calls
- Quiescence (→ overhead)
- Failure atomicity
- Open nesting?
- `set jmp / long jmp?`

Transactional Memory Support in the GCC

Martin Schindewolf¹, Albert Cohen², and Wolfgang Karl¹

¹Universität Karlsruhe (TH),
Institut für Technische Informatik
Lehrstuhl für Rechnerarchitektur

²INRIA Saclay

June 2nd, 2008

