
ORACLE®

Supporting SPDK in Oracle RDBMS

Akshay Shah, Zahra Khatami, Bang Nguyen, Avneesh Pant, Rajarshi Chowdhury and Sumanta Chatterjee

Oracle, VOS Group
April 16, 2019

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

1. **Challenges**
2. SPDK and Oracle Database Architecture
3. OraEnv
4. Oracle Dispatcher
5. Oracle Database Support Model

Challenges

1. **Memory Management**: Oracle instance must manage all the resource allocations within the SPDK libraries.

2. **Security**: There is no read/write access security policy provided for the memory shared across all the secondary processes spawned in SPDK:

   I. Allocates private/shared memory from same area.
   II. Shares memory map with both primary/secondary processes.

Challenges

3. **Recoverability**: We want to prevent memory leaks, detect, and isolate memory corruption incidents.

4. **Restartability**: We want SPDK processes to be restartable.

5. **Scalability**:

✓ Modern RDMA NICs have known scaling limitations in presence of large number of connections and memory regions due to the limited caches on the device.

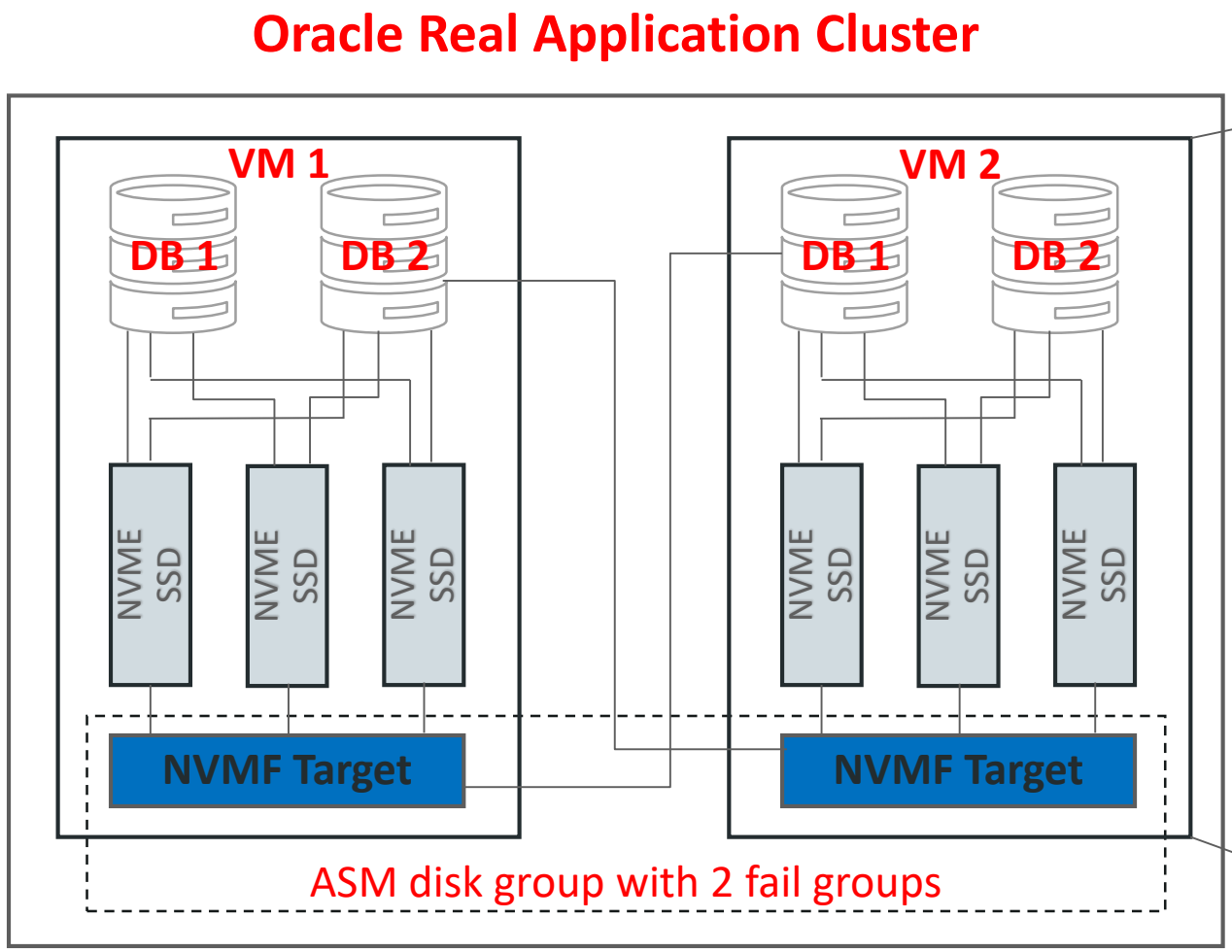
✓ Limited number of the I/O queues on NVMe devices precludes in-band dispatch of the local I/Os from Oracle processes.

ORACLE

Program Agenda

1. Challenges
2. **SPDK and Oracle Database Architecture**
3. OraEnv
4. Oracle Dispatcher
5. Oracle Database Support Model

SPDK and Oracle Database Architecture



Database NVM I/O module		
Device Management	**Local I/O**	**Remote I/O**
Oracle EAL	Oracle NVM dispatcher	Oracle NVM library
Enumerate PCIe devices	SPDK I/O to PCIe ns	SPDK I/O to NVMe-oF ns
Memory Management		Zero-copy RDMA for IB and RoCEE. bcopy for TCP
Rkey/ lkey Management		

SPDK and Oracle Database Architecture

- High-performance I/O subsystem capable of delivering millions of IOPS.
- Guarantees low latency by eliminating overheads in I/O code path.
- Enables multiple databases to securely access the same devices.
- Provides fault tolerance from software bugs and malicious workloads.
- Provides scalable RDMA capability with large number of connections.
- High availability using Oracle RAC, ASM and NVMe-oF technologies.

Program Agenda

1. Challenges
2. SPDK and Oracle Database Architecture
3. **OraEnv**
4. Oracle Dispatcher
5. Oracle Database Support Model

OraEnv

- ✓ Getting rid of DPDK (Sorry! ☹)
- ➢ In spdk/lib/Makefile

```
# If CONFIG_ENV is pointing at a directory in lib, build it.
# Out-of-tree env implementations must be built separately by the user.
ENV_NAME := $(notdir $(CONFIG_ENV))
ifeq ($(abspath $(CONFIG_ENV)),$(SPDK_ROOT_DIR)/lib/$(ENV_NAME))
DIRS-y += $(ENV_NAME)
endif
```

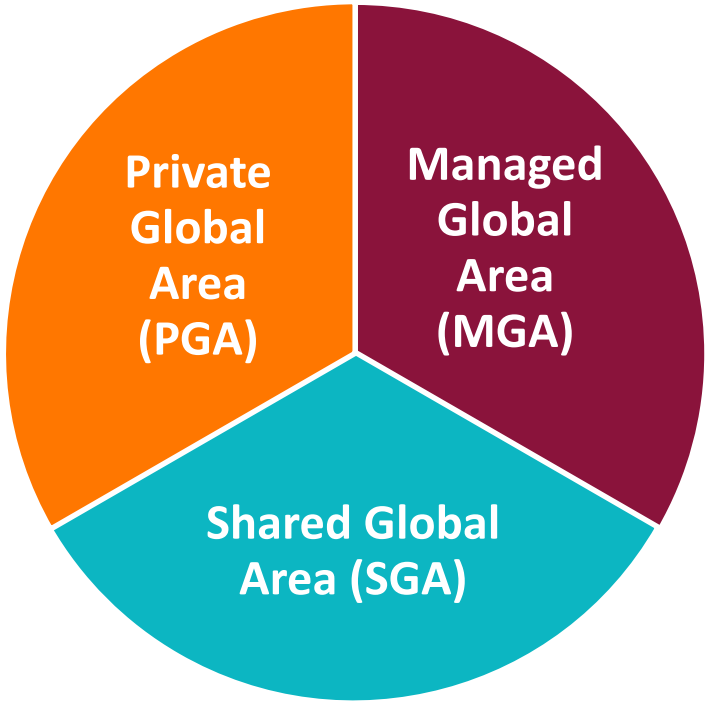
- ✓ Introducing OraEnv: Complete implementation of SPDK Env

```
struct spdk_env_opts {
        const char *name;
        const char *core_mask;
        int shm_id;
        int mem_channel;
        int master_core;
        int mem_size;
        bool no_pci;
        bool hugepage_single_segments;

        bool unlink_hugepage;
        size_t num_pci_addr;
        const char *hugedir;
        struct spdk_pci_addr *pci_blacklist;

        struct spdk_pci_addr *pci_whitelist;


        /** Opaque context for use of the env implementation. */

         void *env_context;
};
```

OraEnv

- Memory managements (allocations/deallocations/physical addresses)
- Providing rkeys/lkeys
- PCIe scanning/enumerations
- lcore initialization
- Enhancements for RDMA transfers with NVMe-oF

OraEnv: Memory Managements



Kgh

- Space is organized into Heaps.
- A heap can either be allocated in space that is private to a particular process or space that is shared by many processes.
- When a user asks for a chunk of space, that chunk will come from a particular heap.
- A heap is composed of a set of contiguous chunks.
- Returns the set of extents contained in the heap.

OraEnv: Memory Managements

Shared memory:

1. Globally shared areas
   - Controllers pages
2. Managed global areas (MGA)
   - Data path

```
/**
* Memory is dma-safe.
*/
#define SPDK_MALLOC_DMA 0x01

/**
* Memory is sharable across process boundries.
*/
#define SPDK_MALLOC_SHARE 0x02
```

```
void *spdk_malloc(size_t size, size_t align, uint64_t *phys_addr, int socket_id, uint32_t flags);
```

```
void *spdk_zmalloc(size_t size, size_t align, uint64_t *phys_addr, int socket_id, uint32_t flags);
```

OraEnv: Memory Managements

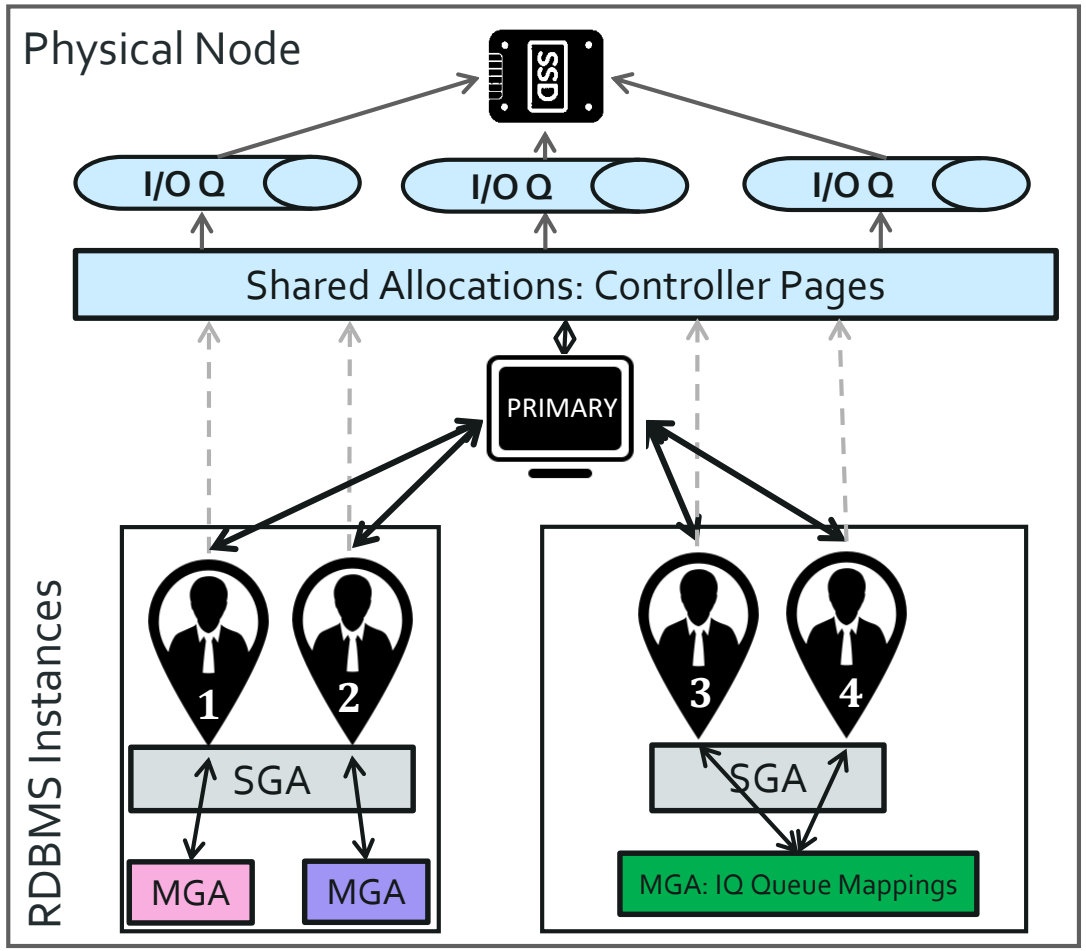
Two types of processes:

1. Primary

✓ The shared memories are requested from a primary process and all the clients have same accesses to them.

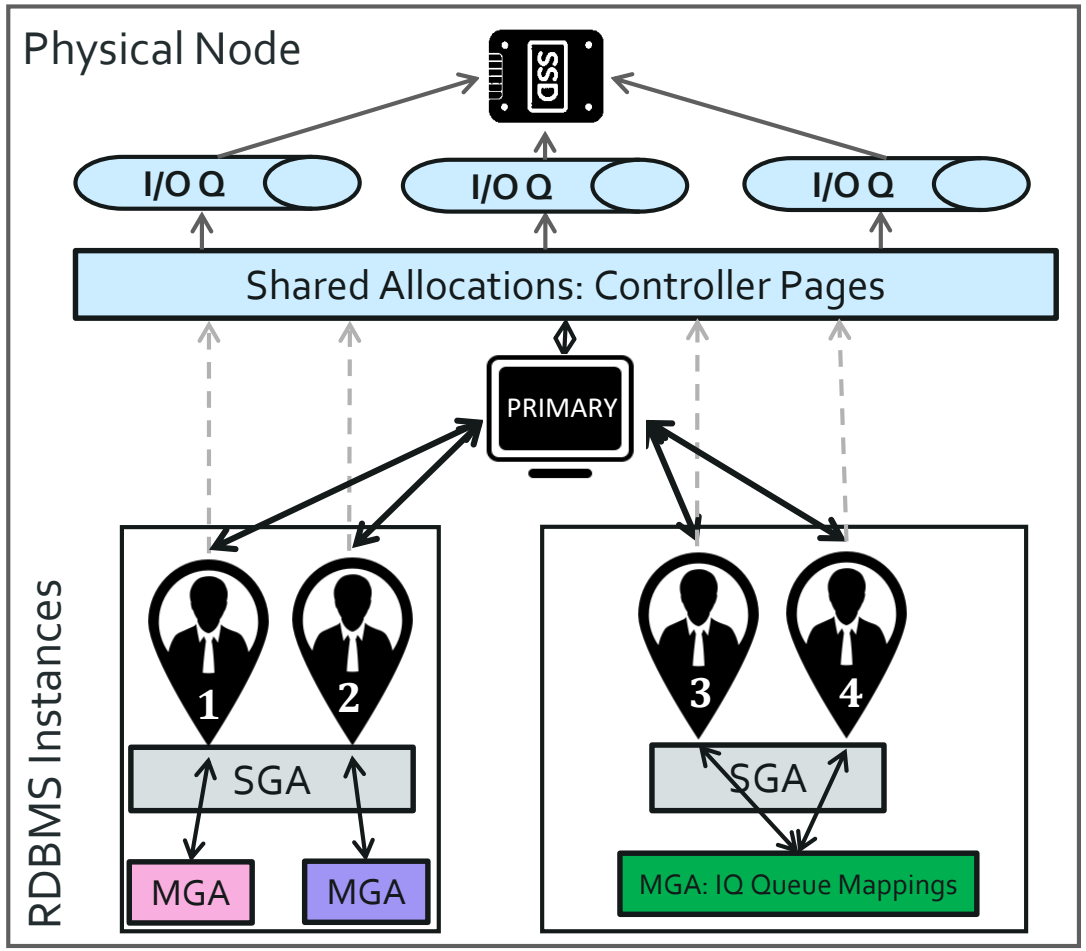
2. Secondary

✓ MGA can be configured to be shared between specific processes



OraEnv: Memory Managements

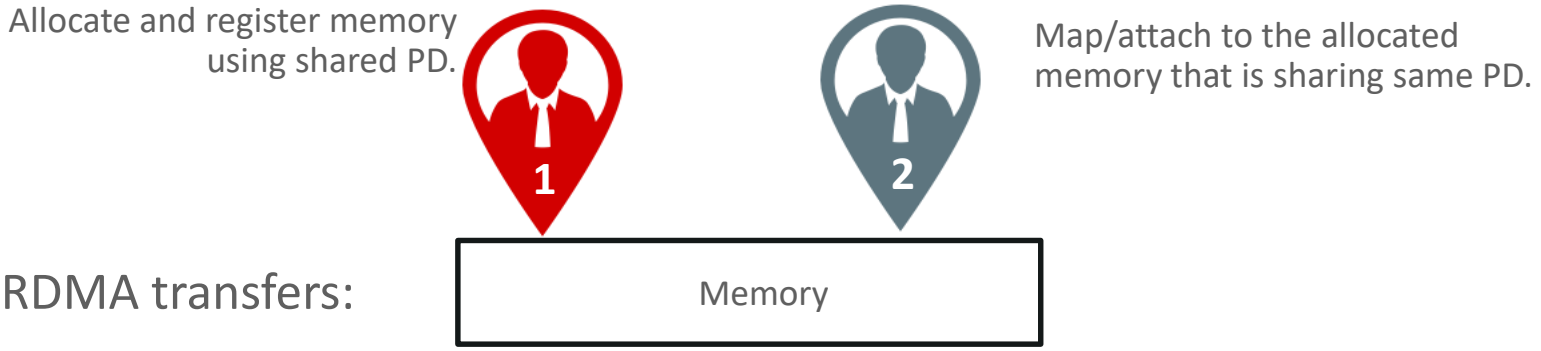
- ✓ **Security**: Clients 3 and 4 are attached to the same MGA section that is isolated from clients 1 and 2. It allows us to share DMA resources across instances.
- ✓ **Recoverability**: Preventing memory leakages and data loss.
- ✓ **Restartability:** For both primary and target processes



OraEnv: Enhancements for RDMA transfers

Registering entire memory ... ch process is not scalable

Using shared PD



- MGA allocations are optimized for RDMA transfers: using shared PD registration.
- Providing lkey/rkeys
- It utilize a single RDMA Protection Domain across all Oracle processes.
- PD is valid as long as one user process attached to it.

OraEnv: Enhancements for RDMA transfers

- New hooks in nvme.h
- An interface for users to provide their own pd/mr/rkeys in both nvme and nvmf libraries.

RDMA Transport Hooks

```
struct spdk_nvme_rdma_hooks {
    struct ibv_pd *(*get_ibv_pd)(const struct spdk_nvme_transport_id *trid,
                                 struct ibv_context *verbs);

    uint64_t (*get_rkey)(struct ibv_pd *pd, void *buf, size_t size);

};
```

Getting pd

```
rctrlr->pd = g_nvme_hooks.get_ibv_pd(&rctrlr->ctrlr.trid, rqpair->cm_id->verbs);
```

```
rc = spdk_mem_map_set_translation(map, (uint64_t)vaddr, size, g_nvme_hooks.get_rkey(pd, vaddr, size));
```

Getting rkey

OraEnv Performance

1. 2 node RoCEE setup with 2x 18-core Xeon Gold 6154 CPU.
2. 1x Intel P4500 PCIe SSD

	OraEnv IOPS	OraEnv MB/s	DPDK IOPS	DPDK MB/s	%	Faster
Perf with PCIe	646422	2525	639311	2497	1.1	OraEnv
Perf with Target (loopback)	623225	2434	623448	2435	--	SAME
Perf with Target (Remote)	388854	3037	397176	3102	2	DPDK

*Performance data is still WIP.

OraEnv: Target

- Fully enterprise secured
- Highly resilient and recoverable
- High availability (HA)

- Getting name spaces
- Analyzing memory consumption and resource utilizations
- Getting the number of clients connected to nvmf_tgt

Program Agenda

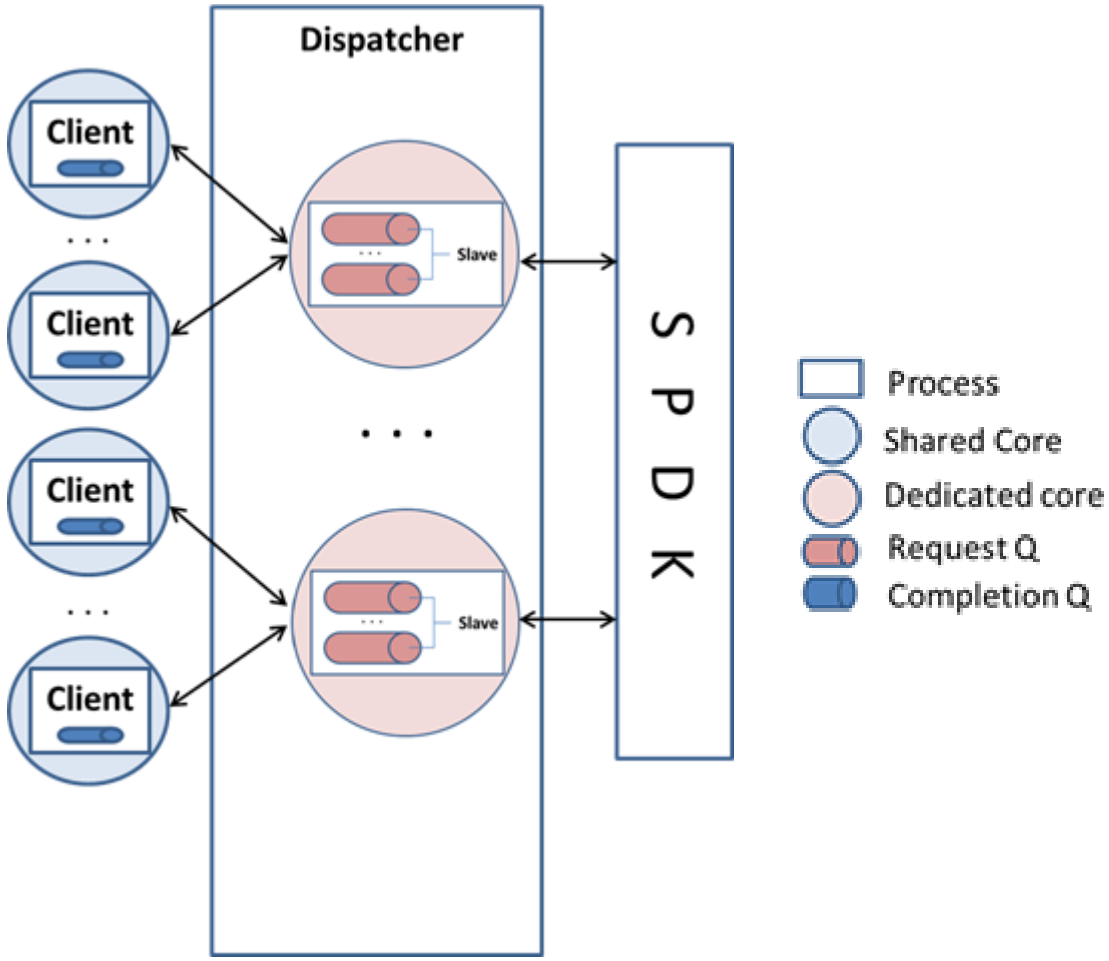
1. Challenges
2. SPDK and Oracle Database Architecture
3. OraEnv
4. **Oracle Dispatcher**
5. Oracle Database Support Model

Oracle SPDK I/O stack

- NVMe devices are discovered as ASM disks.
- ASM fail groups created using local (PCIe) and remote (NVMe-oF) NVMe devices.
- Writes are mirrored to local and remote devices.
- Reads are always issued to local fail group when available.
- Local SGA I/Os are handled by Oracle dispatcher slave using PCIe name space.
- Local PGA I/Os are submitted directly to SPDK using the NVMe-oF name space.
- Remote I/Os are submitted directly to SPDK using the NVMe-oF name space.

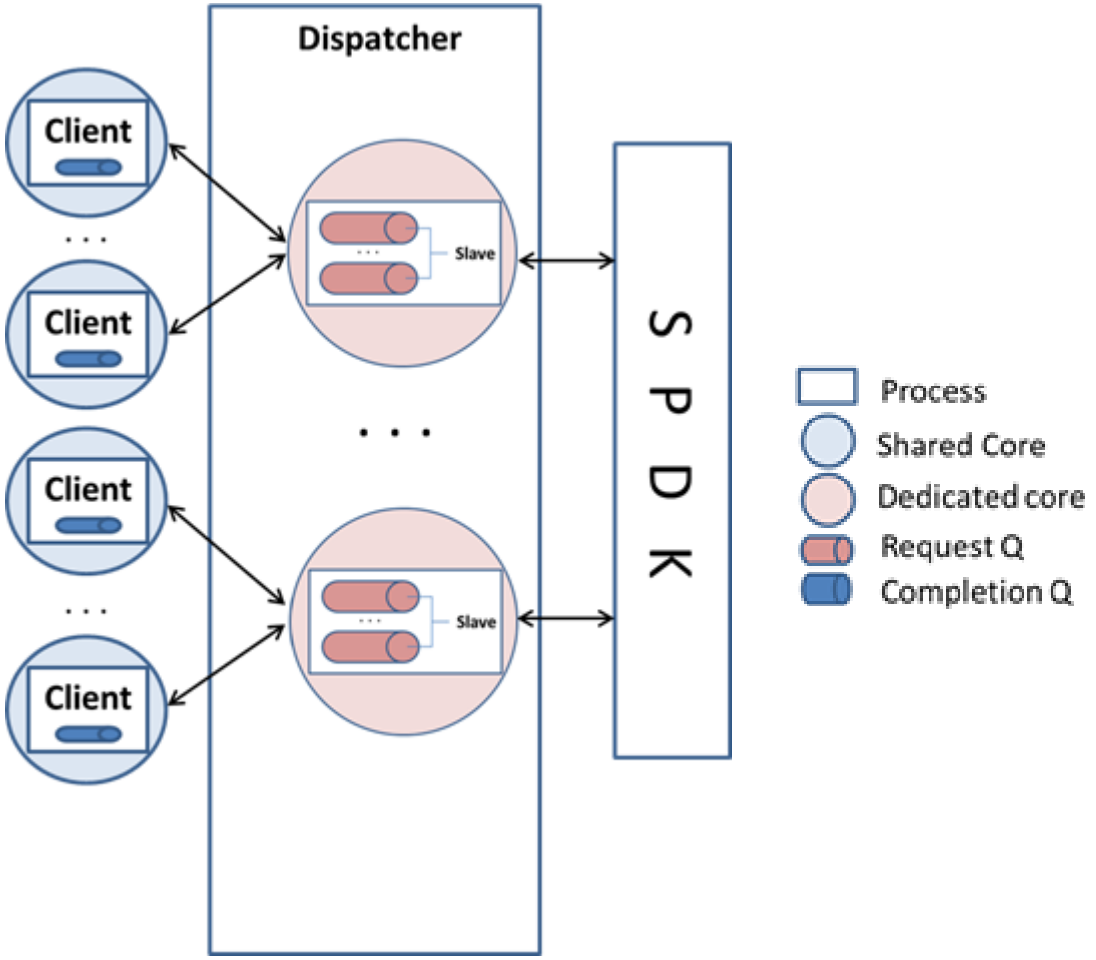
Oracle Dispatcher

- Local IO proxy that runs one or more slaves.
- Each Slave gets its own core.
- Each Slave has one or more request queues.
- Request queue implemented as a lock-free ring.
- Slave processes IO requests from clients.
- Queues IO completions to client completion queues.
- Runs as secondary process to Oraenv primary.



Oracle Client

- Bound to specific Dispatcher Slave/Request queue
- Clients bound to the same Request queue must run on separate cores.
- Clients bound to different Request queues can run on the same core.
- Each client has its own completion queue.
- Submits I/O requests to the Slave request queue.
- Polls I/O completions from its completion queue.



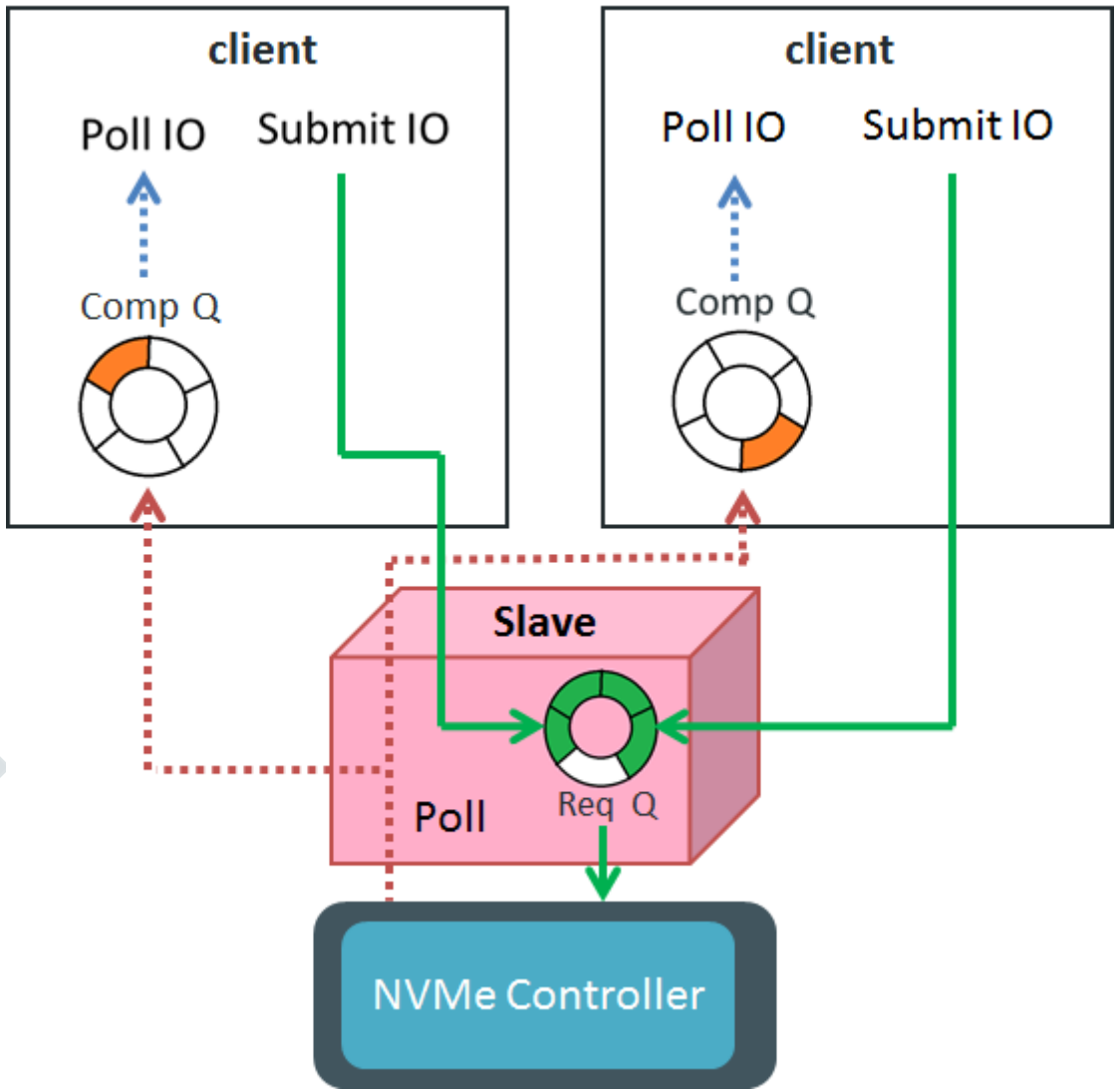
Lock-free Queues

Request Queue

- Multiple producers Single consumer queue.
- IO clients submit requests and Dispatcher Slave processes them.

Completion Queue

- Single producer Single consumer queue.
- Dispatcher Slave queue IO completions and Client reaps them.



Oracle NVM Dispatcher Performance

- 3 node Oracle X6 server based RAC setup.
- 2x 10-core Xeon E5- 2630 v4 processors.
- 6x Samsung PM1725 NVMe PCIe SSDs.
- High redundancy ASM disk group with preferred local (PCIe) fail group.
- Database I/O calibration workload.
- 1.7M random 8 KB read IOPS with 4 dispatcher slaves*
- 2.25M random 8 KB read IOPS with 8 dispatcher slaves*

*Performance data is still WIP.

Program Agenda

1. Challenges
2. SPDK and Oracle Database Architecture
3. OraEnv
4. Oracle Dispatcher
5. **Oracle Database Support Model**

Oracle Database Deployment Model

- Traditionally deployed on-premise in customer data centers.
- Long term release (LTR) supported for multiple (5-7) years.
- Grid Infrastructure (GI) software manages the cluster.
- GI is backward compatible for older database (DB) software releases.
- ASM and Oraenv will be included as part of GI.
- Customers can run multiple DB software versions with the same GI.

SPDK Compatibility and Support

- How do SPDK releases fit into the database deployment model?
- SPDK LTS release is a good start.
- How will critical and security bug fixes be backported to LTS release?
- API and ABI stability is essential to minimize client disruption.
- API and ABI stability limits the complexity involved in backporting fixes.
- How do we achieve the above with SPDK new feature development?


ORACLE®