K-Best Validation

- Very good accuracy for < 8ms
  - Within one timer interval
  - Even when heavily loaded
- Less accurate of > 10ms
  - Light load: ~4% error
  - Interval clock interrupt handling
  - Heavy load: Very high error

Compensate For Timer Overhead

- Subtract Timer Overhead
  - Estimate overhead of single interrupt by measuring periods of inactivity
  - Call interval timer to determine number of interrupts that have occurred
- Better Accuracy for > 10ms
  - Light load: 0.2% error
  - Heavy load: Still very high error

K-Best on NT

- Acceptable accuracy for < 50ms
  - Scheduler allows process to run multiple intervals
- Less accurate of > 10ms
  - Light load: 2% error
  - Heavy load: Generally very high error
Time of Day Clock

- Unix gettimeofday() function
- Return elapsed time since reference time (Jan 1, 1970)
- Implementation
  - Uses interval counting on some machines
    - Coarse grained
  - Uses cycle counter on others
    - Fine grained, but significant overhead and only 1 microsecond resolution

```c
#include <sys/time.h>
#include <unistd.h>

struct timeval tstart, tfinish;
double tsecs;
gettimeofday(&tstart, NULL); P();
gettimeofday(&tfinish, NULL);
tsecs = (tfinish.tv_sec - tstart.tv_sec) +
       1e6 * (tfinish.tv_usec - tstart.tv_usec);
```

K-Best Using gettimeofday

- Linux
  - As good as using cycle counter
- Windows
  - Implemented by interval counting
  - For times > 10 microseconds
  - Too coarse-grained

Measurement Summary

- Timing is highly case and system dependent
  - What is overall duration being measured?
    - > 1 second: interval counting is OK
    - << 1 second: must use cycle counters
  - On what hardware / OS / OS version?
    - Accessing counters
      - How gettimeofday is implemented
    - Timer interrupt overhead
    - Scheduling policy

Devising a Measurement Method

- Long durations: use Unix timing functions
- Short durations
  - If possible, use gettimeofday
  - Otherwise must work with cycle counters
  - K-best scheme most successful

CS 201
Computer Systems Programming II – Memory

Notes based on instructors notes from B&O’s web site: http://csapp.cs.cmu.edu
Random-Access Memory (RAM)

- Key features
  - RAM is packaged as a chip.
  - Basic storage unit is a cell (one bit per cell).
  - Multiple RAM chips form a memory.

- Static RAM (SRAM)
  - Each cell stores bit with a six-transistor circuit.
  - Retains value indefinitely, as long as it is kept powered.
  - Relatively insensitive to disturbances such as electrical noise.
  - Faster and more expensive than DRAM.

- Dynamic RAM (DRAM)
  - Each cell stores bit with a capacitor and transistor.
  - Value must be refreshed every 10-100 ms.
  - Sensitive to disturbances.
  - Slower and cheaper than SRAM.

SRAM vs DRAM Summary

<table>
<thead>
<tr>
<th></th>
<th>Trans. per bit</th>
<th>Access time</th>
<th>Persist?</th>
<th>Sensitive?</th>
<th>Cost</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>6</td>
<td>1X</td>
<td>Yes</td>
<td>No</td>
<td>100x</td>
<td>cache memories</td>
</tr>
<tr>
<td>DRAM</td>
<td>1</td>
<td>10X</td>
<td>No</td>
<td>Yes</td>
<td>1X</td>
<td>Main memories, frame buffers</td>
</tr>
</tbody>
</table>

Conventional DRAM Organization

- $d \times w$ DRAM:
  - $dw$ total bits organized as $d$ supercells of size $w$ bits

Reading DRAM Supercell (2,1)

- Step 1(a): Row access strobe (RAS) selects row 2.
- Step 1(b): Row 2 copied from DRAM array to row buffer.
Reading DRAM Supercell (2,1)

- Step 2(a): Column access strobe (CAS) selects column 1.
- Step 2(b): Supercell (2,1) copied from buffer to data lines, and eventually back to the CPU.

In Class Exercise

- Problem 6.1

Enhanced DRAMs

- All enhanced DRAMs are built around the conventional DRAM core.
  - Fast page mode DRAM (FPM DRAM)
    - Access contents of row with \((RAS, CAS, CAS, CAS, CAS)\) instead of \((RAS, CAS, (RAS, CAS, (RAS, CAS, (RAS, CAS)))\).
  - Extended data out DRAM (EDO DRAM)
    - Enhanced FPM DRAM with more closely spaced CAS signals.
  - Synchronous DRAM (SDRAM)
    - Driven with rising clock edge instead of asynchronous control signals.
  - Double data-rate synchronous DRAM (DDR SDRAM)
    - Enhancement of SDRAM that uses both clock edges as control signals.
  - Video RAM (VRAM)
    - Like FPM DRAM, but output is produced by shifting row buffer
    - Dual ported (allows concurrent reads and writes)
Nonvolatile Memories

- DRAM and SRAM are volatile memories
  - Lose information if powered off.
- Nonvolatile memories retain value even if powered off.
  - Generic name is read-only memory (ROM).
  - Misleading because some ROMs can be read and modified.
- Types of ROMs
  - Programmable ROM (PROM)
  - Erasable programmable ROM (EPROM)
  - Electrically erasable PROM (EEPROM)
  - Flash memory
- Firmware
  - Program stored in a ROM
    - Boot time code, BIOS (basic input/output system)
    - Graphics cards, disk controllers.

Typical Bus Structure Connecting CPU and Memory

- A bus is a collection of parallel wires that carry address, data, and control signals.
- Buses are typically shared by multiple devices.

Memory Read Transaction (1)

- CPU places address A on the memory bus.

Memory Read Transaction (2)

- Main memory reads A from the memory bus, retrieves word x, and places it on the bus.
Memory Read Transaction (3)

- CPU reads word x from the bus and copies it into register %eax.

Memory Write Transaction (1)

- CPU places address A on bus. Main memory reads it and waits for the corresponding data word to arrive.

Memory Write Transaction (2)

- CPU places data word y on the bus.

Memory Write Transaction (3)

- Main memory reads data word y from the bus and stores it at address A.
Disk Geometry
- Disks consist of **platters**, each with two **surfaces**.
- Each surface consists of concentric rings called **tracks**.
- Each track consists of **sectors** separated by **gaps**.

Disk Geometry (Multiple-Platter View)
- Aligned tracks form a cylinder.

Disk Capacity
- **Capacity**: maximum number of bits that can be stored.
  - Vendors express capacity in units of gigabytes (GB), where 1 GB = $10^9$.
- Capacity is determined by these technology factors:
  - **Recording density** (bits/in): number of bits that can be squeezed into a 1 inch segment of a track.
  - **Track density** (tracks/in): number of tracks that can be squeezed into a 1 inch radial segment.
  - **Areal density** (bits/in^2): product of recording and track density.
- Modern disks partition tracks into disjoint subsets called **recording zones**:
  - Each track in a zone has the same number of sectors, determined by the circumference of innermost track.
  - Each zone has a different number of sectors/track.

Computing Disk Capacity
- **Capacity** = (# bytes/sector) x (avg. # sectors/track) x (# tracks/surface) x (# surfaces/platter) x (# platters/disk)
- **Example**:
  - 512 bytes/sector
  - 300 sectors/track (on average)
  - 20,000 tracks/surface
  - 2 surfaces/platter
  - 5 platters/disk
- **Capacity** = $512 \times 300 \times 20000 \times 2 \times 5$
  = 30,720,000,000
  = 30.72 GB
Disk Operation (Single-Platter View)

The disk surface spins at a fixed rotational rate. The read/write head is attached to the end of the arm and flies over the disk surface on a thin cushion of air. By moving radially, the arm can position the read/write head over any track.

Disk Operation (Multi-Platter View)

read/write heads move in unison from cylinder to cylinder

Disk Access Time

- Average time to access some target sector approximated by:
  - \( T_{\text{access}} = T_{\text{avg seek}} + T_{\text{avg rotation}} + T_{\text{avg transfer}} \)
- **Seek time** \( (T_{\text{avg seek}}) \)
  - Time to position heads over cylinder containing target sector.
  - Typical \( T_{\text{avg seek}} = 9\,\text{ms} \)
- **Rotational latency** \( (T_{\text{avg rotation}}) \)
  - Time waiting for first bit of target sector to pass under r/w head.
  - \( T_{\text{avg rotation}} = 1/2 \times 1/\text{RPMs} \times 60\,\text{sec}/1\,\text{min} \)
- **Transfer time** \( (T_{\text{avg transfer}}) \)
  - Time to read the bits in the target sector.
  - \( T_{\text{avg transfer}} = 1/\text{RPM} \times 1/(\text{avg # sectors/track}) \times 60\,\text{secs}/1\,\text{min} \)

Disk Access Time Example

- Given:
  - Rotational rate = 7,200 RPM
  - Average seek time = 9 ms.
  - Avg # sectors/track = 400.
- Derived:
  - \( T_{\text{avg rotation}} = 1/2 \times (60\,\text{secs}/7200\,\text{RPM}) \times 1000\,\text{ms}/\text{sec} = 4\,\text{ms} \)
  - \( T_{\text{avg transfer}} = 60/7200\,\text{RPM} \times 1/400\,\text{secs/track} \times 1000\,\text{ms}/\text{sec} = 0.02\,\text{ms} \)
  - \( T_{\text{access}} = 9\,\text{ms} + 4\,\text{ms} + 0.02\,\text{ms} \)
- Important points:
  - Access time dominated by seek time and rotational latency.
  - First bit in a sector is the most expensive, the rest are free.
  - SRAM access time is about 4 ns/doubleword, DRAM about 60 ns
    - Disk is about 40,000 times slower than SRAM,
    - 2,500 times slower then DRAM.
Modern disks present a simpler abstract view of the complex sector geometry:

- The set of available sectors is modeled as a sequence of b-sized logical blocks (0, 1, 2, ...)

Mapping between logical blocks and actual (physical) sectors

- Maintained by hardware/firmware device called disk controller.
- Converts requests for logical blocks into (surface, track, sector) triples.

Allows controller to set aside spare cylinders for each zone.

- Accounts for the difference in “formatted capacity” and “maximum capacity”.