THE VELOX STACK

Patrick Marlier (UniNE)

UNIVERSITÉ DE NEUCHÂTEL

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

TECHNISCHE UNIVERSITÄT DRESDEN

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

TEL AVIV UNIVERSITY

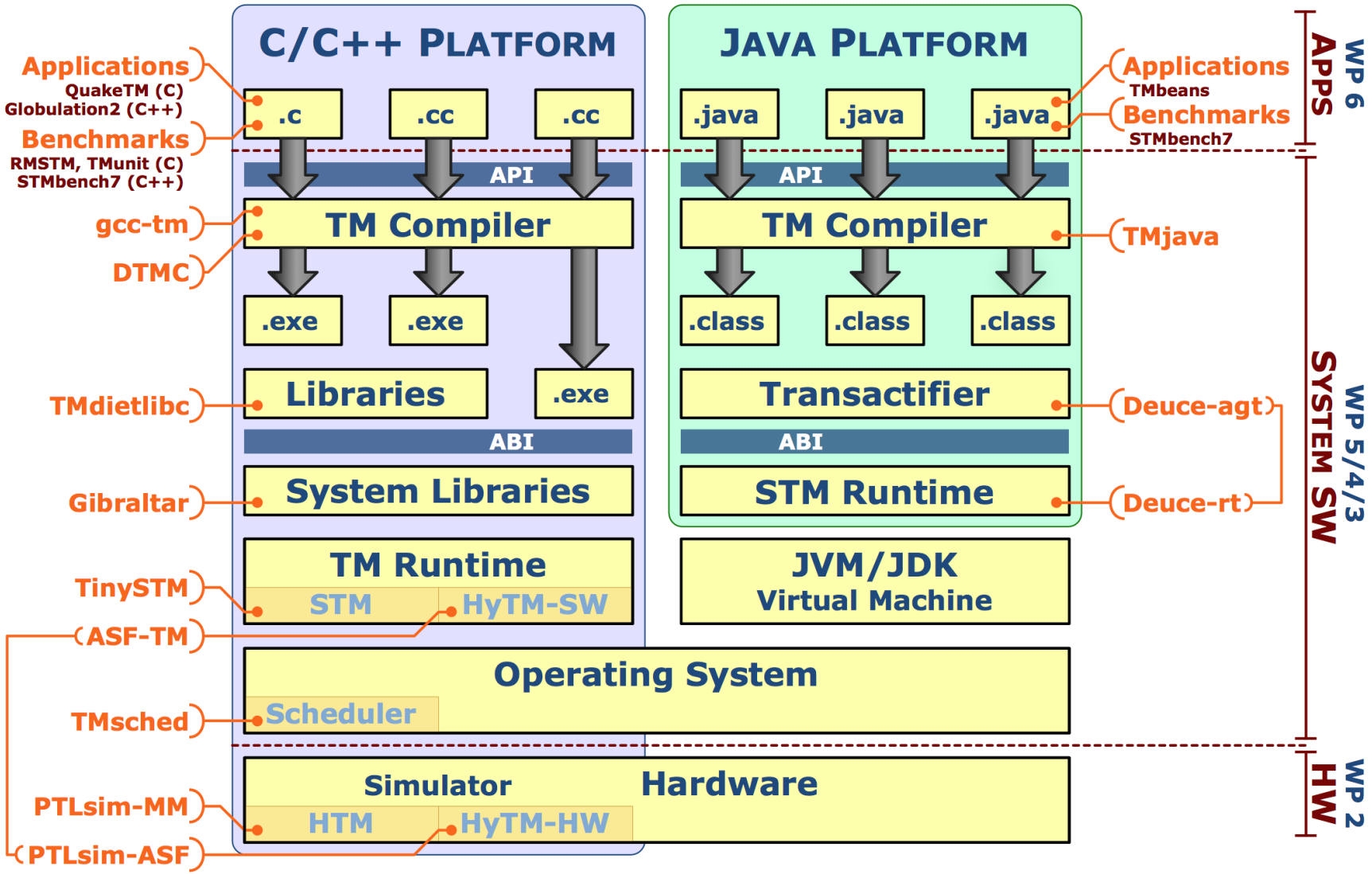
redhat.

AMD Smarter Choice

CHALMERS

UNIVERSITÉ DE NEUCHÂTEL

THE VELOX STACK / OUTLINE



APPLICATIONS

Real applications

- QuakeTM
  - Game server
  - C code with OpenMP
- Globulation 2
  - Real-Time Strategy Game
  - C++ code using STL and Boost library
- RMS-TM
  - Recognition, Mining and Synthesis domain applications
  - C and C++ code

➢ A compiler with TM is required

Benchmarks

- TM-Unit
  - Unit testing of TM library: observe semantic, correctness, performance
- STMBench7
  - C++ code
  - Workloads with different operations on a set of graphs, indexes

TRANSACTIONAL MEMORY API

Based on Draft Specification of Transactional Language Constructs for C++

- Transaction
  - __transaction keyword
- Transaction attributes
  - Transaction (default): __transaction [[atomic]]
  - Transaction with possible irrevocable statement: __transaction [[relaxed]]
- Function attributes
  - undoable function: transaction_safe attribute
  - irrevocable function: transaction_unsafe attribute
  - undoable function with possible irrevocability: transaction_callable attribute
  - uninstrumented function: transaction_pure attribute
  - Wrapped function: transaction_wrap attribute
- Extra statements
  - __transaction_cancel
  - __transaction_abort

C/C++ COMPILERS WITH TRANSACTIONAL MEMORY

unine
UNIVERSITÉ DE NEUCHÂTEL

DTMC

- Using LLVM and LLVM-GCC
- LLVM front-end (LLVM-GCC) with TM-API support
- Transactification using IR instrumentation in LLVM
  - Can generate multiple code paths using different instrumentation or none
  - Can detect aligned memory accesses
  - Link Time Optimization even for transactional accesses
- Research oriented for fast prototyping

GCC

- Fully integrated in GCC : -fgnu-tm
- Complex instrumentation with «after read», «after write», «for write» barriers
- Should be integrated in GCC 4.7
- It is still active so don’t hesitate to contribute, it can make TM mainstream.

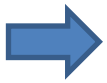
All compilers are API and ABI compatible.

VELOX project

TRANSACTIONAL MEMORY ABI

Based on Transactional Memory Application Binary Interface (TM ABI) Specification 1.0

```
extern long cntr; void increment() {
  __transaction { cntr = cntr + 5; }
}
```



```
extern long cntr;
void increment() {
  _ITM_beginTransaction(...);
  long l_cntr = (long) _ITM_RU8(&cntr);
  l_cntr = l_cntr + 5;
  _ITM_WU8(&cntr, l_cntr);
  _ITM_commitTransaction();
}
```

Code transformation (internal)

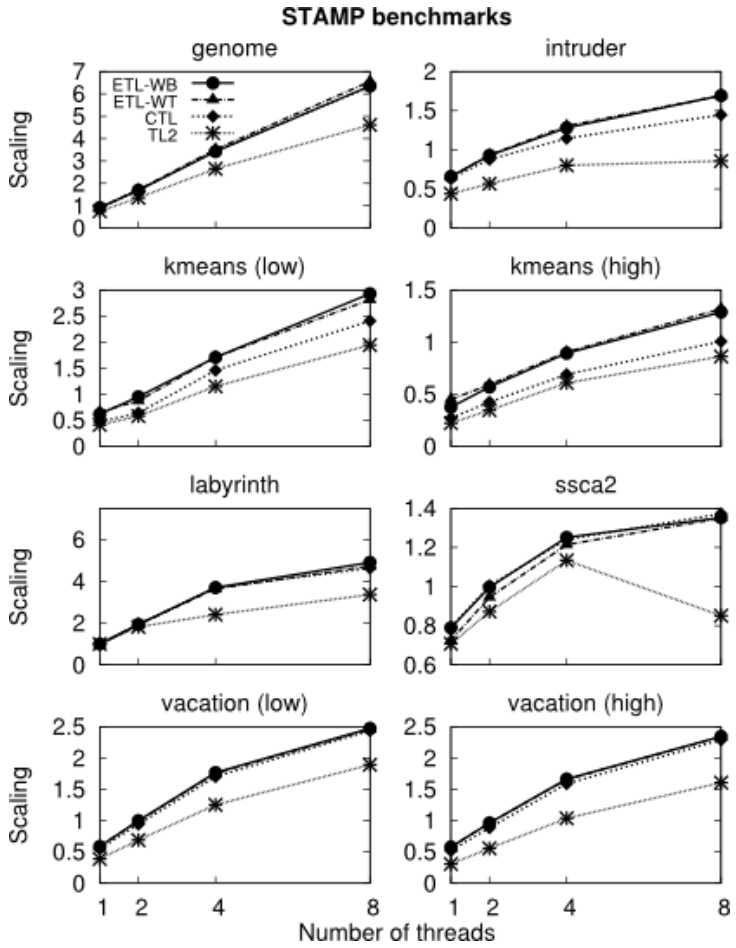
I want to	Transformed to ABI call
Start a transaction	_ITM_beginTransaction()
Read an unsigned int at address X	_ITM_RU4(X)
Write a long value V at address Y	_ITM_WU8(Y, V)
Switch to irrevocable mode	_ITM_changeTransactionMode()
Commit a transaction	_ITM_commitTransaction()

Standardized interface in order to plug any TM library that support TM-ABI

TRANSACTIONAL MEMORY LIBRARY

TinySTM

- Timestamp and lock based STM
- Several flavors with various trade-offs
  - Write-back or write-through
  - Encounter- or commit-time locking
  - Visible reads
  - Serial and concurrent irrevocability
- ABI compatible
- Transaction-safe memory allocator
- Advanced Contention Management
  - Kill transaction and lock stealing
- Extensible via external “modules”



AMD Advanced Synchronization Facilities (ASF)

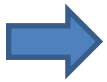
- Extension of the x86_64 ISA
- Designed to be implementable in high-volume microprocessors
- Near cycle-accurate simulator
- Early release instruction, monitor instructions, …

Example of Double Compare And Swap using ASF

```
DCAS:
 MOV R8, RAX
 MOV R9, RBX
retry:
 SPECULATE                  ← Start speculative region
 JNZ retry
 MOV RCX, 1                 ← RCX indicates if DCAS fails or succeeds
 LOCK MOV R10, [mem1]       ← Load mem1
 LOCK MOV RBX, [mem2]       ← Load mem2
 CMP R8, R10                ← Compare mem1 with old value (R8)
 JNZ out                    ← If not equal leave
 CMP R9, RBX                ← Compare mem2 with old value (R9)
 JNZ out                    ← If not equal leave
 LOCK MOV [mem1], RDI       ← Store new value in mem1
 LOCK MOV [mem2], RSI       ← Store new value in mem2
 XOR RCX, RCX               ← Indicates DCAS success
out:
 COMMIT                     ← Commit speculative region
 MOV RAX, R10               ← (restore RAX)
```

Hardware Transactional Memory using pure ASF implementation (with LTO)

```
extern long cntr;
void increment() {
  _ITM_beginTransaction(...);
  long l_cntr = (long) _ITM_RU8(&cntr);
  l_cntr = l_cntr + 5;
  _ITM_WU8(&cntr, l_cntr);
  _ITM_commitTransaction();
}
```

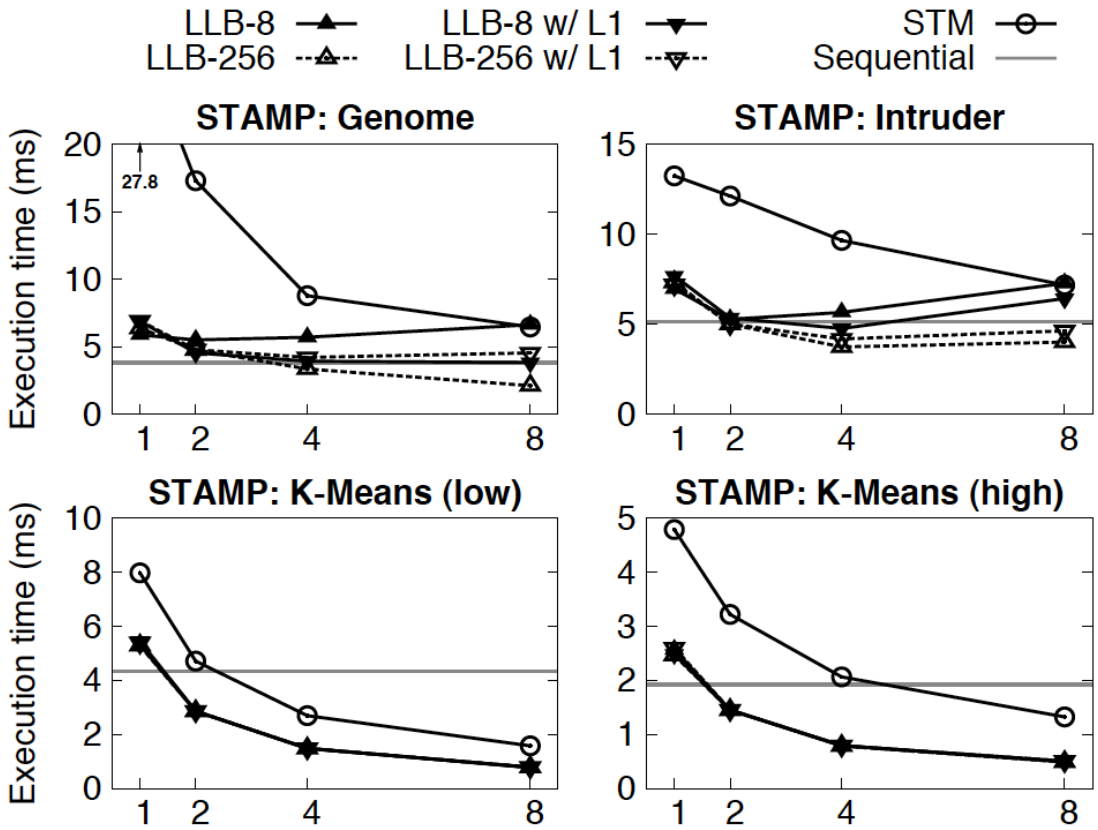


```
SPECULATE
JNZ         retry_or_switch_to_STM
LOCK MOV    RCX, cntr
ADD         RCX, 5
LOCK MOV    cntr, RCX
COMMIT
```

Hybrid Transactional Memory using AMD ASF

- Extension of TinySTM
- Takes advantage that ASF allows the use of non speculative accesses in speculative region
- STM and HTM transactions can execute concurrently
- Overcome HTM limitations

AMD Advanced Synchronization Facilities (pure ASF)



AND A LOT MORE

TM-Dietlibc: TM aware libraries
- Libc functions compatible with transactions

TM-Sched: TM aware kernel
- Time slice extension, transaction scheduling, soft real-time support

Gibraltar: Dynamic instrumentation of binary code
- No need to re-compile libraries and transform legacy code to transactional code

PTLSIM-MM: HTM without need of explicit instrumentation

BeeFarm: FPGA-based multiprocessor system-on-chip with HTM

But also in the **Java side:**
- **STMBench7:** Benchmark for TM
- **TM-Java:** Language extension for TM support
- **Deuce:** JAVA STM run-time and instrumentation framework
- Several STM library: **LSA**, TL2

THANK YOU

UNIVERSITÉ DE NEUCHÂTEL

Questions ?

http://www.velox-project.eu/