DPDK
SUMMIT
NORTH AMERICA

Herd7 - Tool for memory model simulation

HONNAPPA NAGARAHALLI
ARM

JULY 12-13, 2021

Agenda

DPDK SUMMIT NORTH AMERICA

- Why do we need a tool?
- What to expect?
- Herd7
- Memory model simulation with Herd7
- Conclusion

Why do we need a tool?

- Consider a simple thread synchronization problem – Message Passing
  - Writer thread – Update a data structure in shared memory
  - Reader thread – Read the data structure from shared memory
  - Writer uses a guard variable to communicate the readiness of data

```
/* Global variables */
struct payload_s payload;
int guard = 0;

/* Writer */
/* Fill data */
payload.a = 10;
payload.b = 20;

/* Indicate payload is ready */
atomic_store_explicit(&guard, 1, memory_order_relaxed);
```

A

B

```
/* Reader */
/* Read the guard variable */
if (atomic_load_explicit(&guard, memory_order_relaxed) =
        /* payload is set by the writer, read payload */
        have_fun(payload.a, payload.b);
        ……
}
```

C

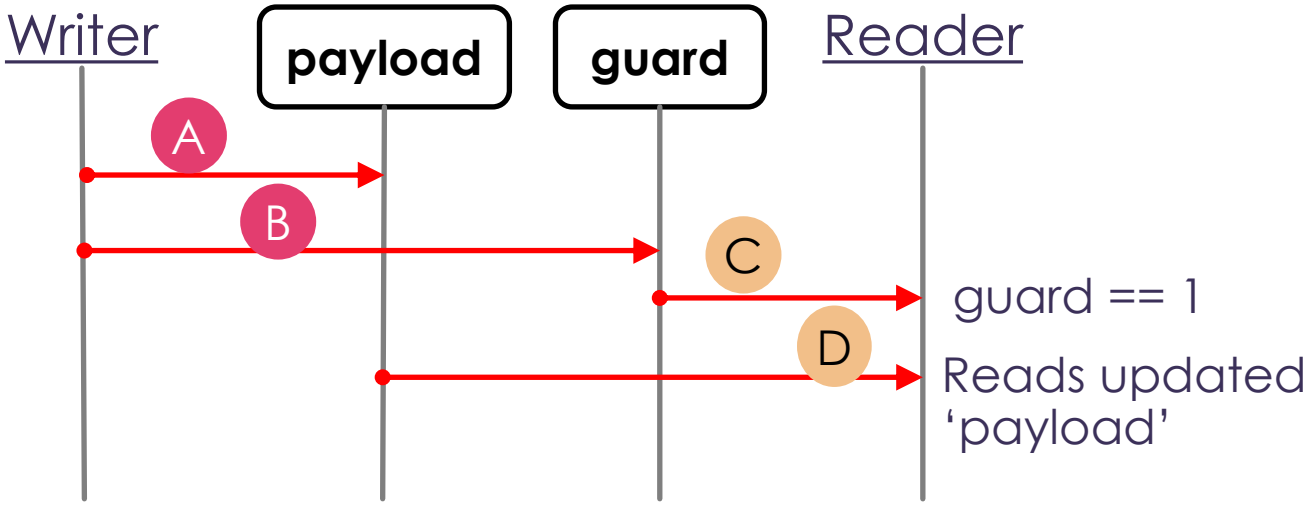
D

2 events in Writer, A & B

2 events in Reader, C & D

Why do we need a tool?

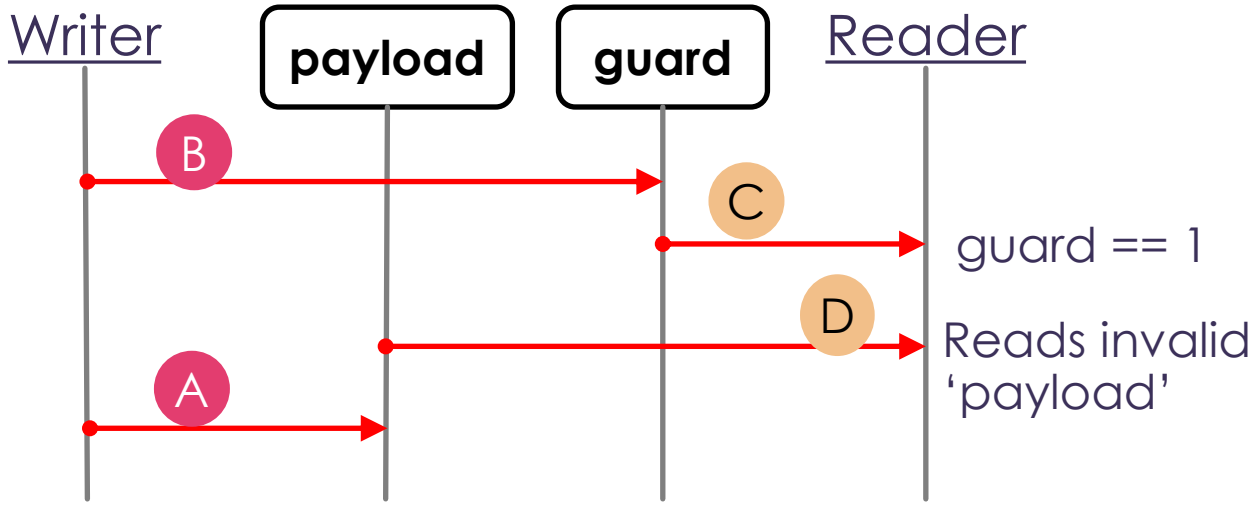
- In a machine with relaxed memory model, events can occur in any order
- Consider one such sequence ABCD



- No barriers required IF this sequence is executed

Why do we need a tool?

- Consider another sequence BCDA



- This shows that 'A' and 'B' need to be execute in order, hence requiring barriers
- Similarly, if we consider sequence ADBC, 'C' and 'D' need to execute in order

Why do we need a tool?

- So, what is the problem?
  - With 4 events, we have 24 possible sequences of events
  - Even with correct memory barriers – we have 6 valid sequences (ACBD, ACDB …)
  - With 5 events – 120 sequences, 6 events – 720 sequences
  - It is not possible to write test cases to validate all these sequences
- Evolving CPU micro-architectures
  - CPU micro-architectures might implement a stronger memory model
  - Increasing performance requirements on CPUs results in more re-ordering
    - https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=84a24bf8c52e66b7ac89ada5e3cfbe72d65c1896
- Hence, we need a tool to validate the correctness of the algorithms

What to expect

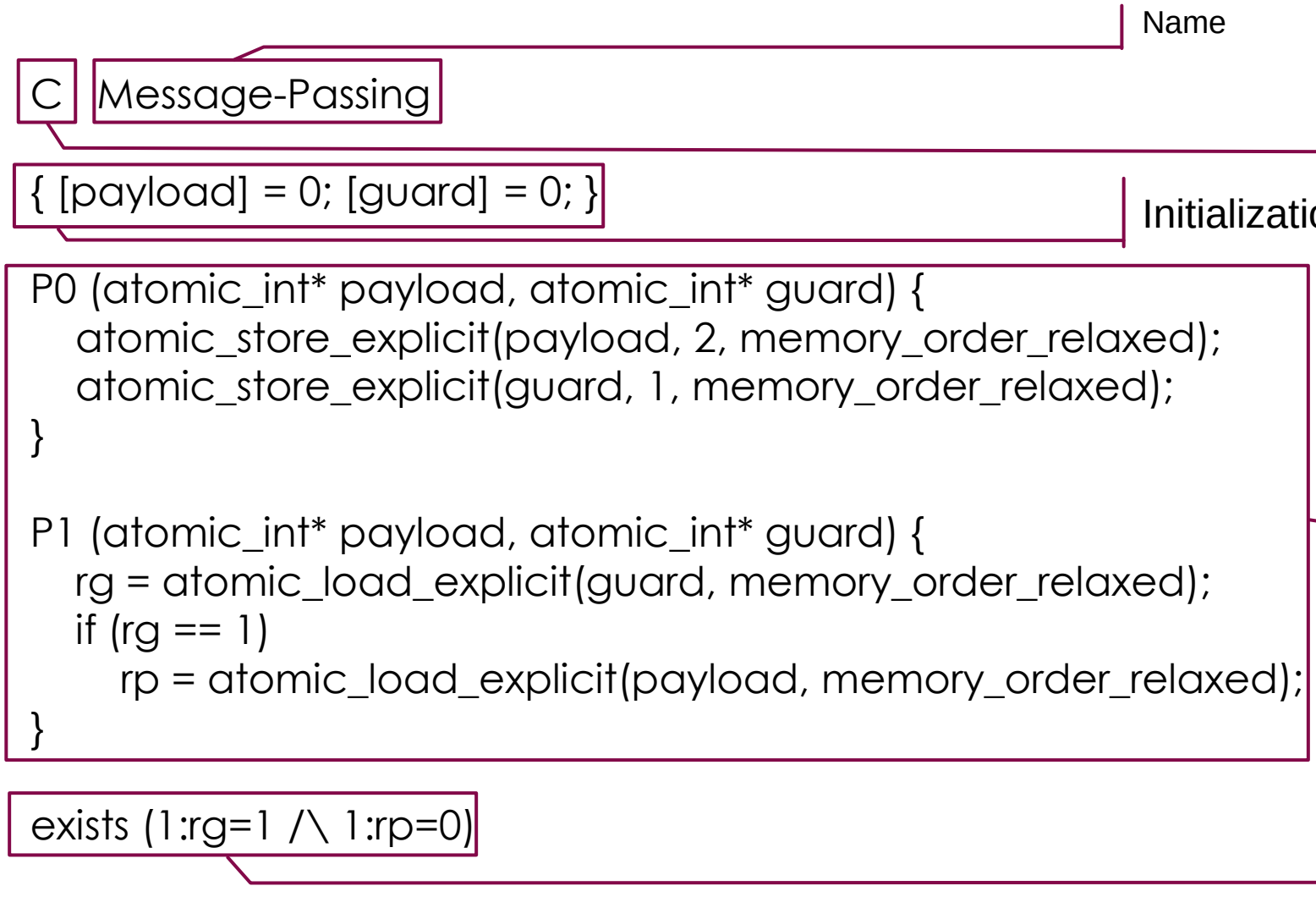
DPDK SUMMIT NORTH AMERICA

- The topic of Memory Model is complex
- I am a novice
- Goal
  - Introduce the tool
  - Reduce the barrier to get started
  - Simple examples, easy to understand, focus on the tool

Herd7 – General Info

- Part of herdtools7 Tool Chain
  - **herd7 – Memory model simulator**
  - diy7 / diycross7 / diyone7 – Litmus test generators
  - litmus7 / klitmus7 – Running the litmus tests on the actual hardware
- Tool page: http://diy.inria.fr/
- Documentation: http://diy.inria.fr/doc/herd.html
- Sources: https://github.com/herd/herdtools7
- Installing the tool on Linux
  - Cover in the backup slides
- Web Interface - http://diy.inria.fr/www/
  - C11 model will be enabled in the future

Herd7 – Representing the algorithm – Litmus test



Programming Architecture
- Supports only small subset of C11 language
- Included – 'if', 'while', pointer dereferencing, simple expressions, atomic functions
- Excluded – 'address-of', compound types, function calls
- Supports Aarch64, PPC, RISC-V, x86

Program Code
- P0, P1 – Threads
- payload, guard – Global variables (shared memory)
- rg, rp – Local variables in thread1

Condition to assert
- Check if (rg ==1 && rp == 0) exists

Herd7 – Analyzing the results

- herd7 -c11 mp.litmus

```
Test Message-Passing Allowed
States 4
1:rp=0; 1:rg=0;
1:rp=0; 1:rg=1;
1:rp=2; 1:rg=0;
1:rp=2; 1:rg=1;
Ok
Witnesses
Positive: 1 Negative: 3
Condition exists (1:rg=1 /\ 1:rp=0)
Observation Message-Passing Sometimes 1 3
Time Message-Passing 0.01
Hash=0d08978cd3e64ee878b665ee423d60df
```

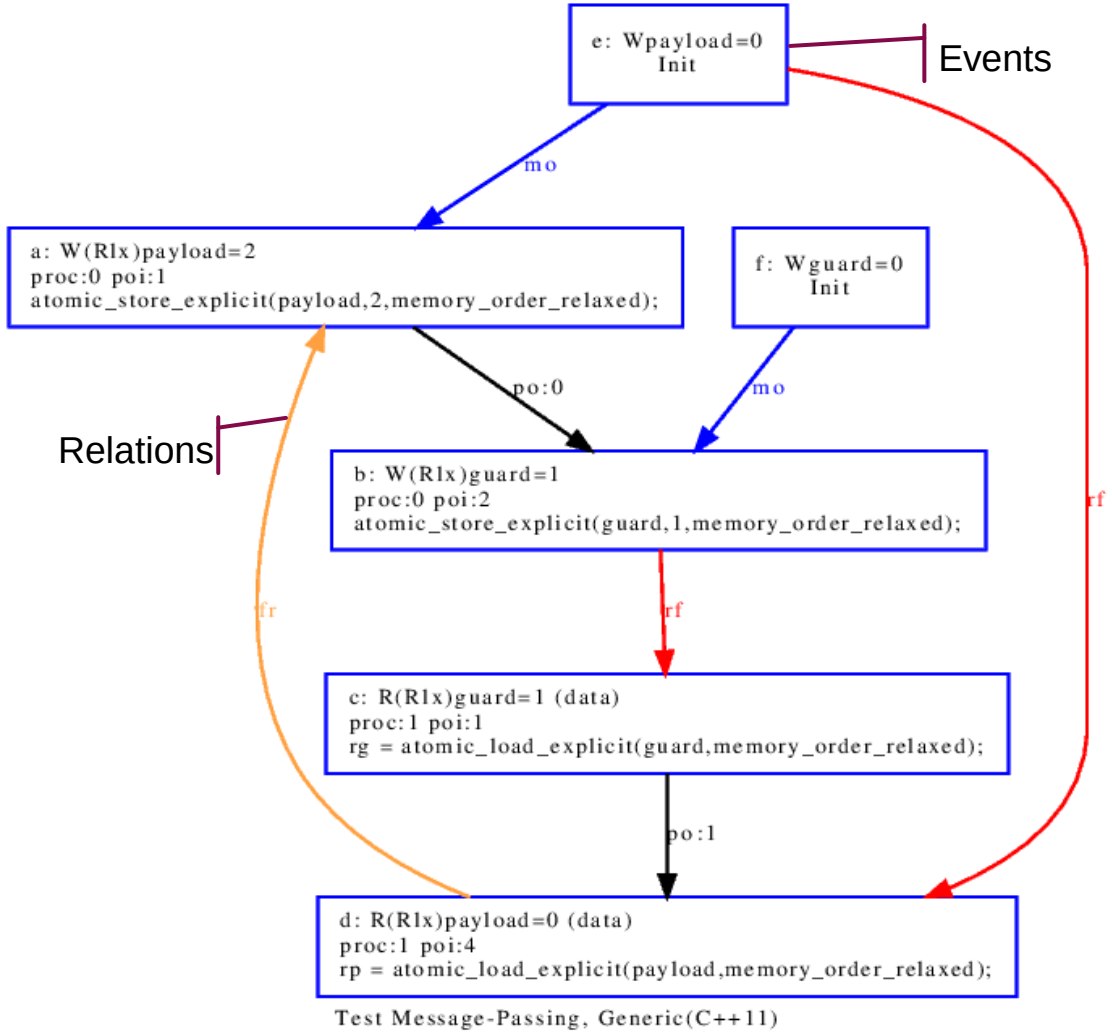
Shows the possible outcomes

Indicates condition exists

Indicates there is 1 sequence of events resulting in the condition.

Herd7 – Fixing the algorithm

- herd7 -c11 **-show prop -squished false -graph free** -evince mp.litmus



- Events
  - Wguard=0 : 0 is written to memory location ‘guard’
  - Rpayload=0 : 0 is read from memory location ‘payload’
  - proc:0 : Event in thread 0
- Relations
  - po: Program Order: Order of the events in the code
  - rf: Read From: Which write event, the read event reads from
  - mo: Modification Order: Ordered history of all writes to a memory location
  - fr: From Read: Indicates all the writes that are overwriting a value read by the read event
- Identifying the problem
  - Event ‘R(Rlx)payload=0’ reads from initialized value of 0
  - Event ‘W(Rlx)payload=2’ occurs after the read event
  - This sequence needs to be blocked

Herd7 – Fixing the algorithm

C Message-Passing

```
{ [payload] = 0; [guard] = 0; }

P0 (atomic_int* payload, atomic_int* guard) {
    atomic_store_explicit(payload, 2, memory_order_relaxed);
    atomic_store_explicit(guard, 1, memory_order_release);
}

P1 (atomic_int* payload, atomic_int* guard) {
    rg = atomic_load_explicit(guard, memory_order_acquire);
    if (rg == 1)
        rp = atomic_load_explicit(payload, memory_order_relaxed);
}

exists (1:rg=1 /\ 1:rp=0)
```

Test Message-Passing Allowed

States 2
1:rg=0; 1:rp=0;
1:rg=1; 1:rp=2;

No

Witnesses
Positive: 0 Negative: 2
Condition exists (1:rg=1 /\ 1:rp=0)
Observation Message-Passing Never 0 2

Time Message-Passing 0.00
Hash=a4450616bef403da2b0fc96f828429a2

Additional things

- Other tools
  - CDS Checker – http://plrg.eecs.uci.edu/software_page/42-2/
  - CPP Mem - http://svr-pes20-cppmem.cl.cam.ac.uk/cppmem/index.html
- Herdtools7
  - Supports architecture specific memory models
  - Does more exhaustive search for the re-ordering candidates (per available literature)
- Validate the DPDK algorithms using any of these tools

Installing Herdtools7

- Environment used
  - Ubuntu 20.04.2
  - 5.4.0-77-generic
- Steps
  - Install Opam – **apt-get install opam**
  - Configure Opam – **opam init**
  - Install herdtools7 suite – **opam install herdtools7**
  - PDF viewer evince – Packaged in Ubuntu

Questions?

DPDK

SUMMIT

NORTH AMERICA