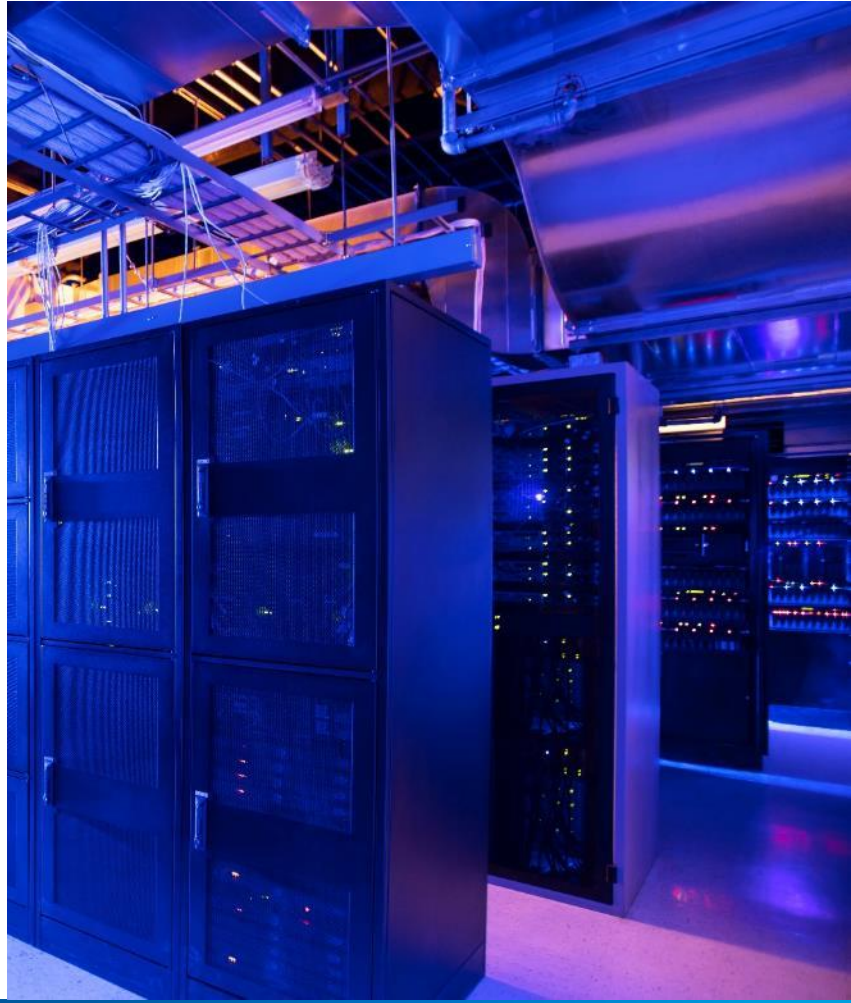
FLEXIBLE AND DYNAMIC RESOURCE USE WITH SPDK

Ben Walker

Darek Stojaczyk

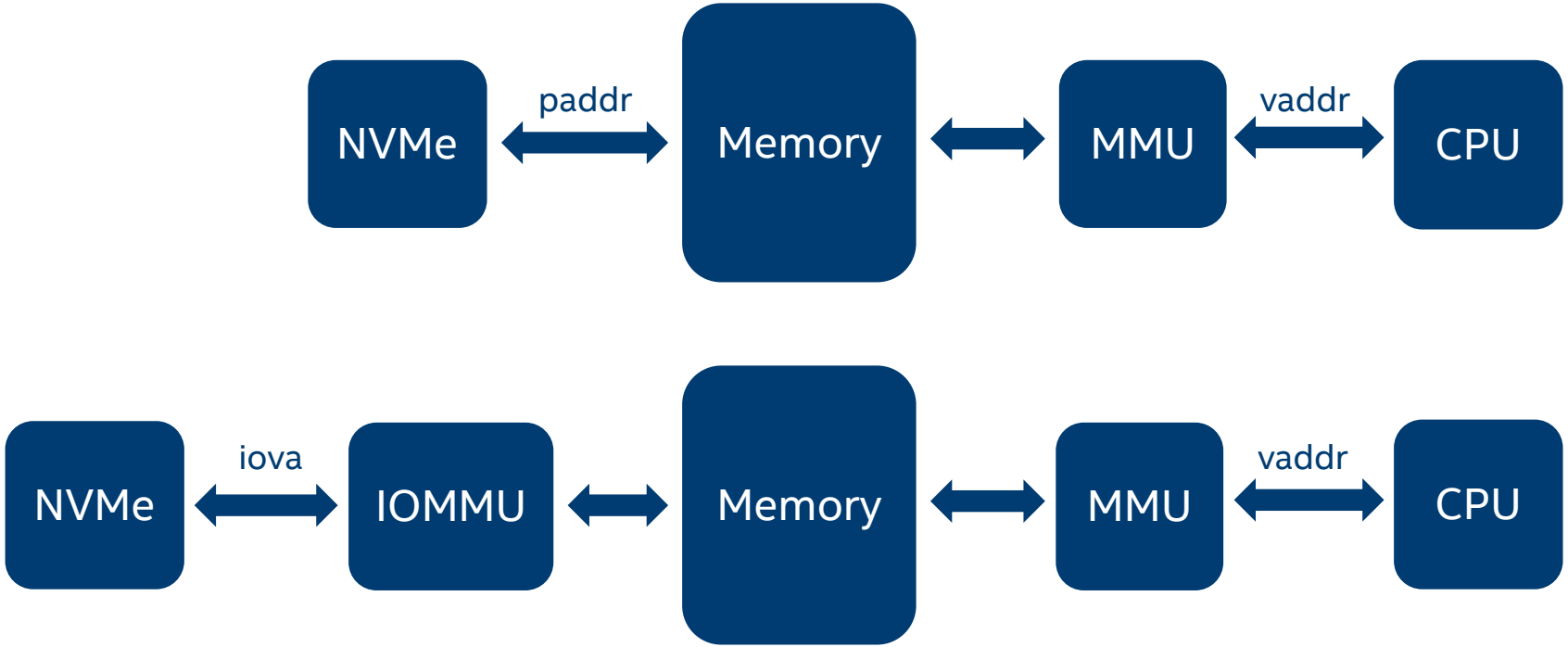
AGENDA

- Direct Memory Access
- Hugepage Management
- Threading Model



DIRECT MEMORY ACCESS

DIRECT MEMORY ACCESS (DMA)



HOW TO DO DMA IN SPDK

`spdk_malloc()`

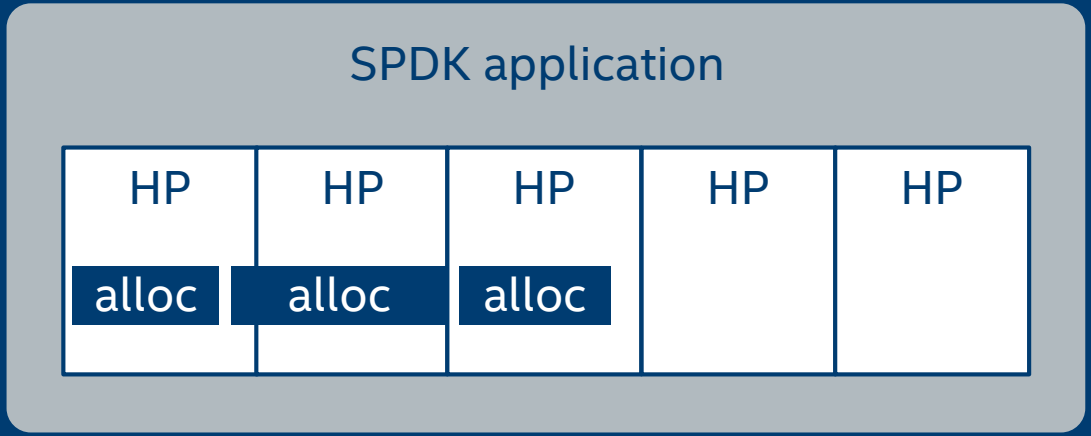
- Backed by hugepages
- Pinned
- Shared between processes in a multi-process group

HUGEPAGE MANAGEMENT

LEGACY MEMORY MANAGEMENT

Pre-reserve big chunk of hugepages

Dynamic allocation within pre-reserved region



Challenge:

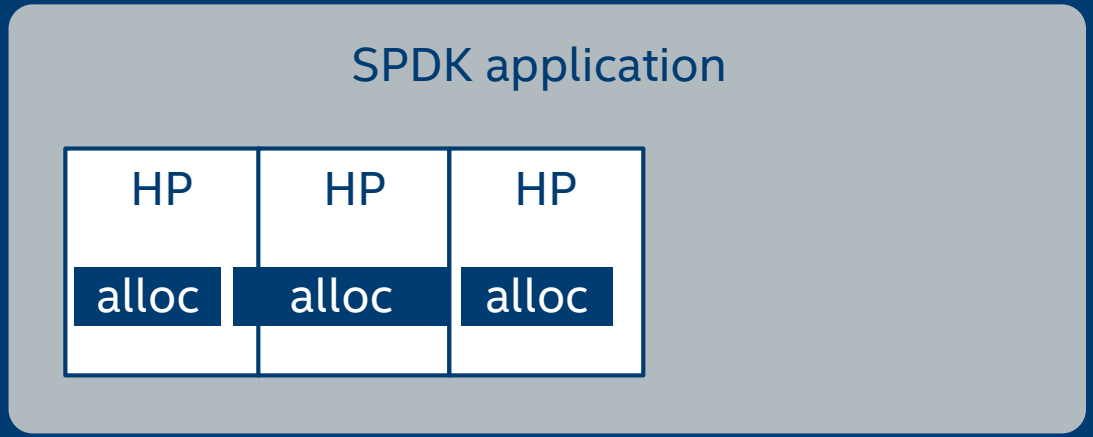
- Need to calculate how much memory the program needs up front

NEW DYNAMIC MEMORY MANAGEMENT

Implemented in DPDK 18.05, hardened in DPDK 18.08

Directly in response to SPDK (and other feedback)

Dynamically allocates hugepages as needed



Available since SPDK 18.10 and DPDK 18.05.1

REGISTER YOUR OWN MEMORY

```
spdk_mem_register(void *vaddr, size_t len)
spdk_mem_unregister(void *vaddr, size_t len)
```

Restrictions:

- *If no IOMMU, must use hugepages*
- *The memory address and length need to be a multiple of 2MB*

MEMORY MAPS

- Translation tables
  - virtual address ➔ uint64_t
- There are multiple registered maps
- Maps are notified when users register memory

```
map = spdk_mem_map_alloc(notify_cb, …)
spdk_mem_map_set_translation(map, vaddr, size, translation)
spdk_mem_map_translate(map, vaddr, *size)
```

MEMORY MAPS

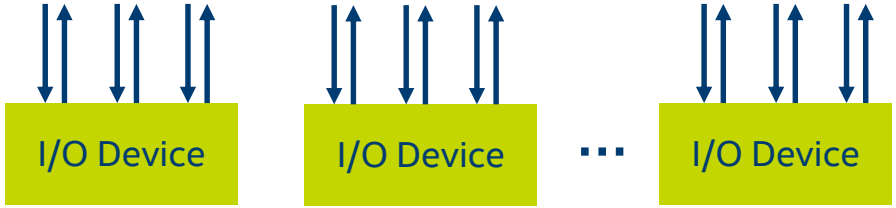
Name	Key	Value	User
PCIe	Virtual address	Physical/bus address	NVMe, I/OAT
RDMA	Virtual address	ibv_mr *	NVMe-oF initiator and target
Vhost-user	Virtual address	Virtual address	Vhost-user PMD

DYNAMIC THREADING

REMINDER: PERFORMANCE VIA CONCURRENCY

Modern CPUs provide many cores

Modern I/O devices provide many independent queues

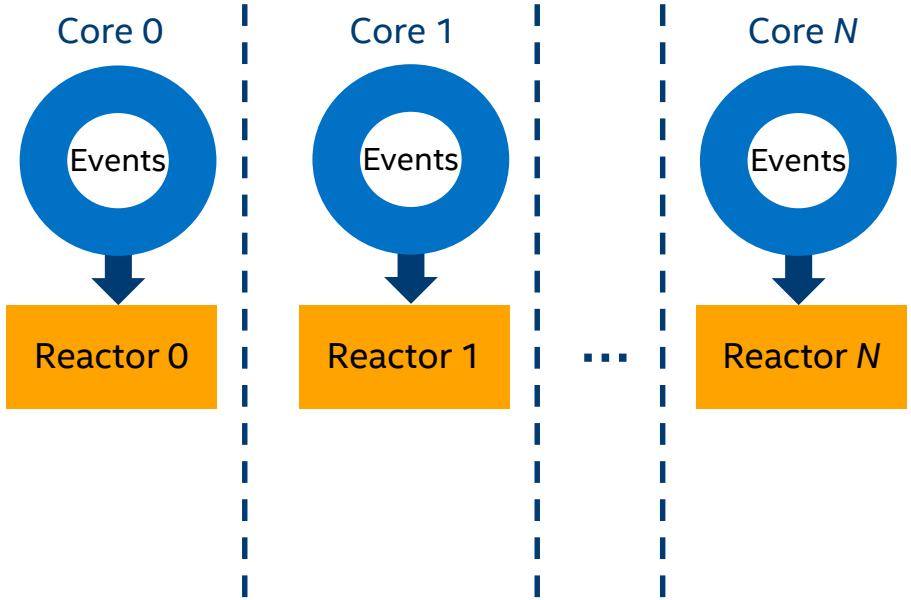


Goal: Architect software to match the hardware

SPDK 18.07 AND EARLIER

System thread running a loop

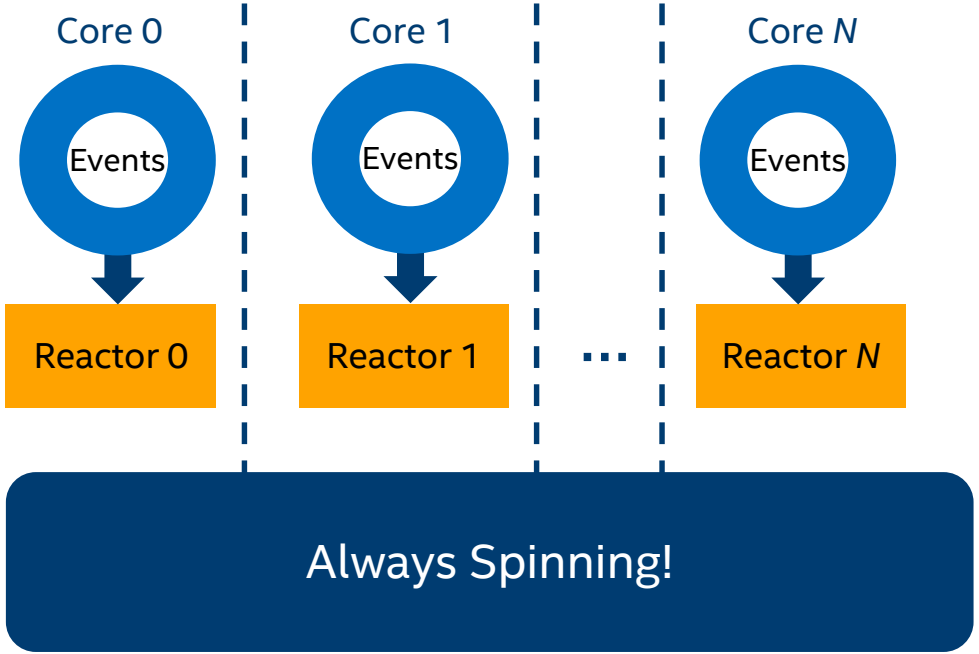
- Pinned to a specific CPU core
- Called a “reactor”
- Polls event ring for incoming work
- Communication via event passing



Implemented in lib/event

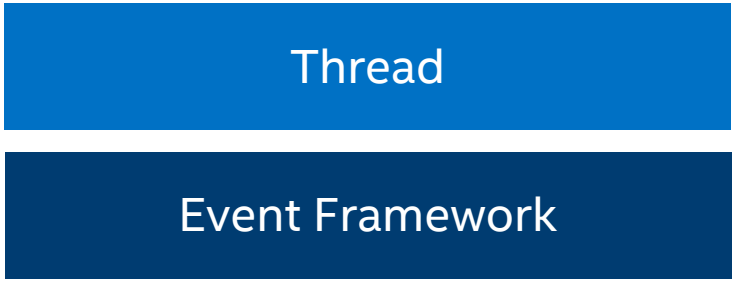
CHALLENGES

- Scaling Up/Down
- Rebalancing
- Integration



THREADING ABSTRACTIONS: GOALS

- Create Abstraction layer between the reactor framework and other SPDK libraries
- Allow applications to plug in their own framework
- Trickling in over the last year – fully abstracted in 18.10



THREADING ABSTRACTIONS: CONCEPTS

- `spdk_thread`
  - A thread spinning in a loop, processing events. We tie data to `spdk_thread`.
- `spdk_msg`
  - An event – a function pointer passed from one thread to another.
- `spdk_poller`
  - A repeated event on a thread with an optional delay between calls.
- `spdk_io_channel`
  - An abstraction for per-thread context.

THREADING ABSTRACTIONS: POLLING

- `spdk_thread_poll`
  - Runs any expired pollers
  - Runs all incoming messages
  - Returns

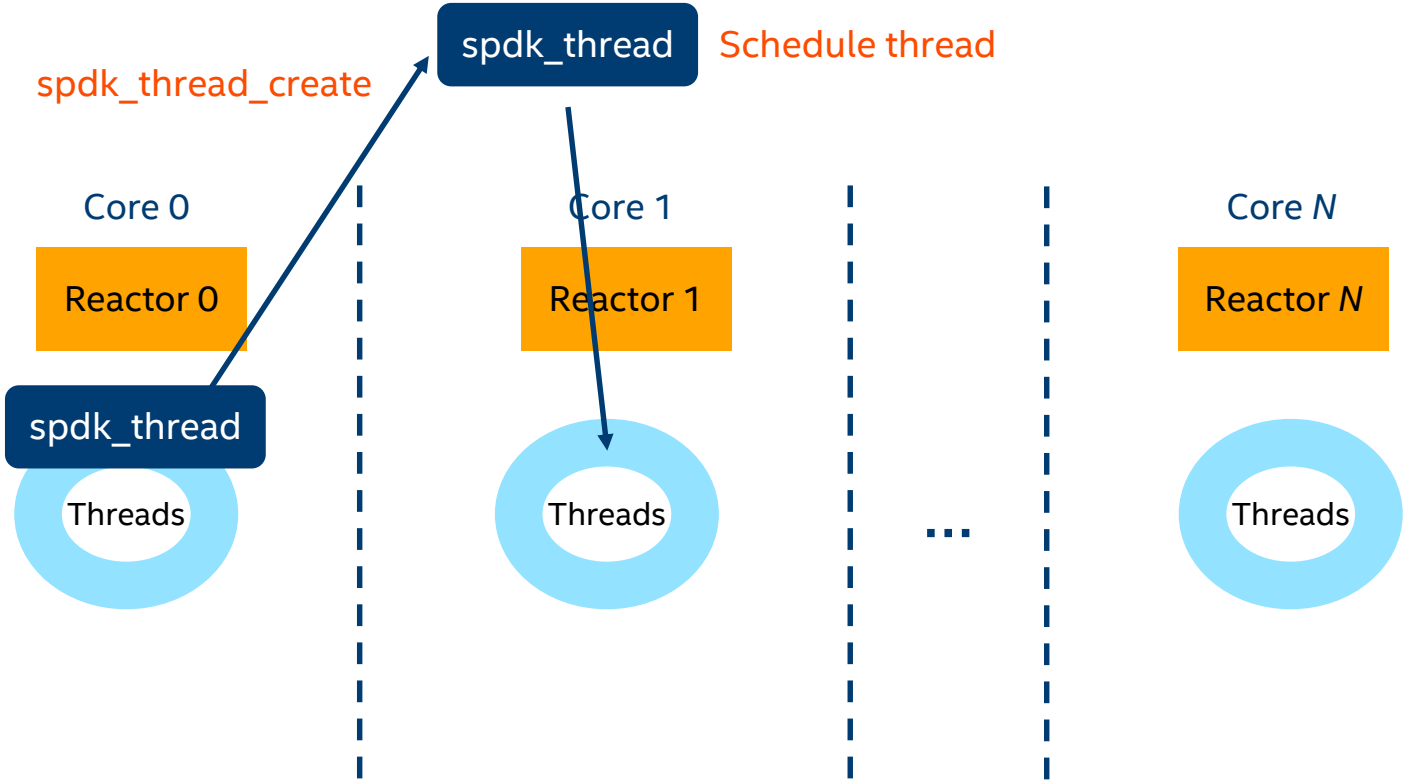
- As of 19.01, reactors do the following on each loop:
  - Call `spdk_thread_poll` on their single `spdk_thread`
  - Check for incoming events (the old style events)

LIGHTWEIGHT THREADING: THE BASICS

Concept	Address Space	Multiplex Method	Scheduler
Process	Isolated	Pre-emptive	OS
Thread	Shared	Pre-emptive	OS
Fiber	Shared	Cooperative	Language Run Time, Framework
Green Thread	Either	Either	Not OS

Today, **`spdk_thread`** is a **Thread**. In 19.04, an **`spdk_thread`** becomes both a **Fiber** and a **Green Thread**.

LIGHTWEIGHT THREADING: THE IMPLEMENTATION



LIGHTWEIGHT THREADING: RECAP

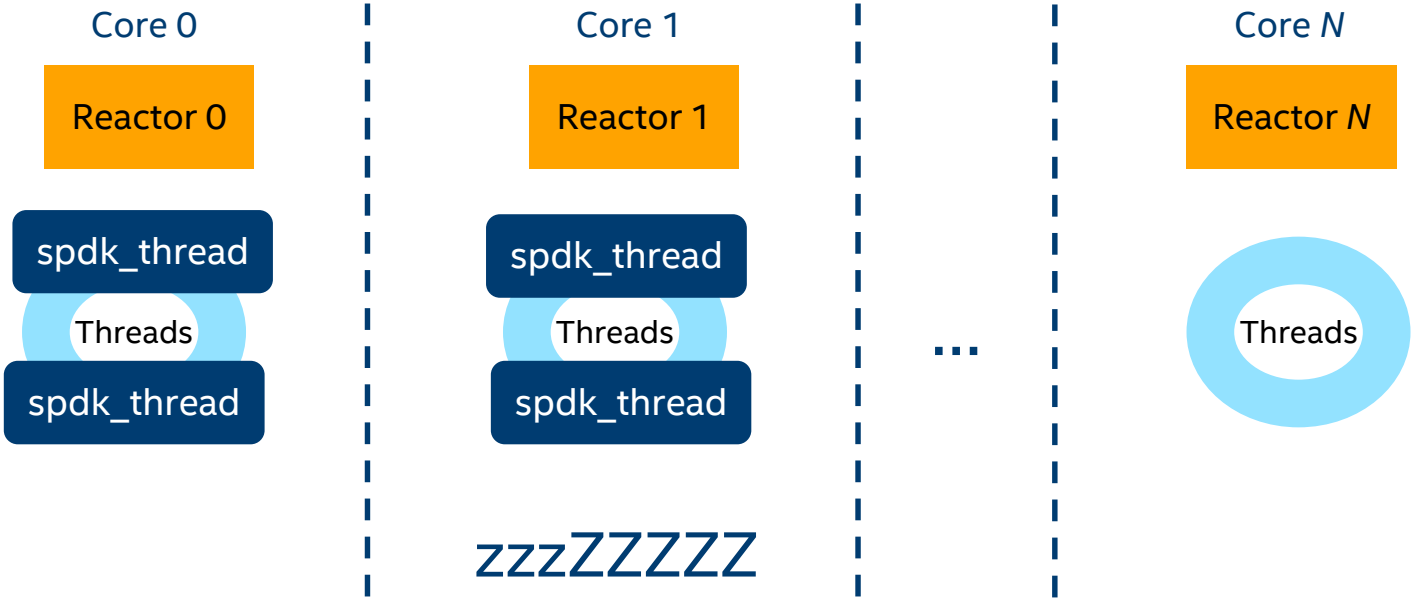
- `spdk_thread_lib_init`
  - Provide a “schedule” callback
- Reactors contain a list of `spdk_threads`
  - Each loop, call `spdk_thread_poll` on each `spdk_thread`
- When thread created, call schedule callback. Get placed on reactor.

LIGHTWEIGHT THREADING: ADVANCED SCHEDULING

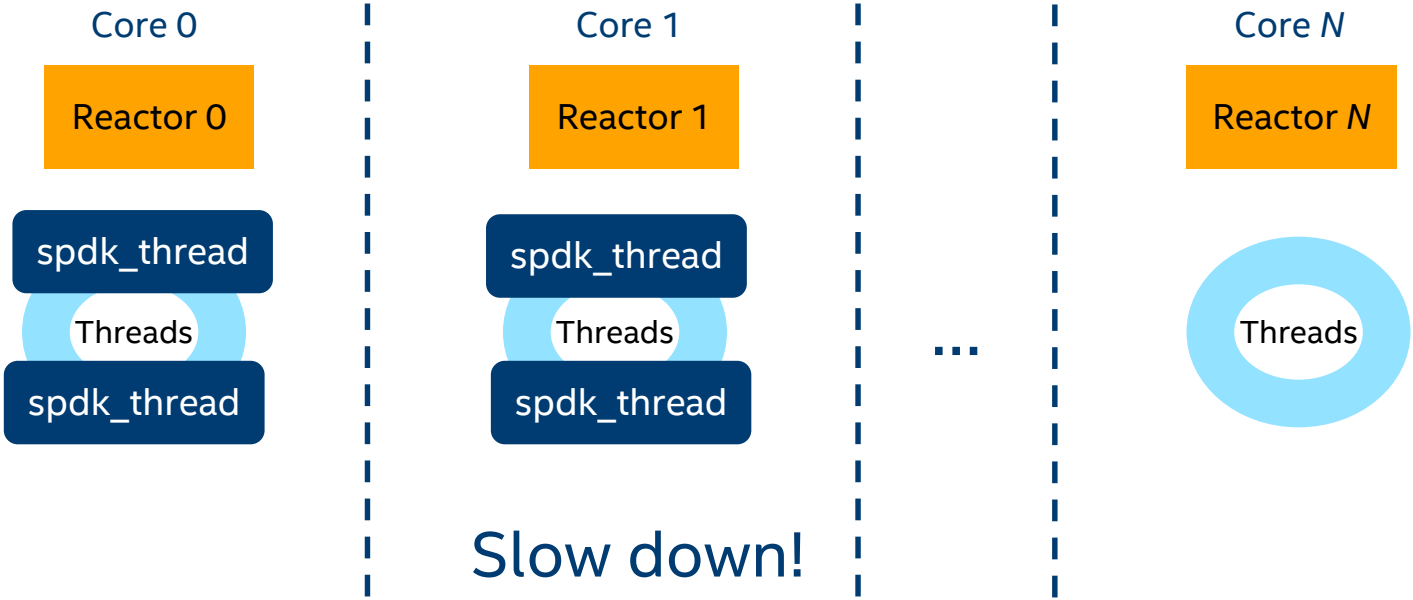
- `spdk_thread_poll` returns whether it did any “work”
  - Each poller reports whether it accomplished anything (true/false)
  - Any message also counts as work

Use Busy Indications to Reschedule Threads!

LIGHTWEIGHT THREADING: ADVANCED SCHEDULING



LIGHTWEIGHT THREADING: ADVANCED SCHEDULING



LIGHTWEIGHT THREADING: ADVANCED SCHEDULING

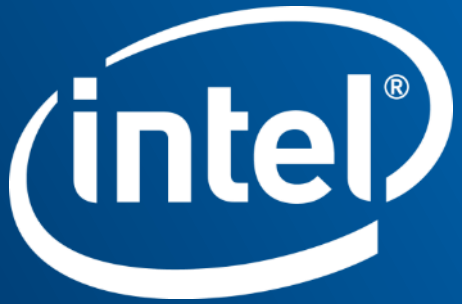
Intel Speed Select Technology

- Intel SST-Base Frequency
- Intel SST-Core Power

Available on Cascade Lake N SKUs

FUTURE DIRECTION

- Smarter thread schedulers
- More configuration parameters (trade latency vs power consumption)
- Thread affinity


intel®

BACKUP


intel