Introduction to the Distributed Software Event Device

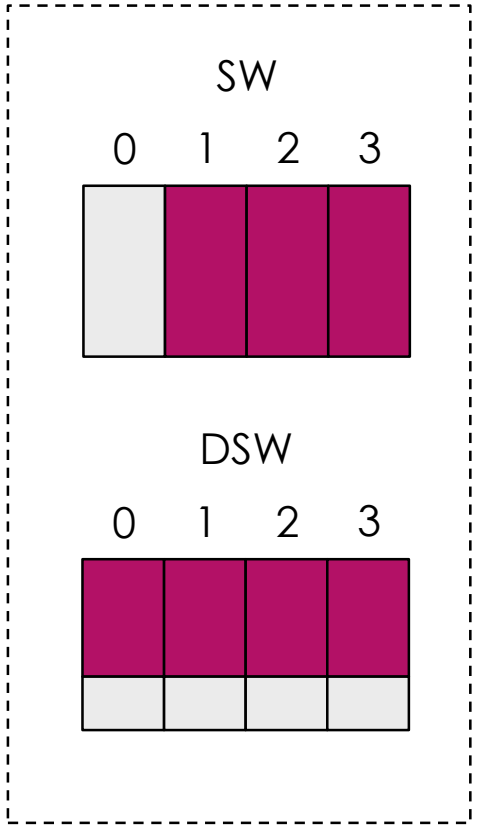
Agenda

DPDK
DATA PLANE DEVELOPMENT KIT

- Overview
- Rationale
- Event Scheduling
- Flow Migration
- Workloads
- Observability
- Performance
- Further Reading

Overview

- A DPDK Event Device
- Software implementation
- Parallel
  - Scheduling work is distributed to all participating lcores
- Queue types: atomic, parallel and single link
  - No ordered or “mixed mode” (CFG_ALL_TYPES)



Why Another Software Scheduler?

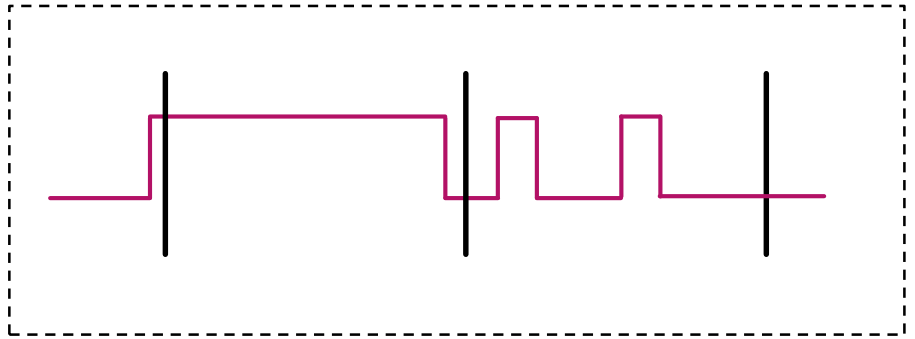
- Avoid using a dedicated scheduler lcore
  - Allow scaling down to few lcores
  - Allow scaling up to more lcores and/or longer pipelines
- Reduce the scheduling overhead (clock cycles/event)
- …both at a cost in load balancing agility
- Complementary to the SW event device

Scheduling

- Scheduling happens at enqueue
- DPDK event rings for transport
- Events go directly port-to-port (usually lcore-to-lcore)
- Procedure
  - Calculate 15-bit flow hash from the event's flow id
  - Look up port id in target queue's flow-hash-to-owning-port-table
- To improve efficiency, events are buffered

Port Load Estimation

- An estimate of a port's load is needed for load balancing
- Dequeue of 0 events: port is now idle
- Dequeue of > 0 events: port is now busy
- At the point of transition, a time stamp is taken
- Periodically, the busy vs idle time is used to calculate a load estimate
- Measurement period is 250 us



Flow Migration

- Purpose: load balancing
  - Maintain flow-hash-to-owning-port tables such that no ports are overloaded
- Every 1 ms: is the load above the threshold?
  - Yes? Try move one of flows the port owns
- The port queries the other ports' load estimates see if there is a suitable candidate port
- To know what flow to move, each port maintains a list of last 128 seen events
- Smallest of the last seen flows is first choice for migration

Migration Procedure

- Maintaining in-order processing guarantees of atomic is the challenge
- “Under-the-hood” signaling schema between the ports
  - Messaging over DPDK rings
  - Control rings are checked during enqueue/dequeue
  - Asynchronous
- This schema requires that there are no unattended ports
- The initiating port will order other ports to “pause” the flow
- The migrated flow will experience a short “hiccup”
- Migration latency depends on enqueue/dequeue call rate
  - Stage processing latency
  - Dequeue burst size
  - Dummy call rate for idle ports
  - Lcore OS thread preemption

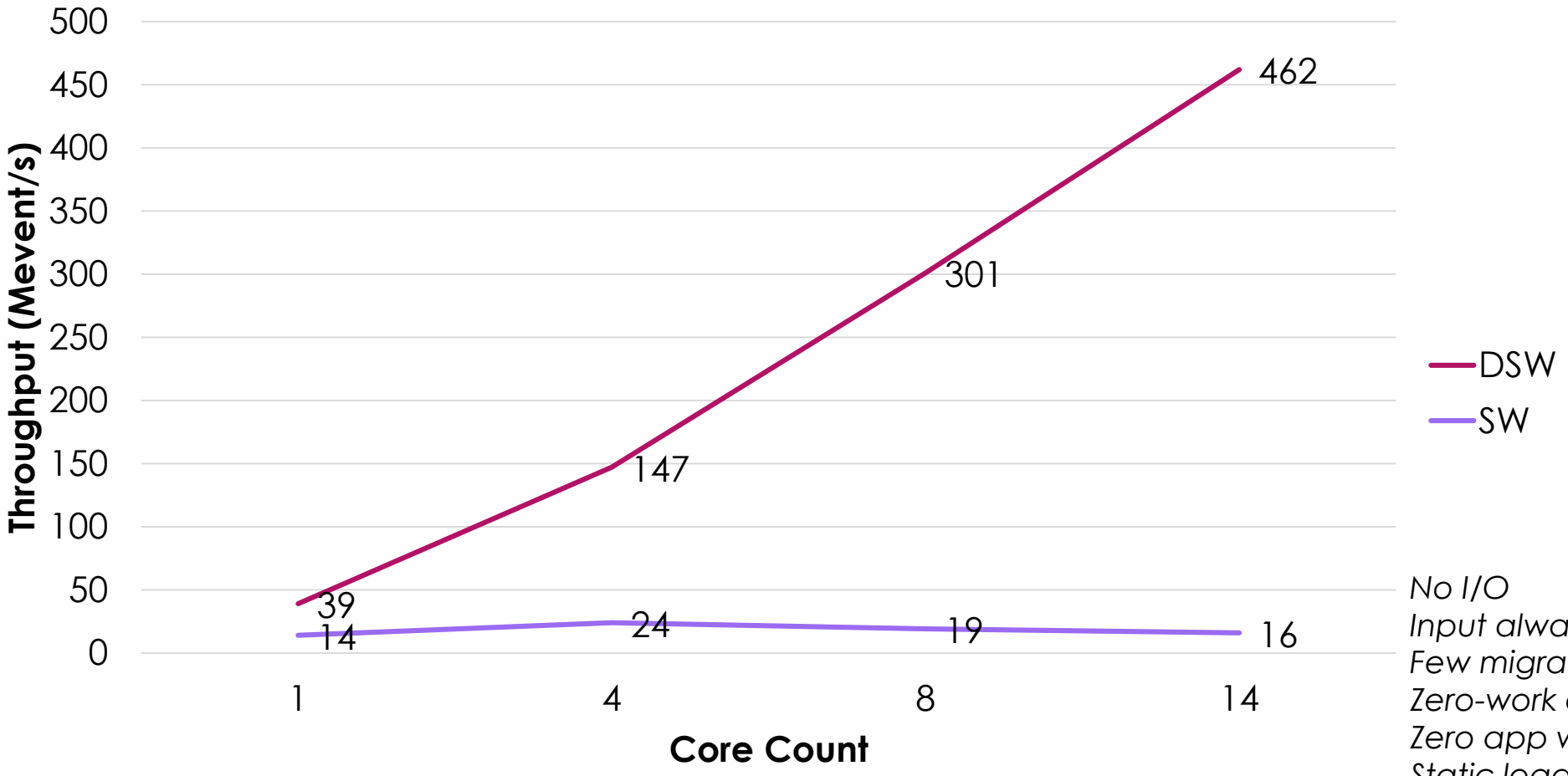
Workload Suitability Speculation

- Many small flows: OK
- Few large flows where the flow event rate change relatively slowly: OK
- Few large very bursty flows: Likely suboptimal load balancing
- What is slowly?
  - Migration rate is ~1 kHz per port

Observability

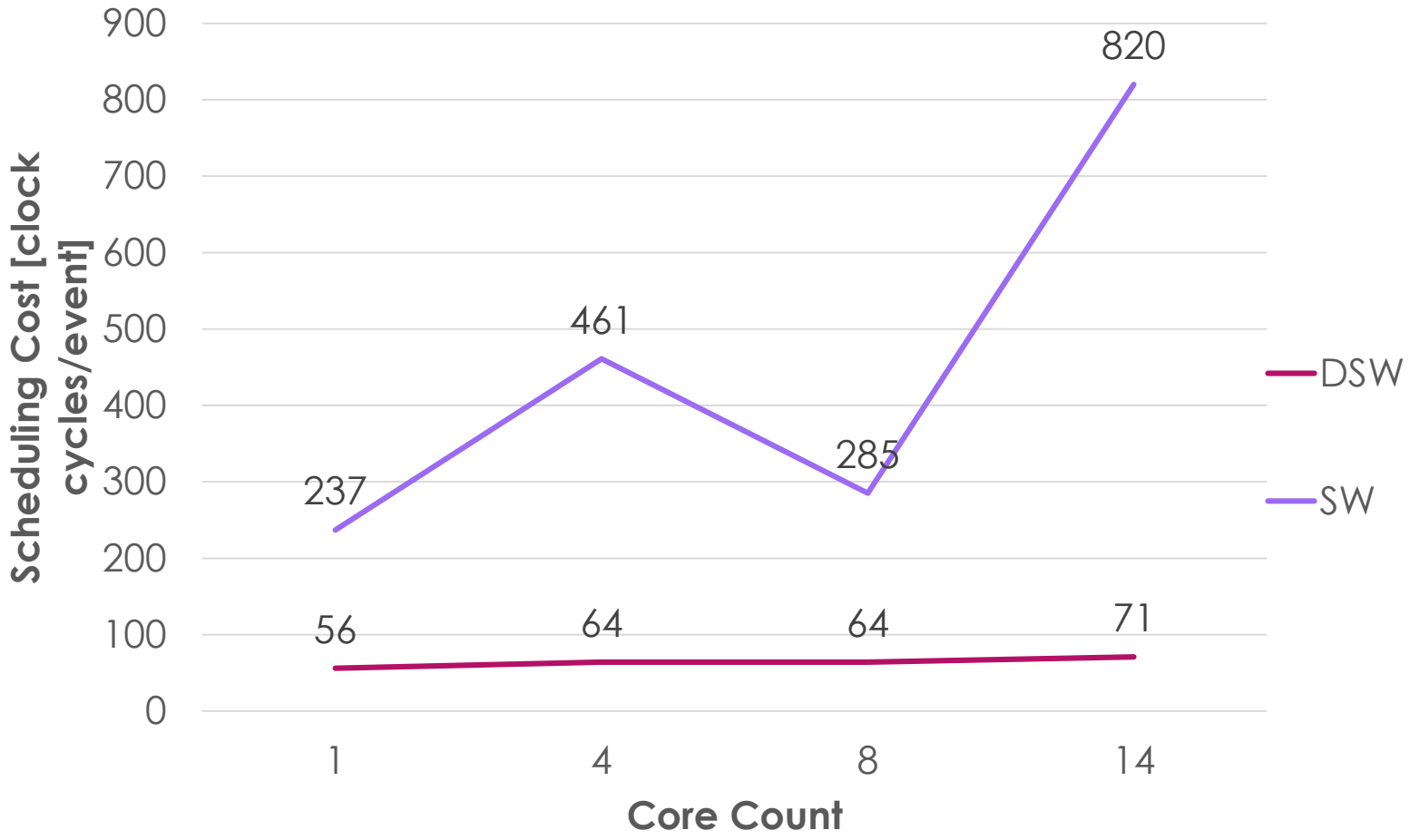
DSW xstats
port_<x>_new_enqueued
port_<x>_forward_enqueued
port_<x>_release_enqueued
port_<x>_queue_<y>_enqueued
port_<x>_dequeued
port_<x>_queue_<y>_dequeued
port_<x>_migrations
port_<x>_migration_latency
port_<x>_event_proc_latency
port_<x>_inflight_credits
port_<x>_load
dev_credits_on_loan

Throughput



No I/O
Input always available
Few migrations
Zero-work one-stage pipeline
Zero app working set
Static load – few migrations
E5-2680v4 @ 2.4 GHz
[14-core Broadwell Xeon]

Efficiency



Five-stage Pipeline
1000 cc work/stage
Takes dedicated core cycles into account

Patch Status

- Patch set
  - http://patchwork.dpdk.org/project/dpdk/list/?series=1116
- Cover letter
  - http://mails.dpdk.org/archives/dev/2018-August/110525.html
- 18.11?

Questions?

Mattias Rönnblom
<mattias.ronnblom@ericsson.com>