Agenda

• AArch64 WFE instruction
• New APIs
• Usage in DPDK
• Results
WFE instruction and supporting components

• WFE = Wait For Event

• When a CPU is in the wait state, it can be woken up by any event

• Events that can wake the CPU include:
  • SEV (send event),
  • loss of an exclusive monitor (in ArmV8).
WFE Instruction and Supporting Components

- A memory location is monitored
- Store to the location triggers core wake-up events
- Wake-up brings core out of low power state
- Spurious wake-ups are possible and must be handled
WFE Working Generic Flow

1. Clear event registers
2. Activate monitoring of location
3. Wait (enter the low power state)
4. Wake up and continue processing
Abstract APIs

• Add the APIs of two memory model flavors
  • `rte_wait_until_equal_relaxed_16/32/64`
  • `rte_wait_until_equal_acquire_16/32/64`

• Abstract API implemented for all architectures
  • AArch64 implementation uses WFE and related instructions
  • Implement as continuous poll loop for other arches not implementing WFE
WFE Usage in Spinlock


• This implementation does not use the new API
  • To save the loading of zero and compare against it and the branch

• WFE may behave differently on different Arm cores, use recommended instruction sequence [1]

---

WFE in Ticket Lock

- Wait for the current ticket number to equal my ticket

```c
--- a/lib/librte_eal/common/include/generic/rte_ticketlock.h
+++ b/lib/librte_eal/common/include/generic/rte_ticketlock.h
@@ -66,8 +66,7 @@
 static inline void
 rte_ticketlock_lock(rte_ticketlock_t *tl)
 {
-    uint16_t me = __atomic_fetch_add(&tl->s.next, 1, __ATOMIC_RELAXED);
-    while (__atomic_load_n(&tl->s.current, __ATOMIC_ACQUIRE) != me)
-        rte_pause();
+    rte_wait_until_equal_acquire_16(&tl->s.current, me);
 }
```

- This example shows how to employ the new API..
WFE in Ring Buffer

- Multiproducer (MP) and multiconsumer (MC) rings
  - Wait for ring tail to be updated by preceding P/C thread(s)
  - Tails have to be updated in the order of moving heads
- Update both generic and C11 ring implementations
- http://patches.dpdk.org/patch/59267/

```diff
diff --git a/lib/librte_ring/rte_ring_generic.h b/lib/librte_ring/rte_ring_generic.h
index 953c6bb..682b8527 100644
--- a/lib/librte_ring/rte_ring_generic.h
+++ b/lib/librte_ring/rte_ring_generic.h
@@ -23,8 +23,7 @@
     * we need to wait for them to complete
     */
     if (!single)
-       while (unlikely(ht->tail != old_val))
-         rte_pause();
+       rte_wait_until_equal_relaxed_32(&ht->tail, old_val);

     ht->tail = new_val;
```
Other examples

• EVENT/OPDL
  • http://patches.dpdk.org/patch/59269/
• ThunderX NICVF
  • http://patches.dpdk.org/patch/59268/
Power efficiency potential of WFE with polling

Mellanox ConnectX-5 driver (mlx5) in DPDK modified to use WFE
DPDK pktgen with 10 Gbps i’face to testpmd on ThunderX2 with mlx5

- 1 receive queue on NIC
Polling: Wasteful of energy!

memcached using OFP + ODP-DPDK

source: Strategies for Improving Tail Latency for Poll-Based Networking, Steve Zekany (Arm intern 2017)
DPDK Power Optimization Research by Intel

Intel reported around 30% reduction in power consumption with L3fwd-power using on-demand CPU power state tuning.

“Based on a US EPA study, they assume that network equipment spends 25% of the time with high traffic (active state) and 75% of the time with low traffic (idle state)”

OpenSHMEM Wait with WFE (single address)

Enabling One-sided Communication Semantics on ARM, Shamis et al., IPDPSW 2017
More use cases

- Datacenter
  - Ethernet Poll Mode Driver (DPDK)
- HPC
  - MPI
    - OpenSHMEM
  - RDMA user level poll mode
- Thread communication over shared memory
- Direct block device I/O (Linux io_uring)
- POSIX asynchronous I/O
- Generic I/O multiplexing facility (epoll in hardware)
Gavin Hu

gavin.hu@arm.com

Thanks