Arm’s Effort For DPDK and Future Plan

GAVIN HU
HONNAPPA NAGARAHALLI

ARM
• Relaxed memory ordering
• __sync vs. __atomic
• WFE and SEV for v8.0
• Atomic instructions for V8.1
• Neon Optimization
• Build system and Documentation
• DTS internal CI
C11 memory ordering

- Aarch64 is relaxed memory ordering
- Program order might be broken for performance optimization
  - By HW
  - By compiler
- This memory reordering is transparent to programmers most of the time
- Still there are use cases the program order should be kept, especially for multi-core/thread execution environments.
  - That is why memory fences are introduced
- Memory fences degrade performance
  - Use less restrictive memory ordering model as possible to mitigate the degradation
Relaxed memory ordering

- Work done So far
  - KNI
  - Ring
  - Hash

- Future Work
  - Vhost
  - Virtio
  - ring

- Remove memory fences in the wrong places
- Lessen the barriers to make them as weak as possible
- Ensure correctness (multi-core safe) by synchronizing to the correct points.
__sync vs __atomic

- Benefits of __atomic
  - Full memory barrier → one-way barrier.
  - Benchmarking shows constant improvements.

```
(gdb) disassemble /s rte_spinlock_lock
Dump of assembler code for function rte_spinlock_lock:
0x0000000000468434 <+0>:    mov    w1, #0x1    // #1
0x0000000000468438 <+4>:    ldxr   w2, [x0]
0x000000000046843c <+8>:    stxr   w3, w1, [x0]
0x0000000000468440 <+12>:   cbnz   w3, 0x468438 <rte_spinlock_lock+4>
0x0000000000468444 <+16>:   dmb    ish
0x0000000000468448 <+20>:   cbz    w2, 0x46845c <rte_spinlock_lock+40>
0x000000000046844c <+24>:   ldr    w2, [x0]
0x0000000000468450 <+28>:   cbz    w2, 0x468438 <rte_spinlock_lock+4>
0x0000000000468454 <+32>:   yield
0x0000000000468458 <+36>:   b      0x46844c <rte_spinlock_lock+24>
0x000000000046845c <+40>:   ret
```

End of assembler dump.
WFE and SEV for V8.0

- **WFE**
  - Suspend execution when the lock is held
  - Get wake up events if the exclusive monitor is cleared by other PEs

- **SEV**
  - Send event explicitly to break WFE

- **Use cases**
  - Spinlock
  - Rw lock
  - Ring
  - Other acquire/release semantics environments
Rte_atomicN Xxx for V8.1

DPDK atomic implementation

- rte_atomicN_xxx APIs are implemented using __sync built-ins.
- These translate to retry loops with load/store exclusive instructions.

Proposed patches

- V8.1 ISA adds atomic instructions in atomic memory operations class.
- rte_atomicN_xxx APIs will be changed to use these new instructions.
Neon optimizations

Library (Planned)
- Vhost lib
- rte_ethdev lib
- rte_hash

Examples
- L3fwd ready
- More to come

PMD
- Intel NIC PMD ready by Arm
- Mellonax PMD ready by MLX
- More to come
Build system and Documentation

Make
- Gcc / clang
- Native / Cross
- Configuration settings

Meson/Ninja
- Native
- Clang
- Configurations

Documentation
- User guides
- Programming guide
DTS and Internal CI

• Work done so far
  • Identify DTS test cases suitable for Arm platform
  • Enable DTS test cases on arm64
  • Integrated into internal CI

• Future work
  • Add More cases
  • Integrated into community lab
Key Takeaways

- Relaxed memory model and examples of optimization
- More performant, scalable rwlock, spinlock, rte ring, atomic APIs
- Neon optimization examples
Arm’s efforts to DPDK and Future Plan

Thanks!

Gavin Hu
gavin.hu@arm.com
Honnappa Nagarahalli
Honnappa.Nagarahalli@arm.com
• BackUp
NEON for AArch64

- An extension to the AArch64 instruction set derived from the AArch32 Advanced SIMD syntax
- Same data processing principles
  - Registers are considered as vectors of elements of the same data type
  - Data types available: signed and unsigned 8-bit, 16-bit, 32-bit, 64-bit, single and double precision floating point
  - Instructions usually perform the same operation in all lanes
KNI optimization

Kernel Module
- Fix the synchronization issues using kernel SMP barriers

User Space
- Fix the synchronization issues using C11 memory model __atomic builtin
Spin lock

- `__atomic` instead of `__sync` builtin
  - Full memory barrier → one-way barrier

- SEV and WFE
  - Tight loop → suspend execution
  - Less stress to memory subsystem
  - Less power
Rw lock

- __atomic instead of __sync builtin
  - Full memory barrier → one-way barrier

- SEV and WFE
  - Tight loop → suspend execution
  - Less stress to memory subsystem
  - Less power
Vhost/virtio optimization Plan

- Relaxed memory ordering
- Spinlock and rwlock
- Prefetch
- Neon optimization
Fixes

- Synchronization to tail update
- Keep deterministic order allowing the CAS retry to work

More to come...

- Set Weak = true to allow spurious failure
- Replace rte_pause with “WFE” instruction, when updating tails.

Optimization

- Relaxed ordering for load and store of head
- Remove duplicate atomic loads