Robust Transactions

Multicore System Model

Christof Fetzer, TU Dresden
OUR GOAL:
SUPPORT LARGE PROGRAMS

CODE

CONCURRENT PROGRAM

SINGLE ADDRESS SPACE
NOT ALL PROGRAMMERS ARE EQUAL...

CODE

CONCURRENT PROGRAM

CODE IS NEVER COMPLETELY CORRECT
FAILURE MODEL

TRANSACTION CRASHES
NEVER-ENDING TRANSACTIONS
PROCESS CRASHES
PERFORMANCE FAILURES
STARVED TRANSACTIONS

USER CODE RELATED
SCHEDULING RELATED
TRANSACTION CRASH

ATOMIC {
    P = MALLOC(100);
    IF (!P) {
        *P = 1234;
    }
    ...
}

“CRASHES" TRANSACTION IF MALLOC FAILED
ETERNAL TRANSACTIONS

NEVER-ENDING TRANSACTION: NEVER TRIES TO COMMIT

ATOMIC {
    WHILE (TRUE)
    ;
}

WILL NEVER TERMINATE
Multicore System Model

(MSM)
SINGLE ADDRESS SPACE

STATE: ADDRESSES -> WORD

Both addresses and words are bounded!
PROCESSES

APPLICATION

A.K.A. NATIVE THREADS

SINGLE ADDRESS SPACE

N PROCESSES

OPERATING SYSTEM

$2^{64} - 1$

$2^0 - 1$
PROCESS CRASH FAILURES

\[ 2^{64}-1 \]

SINGLE ADDRESS SPACE

\[ 2^{32}-1 \]

N PROCESSES

APPLICATION

PROCESS CRASH

OPERATING SYSTEM

KILL - TERM (INATE)
PROCESS PERFORMANCE FAILURES

SINGLE ADDRESS SPACE

OPERATING SYSTEM

APPLICATION SCHEDULER CAN DELAY PROCESSES

SCHEDULER

N PROCESSES

DELAY

$2^{64} - 1$

$2^{20} - 1$
STARVED TRANSACTION

TM EXECUTES

PROCESS CRASHED OR DELAYED?

2^{0-1}

DELAY

N PROCESSES

SCHEDULER

OPERATING SYSTEM
ASYNCHRONOUS SYSTEM MODEL

- Finite, non-zero speed of non-crashed processes
- Relative speed of processes is unbounded
- Processes can crash and no crash detector
- TM cannot decide if a process is just slow, or crashed
Robust STM implementable?
(in MSM)
THREADS

K (GREEN) THREADS

GREEN THREADS MAPPED ON PROCESSES BY TM IMPLEMENTATION

N PROCESSES
THREAD = SEQUENCE OF TRANSACTIONS

THREAD T1:
FOREVER {
    ATOMIC { ... }; // A1
    IF (EXPR) ATOMIC { ... }; // A2
    ELSE ATOMIC {...} // A3
}

USER CODE

TRANSACTION
T1_A1_1

TRANSACTION
T1_A2_2

TRANSACTION
T1_A1_3

TRANSACTION
T1_A3_4

TRANSACTION
T1_A1_5
FAILURE MODEL

LOOK AT THIS FIRST!

- Transaction crashes
- Never-ending transactions
- Process crashes
- Performance failures

User code related
Scheduling related
USER CODE ISSUES

□ TRANSACTION CRASH

```c
ATOMIC {
    IF (!P) {
        *P = 1234;
    }
}
```

SIGNAL OR EXCEPTION THROWN!

COMMIT OR ABORT ON EXCEPTION
FAILURE MODEL

- Transaction Crashes
- Never-ending Transactions
- Process Crashes
- Performance Failures

LOOK AT THESE TWO NEXT!
Universal TM construction
to ensure total progress
BOUNDDED MEMORY

JUST KEEP THE LAST ELEMENT IN LIST!

STATE THREAD

\[ S_3 = F_{T_3_1}(S_2) \]

STATE THREAD

T2

S2

T2

S3

T3

S3

T3

HEAD

CAS
FAILURE MODEL

THE REAL PROBLEM!

- TRANSACTION CRASHES
- NEVER-ENDING TRANSACTIONS
- PROCESS CRASHES
- PERFORMANCE FAILURES

USER CODE RELATED
SCHEDULING RELATED
NEVER-ENDING TRANSACTIONS

HEAD → STATE

$S_1$

$S_2 = F_{T_{2,1}}(S_1)$

ALL THREADS GET STUCK!
BOUNDED NUMBER OF STEPS

\[
S_3' = F_{T_{3-1}}(S_2, TH_3)
\]

STATE THREAD

\[
S_2 \\
T_2
\]

HEAD

CAS

STATE THREAD IMS

\[
S_2 \\
T_3 \\
S_3'
\]

STORE INTERMEDIATE STATE IF NEEDED

AT MOST TH_3 STEPS
NEXT ROUND: CONTINUE

STATE THREAD IMS

S2  T2  S3'

CAS

F_{T3-1}(S3', 1)

ONE MORE STEP

STATE THREAD TH3

S3  T3  NT

STORE NEW THRESHOLD ON TERMINATION
FAILURE MODEL

AS LONG AS MEMORY BOUNDED

- TRANSACTION CRASHES
- NEVER-ENDING TRANSACTIONS
- PROCESS CRASHES
- PERFORMANCE FAILURES

USER CODE RELATED

SCHEDULING RELATED
CONCLUSION

- MSM IS AN ASYNCHRONOUS MODEL, I.E.,
  - NO BOUNDS ON THE ABSOLUTE OR RELATIVE SPEED OF PROCESSES
  - NO PROCESS CRASH DETECTOR
  - BOUNDED MEMORY
  - ONE CAN IMPLEMENT
    - ROBUST STM THAT TAKES CARE OF PROCESS CRASHES, TRANSACTION CRASHES AND NEVER-EDNING TRANSACTIONS AND AVOIDS ZOMBIE TRANSACTIONS