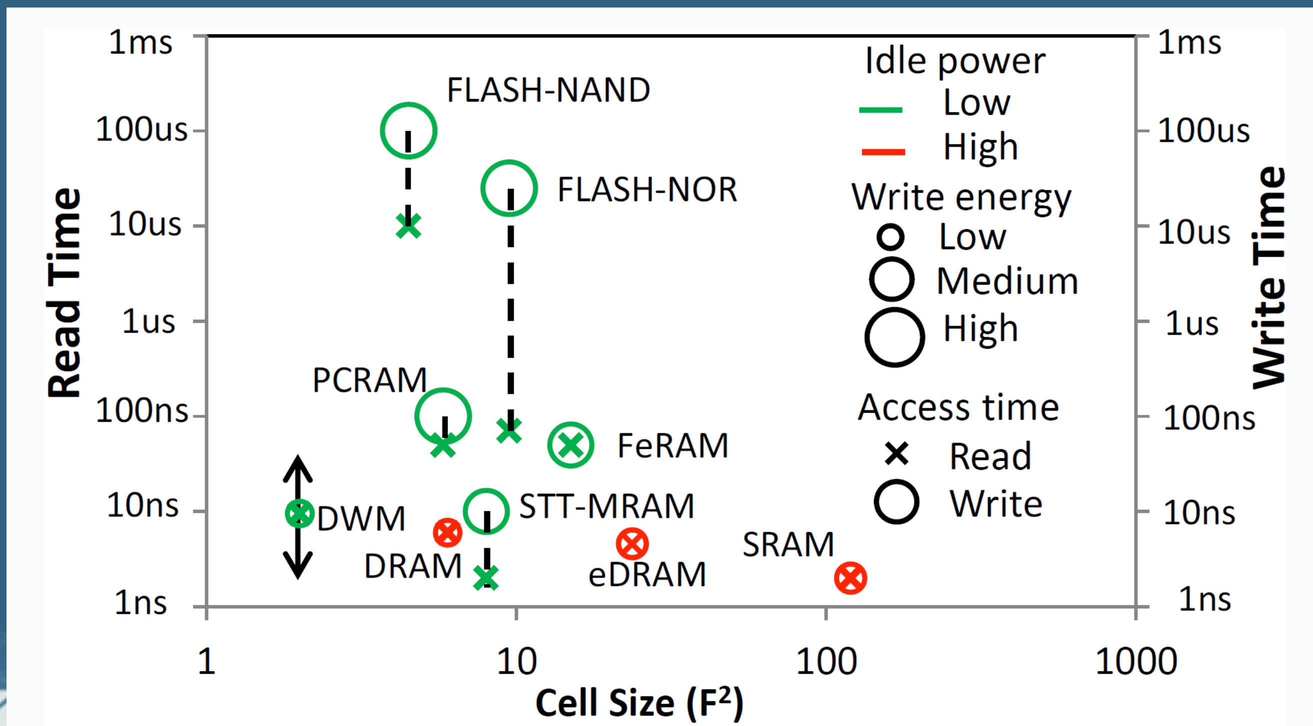
AitF: EXPL: Data Management in Domain Wall Memory-based Scratchpad for High Performance Mobile Devices

PI: Kirk Pruhs
Co-PI: Youtao Zhang
Students:
Max Bender (algorithms),
Lei Zhou (systems)

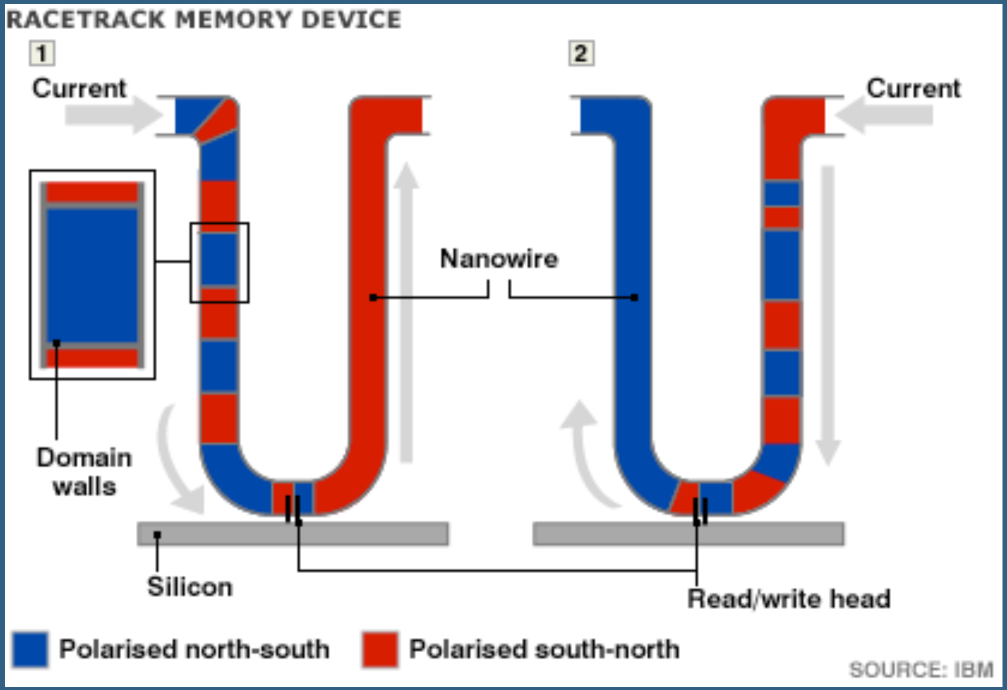
University of Pittsburgh

Motivation: Memory Technologies



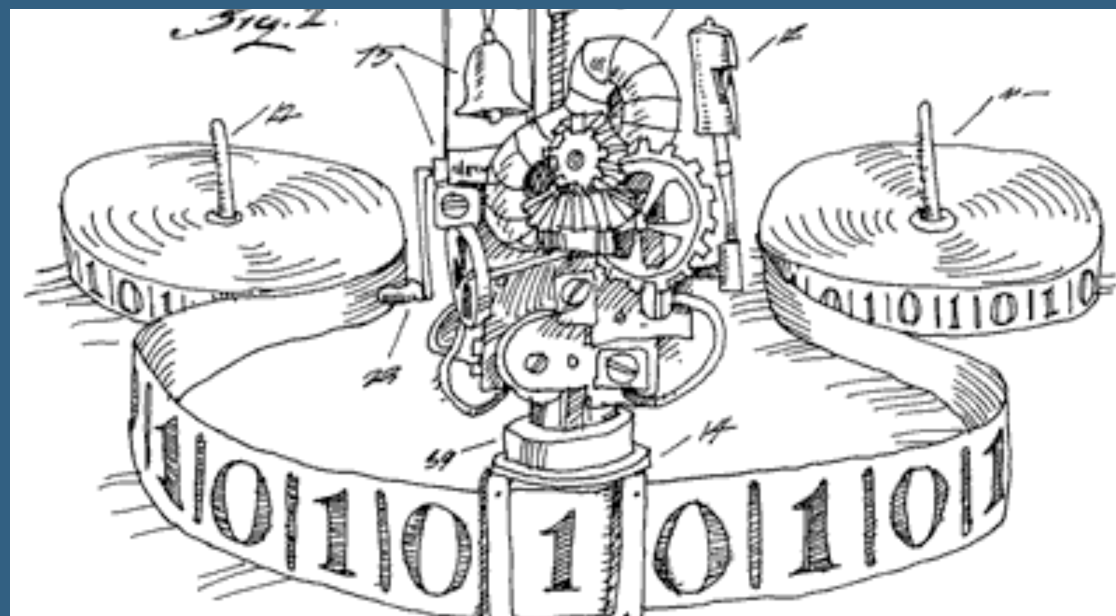
Science 2008

DWM = Domain Wall Memory = Racetrack Memory



- Nickel-iron alloy wires 1-10 microns (millionth of a metre) in length
- Data held in domain walls between regions of different polarisation
- 10 microns length could hold 100 domain walls
- Data is written or read by read/write head on silicon base
- Relevant domain wall shunted to read/write head by applying charge
- Reversing charge moves domain walls back (2)

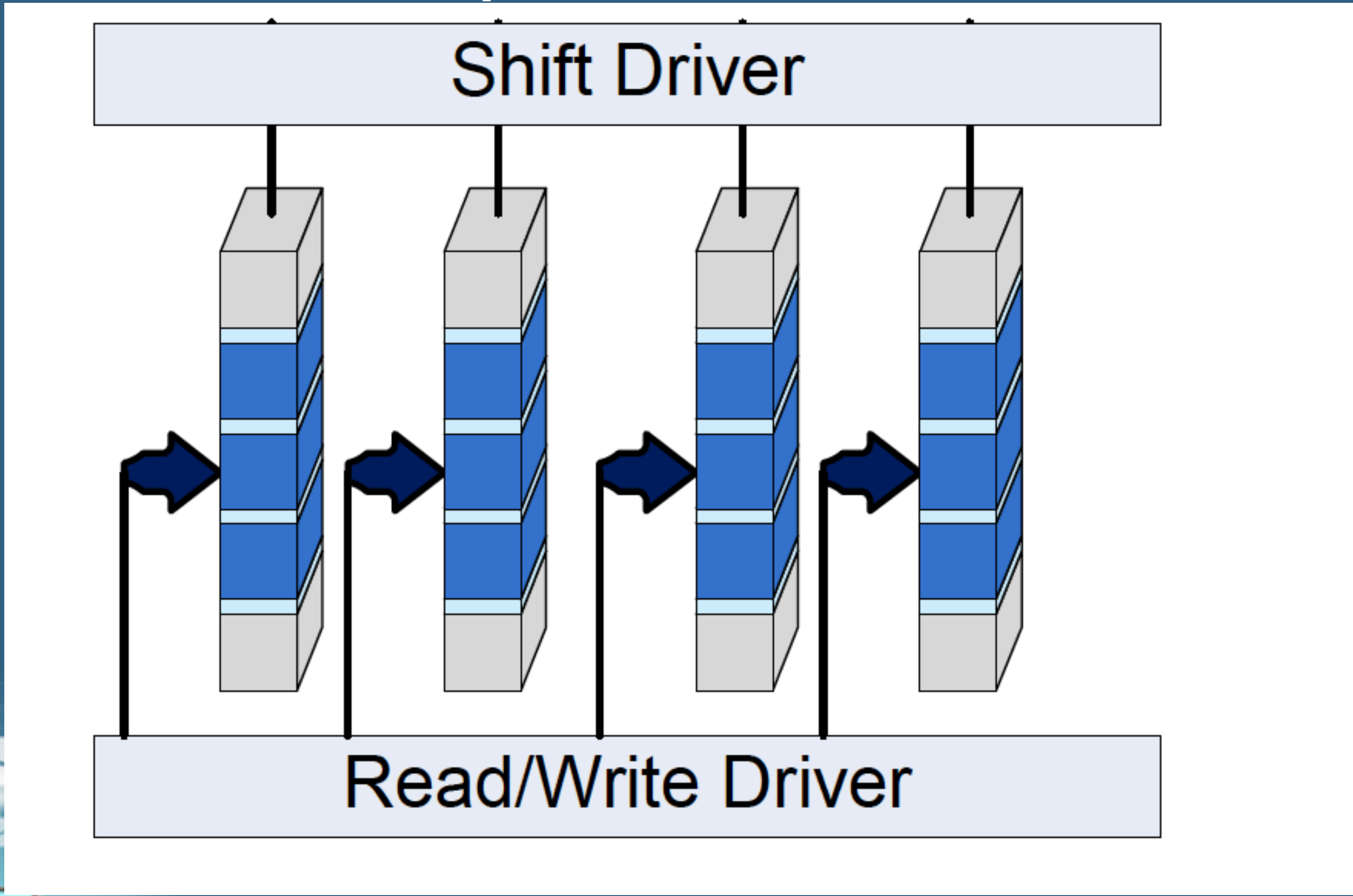
Theoretical View:
DWM = tape with read/write head(s)



Many Modeling Issues (Our Default)

- Number of read/write heads per tape
  - 1
- How words are laid out in memory
  - Supertracks
- Tracks heads have home position or are lazy
  - Lazy
- Used as cache or scratchpad
- Does the compiler instantiate virtual or physical addresses in the program
- Etc.

One Natural Memory Organization: Supertracks



Supertrack of 3 words, each having 4 bits

AitF Proposal Components

1. Algorithms for managing data placement on a single (super) track

2. Algorithms for managing data placement of words on multiple super tracks

3. Experimentation/simulation

Resulting Papers

- Neil Olver, Kirk Pruhs, Kevin Schewior, Rene Sitters, and Leen Stougie: The Itinerant List Update Problem. Under submission.

- XianWei Zhang, Lei Zhao, Youtao Zhang, Jun Yang: Exploit common source-line to construct energy efficient domain wall memory based caches. ICCD 2015: 157-163
- Lei Zhao, Youtao Zhang, and Jun Yang: Mitigating Shift-Based Covert-Channel Attacks in Racetrack Last Level Caches Under submission.

Offline Static Track Management Problem

- Input:
  - sequence of items (memory addresses)
    - e.g. A, B, A, C, A, B, D, A
  - n = number of locations on the track
- Output:
  - Feasible solution = assignment of items to track locations
    - e.g. B is in location 1, C is in location 2, … A is in location n
    - Objective: Minimize the total distance the track has to move to access these items in this order

Example:

- Input: A, B, C, A, B, D
- Feasible solution:

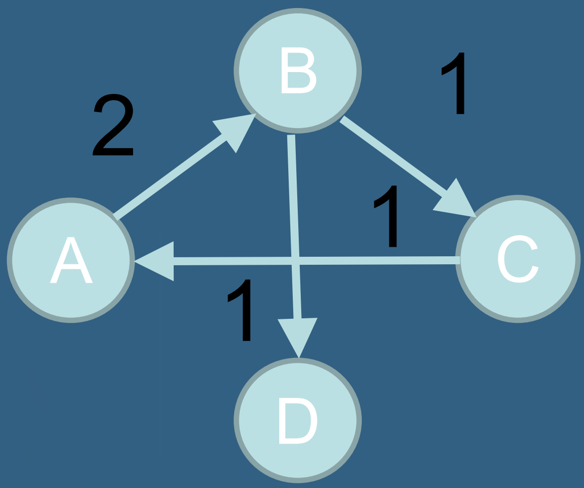
B	A	D	C

- A ⇒ B cost 1
- B ⇒ C cost 3
- C ⇒ A cost 2
- A ⇒ B cost 1
- B ⇒ D cost 2
- Total cost of this layout = 1 + 3 + 2 + 1 = 7

Static Track Management aka Minimum Linear Arrangement Problem

- Track management input: A, B, C, A, B, D

- Minimum linear arrangement input = access graph



Results for Minimum Linear Arrangement

- NP-hard
- Poly-time $\log^2 n$ approximation [Hansen 1989]
- Poly-time $(\log n) \log \log n$ approximation [CHKR 2006, FL 2007]
- No PTAS under complexity theoretic assumptions [AMS2011]

Dynamic Track Management

- Everything the same as static track management except that the possible operations are:
  - Move head one position left or right
  - Swap current items with the item to the left or the right

Classic List Update Result

- If the track head has a home position, then moving the last accessed item to the home position is O(1)-approximate with respect to number of operations

Research Contributions

(After an access or insertion of the ith item there are at most $i - 1$ free exchanges.)

THEOREM 1.
For any Algorithm A and any sequence of operations s starting with the empty set,

$$C_{MF}(s) \le 2C_A(s) + X_A(s) - F_A(s) - m.$$

PROOF.
In this proof (and in the proof of Theorem 3 in the next section) we shall use the concept of a *potential function.* Consider running Algorithms A and MF in parallel on s. A potential function maps a configuration of A's and MF's lists onto a real number Φ. If we do an operation

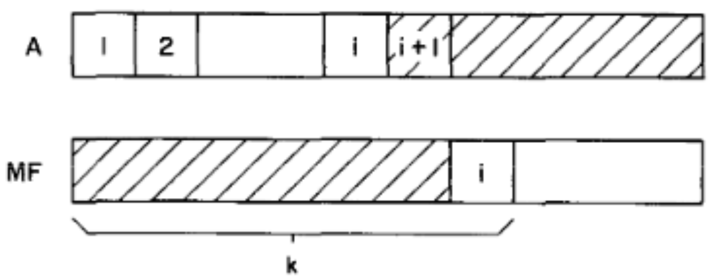


FIGURE 1. Arrangement of *A*'s and MF's lists in the proofs of Theorems 1 and 4. The number of items common to both shaded regions is x_i.

list. (See Figure 1.) Then the number of items preceding

Sleator, Tarjan 1985

Analogous Algorithms For Track Management

1. Move last accessed item to next to last accessed item
2. Move next to last accessed item to last accessed item
3. Move both next to last accessed item and last accessed item towards each other

Intuition question: Which of these algorithms are O(1)-approximate?

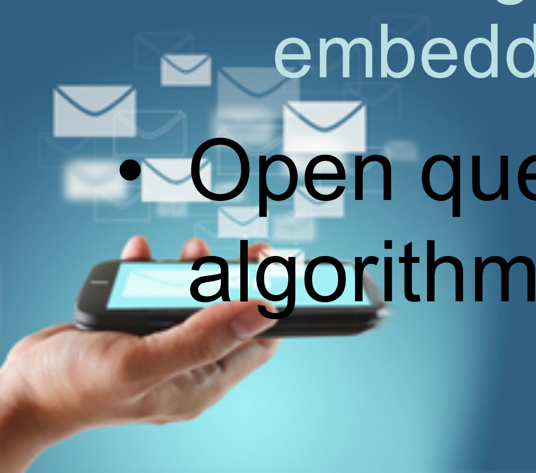
Surprise to Me

- Theorem: Moving the last accessed item to the next to last accessed item is $\Omega(n)$ approximate
  - Access Sequence: 1, n, n-1, n-2, … 2
  - Algorithms' cost $\approx n^2$
  - Optimal cost $\approx n$

- Similar examples showing Ω(n) for other natural algorithms
- Intuition: Dynamic list/track management without a home position is harder because its not clear where in the list/track to aggregate the recently accessed items

Algorithmic Results for Dynamic Track Management

- A log n online lower bound on approximation for online algorithms
- A poly-time $\log^2 n$ offline approximation algorithm
  - Offline is a reasonable assumption if memory is being used as scratchpad memory in embedded system
- Open question: poly-log competitive online algorithm ?



Going Forward

- Track management:
  - find a poly-log approximate online algorithm
  - Circumvent need to use balanced cut as a big hammer
- Multiple track management:
  - Figure out what the “right” problems are
  - Give good algorithms for these problems
  - Experimental simulation studies of these algorithms