**OFI Data Storage / Data Access Subteam Weekly telecom – 10/20/2015**

**DS/DA Shared Documents:** <http://downloads.openfabrics.org/WorkGroups/ofiwg/>

**Agenda**

* roll call, agenda bashing
* Need to elaborate on kernel mode use cases for NVM for the Linux Kernel Maintainers Slide Deck.
* Re-cap for the group an offline discussion between Paul and Frank elaborating on HA use case for storage. When using a midplane, this no longer requires a network, hence a kfi API that does not depend on a given network has tremendous value.

**Continuing Discussion on kernel mode use cases**

- re persistent memory for kernel mode – think about an appliance, which may not have user mode at all.

- slide 5 – NVM is not a fabric. Consider separating the last major bullet into two bullets – Emerging Fabrics, Emerging Use Cases. Eliminate the clause ‘emerging fabrics’ from the 4th subbullet.

- new slide 8 – what does ‘user mode access’ really mean? We haven’t drilled down on it yet.

- NVMe is kernel mode only, providing block access to NVM over PCIe

-

“Here are some other features/functionality to consider with kernel enabling of byte addressable storage:

WRITE ATOMICITY:

Beside PMFS and DAX there are other interesting block over pmem usages that might be important to provide API for. Most modern file systems including ext4 and NTFS assume block atomicity is supported by the underlying block storage stack. This means that power failure on a 512B or 4096B block write better result in read back of a full block of old data or a block full of new data but no torn sector with a mix of old and new data which results in silent data corruption.

Once we talk block over pmem the chance of one of 64, 64B stores will result in a torn 4096B block, is quite high. In the intel pmem.io library we invented the BTT which is a simple write block indirection system that is performant and power fail safe SW only mechanism for handling block based write atomicity. I think the Linux kernel stack also includes the BTT but the kernel API and implications to having such functionality should be considered.

RAS ERROR HANDLING:

Also I have architected most of the SW RAS, Uncorrectable error and poison handling for our nvdimm work. There is significant kernel work that needs to happen to deal with error handling and the added complications of dealing with persistent errors.

RDMA W PMEM:

Also I think I talked about the implications of RDMA with Pmem in the DSDA wg before and we may want to refresh that discussion. Again there are kernel implications to supporting this functionality and it's probably worth a discussion.

Anyway topics we can talk about over time if you are interested...

Chet”

- Write Atomicity – write atomicity for block storage is reasonably well understood, but the implications of data loss during a block write should be considered. The Intel mechanism, called ‘BTT’ is a mechanism for capturing a disk block and updating the LBA table once the block has been successfully written.

- May want to add something to the API to allow access to such a ‘BTT’ (write atomicity) capability.

- For example, such a requirement for atomicity when writing to NVM would want to use Completion Level ‘x’ (data is visible and is within the persistence domain).

- It may be useful to make a distinction between writes to NVM that require persistence (e.g. block mode) and those that do not (e.g. kernel is using NVM simply as large, cheap memory).

-RAS Error Handling – how to deal with error handling and the added complications of dealing with persistent errors.

- Poisoning – the mechanism for indicating that a particular region contains an uncorrectable error. When data, which contains an uncorrectable error, is moved to a new location, the uncorrectable error may seem to disappear, even though there is no valid data there. The poison indicator is set to notify the consumer that the data at that location is invalid, even if it does not reflect an error.

**Update on Doug’s SNIA/RDMA Presentation**

- Problem statement – RDMA solutions today don’t take persistent memory into account; no explicit mechanism for forcing RDMA writes to remote pmem to be durable.

- This is generally true of one-sided operations, (even to non-NVM) – there is no way today to notify the responder that data is now visible/available in his memory.

- The slides graphically illustrate the distinction between visibility and durability.

- But even the simpler ‘data visibility’ problem hasn’t been completely resolved, unless the NIC actually participates in the cache coherency protocol.

**Agenda for next meeting**

- Need to elaborate on kernel mode use cases for NVM for the Linux Kernel Maintainers Slide Deck.

- Re-cap for the group an offline discussion between Paul and Frank elaborating on HA use case for storage. When using a midplane, this no longer requires a network, hence a kfi API that does not depend on a given network has tremendous value.

**Webex Recording:** [**Play recording**](https://cisco.webex.com/cisco/ldr.php?RCID=01b0f73207156e2e15bdf9f7a5e3ea80)

**Next regular telecom:**

Next meeting: Tuesday, 10/27/15

8am-9am Pacific daylight time

**NOTE:** We have switched over to using Webex (courtesy of Cisco). The URL for joining meetings is:

[Join WebEx meeting](https://cisco.webex.com/ciscosales/j.php?MTID=m221d8a20185d84b30daa0096aca0f182)

**Join by phone**

+1-866-432-9903 Call-in toll-free number (US/Canada)

+1-408-525-6800 Call-in toll number (US/Canada)

Access code: 201 212 241