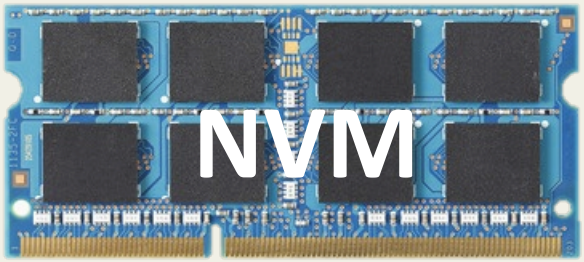
PERSISTENT MEMORY

FAST

BYTE ADDR

NONVOLATILE

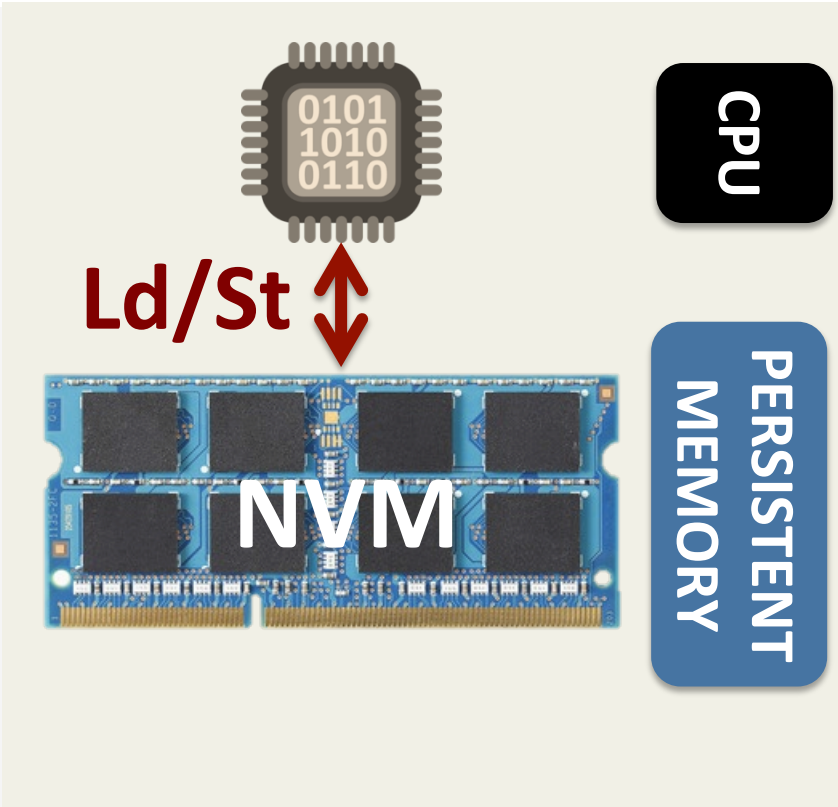


PERSISTENT MEMORY

FAST

BYTE ADDR

NONVOLATILE



Provides an opportunity to manipulate persistent data directly

WHAT HAPPENS WHEN WE RUN AN APPLICATION?

```
void hashtable_update (hashtable_t* ht,
                       void *key, void *data)
{
  list_t* chain = get_chain (ht, key);
  pair_t* pair;
  pair_t updatePair;
  updatePair.first = key;
  pair = (pair_t*) list_find (chain,
                              &updatePair);
  pair->second = data;
}
```

RUN THE CODE…

Persistent Memory System

```
void hashtable_update(hashtable_t* ht,
                      void *key, void *data){
  list_t* chain = get_chain(ht, key);
  pair_t* pair;
  pair_t updatePair;
  updatePair.first = key;
  pair = (pair_t*) list_find(chain,
                        &updatePair);
  pair->second = data;
}
```

System crash can result in permanent data corruption in NVM

CURRENT SOLUTION

```
void TMhashtable_update (TMARCGDECL
hashtable_t* ht, void *key, void*data)
{
  list_t* chain = get_chain (ht, key);
  pair_t* pair;
  pair_t updatePair;
  updatePair.first = key;
  pair = (pair_t*) TMLIST_FIND (chain,
                                &updatePair);
  pair->second = data;
}
```

CURRENT SOLUTION

Manual declaration of persistent components

```
void TMhashtable_update (TMARCGDECL
hashtable_t* ht, void *key, void *data)
{
  list_t* chain = get_chain (ht, key);
  pair_t* pair;
  pair_t updatePair;
  updatePair.first = key;
  pair = (pair_t*)TMLIST_FIND(chain,
                              &updatePair);
  pair->second = data;
}
```

Needs a new implementation

Prohibited Operation `=`

Third party code can be inconsistent

CURRENT SOLUTION

Manual declaration of persistent components

```
void TMhashtable_update (TMARCGDECL
hashtable_t* ht, void *key, void *data)
{
  list_t* chain = get_chain (ht, key);
  pair_t* pair;
  pair_t updatePair;
  updatePair.first = key;
  pair = (pair_t*)TMLIST_FIND(chain,
                              &updatePair);
  pair->second = data;
}
```

Needs a new implementation

Prohibited Operation `=`

Third party code can be inconsistent

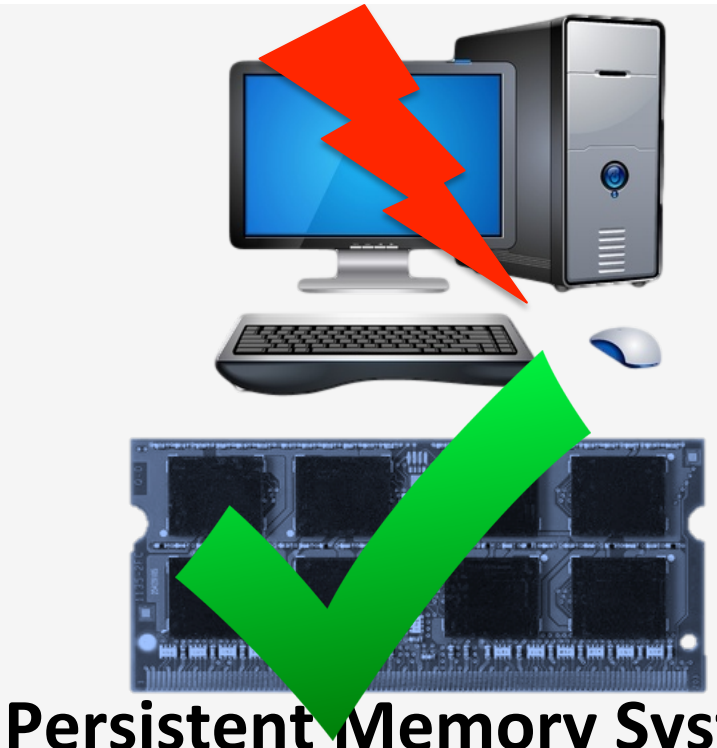
Burden on the programmers

Our Solution:
ThyNVM

NO MODIFICATION IN THE CODE

```
void hashtable_update (hashtable_t* ht,
                       void *key, void *data)
{
  list_t* chain = get_chain (ht, key);
  pair_t* pair;
  pair_t updatePair;
  updatePair.first = key;
  pair = (pair_t*) list_find (chain,
                              &updatePair);
  pair->second = data;
}
```

RUN THE EXACT SAME CODE…



Persistent Memory System

```
void hashtable_update(hashtable_t* ht,
                      void *key, void *data){
  list_t* chain = get_chain(ht, key);
  pair_t* pair;
  pair_t updatePair;
  updatePair.first = key;
  pair = (pair_t*) list_find(chain,
                        &updatePair);
  pair->second = data;
}
```

Software transparent memory crash consistency

ThyNVM

A new hardware-based checkpointing mechanism

- **Checkpoints** at *multiple granularities* to minimize both latency and metadata
- **Overlaps** *checkpointing* and *execution*
- **Adapts** to *DRAM and NVM* characteristics

Performs within 4.9% of an *idealized DRAM* with zero cost consistency

WEDNESDAY 10:20 am

ThyNVM

Enabling Software-Transparent Crash Consistency in Persistent Memory

Jinglei Ren, Jishen Zhao, **Samira Khan**,
Jongmoo Choi, Yongwei Wu, and Onur Mutlu