An NVM Carol

Margo Seltzer
UBC

Virendra Marathe, Steve Byan
Oracle Labs

UBC
Oracle Labs
NVM Technologies

PCM

STT-RAM

ReRAM

Carbon Nanotubes

https://en.wikipedia.org/wiki/Phase-change_memory
http://nantero.com/technology/
3D Xpoint

PCM

STT-RAM

ReRAM

Carbon Nanotubes

https://en.wikipedia.org/wiki/Phase-change_memory
http://nantero.com/technology/
NVM Characteristics
NVM Past

Bubbles and CCD memories—Solid state mass storage

by J. EGIL JULIUSSEN
Texas Instruments Incorporated
Dallas, Texas

INTRODUCTION
In the last year significant technical advances have taken
place in the development of magnetic bubble memories
(MBBM) and charge coupled devices (CCD). The first bubble
memory chip has been introduced with larger and better
very important for small systems. As chips they can also be
packaged directly with the CPU on PC boards.

The characteristics of today’s major CCD and bubble
memory components are shown in Table 1. The 93K bit
MBBM chip has an average access time of 4 milliseconds
and has a transfer rate of 56K bits per second. The dual-tube

Abstract
Making Smalltalk a Database System
George Copeland
Servio Logic Corporation
David Mager
Oregon Graduate Center
and
Servio Logic Corporation

Servio Logic Corporation in
making system for database

Dynamic Storage Allocation in the Atlas Computer,
Including an Automatic Use of a Backing Store
John Fatheringham
Perrottii Electric, Inc., Pleasenew, New York

1. Introduction
This paper is concerned with the method of address in-
terpretation in the Atlas computer. The Atlas has been
characters within a word for certain special functions, and
the leading digit of these there is also used for identifying
the half-word operand for 24-bit functions such as an index
register operation. The convention '1' bits address is used

1. Green, B. Moxon, J. The design and implementation of an In-
formation processing system. 2. Dijkstra, E. W. 1968
NVM Past: Bubble Memory
The Quote Game

1. **Bubble Memory** has the potential of replacing persistent storage.
2. The **Bubble Memory** market is finally ready to start to fulfill some of the expectations its enthusiasts have been predicting for a number of years.
3. There are several aspects of **NVM** that make existing database architectures inappropriate for them.
4. However, favorable factors are now emerging that will propel the emerging **NVM** business onto a rapid growth trajectory.
5. By exploiting this **Bubble Memory** for relational database management systems, we introduce efficient support for permutation, sorting and searching for data.
6. We present four alternative implementations to incorporate **NVM** into the processing stack of a query processor.
7. Some of the researchers in this field look forward to the ultimate replacement of persistent storage by **new Bubble Memory**.
8. Since the **Bubble Memory** is intrinsically similar to the data model, and adapted to the access requirements, we believe the overall system is simpler both in operation and in programming.
Bubble Memory: Use Cases
Bubble Memory: What Happened

![Graph showing price and density trends over years for dram and disk memory.]

- **Price** (Y-axis)
- **Density** (X-axis)
- **Year** (X-axis)
- **dram** (solid line)
- **disk** (dotted line)
Lessons

TULIP MANIA: THE FIRST ECONOMIC BUBBLE

NVM Past
Single-Level Store: Virtual Addressing
Single-Level Store: Persistence
Single-Level Store: Research

Fig. 4—Main core store control.

Fig. 5—Flow diagram of main core store control.
Persistent Objects: Why?

There was a long discussion on this topic. Some of the participants thought this whole area was misguided, while others thought it was a highly significant research area. In fact, virtually all participants held a strong opinion one way or the other on this research area. About half thought it held no promise of significant results while the other half thought it was an area where major results could be expected. This seeming contradiction can be explained by the fact that the proponents and detractors invariably discovered they were talking about different capabilities when they used the words "object-oriented data base" (OODB).
Persistent Objects: What
Persistent Foo

PS-Algol: “It should be possible to add persistence to an existing language with minimal change to the language.”

procedure open.database(
  string database.name,
  password,mode->pntr)

fp = os_fopen(Mmap_mapfile, "r");
mmapAddr = mmap(addr, len, proto,
  flags | MAP_SHARED_VALIDATE | MAP_SYNC | fd, offset);
mmapAddr is now a pointer referencing all persistent data.

Returns a pointer that is then used as the root to all persistent data stored in the database.

Language-Based Database
Persistent Foo and NVM

Then

Now

NVM Past
M. Herlihy, J. E. B. Moss, Transactional memory: architectural support for lock-free data structures (1993)
HTM Research

Hardware Transactional Memory

- HTM: Transactional memory can be implemented in hardware using the cache and cache coherency mechanism.
- Uncommitted transactional data is stored in the cache.
- Transactional in the cache has an additional bit.
- HTM has very but has size limitations.

Hardware Transactional Memory (HTM)
- Conflict manager
  - Decide if an instruction can be executed
- Cache protocol
  - Adapted with TM context

Microsoft Research

To Lock, Swap or Elide:
On the Interplay of Hardware
Transaction Indexing

Hardware Limitation:
Cache Capacity

- HTM uses the cache to hold all transactional data.
- Therefore, HTM aborts transactions larger than the cache.
- Restricting transaction size is awkward and not modular.
  - Size will depend on associativity, block size, etc.
  - In addition to cache size.
  - Cache configuration change from processor to processor.
STM Performance Issues

- Memory management overhead
  - Garbage collection, object cloning/buffering of writes
  - Multiple pointer chasing required to access object data

- Validation overhead
  - Visible readers require 2N CASs to read N objects
  - Invisible readers need to perform bookkeeping and validation – O(N^2) operation with N objects

Pastramy
Persistent and highly available Software Transactional Memory
Paulo Romano, Nuno Canasoro, Josie Cachopo, Luis Rodrigues

Hybrid NORec - Opacity

1. Read X
2. Read Y
3. Func(X, Y)

Verify clock
(puts in HTM)

Lock clock

ABORT

1. X = 4

2. Y = 8

Hybrid NORec - Sandboxing

1. Read X
2. Read Y
3. Func(X, Y)

Verify clock

Lock clock

Func uses:
1. new X
2. old Y

Unsafe (hopes HTM aborts)
NVM Today

Storage Allocators
Data Structures
Logging
File Systems
Database Architectures
Programming Model
ISA and System Changes

NVM Present
NVM Today: Cross Cutting Themes

- Synchronization, Consistency, and Programming Models
- Persistent Storage Components
- Warm Restart

Storage Allocators
Data Structures
Logging
File Systems
Database Architectures
Programming Model
Storage
Synchronization, Consistency, and Programming Models
Persistent Storage Components
Warm Restart
NVM Present
NVM Today
Synchronization, Consistency, and Programming Models
The Consistency Problem

Free Space

Free Pointer
User memory
NVM Today
Synchronization, Consistency, and Programming Models
The Consistency Problem

Rules for writing NVM:
1. Data is persistent only after it leaves the cache
2. Data can leave the cache in any order it wants
3. Cache flush operations force data out of the cache
4. Barrier instructions (e.g., sfence) enforce ordering between stores
NVM Today
Synchronization, Consistency, and Programming Models
The Consistency Problem

Free Space

Free Pointer

User memory

Alloc
NVM Today
Synchronization, Consistency, and Programming Models
The Consistency Problem

Free Space

Free Pointer

User memory

Alloc

NVM Present
NVM Today
Synchronization, Consistency, and Programming Models
The Consistency Problem

Free Space

Free Pointer

User memory

NVM Present
Rules for writing NVM:
1. Data is persistent only after it leaves the cache
2. Data can leave the cache in any order it wants
3. Cache flush operations force data
4. Barrier instructions (sfence) enforce ordering between stores
5. Sfence is expensive!
The Barrier Game
The Barrier Game

Begin
Write A
Write B
Write C
Commit

Atomic
Commit

DRAM
NVM Today
Synchronization, Consistency, and Programming Models
Programming Models

NVM Present
typedef persistent struct mystruct mystruct;
persistent struct mystruct {
    nvm_mutex mutex;
    int count;
};
nvm_desc desc; // region descriptor
int increment(mystruct ^my){
    int ret;
    @ desc {
        nvm_xlock(%my=>mutex);
        my=>count@++;
        ret = my=>count;
    }
    return ret;
}
NVM Today
Synchronization, Consistency, and Programming Models
Programming Models

NVM Direct (Bill Bridge)

typedef persistent struct mystruct mystruct;
persistent struct mystruct {
    nvm_mutex mutex;
    int count;
};
nvm_desc desc; // region descriptor
int increment(mystruct ^my){
    int ret;
    @ desc {
        nvm_xlock(%my->mutex);
        my->count@++;
        ret = my->count;
    }
    return ret;
}

/* Determine if address is in pmem. */
is_pmem = pmem_is_pmem(pmemAddr, PMEM_LEN);

/* Store string to persistent memory. */
strcpy(pmemAddr, "hello, persistent memory");

/* Flush string to persistent memory. */
if (is_pmem)
    pmemPersist(pmemAddr, PMEM_LEN);
else
    pmem_sync(pmemAddr, PMEM_LEN);
NVM Today
Synchronization, Consistency, and Programming Models
Programming Models: Then and Now

Then
• Objects
• Databases
• High level interfaces

Now
• Allocators
• Data structures
• Low level interfaces
NVM Today
Persistent Storage Components

Memory Bus
NVM Today
Persistent Storage Components
What do people do?
NVM Today
Persistent Storage Components
NVM Today
Warm Restart
NVM Today
Performance Reality

Bandwidth for 2048 byte Random Reads

1024-byte 2:1 Random R:W

Mason (UBC) & Gavrilovska (Georgia Tech)
NVM Today
Performance Reality

Polybench Tests for DRAM versus PMEM

Normalized CPU time

Mason (UBC) & Gavrilovska (Georgia Tech)
NVM Future
NVM Future: Another Tier
NVM Future: Availability
NVM Future: Killer App

"Looks like another killer app."
Thank You!

ORACLE®