SILICON VALLEY In-Memory Computing SUMMIT 2017

IN-PERSISTENT-MEMORY COMPUTING WITH JAVA

ERIC KACZMAREK
INTEL CORPORATION

LEGAL DISCLAIMER & OPTIMIZATION NOTICE

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.

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

IN-MEMORY COMPUTING

- Many sessions to discuss its value and approaches ☺️
- Definition from a quick search

“In-memory computing is the storage of information in the main random access memory (RAM) of dedicated servers rather than in complicated relational databases operating on comparatively slow disk drives”** - Techopedia.com

I/O

Is this the name/title of the interface? If not, "interface" should be lower case. My guess is that is should read "Behind a SATA/SAS/PCIE interface"

- Behind a SATA/SAS/PCIE interface
- High access latencies
- Low read/write bandwidth
- Data stored as a stream of bytes

TRADITIONAL STORAGE STILL RELEVANT

- Cheap
- High capacity
- Durable

SIDE BY SIDE STORAGE AND MEMORY

	Storage (HDD, SSD, NVMe)	Memory (DRAM)
Capacity	Terabytes	Gigabytes
Durability	Yes	No (through software)
Access	Stream of bytes	Random data access
Bandwidth	~3GBps	~60GBps
Latencies	Microseconds	Nanoseconds

Complement each other

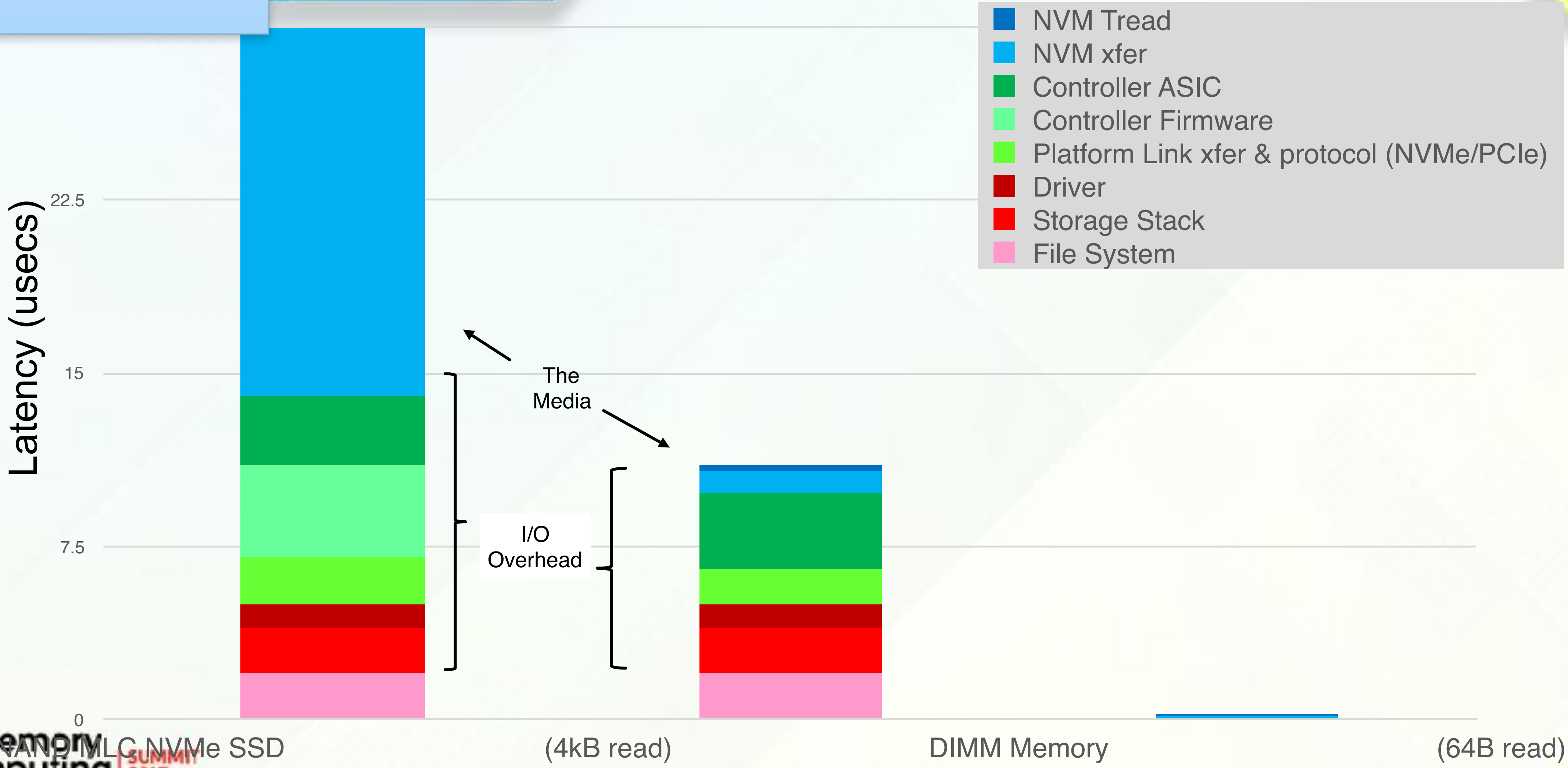
WHAT IF WE CAN HAVE THE BEST OF BOTH

	Storage (HDD, SSD, NVMe)	Memory (DRAM)
Capacity	Terabytes	Gigabytes
Durability	Yes	No (through software)
Access	Stream of bytes	Random data access
Bandwidth	~3GBps	~60GBps
Latencies	Microseconds	Nanoseconds

There was a "click to add" still showing in the background. I removed the blank text box that was housing it.

Vincent, Amber

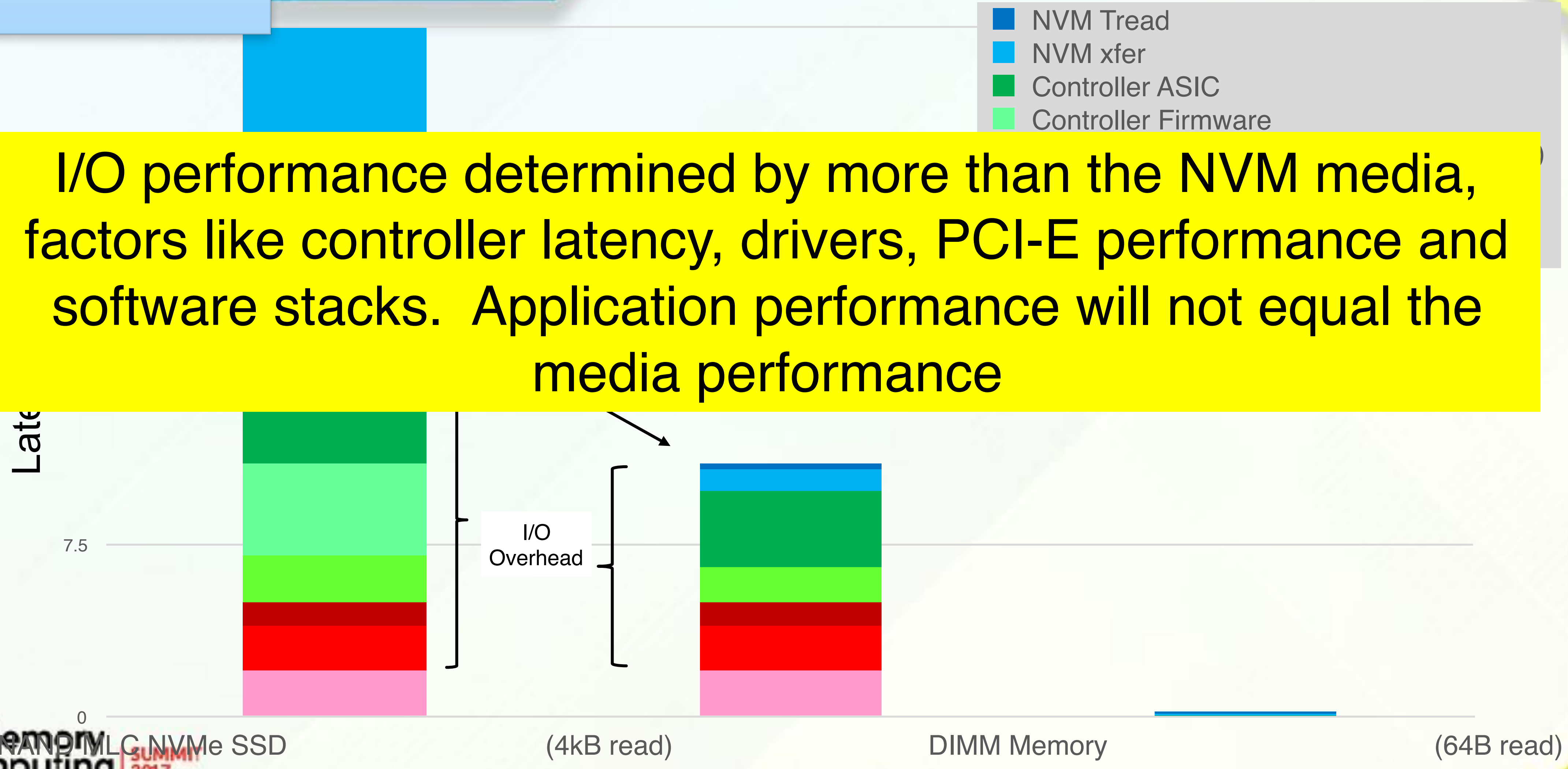
H A TRADITIONAL STORAGE



There was a "click to add" still showing in the background. I removed the blank text box that was housing it.

Vincent, Amber

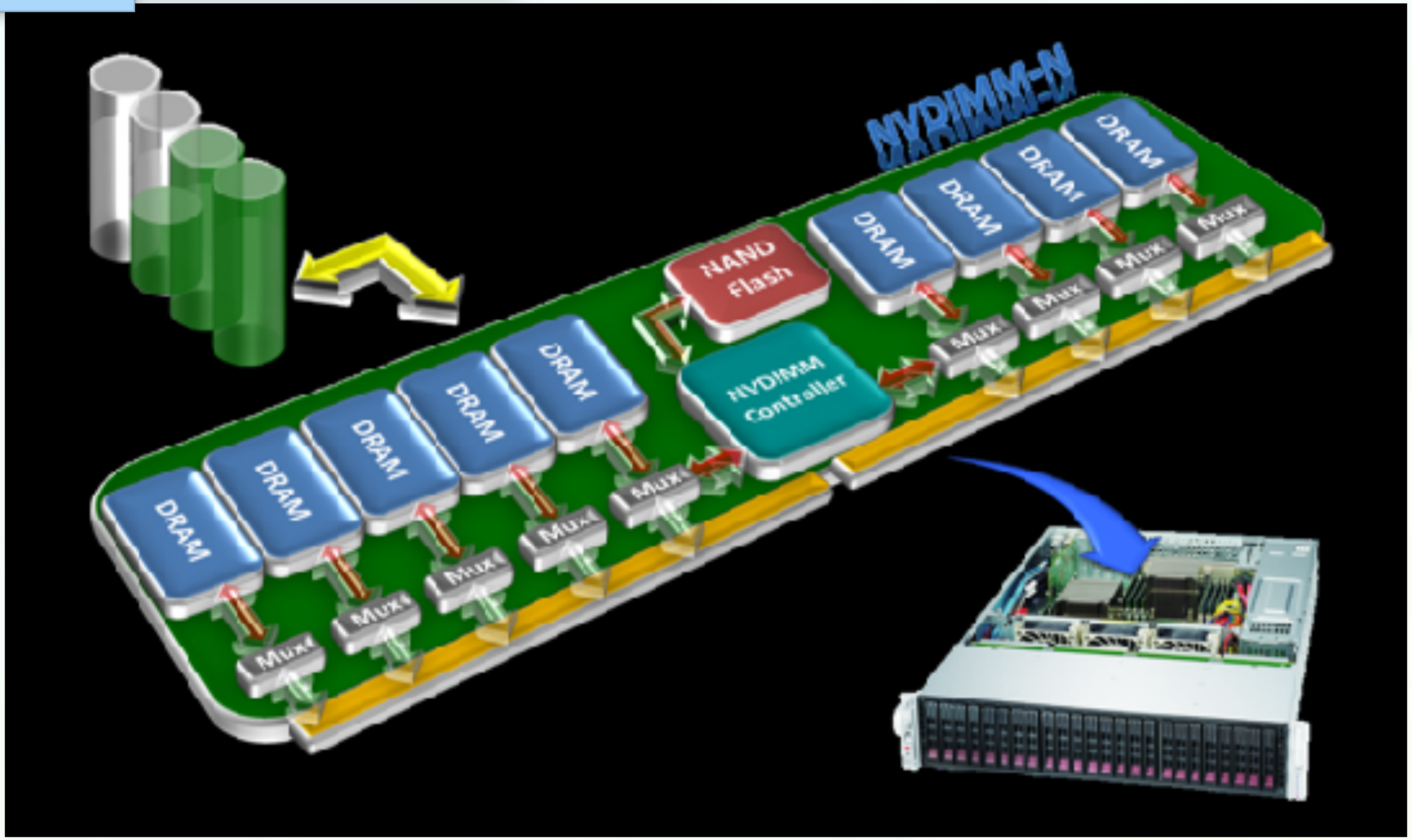
H A TRADITIONAL STORAGE



I/O performance determined by more than the NVM media, factors like controller latency, drivers, PCI-E performance and software stacks. Application performance will not equal the media performance

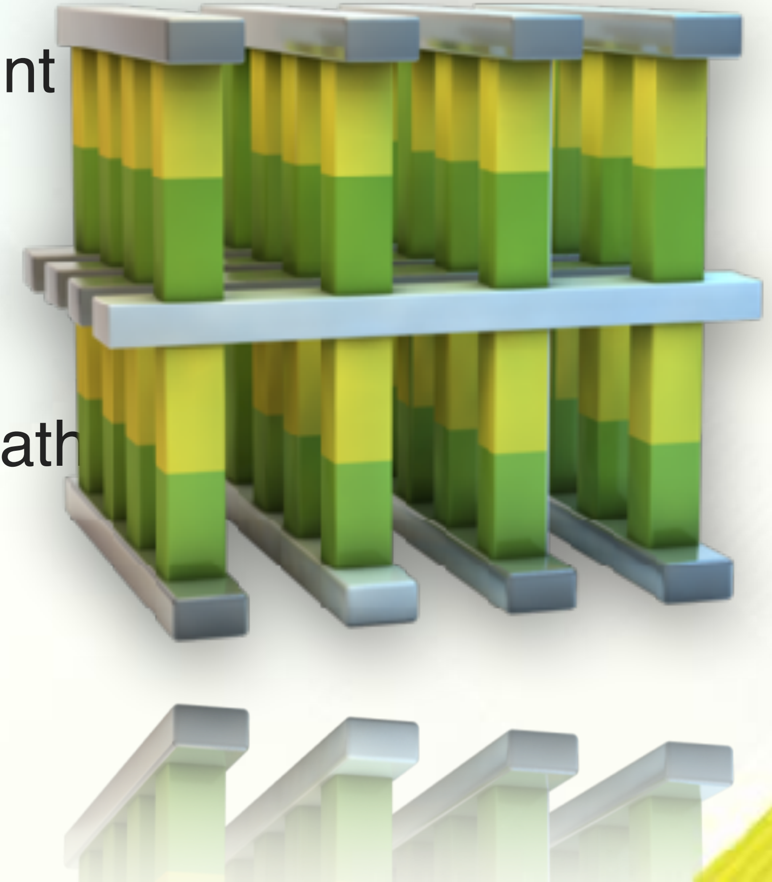
I removed the text box with "click to add content"

G PERSISTENT MEMORY



PERSISTENT MEMORY DEFINITION AND VALUE

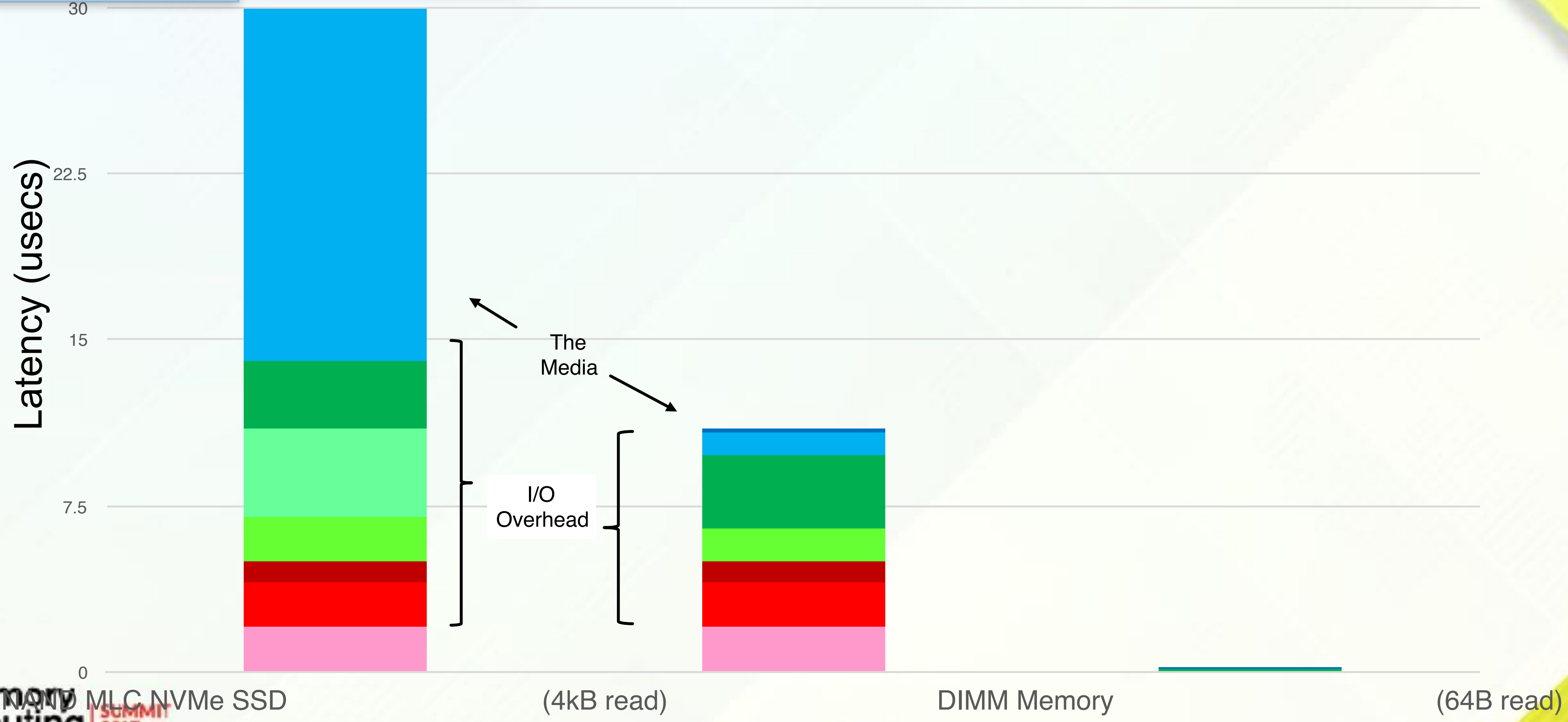
- Memory-like performance – not a DRAM replacement
- Byte addressable – no DRAM footprint
- Durable across applications or system restarts
- Large capacity (terabytes)
- Direct user mode access – no kernel code in data path



I removed the text box with "click to add content"

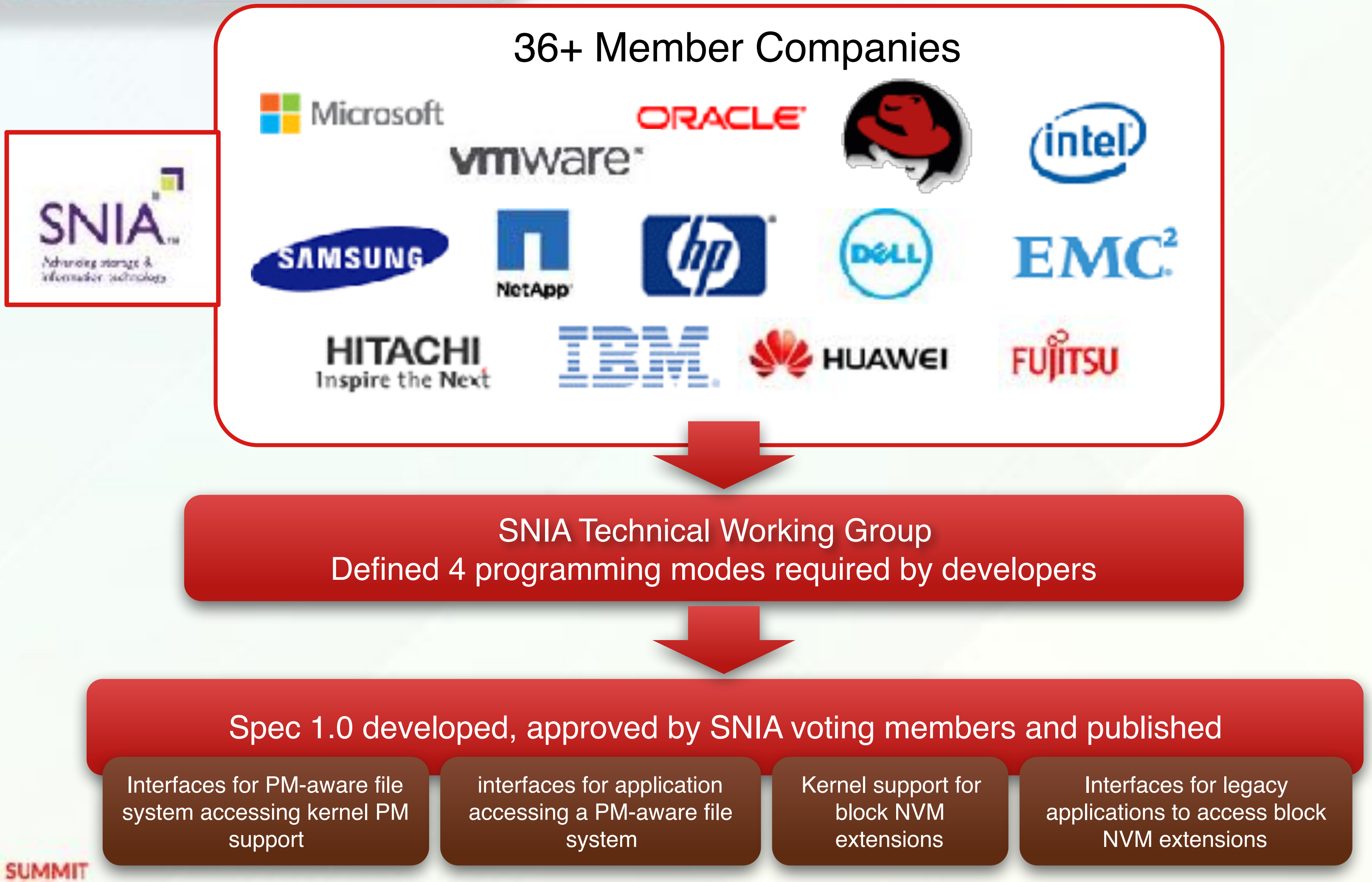
Vincent, Amber
Made grey box slightly bigger for

T MEMORY



In-Memory Computing SUMMIT 2017

OPEN INDUSTRY PROGRAMMING MODEL



36+ Member Companies

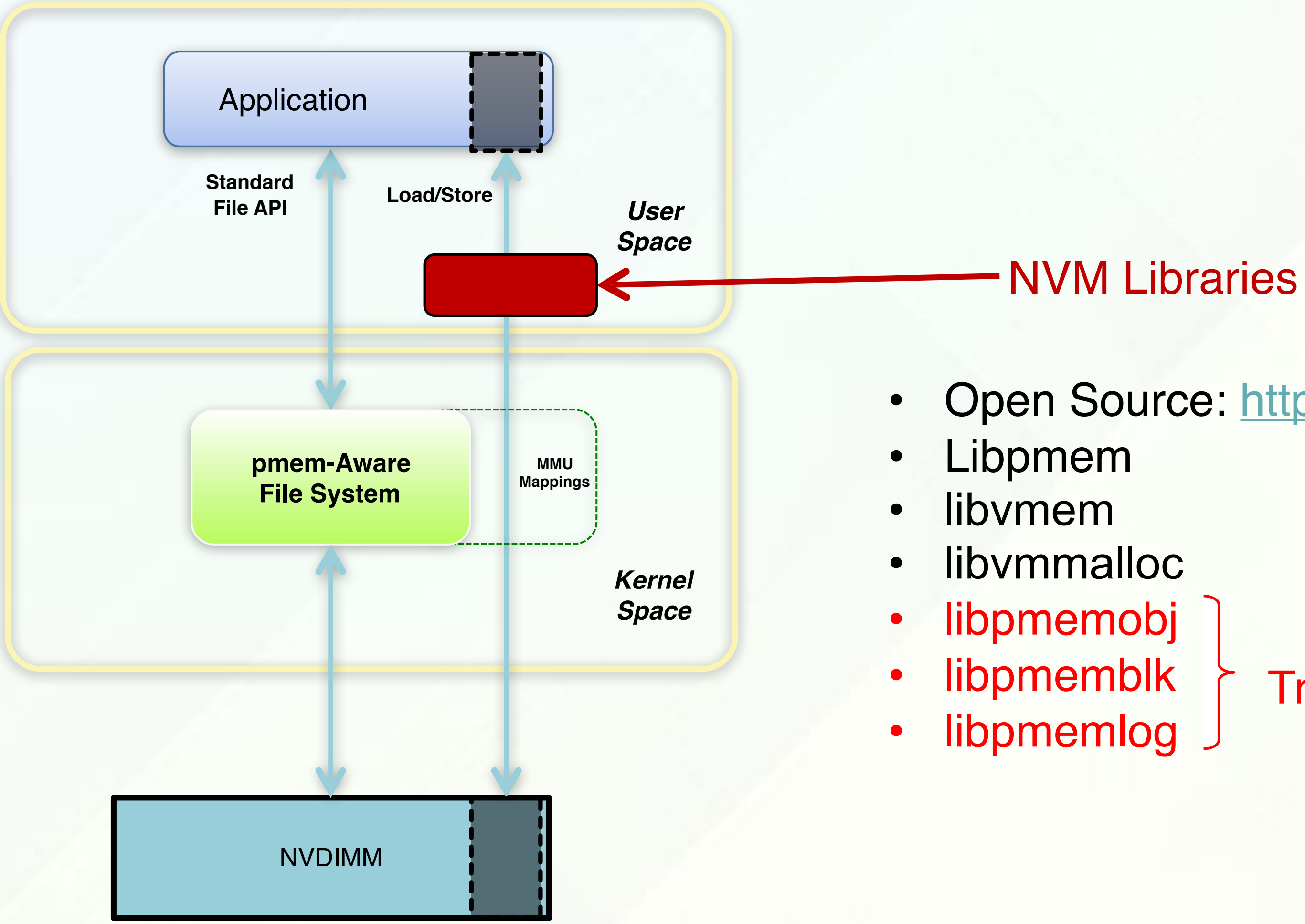
SNIA Technical Working Group
Defined 4 programming modes required by developers

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

C/C++ OPEN SOURCE NVM LIBRARY



- Open Source: http://pmem.io
- Libpmem
- libvmem
- libvmmalloc
- libpmemobj
- libpmemblk
- libpmemlog

Transactional

WHAT ABOUT JAVA

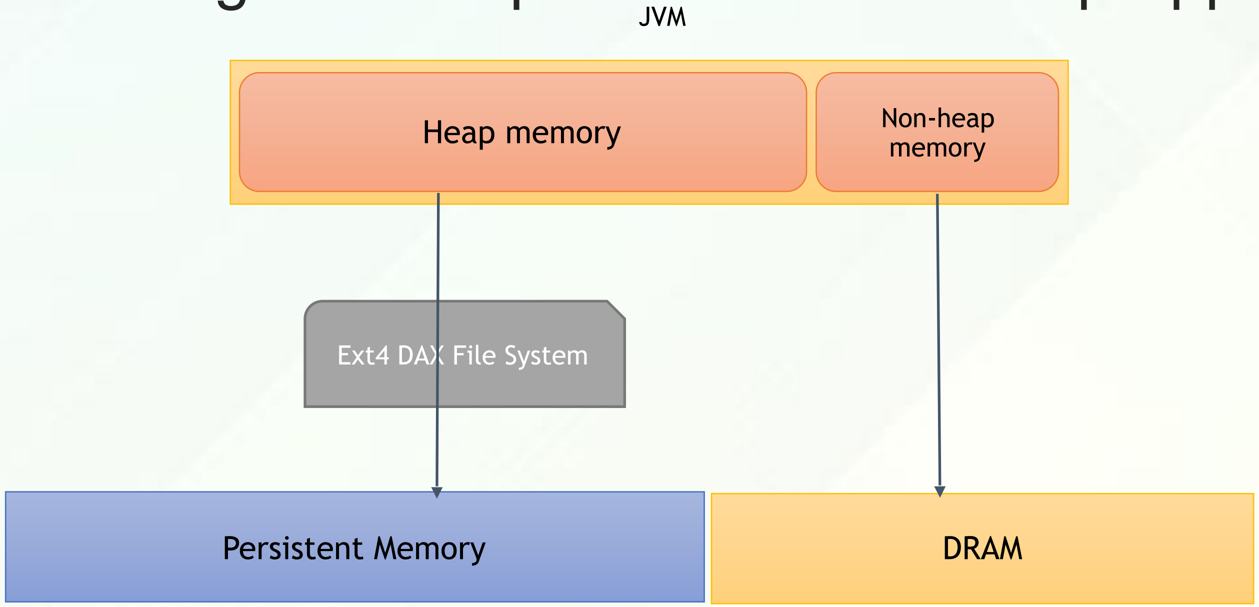
- Broad range of middleware developed in Java or Scala (JVM languages)
  - Ex: Apache Spark, Apache Cassandra, Apache Ignite, Hazelcast IMDG, etc.
- Java abstracts hardware from the developer
- Hooks or hardware access possible through JNI, Unsafe, etc. but
  - Not portable
  - Performance overhead (data marshaling, thread safety, etc.)
  - Might not be supported in future releases

EXPOSING PERSISTENT MEMORY TO JAVA

- Entire Java heap in Persistent Memory
- Heterogeneous Java Heap
- Persistent Collections for Java (PCJ)

ENTIRE JAVA HEAP IN PERSISTENT MEMORY

- Heap memory allocation on persistent memory
- No code changes
- java -Xmx32g -Xms16g -XX:HeapDir=/XPointFS/heap ApplicationClass

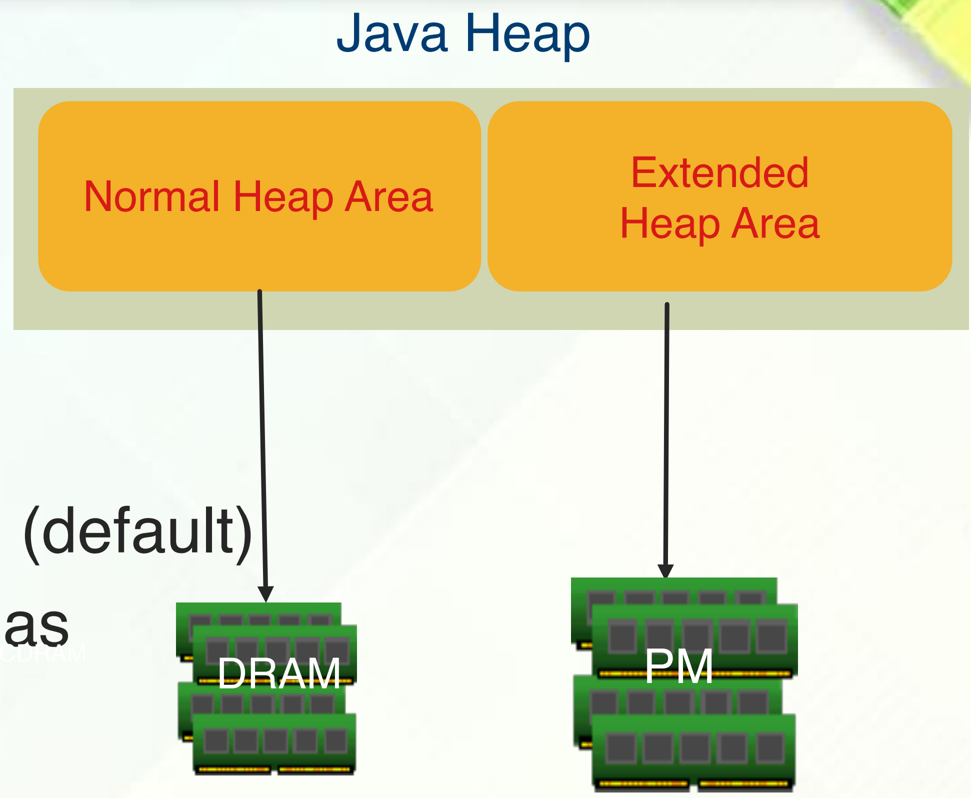


ENTIRE JAVA HEAP IN PERSISTENT MEMORY USE CASES

- In multi-JVM deployments to prioritize Java VMs. (ex: Oracle Fusion Apps)
- Applications which can benefit from large memory
- OpenJDK JEP: https://bugs.openjdk.java.net/browse/JDK-8171181

HETEROGENEOUS JAVA HEAP

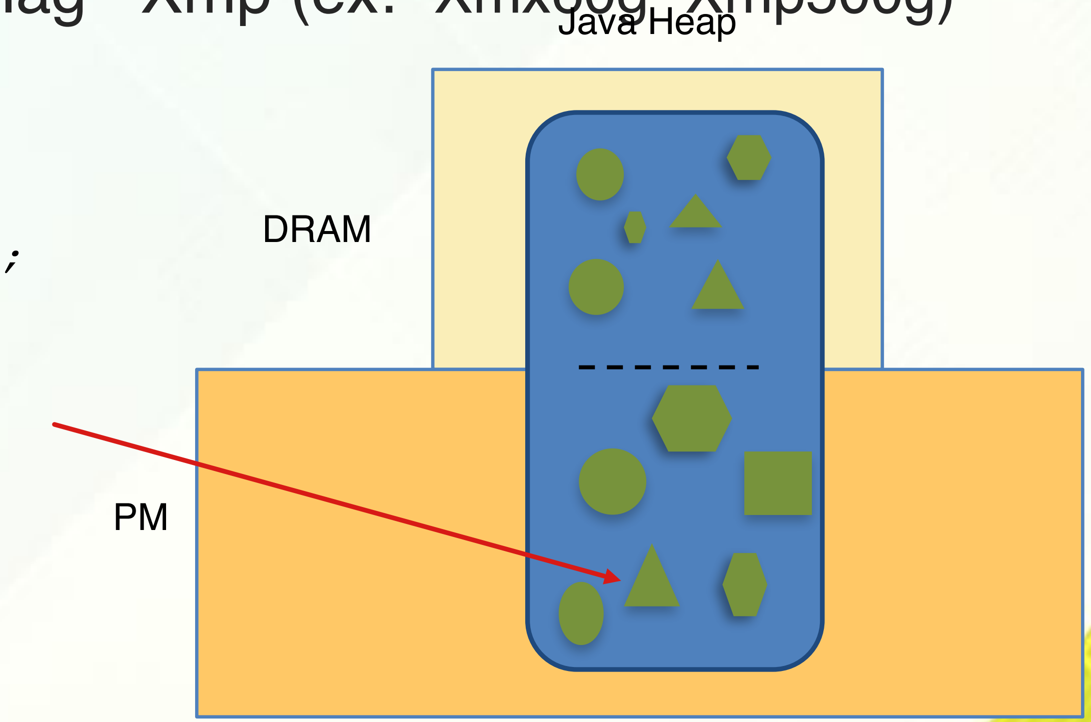
- User directed allocation
- Frequently accessed objects reside in DRAM (default)
- Garbage collection (G1 GC) collects both areas



HETEROGENEOUS JAVA HEAP INTERFACE

- Extended HeapArea size specified through flag –Xmp (ex: -Xmx60g -Xmp500g)
- APIs for setting allocation context

```
// Set the allocation target to PM
Heap.setAllocationTarget(AllocationTarget.PM);
HashMap user_records = new HashMap(1000000);
// reset allocation target to DRAM.
Heap.resetAllocationTarget();
```



Where is data persistence?

PERSISTENT COLLECTIONS FOR JAVA

- Library of persistent classes
- Custom persistent classes
- Low-level Memory Regions

https://github.com/pmem/pcj

LIBRARY OF PERSISTENT CLASSES

Primitive arrays (e.g. PersistentByteArray, mutable and immutable)

PersistentArray<E extends PersistentObject> (mutable and immutable)

PersistentTuple<T1 extends PersistentObject, …> (mutable and immutable)

PersistentArrayList<E extends PersistentObject>

PersistentHashMap<K extends PersistentObject, V extends PersistentObject>

PersistentLinkedList<E extends PersistentObject>

PersistentLinkedQueue<E extends PersistentObject>

PersistentSkipListMap<K extends PersistentObject, V extends PersistentObject>

PersistentFPTreeMap<K extends PersistentObject, V extends PersistentObject>

PersistentSIHashMap<K extends PersistentObject, V extends PersistentObject>

ObjectDirectory - indefinitely reachable root map of <String, T extends PersistentObject>

Primitive types (as field and array element values, no separate clas

Boxed primitives (e.g. PersistentLong)

PersistentString

PersistentByteBuffer

PersistentAtomicReference<T extends PersistentObject>

Development in Progress

LIBRARY OF PERSISTENT CLASSES

- State stored on persistent heap
- Instances behave like regular Java objects, just longer-lived
- Reachability-based lifetime
- Easy-to-understand data consistency model (transactional)

USING PERSISTENT COLLECTIONS

```
PersistentIntArray data = new PersistentIntArray(1024);
ObjectDirectory.put(”MyApplicationData", data);
// no serialization, reference to array is written
data.set(0, 123);

// Restart JVM or system
PersistentIntArray data1 =
ObjectDirectory.get("MyApplicationData",PersistentIntArray.class);
assert(data.get(0) == 123);
```

SUPPORT FOR CUSTOM PERSISTENT CLASSES

- Extending built-in persistent class
- Creating a new persistent class

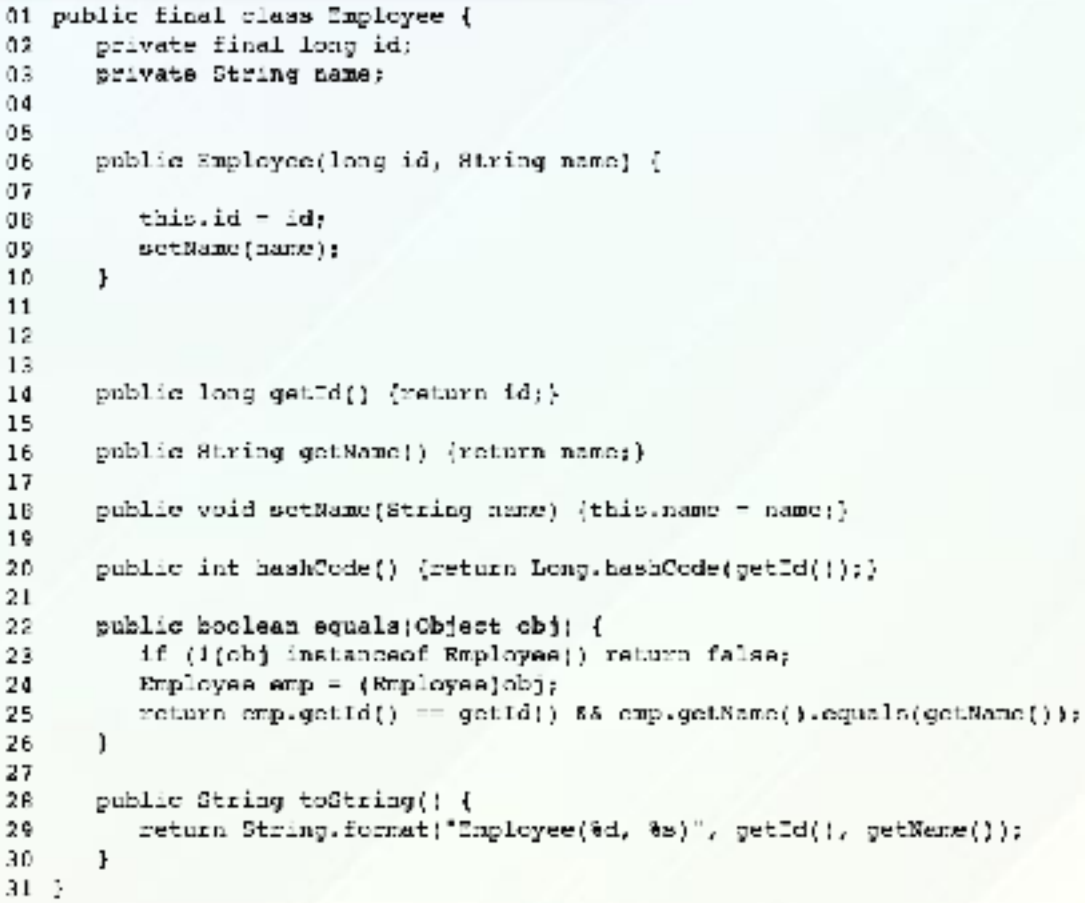
EXTENDING BUILT-IN PERSISTENT CLASS

```
public class Employee extends PersistentTuple2<PersistentLong, PersistentString> {
   public Employee(PersistentLong id, PersistentString name) {
      setId(id);
      setName(name);
   }
   public PersistentLong getId() {
      return _1();
   }
   public void setId(PersistentLong id) {
      _1(id);
   }
```

```
public PersistentString getName() {
      return _2();
   }
   public void setName(PersistentString name) {
      _2(name);
   }
   public String toString() {
      return String.format("Employee(%s, %s)", getId(), getName
   }
}
```

CREATING A PERSISTENT CLASS

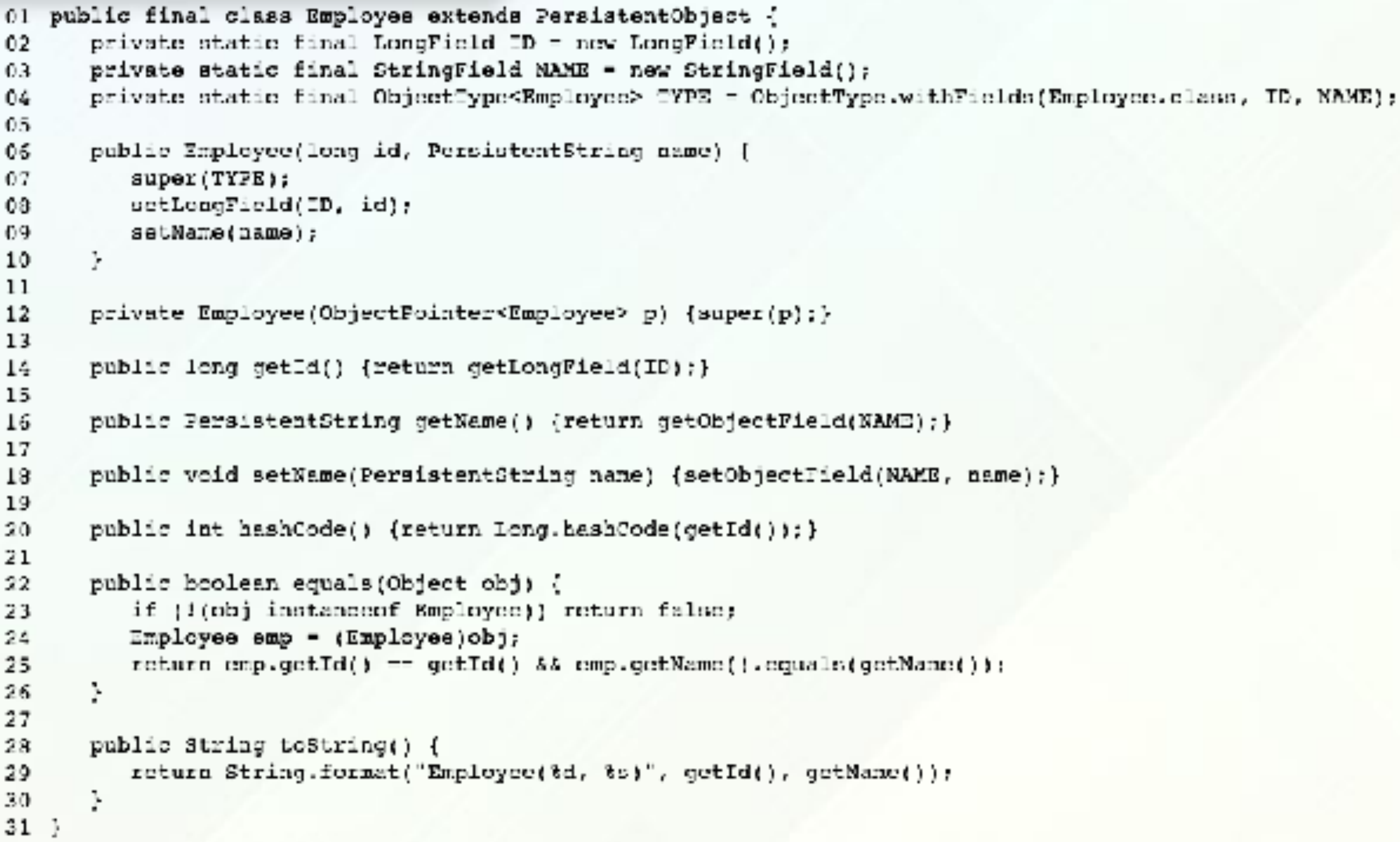
Non-persistent



```java
public final class Employee {
   private final long id;
   private String name;


   public Employee(long id, String name) {

      this.id = id;
      setName(name);
   }



   public long getId() {return id;}

   public String getName() {return name;}

   public void setName(String name) {this.name = name;}

   public int hashCode() {return Long.hashCode(getId());}

   public boolean equals(Object obj) {
      if (!(obj instanceof Employee)) return false;
      Employee emp = (Employee)obj;
      return emp.getId() == getId() && emp.getName().equals(getName());
   }

   public String toString() {
      return String.format("Employee(%d, %s)", getId(), getName());
   }
}
```

CREATING A PERSISTENT CLASS

Persistent



```java
public final class Employee extends PersistentObject {
    private static final LongField ID = new LongField();
    private static final StringField NAME = new StringField();
    private static final ObjectType<Employee> TYPE = ObjectType.withFields(Employee.class, ID, NAME);

    public Employee(long id, PersistentString name) {
        super(TYPE);
        setLongField(ID, id);
        setName(name);
    }

    private Employee(ObjectPointer<Employee> p) {super(p);}

    public long getId() {return getLongField(ID);}

    public PersistentString getName() {return getObjectField(NAME);}

    public void setName(PersistentString name) {setObjectField(NAME, name);}

    public int hashCode() {return Long.hashCode(getId());}

    public boolean equals(Object obj) {
        if (!(obj instanceof Employee)) return false;
        Employee emp = (Employee)obj;
        return emp.getId() == getId() && emp.getName().equals(getName());
    }

    public String toString() {
        return String.format("Employee(%d, %s)", getId(), getName());
    }
}
```

LOW-LEVEL MEMORY REGIONS

- Interface from OpenJDK Panama project
- Get and set for byte, short, int, long (on persistent memory)
- Heap API to allocate and free MemoryRegions
- Developers can
  - Retrofit existing code at low-level
  - Create their own abstractions
- Three versions
  - RawMemoryRegion -- useful for volatile use or when caller provides data consistency externally
  - FlushableMemoryRegion -- includes flush() method and fail-safe isFlushed() state
  - TransactionalMemoryRegion -- writes are transactional

APPLICATIONS OF PERSISTENT MEMORY

APPLICATIONS OF PERSISTENT MEMORY

All of In-Memory Computing Applications?

EVERYTHING SOUNDS SO EASY…

- Not so…
- Software innovation – new programing paradigm: "To persist or not to persist"
  - Think early days of the smart phone
  - Any write could be your last write – do you need the data when the application restarts?
  - More than just large memory
- Existing software needs to be re-architected – to unlock features and performance
  - Apache Cassandra, Apache Spark
- Traditional memory still in every system – applications need to be aware

CALL TO ACTION

- Innovate on persistence – discover usages!!
- Feedback on Java persistent programing model

https://software.intel.com/en-us/persistent-memory

JOIN THE DISCUSSION

- Learn about the Persistent Memory programming model - http://www.snia.org/forums/sssi/nvmp
- Join the pmem NVM Libraries Open Source project - http://pmem.io
- Read the documents and code supporting ACPI 6.1 and Linux NFIT drivers
  - http://www.uefi.org/sites/default/files/resources/ACPI_6.1.pdf
  - https://github.com/pmem/ndctl
  - http://pmem.io/documents/
  - https://github.com/01org/prd
- Intel Architecture Instruction Set Extensions Programming Reference
  - https://software.intel.com/en-us/intel-isa-extensions
- Intel 3D XPointTM Memory
  - https://software.intel.com/en-us/persistent-memory

Thank You