SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2016

Building on The NVM Programming Model – A Windows Implementation

Chandra Konamki
Sr Software Engineer, Microsoft

Paul Luse
Principal Engineer, Intel

Outline

- Open NVM Programming Model
- NVML Overview
  - Abstraction
  - Value Added Functionality
  - Performance
- Windows Implementation
  - Team
  - Goals
  - Status

Open NVM Programming Model

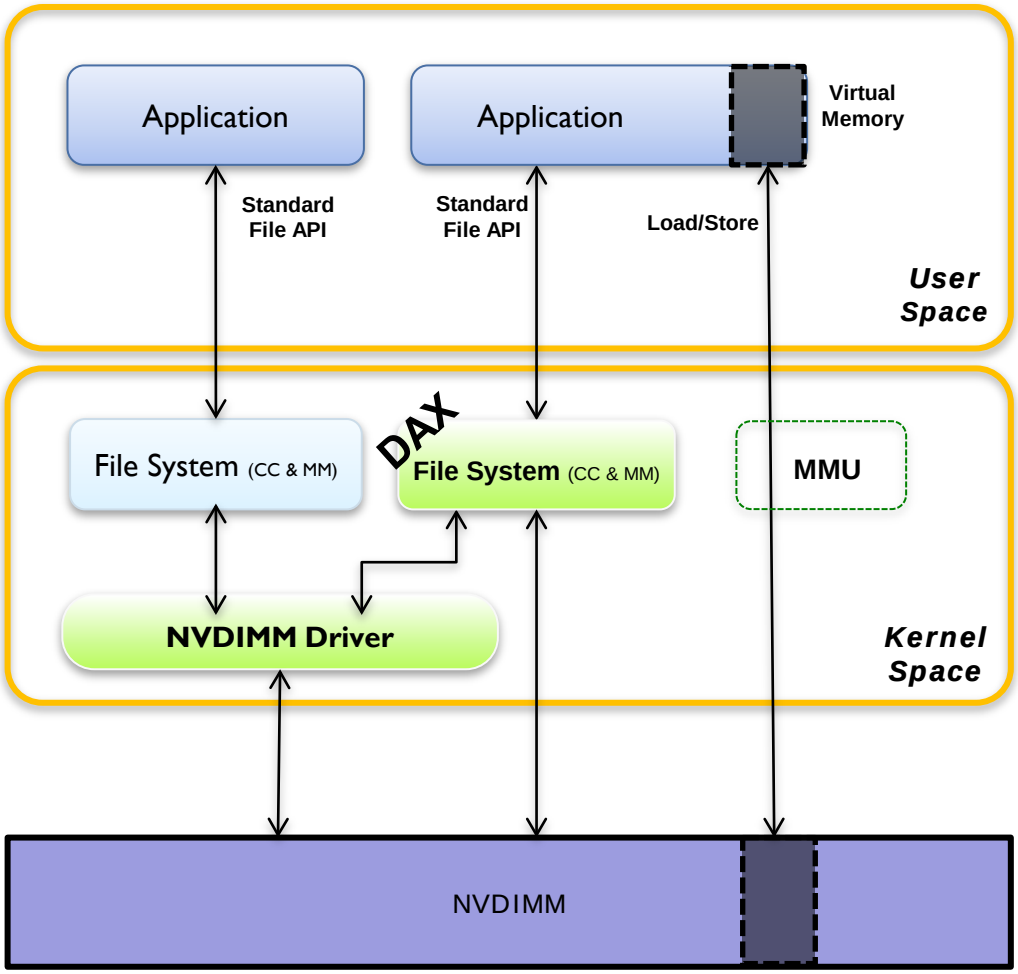
SNIA Technical Working Group
Defined 4 programming modes required by developers

Spec 1.0 developed, approved by SNIA voting members and published

- Interfaces for PM-aware file system accessing kernel PM support
- interfaces for application accessing a PM-aware file system
- Kernel support for block NVM extensions
- Interfaces for legacy applications to access block NVM extensions

http://snia.org/sites/default/files/NVMProgrammingModel_v1.pdf

Open NVM Programming Model



OS Enablement

- Windows PM Support
  - PM support available in all Windows SKUs since the Windows Anniversary Update and Server 2016
    - DAX mode support in NTFS
- Linux Support
  - PM support is available in kernel since 4.4
    - DAX mode support in ext4
- Supports JEDEC-defined NVDIMM-N devices
  - These are commercially available PM devices

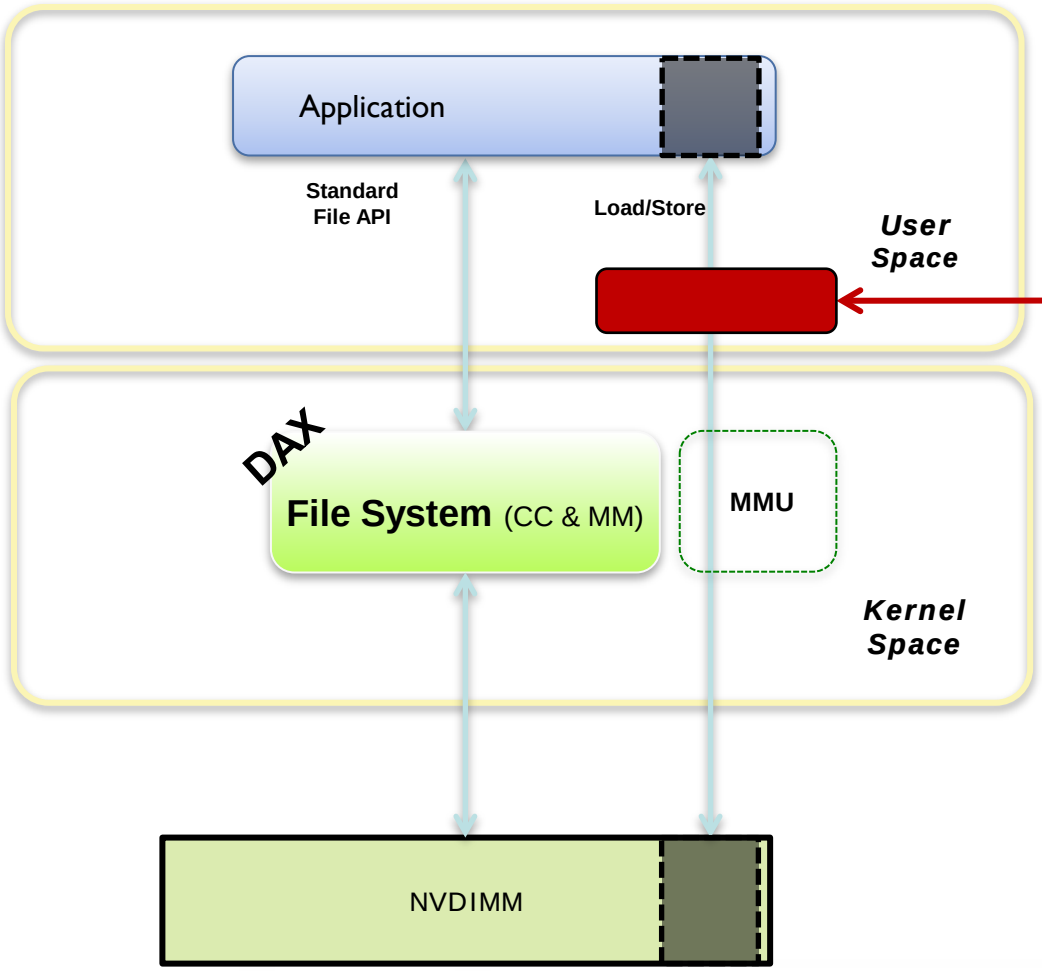
HPE NVM solution for WS2012-R2

- Provides an early access to NVM technologies
  - Kernel support for NVM disk & volumes, standard file APIs
  - Interfaces for PM direct access aware (load/store) applications
    - Semantically identical replacements for CreateFileMapping, MapViewOfFile, etc.
  - Subset of NVML 1.0: libpmem, libpmemblk, & libpmemobj
    - Based on Intel NVML POC code (github.com/krzycz/nvml/tree/win-poc-rebased)
  - No DAX file system support
- HPE whitepaper with more information:
  http://h20195.www2.hpe.com/V2/GetDocument.aspx?docname=4AA6-4681ENW&cc=us&lc=en

Non-Volatile Memory Library (NVML)

- Builds on open persistent memory programming model
  - Includes C/C++ language libraries for:
    - hardware independence
    - transaction support, log file and object store support
    - memory allocation and high availability
  - Facilitates persistent memory application development
    - Using NVML is a convenience, not a requirement
- Library development to address common pain points
  - Libraries for most common asks
  - Open source; Linux & Windows
  - Multiple language bindings and remote replication over time

NVM Libraries



Open Source: http://pmem.io

- libpmem
- libpmemobj
- libpmemblk
- libpmemlog

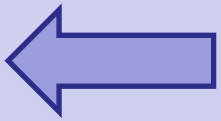
} Transactional

- libvmem
- libvmmalloc

Functionalities of NVM libraries

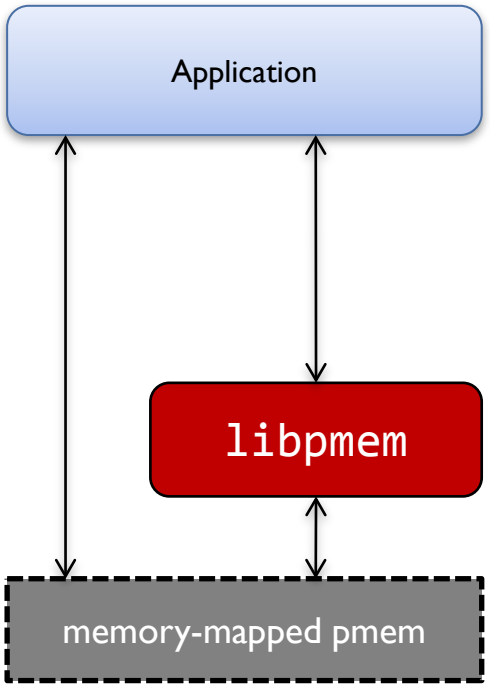
Library	Persistent	Transactions	Allocator	pmem-Aware Locks & Lists
pmem	✓			
pmemblk	✓	✓ Array element		
pmemlog	✓	✓ Log entry	✓	
pmemobj	✓	✓ General purpose	✓	✓
vmem			✓	
vmmalloc			✓	

Standard Load / Store Persistence

```
handle = CreateFile(…);
CreateFileMapping(…);
pmem = MapViewOfFile(…);

strcpy(pmem, “persistent memory");     > 8 bytes, not atomic

FlushViewOfFile(pmem, 18);
FlushFileBuffers(&handle);
```

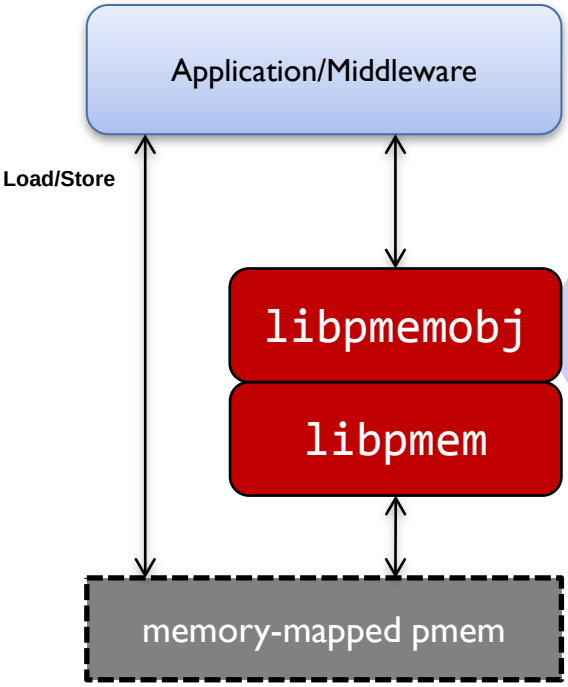
libpmem Load / Store Persistence

```
handle = CreateFile(…);
CreateFileMapping(…);
pmem = MapViewOfFile(…);

strcpy(pmem, “persistent memory");

pmem_persist(pmem, 18);
```



greater than 8 bytes, still not atomic

libpmemobj: Transactional Object Store

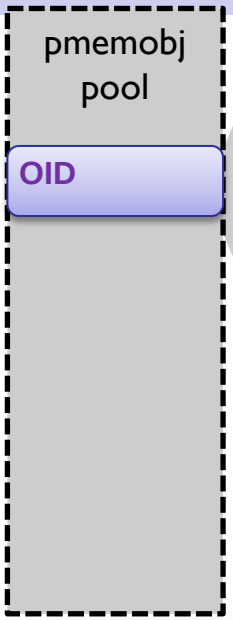


Transactional API:

- Open a file
- Begin/end TX
- Allocate/free
- Commit/abort

Simple libpmemobj Transaction

```
TX_BEGIN_LOCK(pop, TX_LOCK_MUTEX, &op->mylock) {

        TX_STRCPY(op->name, "persistent memory");

} TX_END
```



Object definition:

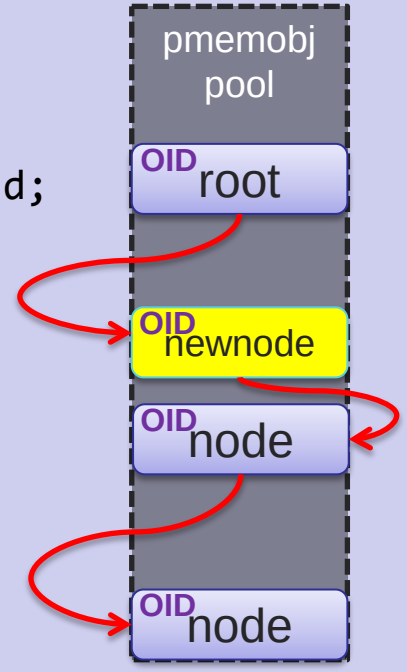
```
struct myobj {
    PMEMmutex mylock;
    char name[NAMELEN];
};
```

Now, we're atomic!

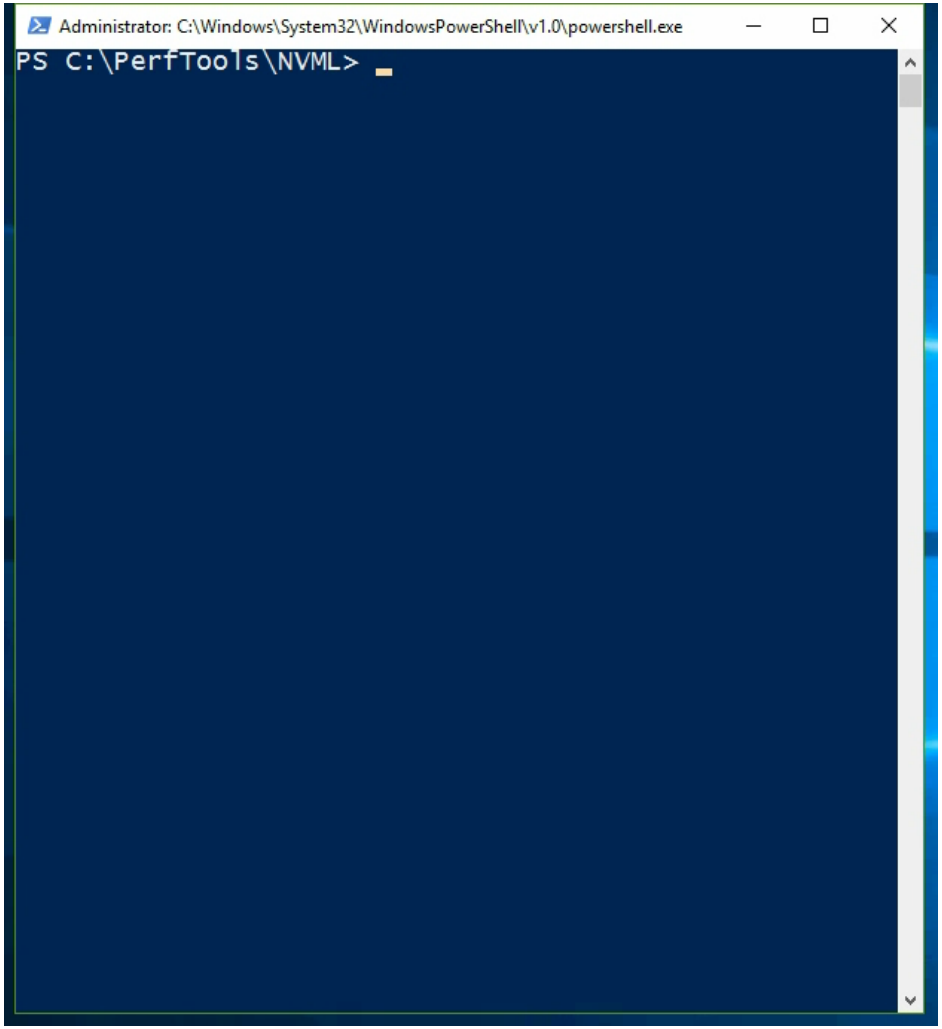
Macro Magic in libpmemobj

(the *assembly* language of pmem programming)

```
TX_BEGIN_LOCK(Pop, TX_LOCK_MUTEX, &D_RW(rootoid)->listlock) {
        OID_TYPE(struct node) newnodeoid =
                                TX_ZALLOC(struct node, 0);

        D_RW(newnodeoid)->data = data;
        D_RW(newnodeoid)->nextoid = D_RO(rootoid)->headoid;
        TX_ADD(rootoid);
        D_RW(rootoid)->headoid = newnodeoid;

} TX_ONABORT {
        perror("transaction failed");
        /* ... */
} TX_END
```



Performance comparison



- Benefits
  - No user to kernel switch
  - More granular
- Libpmem optimized memcpy
  - Smart use of flush vs. non-temporal store

Booth demo by Tobi at IDF, using MSFT internal testing tool, libpmem (prototype) running on Windows Server 2016 (Build 14393).

The Windows Implementation

- Scope
  - NVML 1.0 (excludes remote libraries)
- Objectives
  - Maintain identical APIs for all OSes
  - Maximize shared code and documentation
  - Common repository
  - Foster an Open community

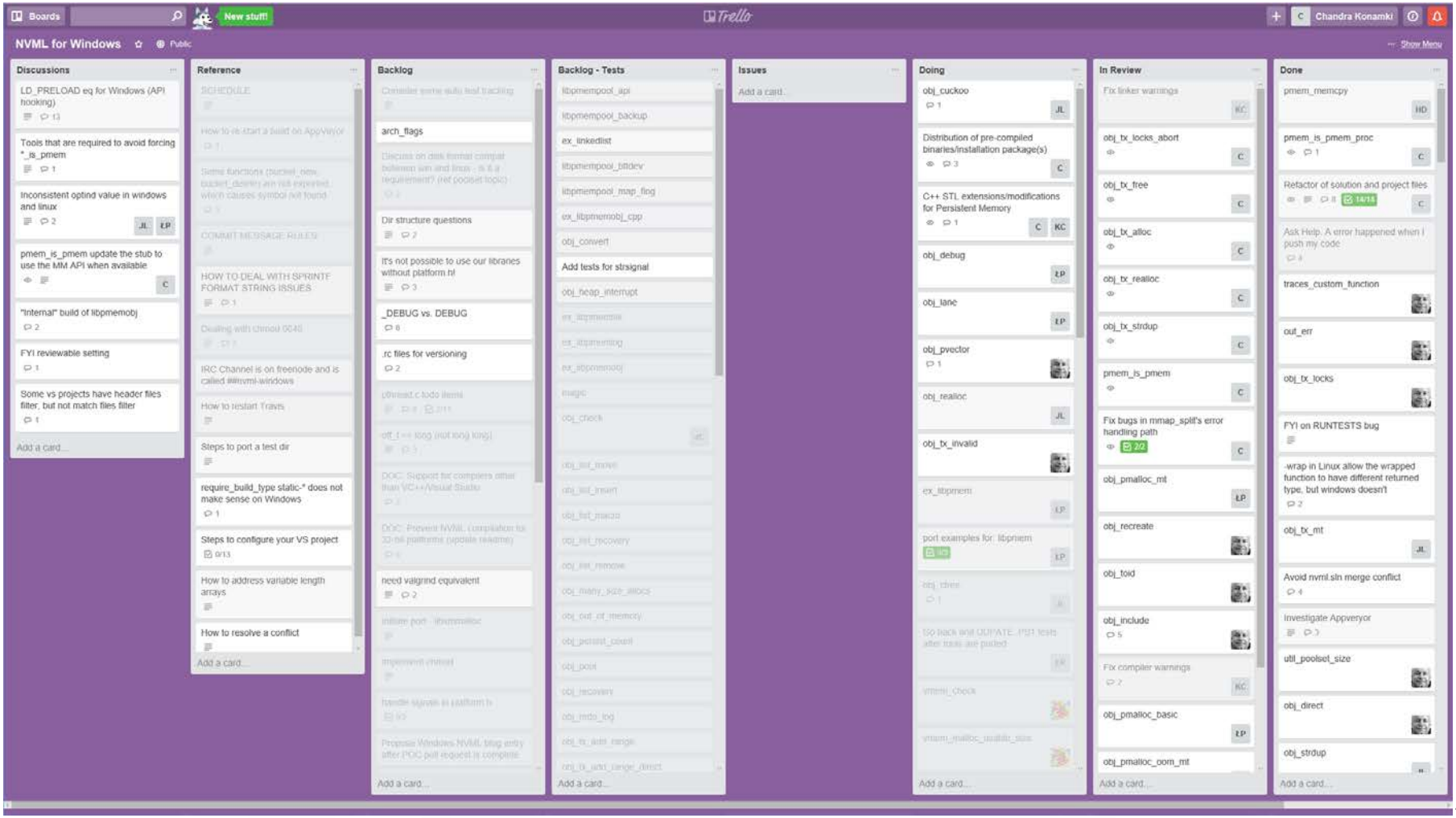
The Community

- Currently: HP Enterprise, HP Labs, Intel, Microsoft
  - We invite and welcome more developers!
  - We are time zone friendly, current community has members from: Texas, Washington, Arizona, Poland, Brazil, Taiwan, California
- Goal
  - Code complete Q4’16

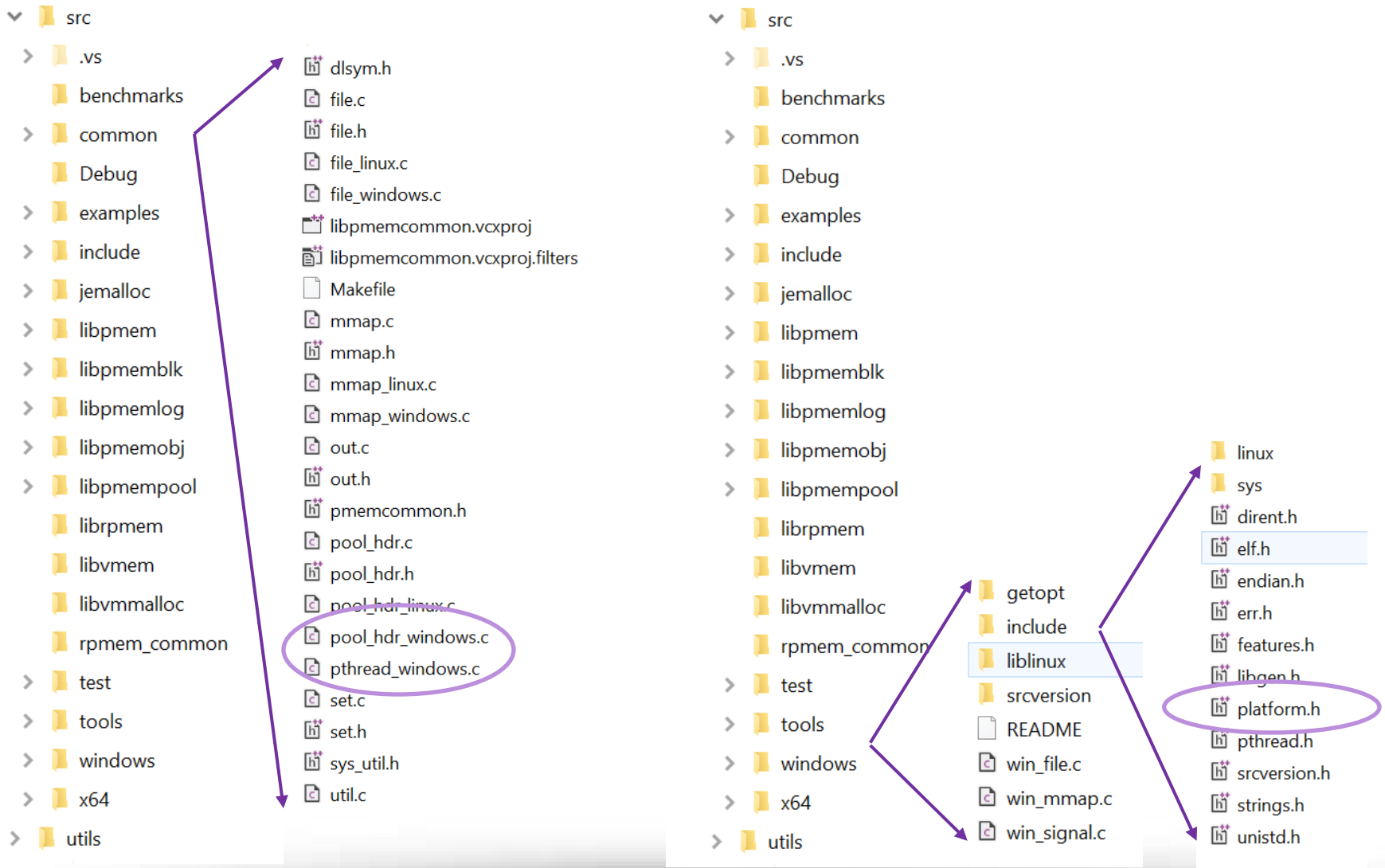
Tools and Process

- Tools
  - Common for all OSes: Github, Reviewable, home grown unit test framework (mostly)
  - Extra for Windows: Visual Studio 2015, AppVeyor, Trello
- Process
  - 1 short weekly meeting
  - IRC on freenode: ##nvml-windows
  - Collaboration (& backlog) using Trello
    - https://trello.com/b/IMPSJ4Iu/nvml-for-windows

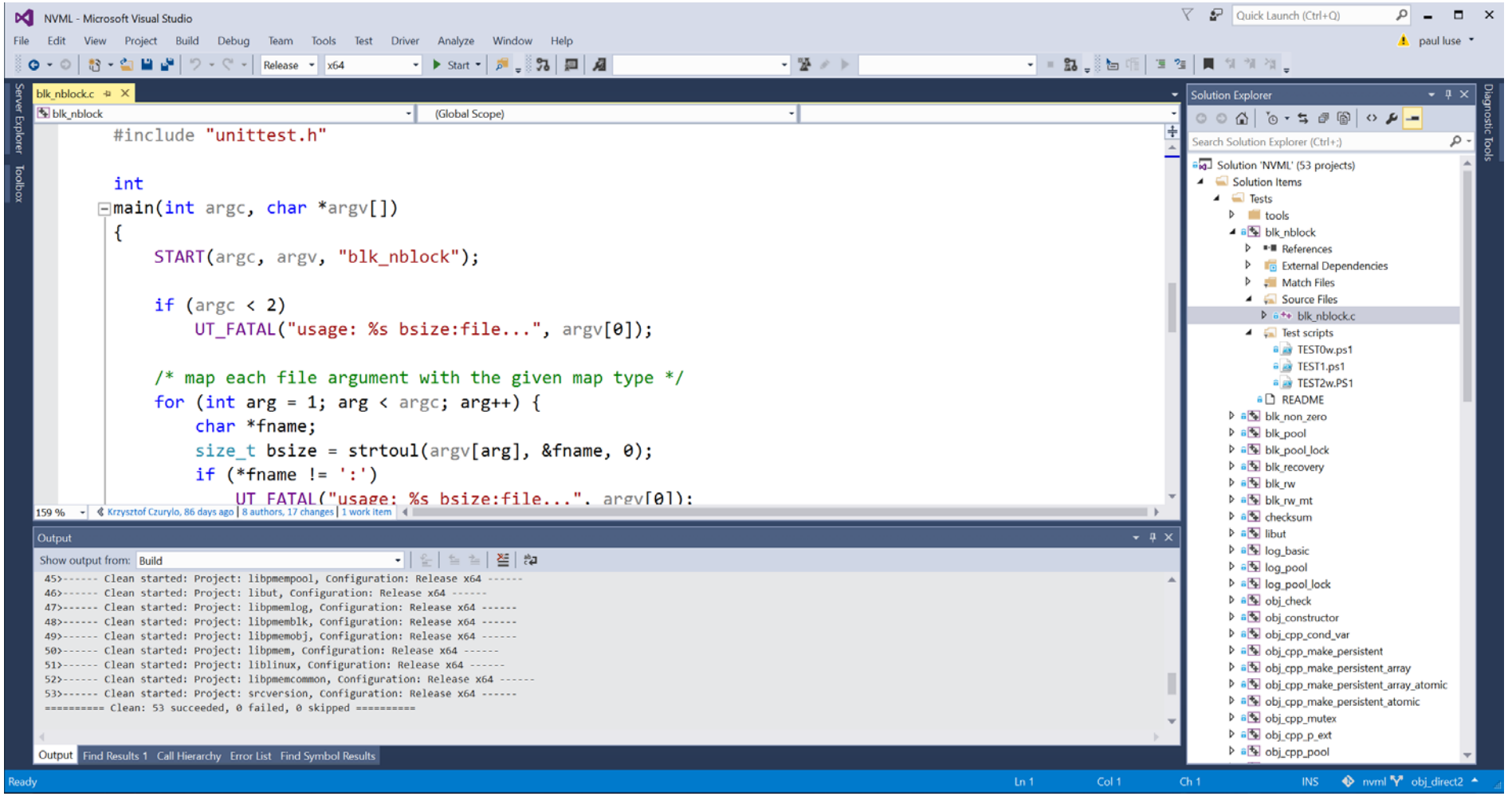
Trello: Collaboration



Multi-OS Support: Repo Layout



Visual Studio Solution



Status and Plans

- Status
  - Active development, nearing the end of completing test code for Windows which might enable additional backlog items for Windows specific changes
- Plans
  - Continue to maintain http://pmem.io
  - Windows version of NVML will be available on github

References

- SNIA NVM Programming Model
  - http://www.snia.org/forums/sssi/nvmp
- Intel Architecture Instruction Set Extensions Programming Reference
  - https://software.intel.com/en-us/intel-isa-extensions
- Open Source NVM Library work
  - http://pmem.io

Disclaimer

- By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.
- You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.
- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
- A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.
- Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.
- The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
- This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.
- Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.
- Intel, the Intel logo, are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.
-
- *Other brands and names may be claimed as the property of others.

Software Disclaimer & Optimization Notice

- By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.
- You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.
- INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
-
- Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.
- This document is provided by Microsoft "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.
- This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804