Persistent Memory Workshop

libpmemobj-cpp hands-on

Contributors:
Jim Fister (SNIA)
Stephen Bates (Eideticom)
Andy Rudoff (Intel)
Igor Chorążewicz (Intel)

https://github.com/pmemhackathon/2019-04-08

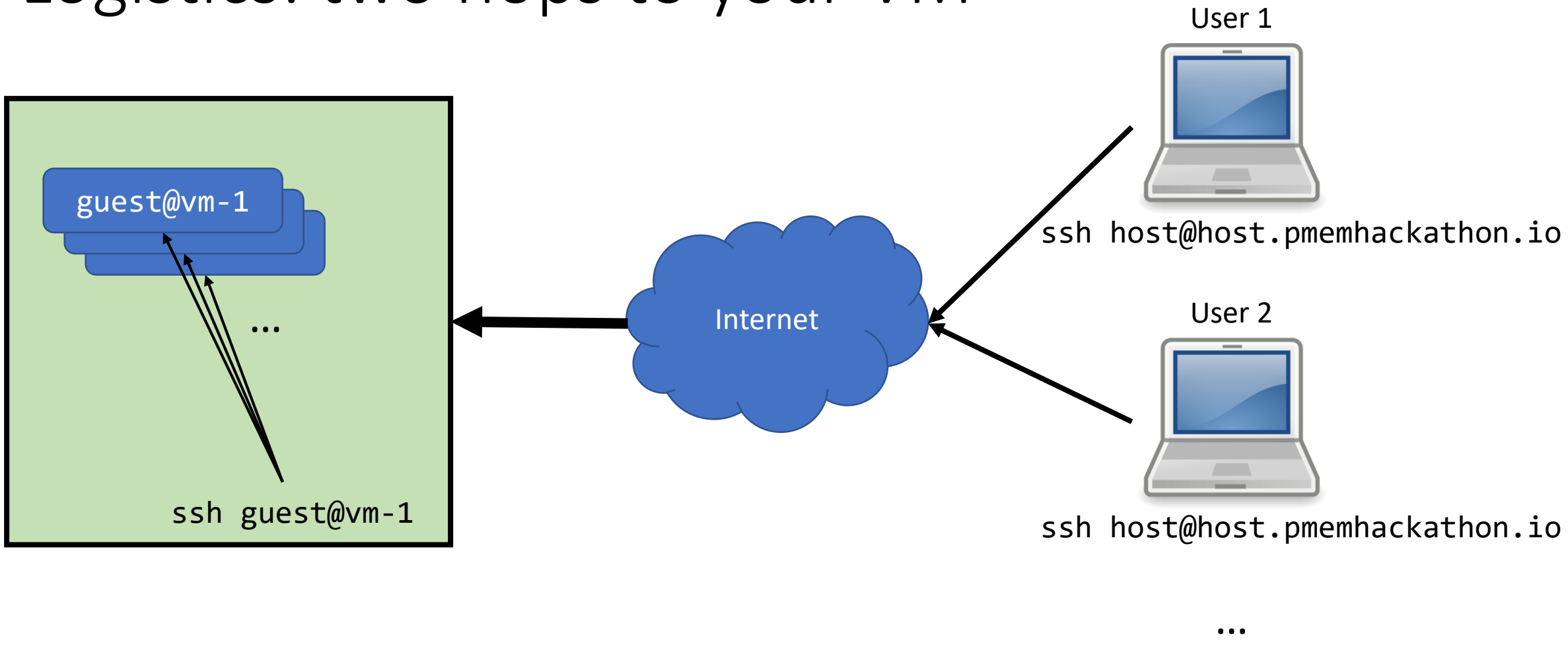
Agenda

- Logistics
  - How to login to your VM & get it ready
- Persistent Memory Platform Support
  - Platform level support
  - Checking out your kernel
  - Finding and configuring your pmem
- Persistent Memory Programming
  - Installing libraries and toold (pmdk, libpmemobj-cpp, valgrind)
  - Using libpmemobj-cpp
  - Finding bugs related to persistent memory programming

What Does “Hackathon” Mean To Us?

- Main goal is to show you how to find, configure, and program pmem
  - All slides are in the GitHub repo
  - All shell commands we type are in the GitHub repo
    - You probably don’t need to write them down
    - You probably don’t even need to type many of them, just cut & paste into the shell
  - Go to https://github.com/pmemhackathon/2019-04-08 to see today’s repo
    - But in a minute, we’ll demonstrate cloning the repo to your VM
- Mostly we will show you how to install stuff and get you going
  - After installing samples, try them out, or write your own
  - We’ll walk through some for everyone, then will walk around & help you

Logistics: two hops to your VM



Make a local clone of the hackathon repo

```
$ cd
$ git clone https://github.com/pmemhackathon/2019-04-08
Cloning into '2019-04-08'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 14 (delta 1), reused 14 (delta 1), pack-reused 0
Unpacking objects: 100% (14/14), done.
$ cd 2019-04-08
$ more README.txt
```

Most of the shell commands we type during demos are in this README.txt

Does your System Support Persistent Memory?

- Does my platform support persistent memory?
  - Your vendor determines this. Buy a system meant for it.
    - Don’t just buy an NVDIMM and plug it into a random system – you need platform support (like BIOS, ADR, power supply). You want validated configurations.

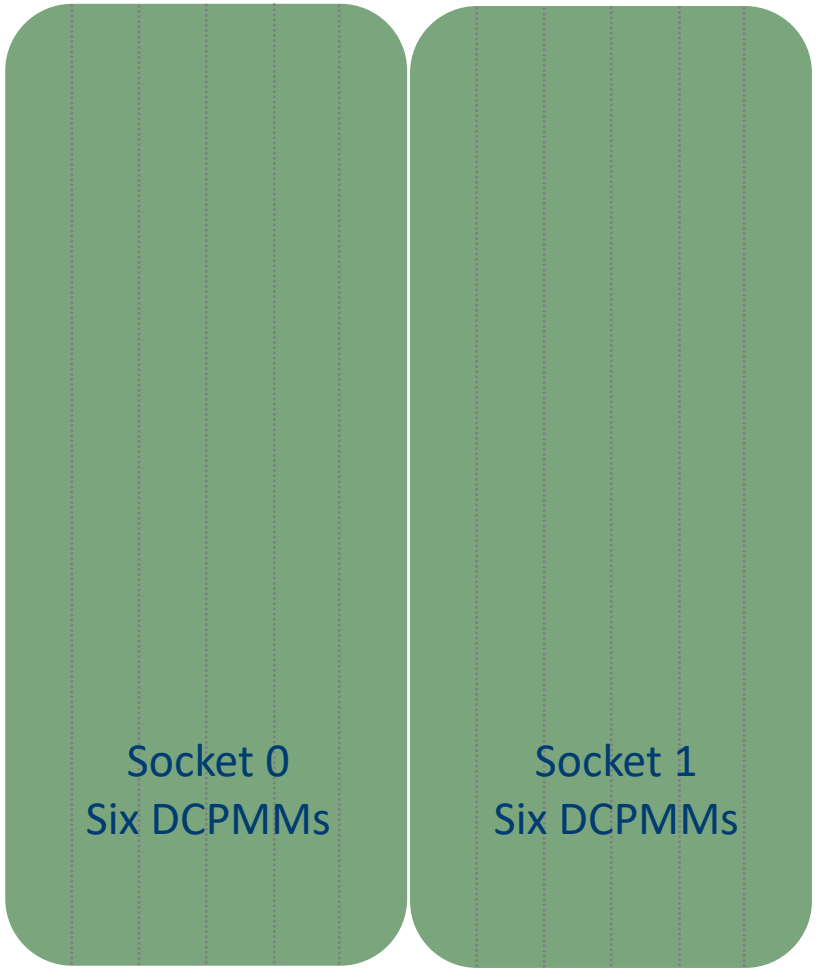
```
ndctl list –BN     # check the "provider" field for ACPI.NFIT
```

- Does my OS support persistent memory?
  - Major OS vendors (and Linux distros) will tell you which version supports it
  - Linux kernel support is enabled in the config file used to build the kernel

```
uname -r         # see kernel currently running
grep -i pmem /boot/config-`uname -r`
grep -i nvdimm /boot/config-`uname -r`
```

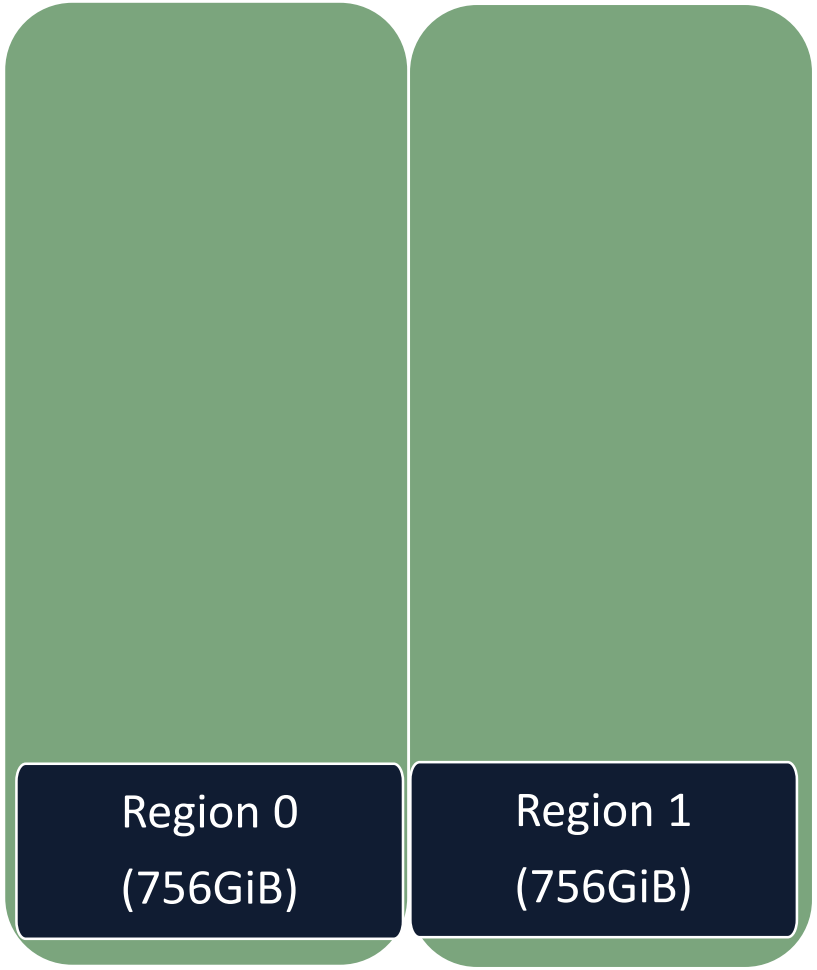
Example: Provisioning Intel® Optane DC Persistent Memory Modules

Hardware



Persistent Memory Modules
(Interleave sets)

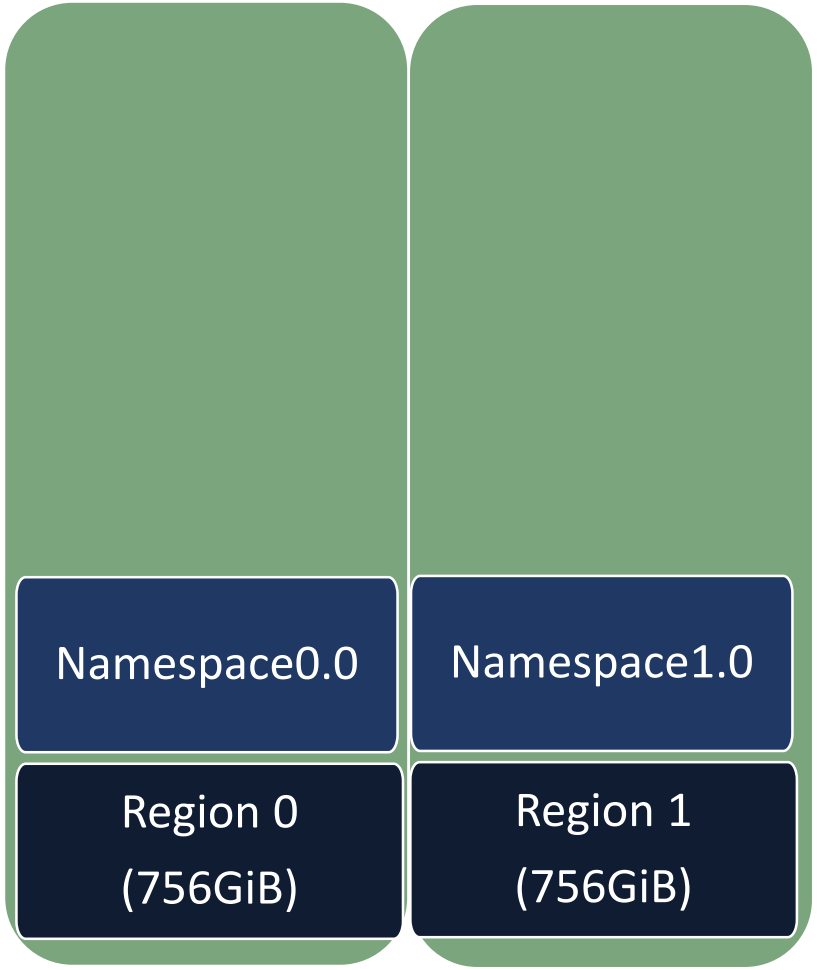
Hardware



Persistent Memory Modules
(Interleave sets)

```
# ipmctl create -goal PersistentMemoryType=AppDirect
```

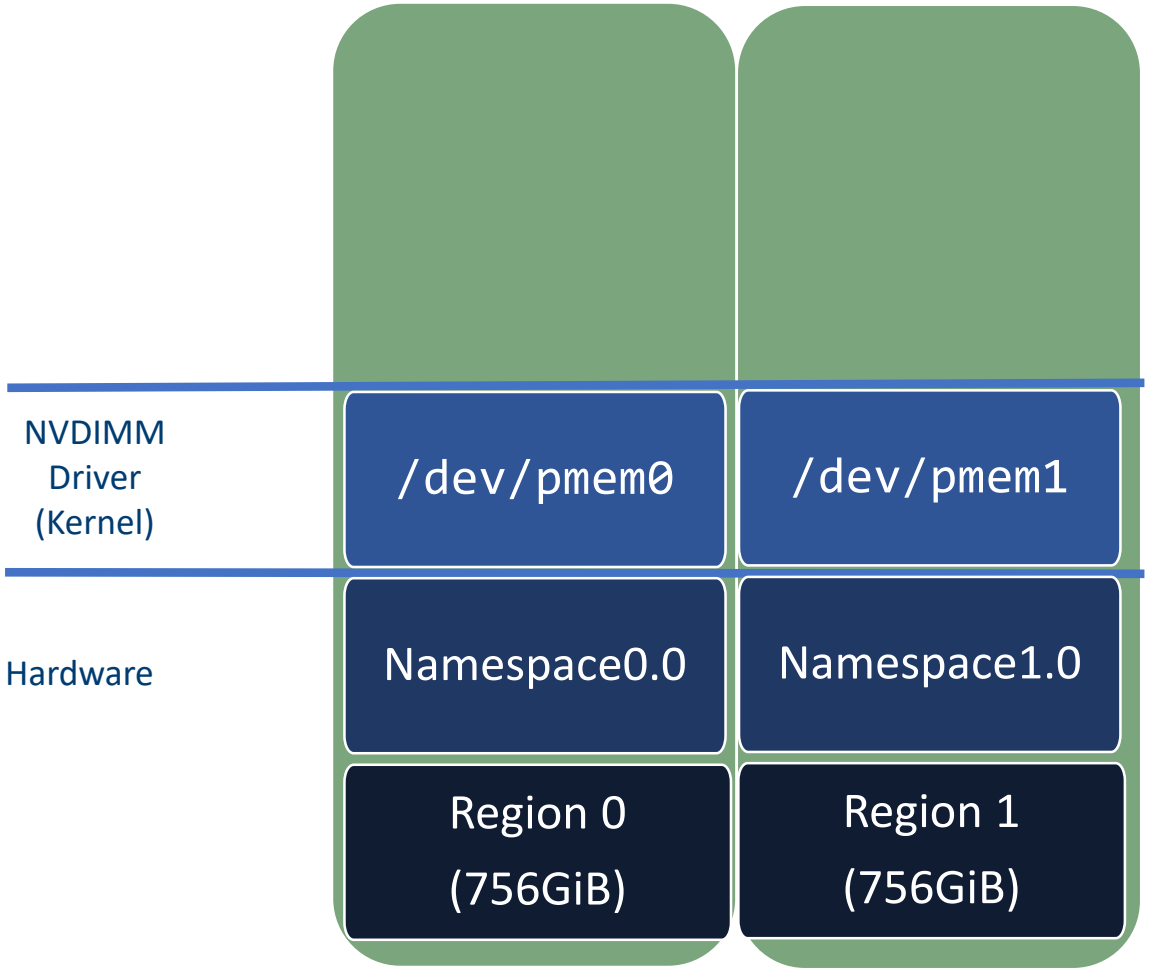
Hardware



Persistent Memory Modules
(Interleave sets)

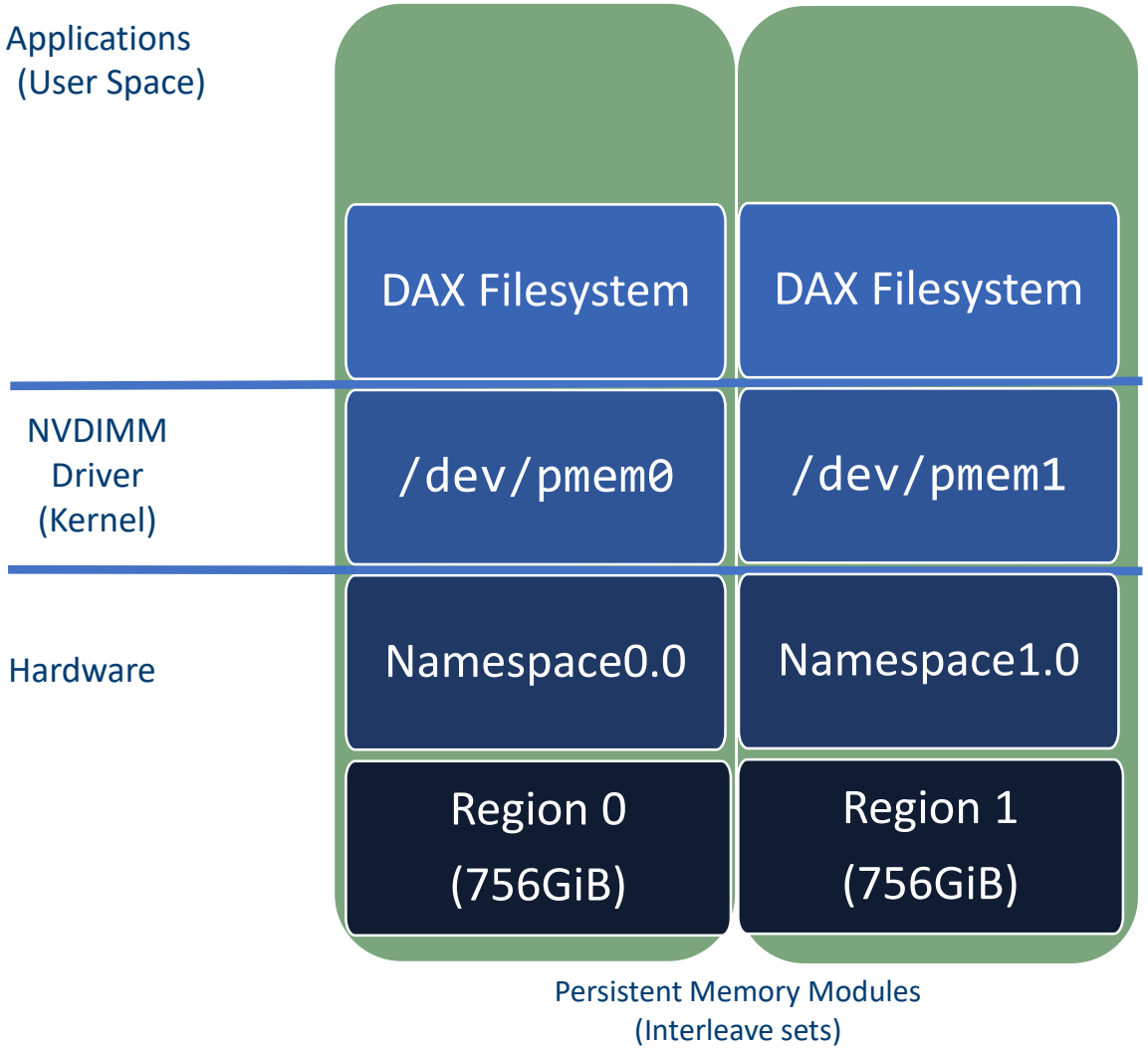
```
# ndctl create-namespace
# ndctl create-namespace
```

```
# ipmctl create -goal PersistentMemoryType=AppDirect
```




```
# ndctl create-namespace
# ndctl create-namespace
```

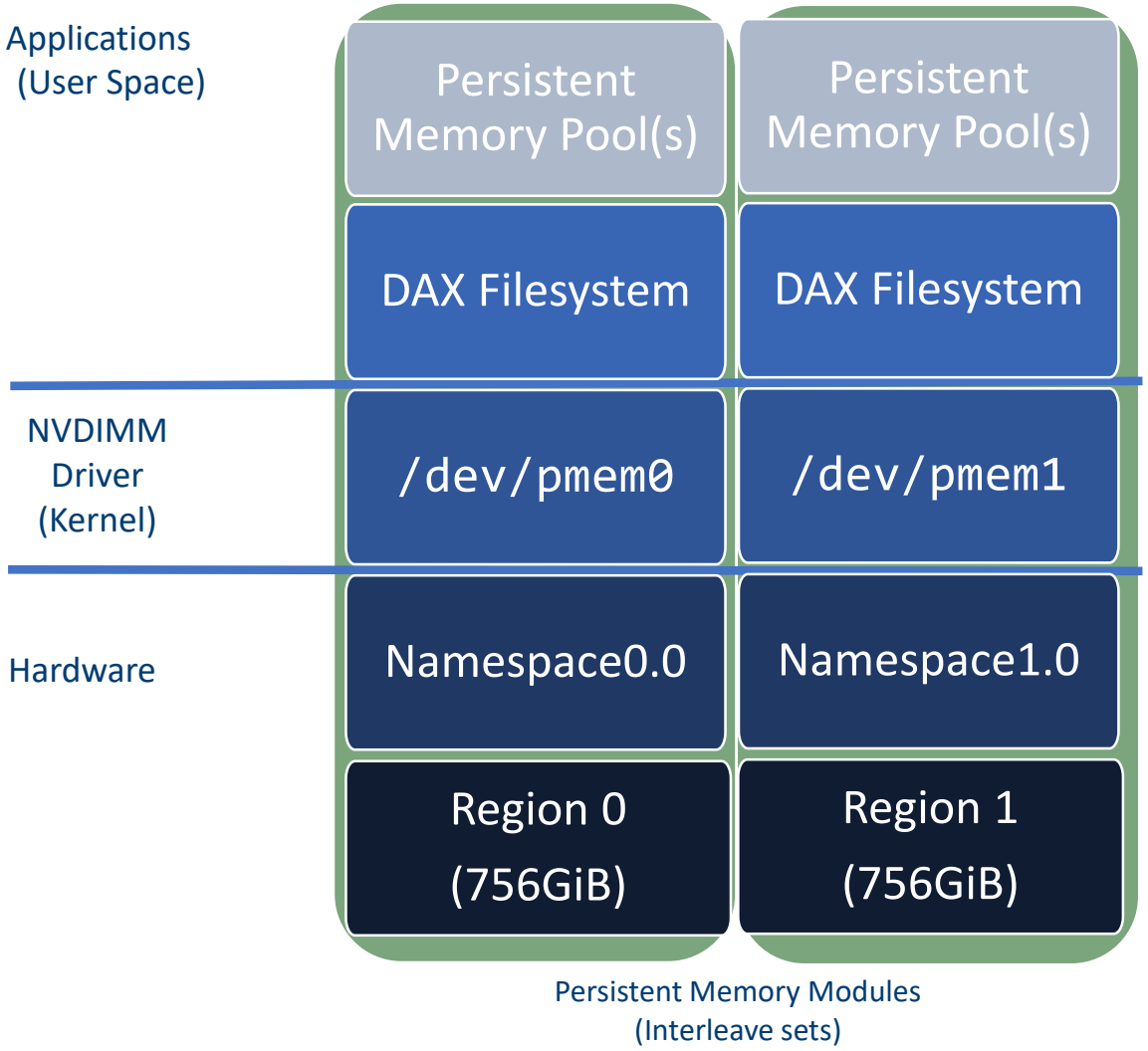
```
# ipmctl create -goal PersistentMemoryType=AppDirect
```




```
# mkfs.ext4 /dev/pmem0
# mkfs.ext4 /dev/pmem1
# mount -o dax /dev/pmem0 /mnt/pmem-fsdax0
# mount -o dax /dev/pmem1 /mnt/pmem-fsdax1
```

```
# ndctl create-namespace
# ndctl create-namespace
```

```
# ipmctl create -goal PersistentMemoryType=AppDirect
```

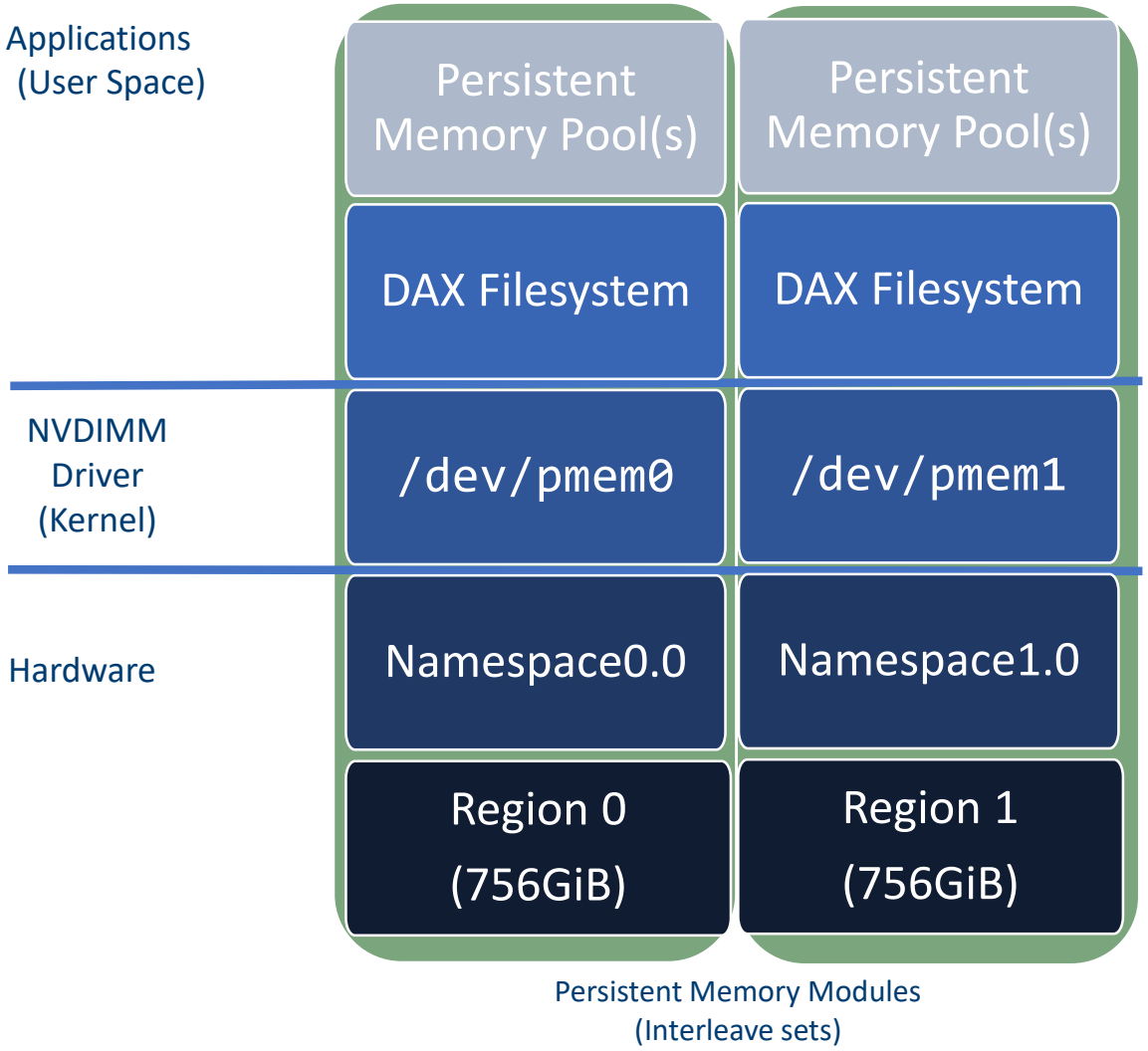



```
$ pmempool create /mnt/pmem-fsdax0/pool0
$ pmempool create /mnt/pmem-fsdax1/pool1
```

```
# mkfs.ext4 /dev/pmem0
# mkfs.ext4 /dev/pmem1
# mount -o dax /dev/pmem0 /mnt/pmem-fsdax0
# mount -o dax /dev/pmem1 /mnt/pmem-fsdax1
```

```
# ndctl create-namespace
# ndctl create-namespace
```

```
# ipmctl create -goal PersistentMemoryType=AppDirect
```




```
$ pmempool create /mnt/pmem-fsdax0/pool0
$ pmempool create /mnt/pmem-fsdax1/pool1

# mkfs.ext4 /dev/pmem0
# mkfs.ext4 /dev/pmem1
# mount -o dax /dev/pmem0 /mnt/pmem-fsdax0
# mount -o dax /dev/pmem1 /mnt/pmem-fsdax1

# ndctl create-namespace
# ndctl create-namespace
```

Vendor Neutral

```
# ipmctl create -goal PersistentMemoryType=AppDirect
```

Vendor Specific

In your VM…

```
$ sudo ndctl list -u
$ sudo ndctl create-namespace –f –e namespace0.0 --mode fsdax

$ ls –l /dev/pmem*

$ sudo mkfs.ext4 /dev/pmem0

$ sudo mkdir /mnt/pmem-fsdax
$ sudo mount -o dax /dev/pmem0 /mnt/pmem-fsdax

$ sudo chmod 777 /mnt/pmem-fsdax    # open up perms for this hackathon

$ df –h
... other file-related stuff works as expected…
```

Essential Programming Background

- Lots of ways to use pmem with existing programs
  - Storage APIs
  - Libraries or kernels using pmem transparently
  - Memory Mode
- This hackathon doesn’t cover the above (too easy!)
  - We assume you want direct access to pmem
  - We show code, but also concepts
  - There are lots of paths you can take, these are just examples

Programming Examples For This Hackathon

- Simple hashtable implementation
- Processing data on persistent memory using map reduce

Resources

- PMDK Resources:
  - Home: https://pmem.io
  - PMDK: https://pmem.io/pmdk
  - PMDK Source Code : https://github.com/pmem/PMDK
  - Google Group: https://groups.google.com/forum/#!forum/pmem
  - Intel Developer Zone: https://software.intel.com/persistent-memory
  - libpmemobj-cpp: https://github.com/pmem/libpmemobj-cpp
  - valgrind: https://github.com/pmem/valgrind
- NDCTL: https://pmem.io/ndctl
- SNIA NVM Programming Model: https://www.snia.org/tech_activities/standards/curr_standards/npm
- Getting Started Guides: https://docs.pmem.io

Installing libraries

- Many libraries discussed during this hackathon are upstream
- Product & libraries are fairly new
  - Lots of recent enhancements, especially around performance
- We show installing the libraries by cloning & building the source
  - This gets you the latest & greatest
  - Also gets you in-tree tools & examples
  - Encourages you to submit issues to GitHub

- For products, we would expect apps to check dependencies
  - Ensuring the libraries installed on the system were validated for that app

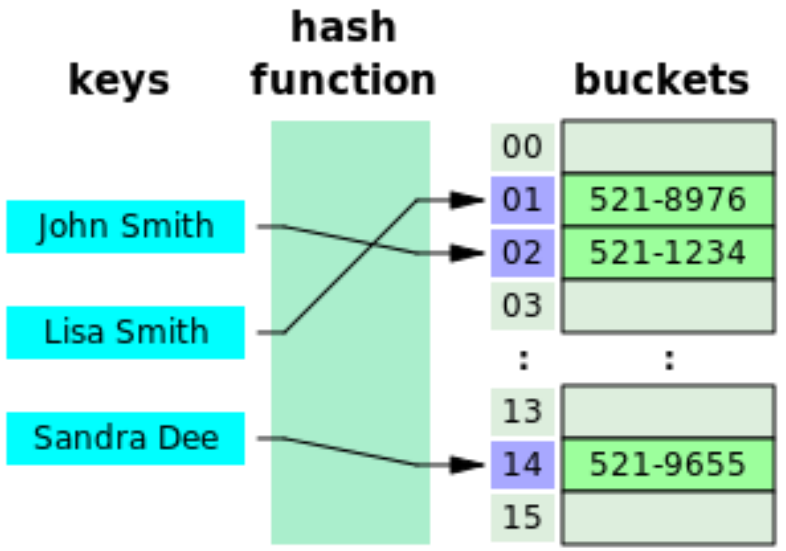
Using libpmemobj-cpp

- Introduction and documentation:
  - http://pmem.io/pmdk/cpp_obj/
- C++ containers
  - http://pmem.io/2018/11/02/cpp-array.html
  - http://pmem.io/2019/02/20/cpp-vector.html
  - More containers under development

Cuckoo hashmap example

Hashtable

- Associative data structure that maps keys to values
- Uses a hash function to compute and index into an array of values.
- If two keys produces the same hash – a collision occures

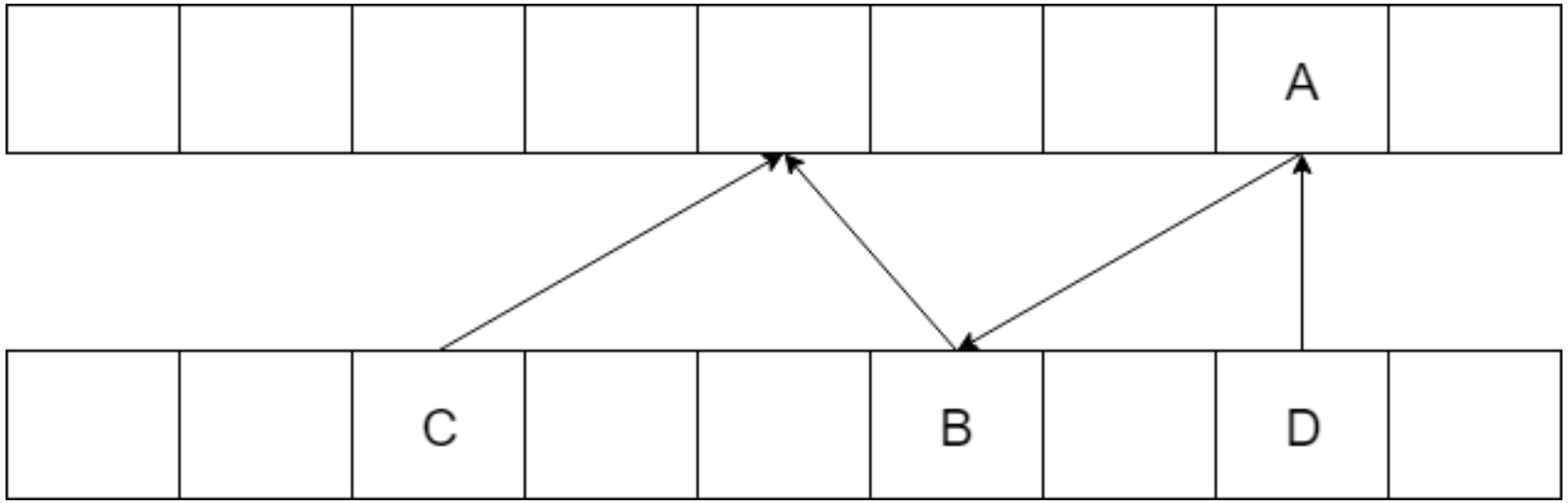


By Jorge Stolfi - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6471238

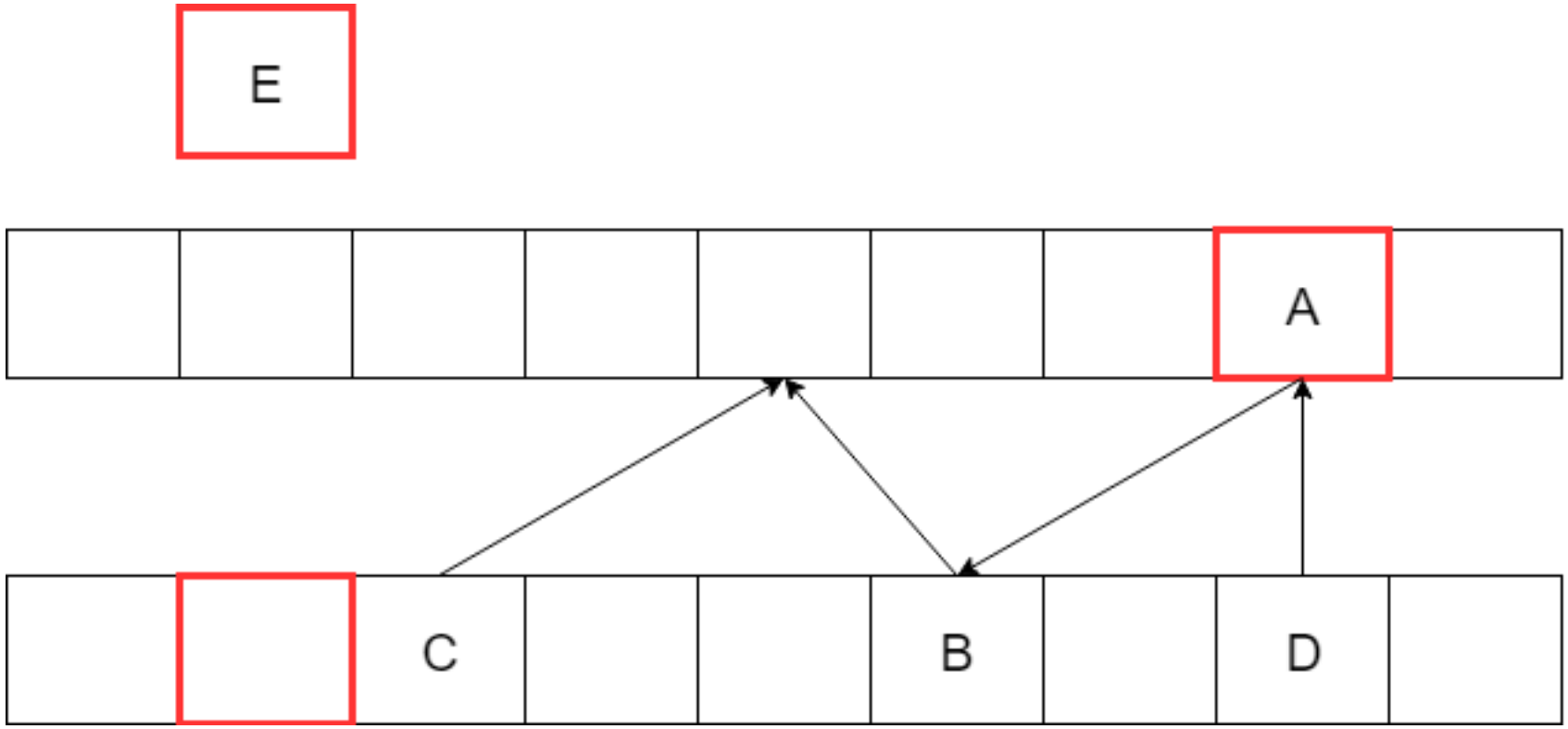
Cuckoo hashing

- Scheme for resolving hash collisions
- Guarantees worst-case constant lookup time
- Inserting new element may push older one it’s alternative location
- Uses two or more hash functions
- Hash table is split into two smaller tables of equal size and each hash function provides an index into one of these two tables

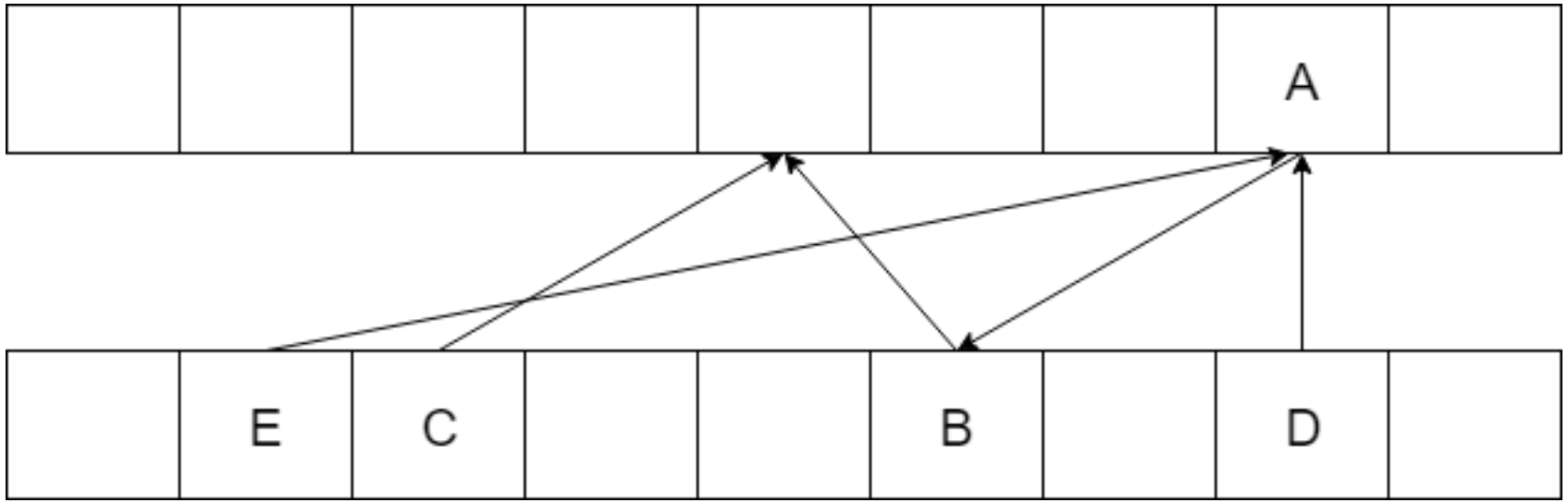
Cuckoo hashing example



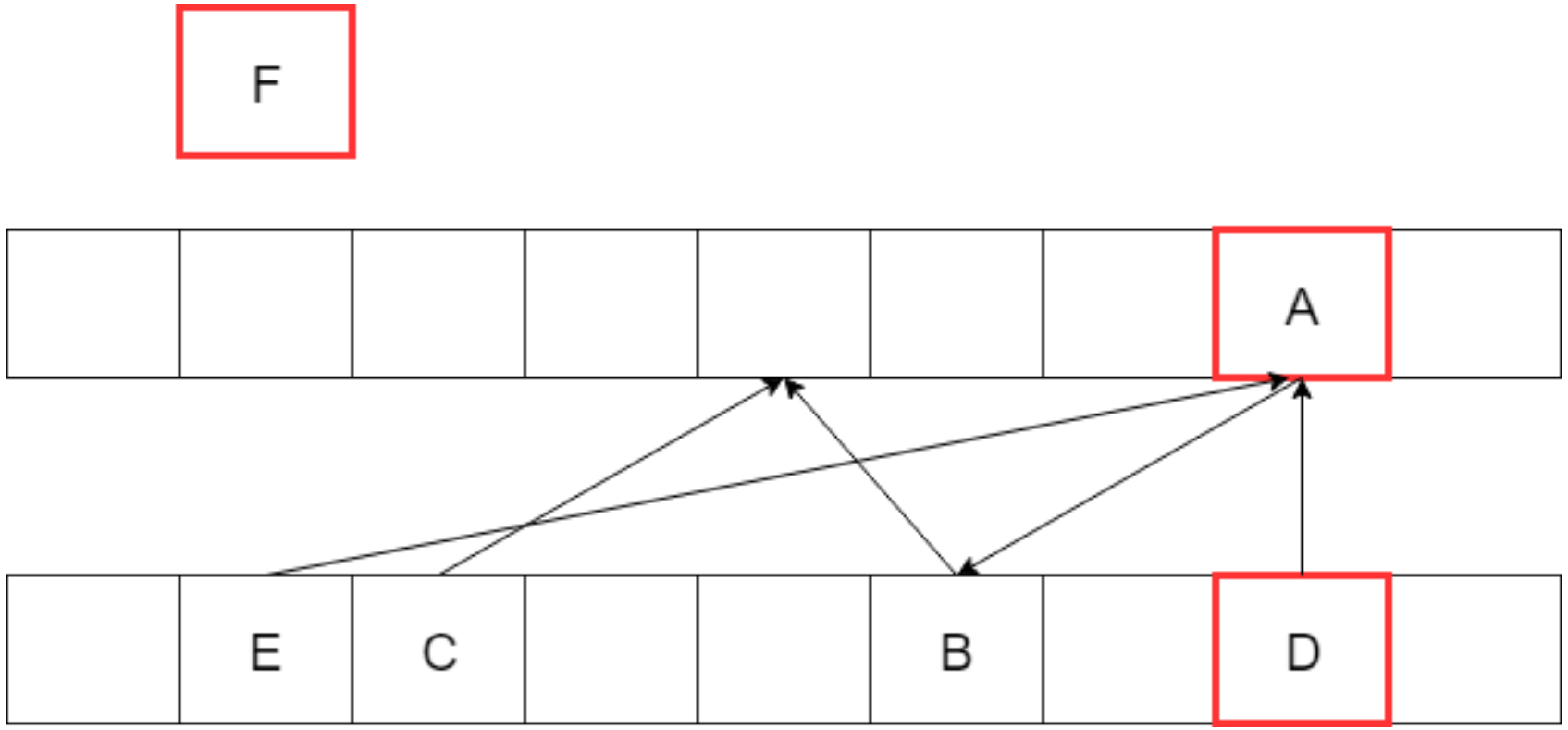
Cuckoo hashing example



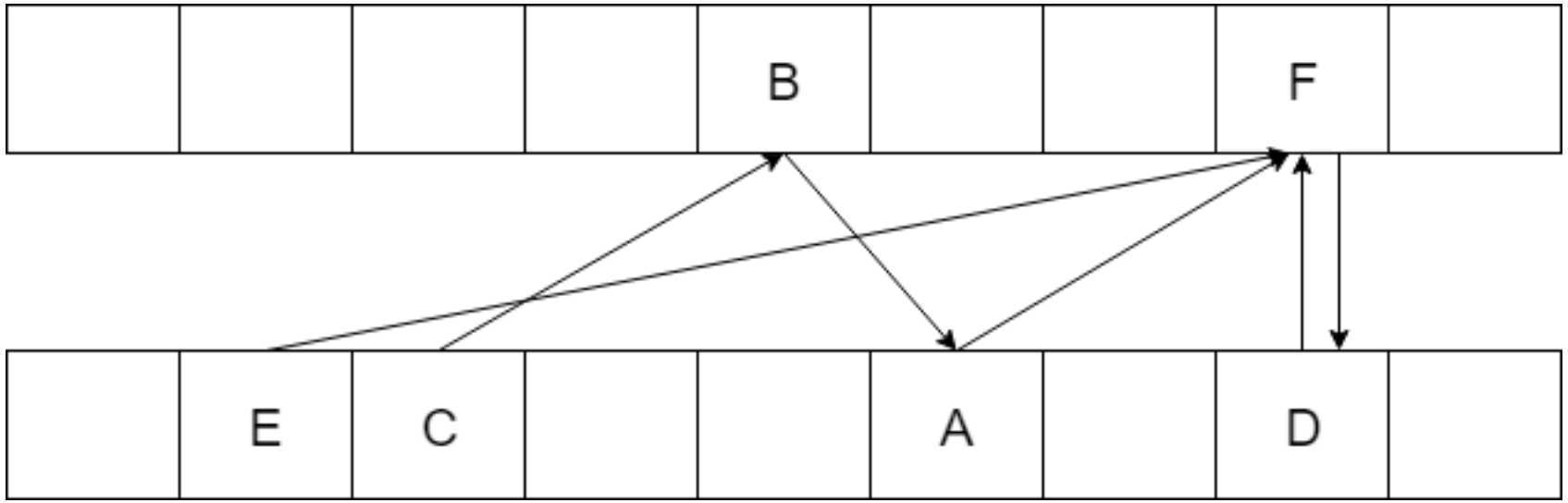
Cuckoo hashing example



Cuckoo hashing example



Cuckoo hashing example



Cuckoo hashmap example

- We will implement a simple hashmap using persistent containers
- Hashmap will have a following API:
  - insert
  - at
  - begin()/end()
  - cbegin()/cend()
- usage (also in README.txt):

```
$ simplekv-simple pool [get key|insert key value|print]
```

Finding bugs related to persistent memory programming

Pmemcheck – persistent memory error detector

- Checks for non-persistent stores
- Checks for overwrites
- Checks for stores made outside of a transaction
- Checks for snapshotting the same object in two different threads
- Can be found here: https://github.com/pmem/valgrind

Pmemcheck – installation and usage

- Installation

```
$ git clone https://github.com/pmem/valgrind
$ cd valgrind
$ ./autogen.sh
$ ./configure [--prefix=/where/to/install]
$ make install
```

- Usage

```
$ valgrind –-tool=pmemcheck [valgrind options] <your_app> [your_app options]
```

Optimizing data for persistent memory

Data oriented design

- The approach is to focus on the data layout, separating and sorting fields according to when they are needed.
- In general – allows for better utilization of CPU cache
- For persistent memory – allows for optimized snapshotting

Data oriented desing in simplekv example

- Using array of indexes and storing key/values in a vector – when iterating over all elements, only one snapshot is required

- Separating key and value – using two vectors instead of one (snapshotting key may not be required while iterating)

Map reduce example

- This example uses MapReduce to count words in text files
- MapReduce is implemented using:
  - std::transform - https://en.cppreference.com/w/cpp/algorithm/transform
  - std::accumulate - https://en.cppreference.com/w/cpp/algorithm/accumulate
- usage (also in README.txt):

```
$ simplekv-word-count pool file1.txt file2.txt ...
```

Links to More Information

More Developer Resources

- Find the PMDK (Persistent Memory Development Kit) at http://pmem.io/pmdk/
- Getting Started
  - Intel IDZ persistent memory- https://software.intel.com/en-us/persistent-memory
  - Entry into overall architecture - http://pmem.io/2014/08/27/crawl-walk-run.html
  - Emulate persistent memory - http://pmem.io/2016/02/22/pm-emulation.html
- Linux Resources
  - Linux Community Pmem Wiki - https://nvdimm.wiki.kernel.org/
  - Pmem enabling in SUSE Linux Enterprise 12 SP2 - https://www.suse.com/communities/blog/nvdimm-enabling-suse-linux-enterprise-12-service-pack-2/
- Windows Resources
  - Using Byte-Addressable Storage in Windows Server 2016 -https://channel9.msdn.com/Events/Build/2016/P470
  - Accelerating SQL Server 2016 using Pmem - https://channel9.msdn.com/Shows/Data-Exposed/SQL-Server-2016-and-Windows-Server-2016-SCM--FAST
- Other Resources
  - SNIA Persistent Memory Summit 2018 - https://www.snia.org/pm-summit
  - Intel manageability tools for Pmem - https://01.org/ixpdimm-sw/