Introduction

- Monitor: A tool used to observe system activities

- Use:
  - Workload Characterization
  - Finding Performance Bottlenecks
  - Model Validation
  - System Debugging
  - Security Checks
Terminology

- **Event**
  - Any change in the system state
  - Examples: context switch, packet arrival, user-loggin, system shutdown, etc.

- **Trace**
  - A sequence of events logged, normally with a timestamp on each event.

- **Resolution/Granularity**
  - How frequently do the events occur or captured?
  - Examples:
    - an instruction in almost every cycle
    - a cache miss in every 100-1000 cycles
    - a context switch in every 1 ms
    - a system shutdown in every two-week period

Terminology (cntd.)

- **Overhead**
  - The extra amount of resources (time, space, cost, power, etc.) required during monitoring.

- **Perturbation**
  - The impact of monitoring on the system’s behavior.
  - Example: A network monitor program that uses the same network as the one it’s monitoring.
Monitory Types

- Hardware Monitors
- Software Monitors
- Hybrid Monitors

Software Monitors

- A piece of code, triggered and executed either periodically or upon the occurrence of an event

- Examples:
  - Operating System Level Event Counters
    - Process creation/termination
    - Page faults and memory allocations
    - Packets sent/received
  - Web Server Logger
    - Recording a log entry for each incoming request
  - Cluster Heartbeat
    - Activated periodically to send a message to each node in a cluster to figure out whether the node is still alive
Software Monitors

- **Advantages**
  - **Flexible**
    - The definition of events can be arbitrarily complex
      - cache miss, page fault, driver load
  - **Extensible**
    - New events can be defined and monitored dynamically.

- **Disadvantages**
  - High overhead
  - High perturbation

- Don’t use software monitors for ultra-high frequency events
  - Examples: instructions retired, cache misses

Hardware Monitors

- Specially-designed hardware equipment attached to the monitored device.
  - No software resources is consumed during monitoring.
  - Examples: Hardware Performance Counters, Power Analyzers, Thermal Sensors

  or

- Device firmware (software) monitors events that occur in the device.
  - Still no software resources is consumed on the host computer. But the device itself is executing some software to do the monitoring.
  - Example: Network card collecting statistics on the amount of information being transmitted/received.
Hardware Monitors

- **Advantages**
  - **High Resolution**
    - The software is not involved
  - **Reliable**
    - The monitor hardware is supposed to be simple and built by experts.
  - **Available**
    - In the case of software failures, hardware monitors still work

- **Disadvantages**
  - **Inflexible**
    - Only a fixed set of events can be monitored.
    - High-level events are difficult to monitor by a hardware monitor (example: hardware doesn’t know anything about user log-in)
  - **High Cost**
    - Custom-made hardware is usually very expensive.

Hybrid Monitors

- **Combination of Hardware and Software Monitors**
  - Low-level events gathered by the hardware monitor
  - Software pre-processes the data generated by the hardware monitor on the fly, to compose higher-level events.

- **Example**
  - Hardware monitors all activities on memory bus
  - Software tracks processes that generate these activities and infers a lock contention
Discussion

- Which type of monitor would you use for the following event types?
  - Number of instructions executed by CPU
  - Number of Interrupts
  - Number of bytes received from a network device
  - Query completion for a DBMS
  - Hard disk bad sector found
  - Network connection timeout

Monitors Trigger Type

- Event-Driven
  - Monitors is triggered by the occurrence of an event.

- Timer-Driven (Sampling Monitors)
  - Monitor is triggered upon fixed time intervals.

- Event-Sampling
  - Monitor is triggered once in every N occurrence of an event.
Trigger Points

- Pre-defined Points
  - There are a fixed set of event-handling points in system execution, where the monitor can be explicitly triggered by the system.
  - Examples: page fault, timer interrupt, context-switch, packet arrival (interrupt), data-base transaction commit or abort

- Arbitrary Points
  - The program or subsystem needs to be instrumented with triggers (or probe points)
  - Examples: monitoring all function calls to build a call-graph, assertion checking

Instrumentation

- Static
  - The program source code is changed, manually or dynamically to include new probe-points.
  - The source code must be available.
  - The program must be recompiled and re-executed for turning monitoring on and off.

- Dynamic
  - The probe points are dynamically inserted into an executable binary, by manipulating the sequence of instructions.
  - No need for source code
  - Flexible as the instrumentation can be turned on and off easily.

- Famous Instrumentation Systems
  - Valgrind
  - DTrace