FSort: External Sorting on Flash-based Sensor Devices

Panayiotis Andreou, Orestis Spanos,
Demetrios Zeinalipour-Yazti, George Samaras
University of Cyprus, Cyprus

Panos K. Chrysanthis
University of Pittsburgh, USA

6th Intl. Workshop on Data Management for Sensor Networks (DMSN'09), in conjunction with VLDB'09, Lyon, France, 2009

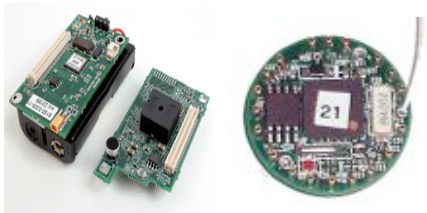
http://www.cs.ucy.ac.cy/~dzeina/

Motivation

- Wireless Sensor Devices have evolved from **primitive devices** into **powerful computing devices**.
- More **Computation Power** and More **Storage!**

2002

Applications: Remote Logging / Environmental Monitoring



Mica2 / Mica2Dot:

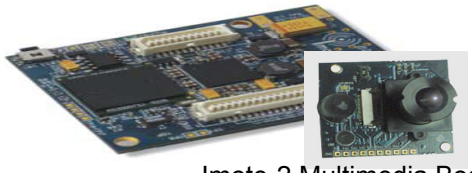
128KB Flash

2005

Applications: Digital Image Processing, etc.

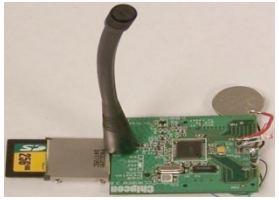
SunSpot:

4MB Flash



Imote-2 Multimedia Board

Imote-2: 32MB Flash



RISE: 1GB Flash

2009

Applications: People-Centric Sensing (sensing data by people for people)

IPhone:

32GB Flash

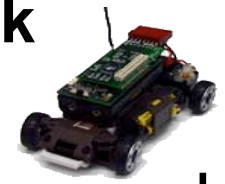
(GPS, camera, mic ambient light, 3-axis accelerometer, digital compass, …)

HTC Magic

512MB Flash

Motivation

- As **data storage mediums** continue to grow, **Sensor Device Developers** tend to **store more data in-situ**, e.g.,
  - To increase the **Sensing Fidelity**
    - Capture data locally, preprocess it, then transmit the useful information over to the sink, e.g., sound detection application
  - To overcome the problem of a **Distant Sink**
    - Applicable in Mobile Sensor Networks

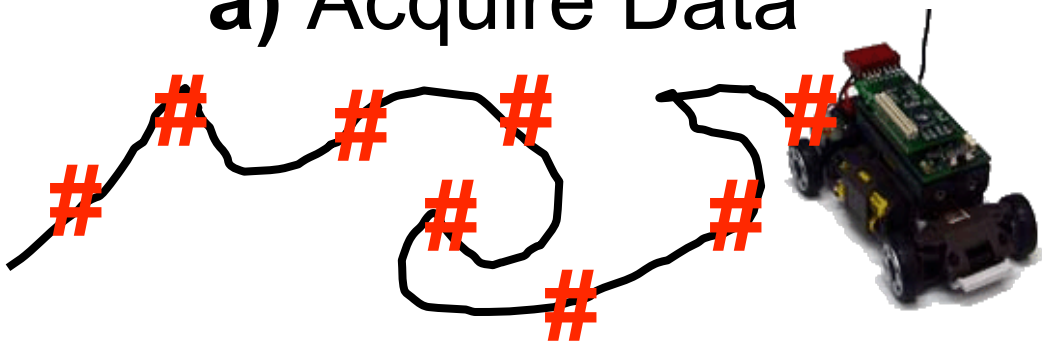


MilliBots (CMU)

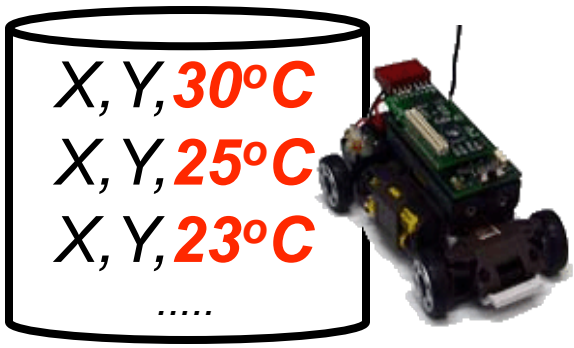
- Storing large quantities of data locally on each sensor calls for **database techniques …**
  - Indexing (e.g., “MicroHash”, In USENIX FAST’05)
  - Architectures (e.g., FlashDB, In IPSN’07)
  - External Sorting (e.g., FSort, In DMSN’09)
  - ….

Sorting on Sensors: Why is it Necessary?

a) Acquire Data



b) Store it Locally



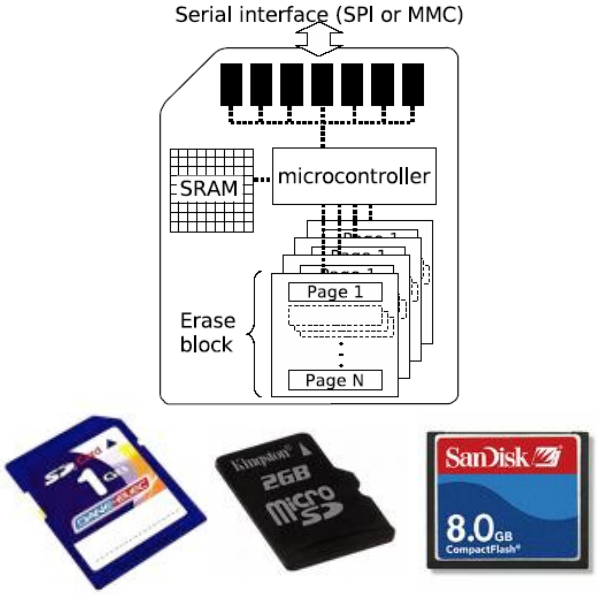
C) *"Find GPS locations where the temperature is in the range A to B"*

- ***Problem:*** *Temperature Data needs be* ***sorted on the given attribute*** *or some* ***index structure*** *might be necessary (e.g., B+tree)*
- **Other Examples: JOIN** the readings of two sensors, **Sort** local Images, **Remove** duplicates, .

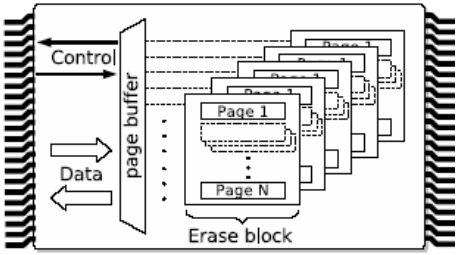
Background: Flash Memory

- The most prevalent storage medium used for Sensor Devices is **Flash Memory** (NAND Flash)
- Fastest Growing Memory Market (‘05 $8.7B, ’06 $11B)

Removable Flash

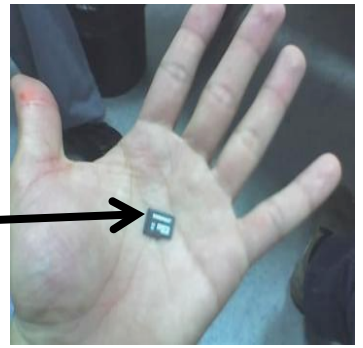


Surface Mount Flash



Background: Flash Memory

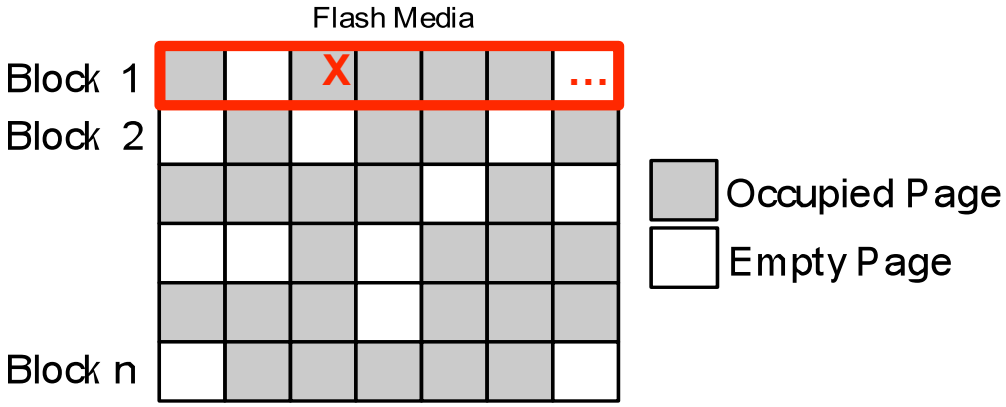
- **Simple Cell Architecture**
  - ➔ High capacity in a **small surface**
  - ➔ Economical Reproduction



- **Fast Random Access** (50-80 µs) compared to 10-20ms in Magnetic Disks

- **Shock Resistant & Power Efficient**
  - ➔ Appropriate for Sensor and Mobile Devices

Background: Characteristics of Flash



1. Delete-Constrain: **Erasure of a page can only be performed at a block granularity (i.e. 8KB~64KB)**
2. Write-Constrain: Data is written to flash on a **page granularity** (256B~512B) to an **empty page** (delete page if necessary).
3. Wear-Constrain: **Each page can only be written a limited number of times (typically 10,000-100,000)**
4. Asymmetric Read/Write: **Reading Faster (Cheaper) than Writing (23mJ vs. 763mJ on the RISE sensor)**

Presentation Outline

1. Motivation and Background

2. **Overview of Sorting on Sensor Devices**

3. The FSort External Sorting Algorithm

4. Experimental Evaluation

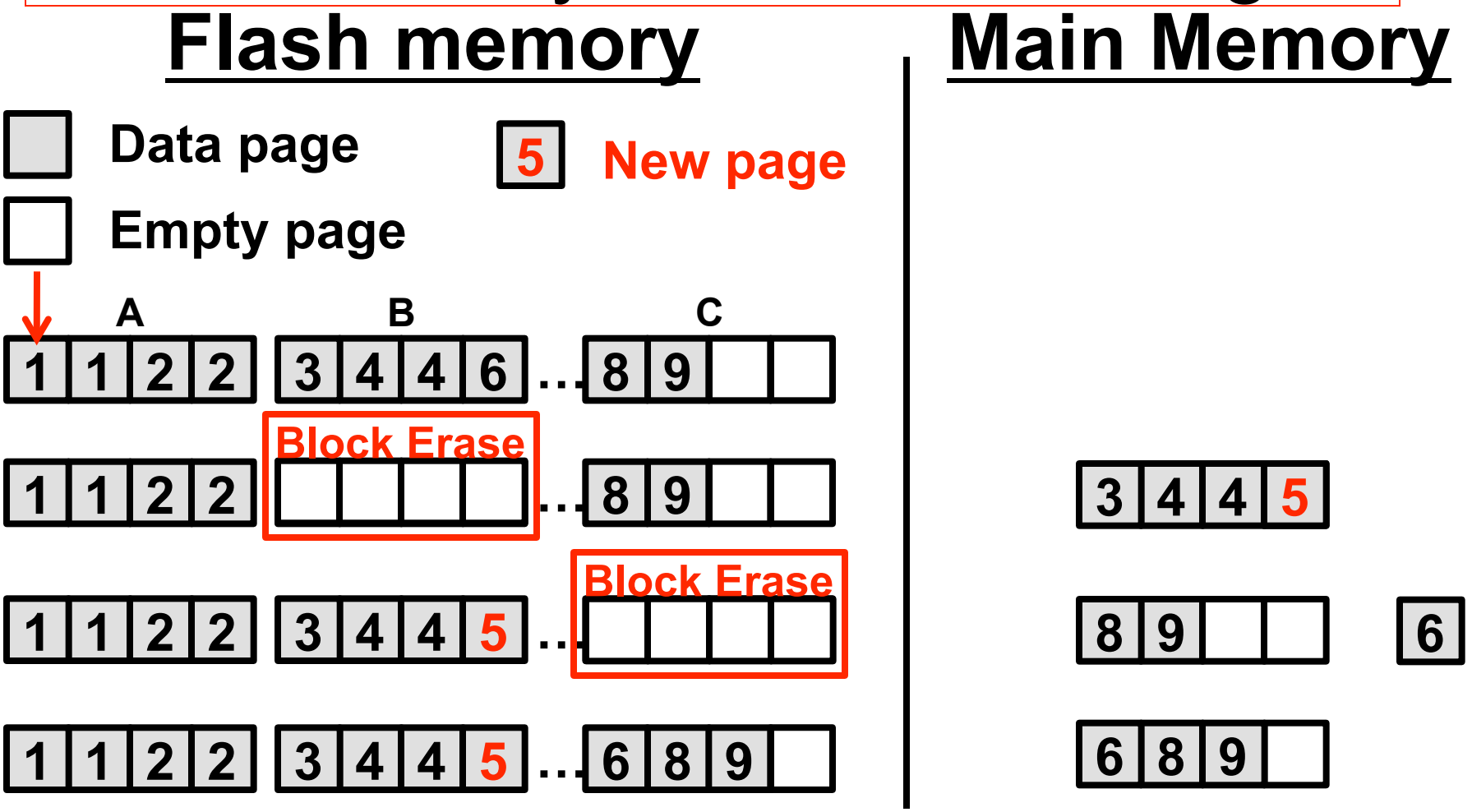
5. Conclusions and Future Work

Sorting on Sensor Devices

Sorting can be performed in two modes:

- **Online Sorting**
  - Keep a target sequence **sorted at all times**.
  - i.e., upon reception of a **new data record (e.g., after data acquisition)**, sort sequence using the **InsertionSort** idea.
- **Offline Sorting**
  - Log data records in an **Unsorted File** until **Sorting function** is requested.
  - **Problem:** Unsorted File might be too large to fit in main memory, thus In-Memory Sorting (using quicksort, mergesort, etc) is not feasible
  - **Solution:** Use an External Sorting Algorithm

Inefficiency of Online Sorting



Online Sorting is Inefficient! For 1 write it took us:

6 Reads + 7 Writes + 2 Block Erases

Decreased Performance + Decreased Lifetime

Presentation Outline

1. Motivation and Background
2. Overview of Sorting on Sensor Devices
3. **FSort External Sorting Algorithm**
4. Experimental Evaluation
5. Conclusions and Future Work

FSort Algorithm

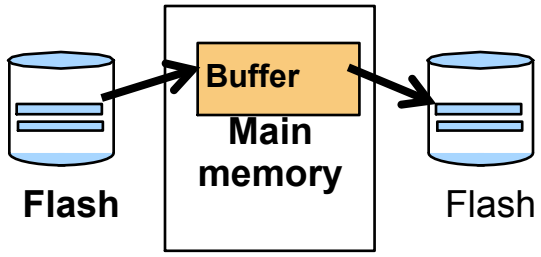
- **FSort** is an **external sorting** algorithm appropriate for flash-media on sensor devices

Characteristics

- **Offline External Sorting Algorithm**
  - It sorts on-demand, **arbitrary large sequences** with a **fixed-sized (**main) memory.
- **Appropriate for Sensor Devices**
  - Its performance is boosted by the **temporal coherence** in sensor readings (i.e., values that don't change quickly over time).
- **Designed for Flash**
  - It deletes blocks sequentially
  - It minimizes read/write/block erase operations.

FSort: Outline of Operation

FSort consists of two phases:



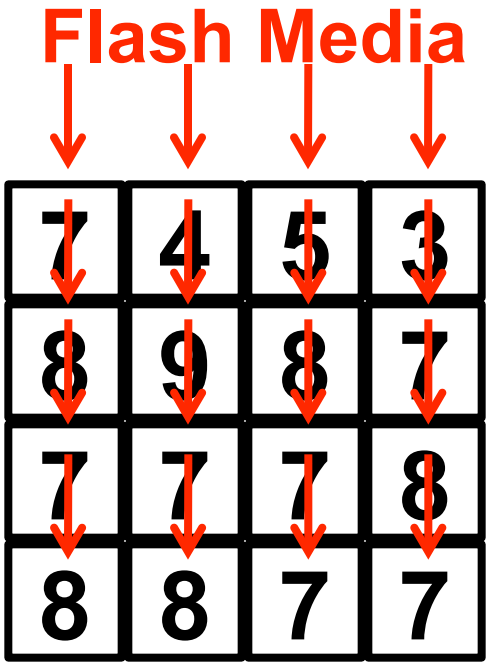
- **Sorting phase:**
  - Perform a Scan of the Flash media using **M block iterators.**
  - Use the **Replacement-Selection** concept to produce **Sorted Runs (Sequences)**
  - **Append** the Sorted Runs to the **end of the Flash Media**
- **Merging phase**
  - **Merge the sorted runs** recursively until a **single sorted output is produced.**

FSort: Sorting Phase

a) Initialize **B-1 Block Iterators.**

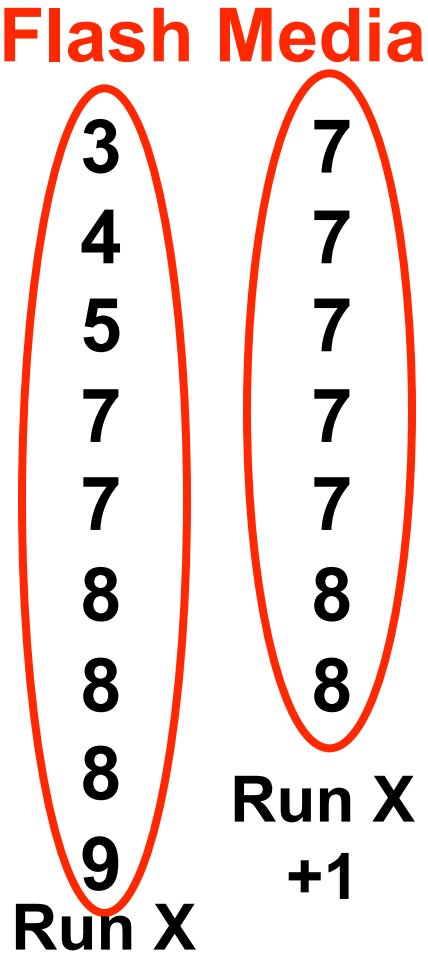
b) Write **next smallest elem. >=** current elem. to run **X**

- If above not possible, output next elem. to run **X+1**



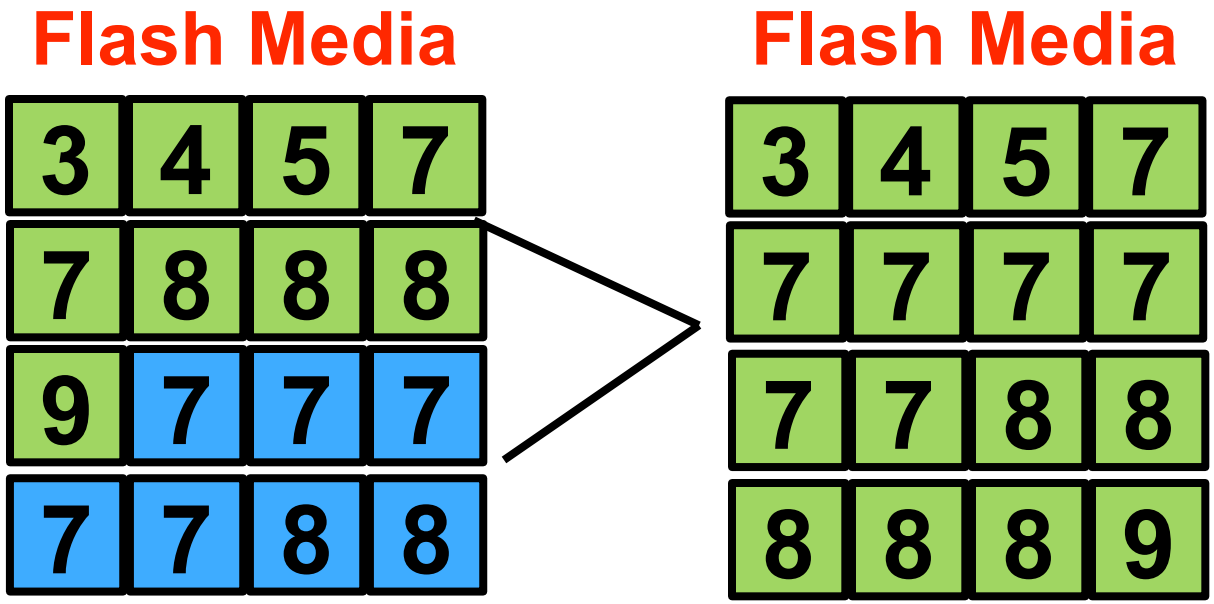
Main Memory

7	4	5	3



FSort: Merging Phase

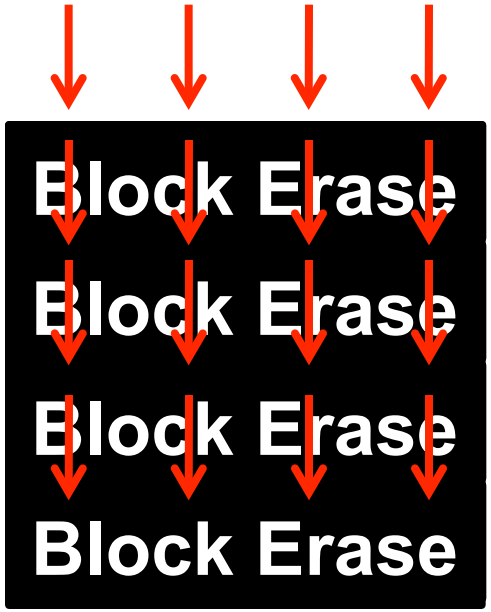
Example: B Buffer Pages (Input: B-1, Output: **1**)



Number of Passes over Data? 2 times

FSort and the Delete Constrain

- A block is **deleted** only whenever **all pages** within the given **block are empty**
  - This satisfies the Flash Delete Constrain



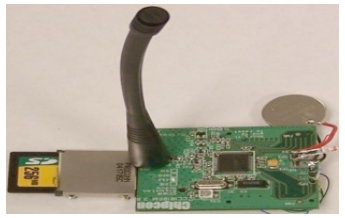
Additionally, due to **temporal coherence** (i.e., values within a block are similar) **blocks are usually freed up quickly.**

Presentation Outline

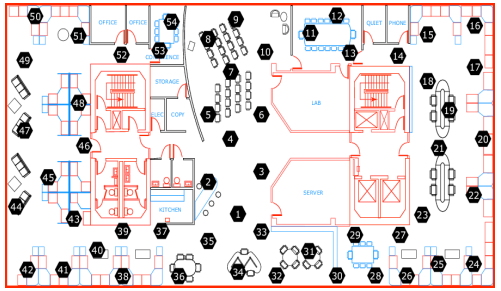
1. Motivation and Background
2. Overview of Sorting on Sensor Devices
3. FSort External Sorting Algorithm
4. **Experimental Evaluation**
5. Conclusions and Future Work

Experimental Evaluation

- **Testbed: Trace-driven Simulator**
  - Written in C++ (idea also prototyped in LiteOS/C on IRIS Sensor)
  - Emulates Flash (Page=512B, Block=16KB)
  - Measures:
    - # of read (1.17mA), write (37mA), erases (18mA)
    - Energy Consumption Energy = Volts x Amperes x Seconds



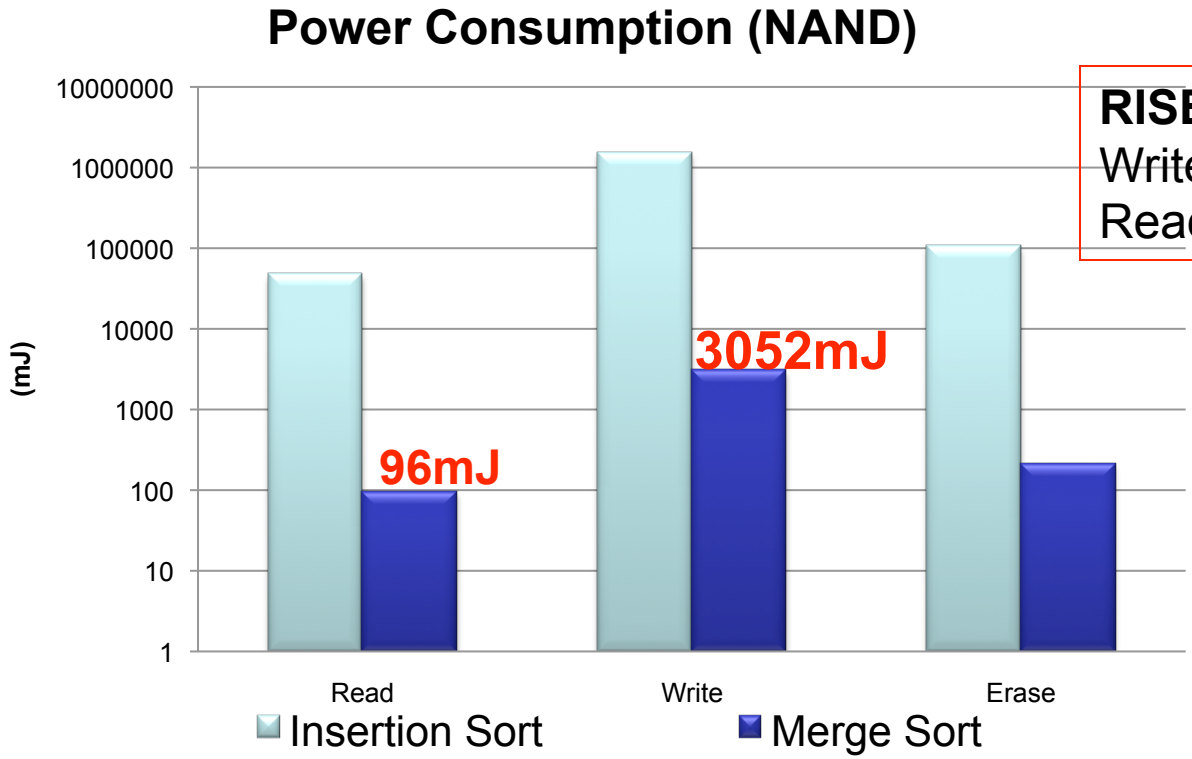
- **Dataset:** Intel Research Berkeley
  - Temperature, Humidity, Light, Vol, etc.
  - 2.3 million readings



- **Compared Algorithms**
  - **Online vs. Offline Sorting**
    - InsertionSort vs. External Merge Sort
  - **Comparison of Offline Sorting**
    - External Merge Sort vs. FSort

Online vs. Offline Sorting

Sorting 1000 records (1 record / page)



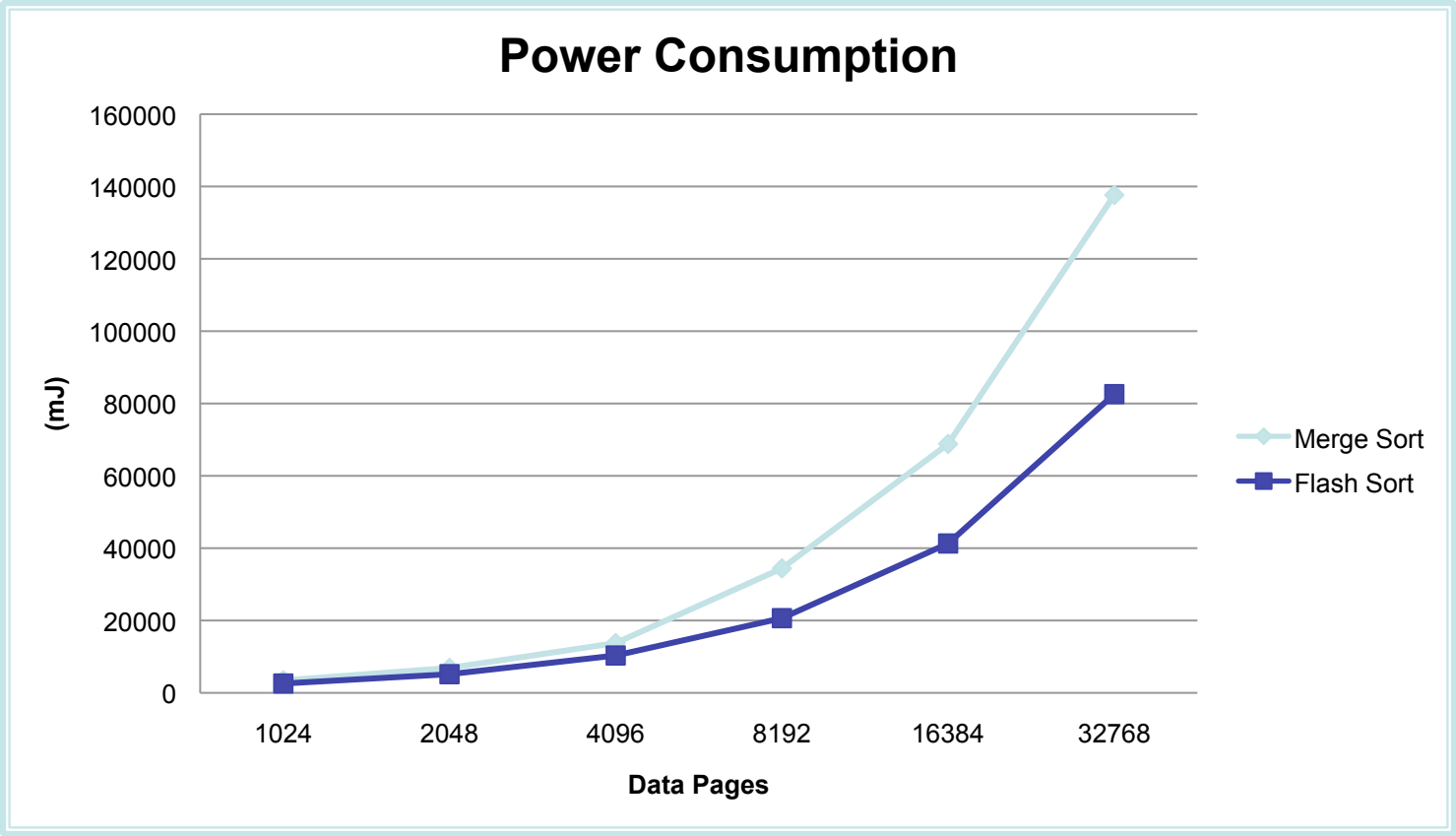
RISE Energy Model
Write: 37mA
Read: 1.17mA

a) **Online Sorting** is 2 Orders **slower** (less efficient) than **Offline Sorting**

b) **Same** number of **Read/Write** operations**, yet** writing is overall **more Expensive**

External MergeSort vs. FSort

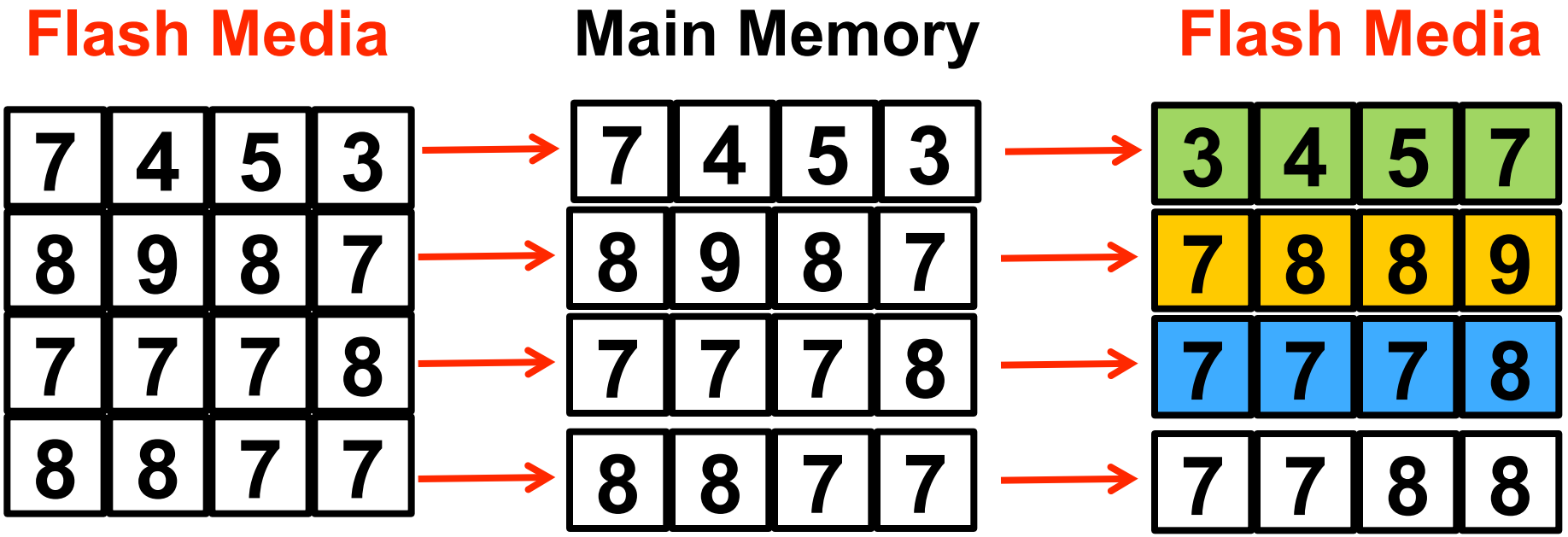
Task: Sort records (on temperature) after adding 1000 measurement to local flash media.



FSort is 25-40% more efficient than External MergeSort.

External Merge Sort

Example: B Buffer Pages (Input: B-1, Output: **1**)



Drawback of External Merge Sort: Small Runs

➔ Here **4** rather than **~8 (Fsort)**,

➔ In our experiments we found that they are **2-3 times smaller** than FSort

Presentation Outline

1. Motivation and Background
2. Overview of Sorting on Sensor Devices
3. FSort External Sorting Algorithm
4. FSort Experimental Evaluation
5. **Conclusions and Future Work**

Conclusions and Future Work

- We have shown that FSort improves **response time** and **energy consumption** for sorting **by 25 - 40%** compared to EMS.
- **In the future we plan to:**
  - Formally **analyze** the **performance** of the FSort algorithm.
  - **Empirically assess its performance** on real **devices** and **storage** mediums (External Flash cards, Smart-phones, SSDs, …)
  - Study the utilization of **complementary structures** for sorting (e.g., indexes)

FSort: External Sorting on Flash-based Sensor Devices

Thank you!

Questions?

This presentation will be available at:
http://www.cs.ucy.ac.cy/~dzeina/