Mobile Computing
#MC02 Device Databases

CS60002: Distributed Systems
Winter 2006-2007

Administrative Stuff

- Student papers
  - Ask me after class if interested
  - Topics not limited to those covered in lectures
- Masters projects
  - Four potential projects
  - Ask me after class if interested

Parts mentioned in MC01

Today

- Device databases
  - Flash, OR/direct

Tomorrow

- Synchronization
  - Algorithms
- Push/notifications
  - Scale to MM
- Handheld design
  - CPU, RTOS, battery
- Core Mobile Apps
  - Email/IM, PDA,browse
- IP Protocols
  - IMS, SIMPLE/XMPP
- Broadcast
  - Algorithms
- Device Management
  - Software & Config

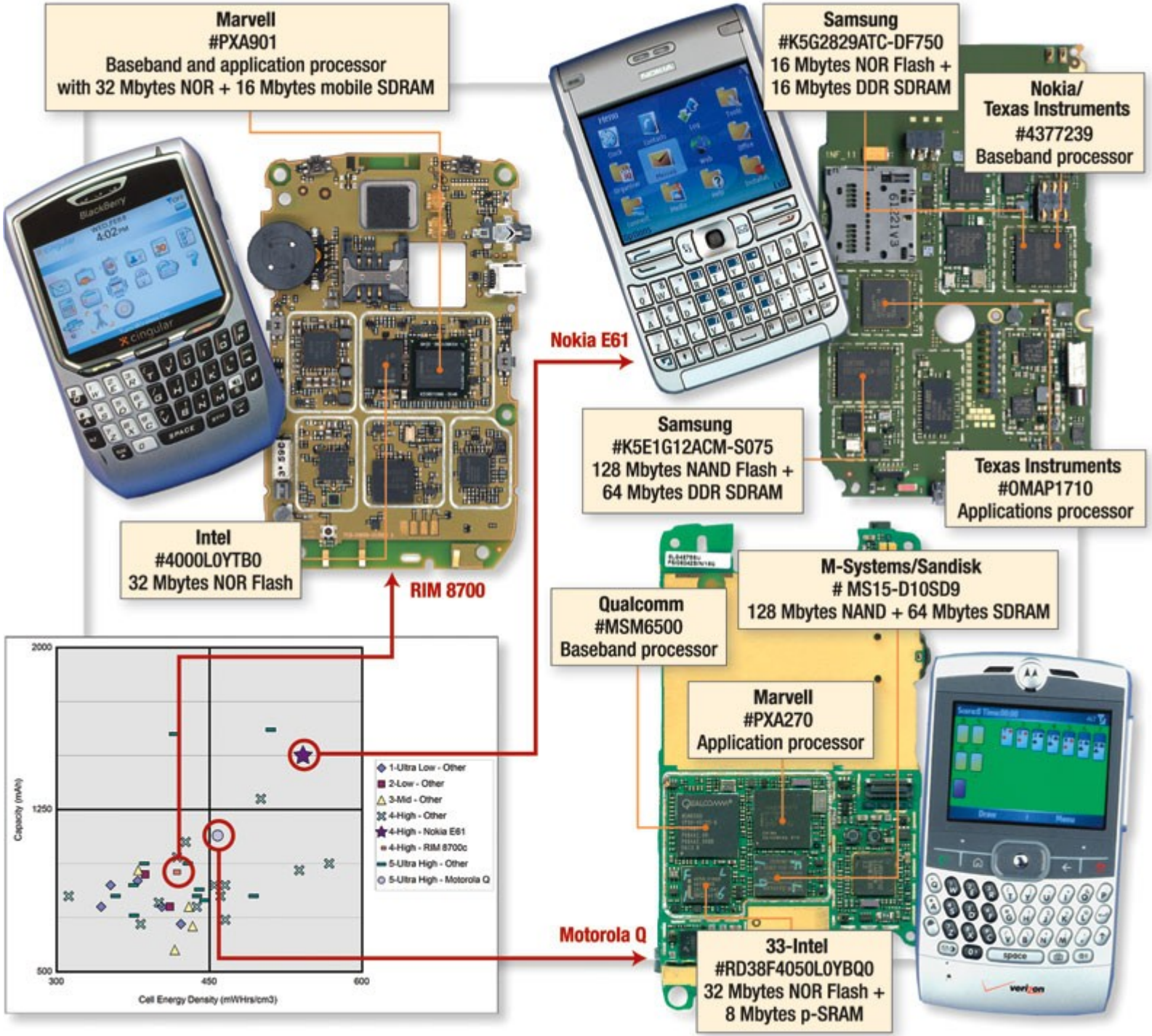
Data on Mobiles

- What is stored?
  - Contacts
  - MP3s, Ringtones, Photos
  - SMS, Call logs
  - Email, calendar, tasks, ..
  - Code (radio, app)
  - Browser cache
  - Configuration data
- How is it stored?
  - File Systems
    - Block file systems
    - Flash file systems
  - Databases
    - On file systems
    - MMDB for Flash
  - New hardware
    - Uses NAND Flash

What is Flash memory?

Flash Memory

- A Type of EEPROM
  - Used in mobiles, digicams, SD cards, USB drives, ..
  - Poised to replace HDD in laptops & elsewhere
  - Comes in two kinds: NOR Flash and NAND Flash
- Properties common to NAND and NOR
  - Non-volatile
  - Erase/write by blocks only
  - Finite number of erase/write cycles
  - Compact, energy efficient and inexpensive
    - Although NAND is more so



Source: mobilehandsetdesign.com

NOR vs. NAND

- NOR Flash
  - Random access
  - Slow write/erase
  - Replaces EEPROM
  - Suitable for code
  - Relatively reliable
- NAND Flash
  - No random access
  - Faster write/erase
  - Replaces HDD
  - Suitable for data
  - Single-bit errors

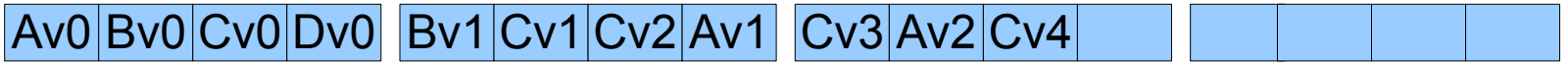
Sector write/erase

- Can only erase in sectors
  - Write operations can only set 1's to 0's
  - Must set 0's to 1's by entire blocks
  - Sector size varies from kilobytes to megabytes
  - Erase cycles are long for NOR flash
- Sectors vs. Pages in NAND
  - Page: Unit of read. Typically 512 bytes to 4KB
  - Sector: Unit of erase. Typically 32 to 128 pages

Wear leveling

- Finite number of write/erase cycles
  - Typical range: 10,000 to 1,000,000
  - May be acceptable for, say, an MP3 archive
    - (assuming reasonable implementation, of course)
  - But not for the global allocation map of an RDBMS
- Wear leveling distributes writes across sectors
  - Isolate applications from wear leveling
  - Needs work to support traditional file systems

Block File System on Flash



- Challenges
  - Erase by sectors. Must move active pages.
  - Maintain Logical Block <=> Sector+Page Mapping
    - Direct map (Sector+Page => Block) in Flash
    - Reverse map (Block => Sector+page) in RAM+Flash
  - Wear leveling
    - What if block C is hot and D is cold?

Flash File Systems

- Why bother?
  - Example: Global allocation map
  - Example: Delete a file
- Log structured file systems
  - New versions of pages written to end of log
  - Variable length inodes
  - (Logical inodes => Sector+page) map in RAM
- Implementations
  - JFFS, YAFFS, TrueFFS

Reading Exercise

- Please read survey Paper

  Eran Gal and Sivan Toledo: ”*Algorithms and Data Structures for Flash Memories*,” ACM Computing Surveys, Vol. 37, No. 2, June 2005, pp. 138-163

  - Section 2: Block mapping technique
    - Except Section 2.3
  - Section 3: Flash specific file systems
    - Except Sections 3.2, and 3.5-3.8
- Unfortunately, most techniques are proprietary
  - Open source (YAFFS, JFFS2, ..) is your best bet
  - Keep an eye on hardware advances

Device Databases

- Two distinct approaches
  - Retrofit small footprint RDBMS
  - Build from scratch
- Retrofit RDBMS
  - Mature, optimized
  - API and applications
  - Share code & impl
  - Schema & Structure
  - Query processing
- Build from scratch
  - Optimize for Flash
  - Little/no serialization
    - Closer to heap
  - Enable new modes

Object-relational mapping

- Example
  - Java serialization
  - Hibernate.org
- Strengths
  - Leverages mature RDMBS technology
  - Tailor objects to applications
- Limitations
  - May lead to inefficiency and bloated code
  - Does not enable any new paradigms

Small footprint RDBMS

- Examples
  - Apache Derby (CloudScape)
  - Sybase iAnywhere
  - SQLite.org (in Google Gears, Adobe AIR & WebKit)
  - Berkeley DB/SleepyCat

(Example: Searching contacts)

Direct storage of objects

- Flash is non-volatile
  - Why not leave the heap as is between reboots?
  - Not as simple as it appears
- Challenges
  - Flash idiosyncrasies
    - Block erase/writes, and wear leveling
    - Serial reads or page prefetches for NAND Flash
      - Slow erase cycles pretty much rules out NOR Flash
  - Transactional properties
    - How to ensure atomicity?

XIP: Execute-in-place

- Where do you store the code?
  - Need random access read
  - Rarely updated
  - Ideal for NOR Flash
- But, NAND Flash is cheaper than NOR Flash
  - Problem: Reads are serial
  - Solution: SRAM cache with prefetch logic
  - Example: Samsung OneNAND

Recap

- File Systems
  - Block file systems
  - Flash-specific file systems
  - Reading assignment: ACM Survey Paper
- Databases
  - On file systems
  - MMDB for Flash
- Hardware advances
  - Enable XIP for NAND Flash