Providing Transparent Transactions on C/C++ Memory Locations with TARIFA

Martin Süsskraut and Ulrich Müller
{ms67, um766083}@inf.tu-dresden.de

Systems Engineering Group
Technische Universität Dresden
http://wwwse.inf.tu-dresden.de/

High Assurance Systems Engineering Conference 2005
Retrys

- systems under high load suffer from resource depletion
- lead to transient failures
- best practice: retry

Our Suggestion

Use transactions – they provide atomicity.
- automatic integration of retrys
- recovery blocks
Thread Synchronization

- trade off: granularity of mutexes
- choosing the right mutex for guarding could be a painful process

Our Suggestion

Use transactions – they provide isolation.
Motivation
Software Transactional Memory

Transactions

- well known from data bases
- provide Atomicity, Consistency, Isolation, and Durability (ACID)

Java

- successful language integration [HF03]
- can be faster than well engineered mutexes

C/C++

- Software Transactional Memory (STM) libraries [Fra03] exist
- but no language integration
How to embed a STM into C/C++?

- no embedding at all: using native language library
- extend a C/C++ compiler
- dynamically at runtime (using an interpreter)
- instrument the compiler output

```
1 atomic {
2     a += b;
3 }
```

```
1 t = STMStart ( );
2 STMWrite(t, &a, STMRead(t, &b) + STMRead(t, &a));
3 STMCommit(t);
```
Our Approach

On assembler level

- treat every assembler operation independently
- put every memory access under control of a STM library
Conclusion

Pros:
- nearly no overhead compared with direct use of an STM library
- does not depend on a certain STM library
- can be used:
  - automate retrans
  - substitute mutexes

Challenges:
- calls to uninstrumented functions
- portability
Keir Fraser.
Practical lock freedom.

Tim Harris and Keir Fraser.
Language support for lightweight transactions.