Bab 6. Monitoring

Dr. Ir. Yeffry Handoko Putra, M.T

Definisi Monitor

- Suatu tool untuk mengamati kinerja sistem
- observe the performance of systems→ collect performance statistics→ analyze the data→ and display results
- Reason:
  - A system programmer may use a monitor to **find the frequently used segments** of the software and optimize their performance.
  - A systems manager may use a monitor to **measure resource utilizations** and to find the **performance bottleneck**.
  - A systems manager may also use a monitor to tune the system. The system parameters can be adjusted to improve the performance.
  - A systems analyst may use a monitor to **characterize** the workload. The results may be used for capacity planning and for creating test workloads.
  - A systems analyst may use a monitor to **find model parameters**, to validate models, and to develop inputs for models.

Terminologi Monitor

- **Event**
- **Trace**
- **Overhead / artifact**
- **Domain**
- **Input Rate**
- **Resolution**
- **Input Width**

Event:

A change in the system state is called an event. Examples of events are process context switching, beginning of seek on a disk, and arrival of a packet

Trace

A trace is a log of events usually including the time of the event, the type of event, and other important parameters associated with the event

- **Overhead:** Most monitors slightly perturb the system operation. They may consume system resources, such as CPU or storage. For example, the data collected by the monitor may be recorded on the secondary storage. This consumption of system resources is called overhead. One of the goals of monitor design is to minimize the overhead. Sometimes, the term **artifact** is also used to denote the overhead.
- **Domain:** The set of activities observable by the monitor is its domain. For example, accounting logs record information about CPU time; number of disks, terminals, networks, and paging I/O's; number of characters transferred among disks, terminals, networks, and paging device; and clasped time for each user session. These constitute the domain of the accounting logs.
- **Input Rate:** The maximum frequency of events that a monitor can correctly observe is called its input rate. Generally, two input rates are specified: burst mode and sustained. Burst-mode rate specifies the rate at which an event can occur for a short duration. It is higher than the sustained rate, which the monitor can tolerate for long durations.

- **Resolution:**
  The coarseness of the information observed is called the resolution. For example, a monitor may be able to record time only in units of 16 milliseconds. Similarly, the size of buckets used in a histogram may determine the resolution of the histogram.

- **Input Width:**
  The number of bits of information recorded on a event is called the input width. This, along with the input rate, determines the storage required to record the events.

Monitor Classification

Depend of triggering

1. Event-driven Monitoring
2. Sampling monitoring

Depend on displaying ability:

1. On-line monitor
2. Batch Monitor

Software/hardware

1. Software Monitor
2. Hardware Monitor

Software Monitor

1. Activation Mechanism: ***how to trigger the data collection routine***
   a. Trap instruction
   b. Trace mode
   c. Timer interrupt
2. Buffer size
3. Number of buffers
4. Buffer overflow
5. Data compression and analysis
6. On/off Switch
7. Language
8. Priority
9. Abnormal Event of monitoring

- Trap instruction:
  a software interrupt mechanism that transfers control to a data collection routine
  The monitor measures elapsed time for various operating system services using trap instructions at the beginning and at the end of the service code. Thus, to measure I/O service time, a trap instruction placed at the beginning of the I/O service-call-handling routine enables the monitor to record the time clock. After finishing the I/O, another trap instruction reads the clock, subtracts the beginning value, and thus obtains the time spent in the service routine.
- Trace mode: the instruction execution is interrupted after every instruction, and the control is passed on to a data collection routine
  For example, this approach can be used to develop an instruction-trace monitor to produce a program counter histogram (a histogram of instruction addresses). The monitor records the address (program counter) and returns control to the user program..

- timer interrupt
  service provided by the operating system is used to transfer control to a data collection routine at fixed intervals (called sampling) specially suitable for frequent events since the overhead is independent of the event rate. The overhead per activation, input width, and rate of variation of the sampled quantity determine the desired sampling rate. If a counter is being sampled, the sampling should be done so that the probability of counter overflows between sampling is minimized.

HARDWARE MONITORS

1. *Probes*: High-impedance probes are used to observe signals at desired points in the system hardware.
2. *Counters*: These are incremented whenever a particular event occurs.
3. *Logic Elements*: Signals from many probes can be combined using AND, OR, and other logic gates. The combinations are used to indicate events that may increment the counters.
4. *Comparators*: These can be used to compare counters or signal values with preset values.
5. *Mapping Hardware*: This allows histograms of observed quantities to be computed. It consists of multiple comparators and counters.
6. *Timer*: Used for time stamping or for triggering a sampling operation.
7. *Tape/Disk*: Most monitors have built-in tape/disk drives to store the data.

SOFTWARE VERSUS HARDWARE MONITORS

TABLE 7.1 Comparison of Hardware and Software Monitors

Criterion	Hardware Monitor	Software Monitor
Domain	Difficult to monitor operating system events.	Difficult to monitor hardware events unless recognizable by an instruction.
Input rate	Sampling rates of 10^5 per second possible.	Sampling rate limited by the processor MIPS and overhead required.
Time resolution	10 nanoseconds is possible.	Generally 10 to 16 milliseconds.
Expertise	Requires intimate knowledge of hardware.	Requires intimate knowledge of software.
Recording capacity	Limited by memory and secondary storage. Not a problem currently.	Limited by overhead desired.
Input width	Can record several simultaneous events.	Cannot record several simultaneous events unless there are multiple processors.
Monitor overhead	None	Overhead depends upon the input rate and input width. Less than 5% adequate and more than 100% possible.
Portability	Generally portable.	Specific to an operating system.
Availability	Monitoring continues even during system malfunction or failure.	Cannot monitor during system crash.
Errors	Possible to connect the probes to wrong points.	Once debugged, errors are rare.
Cost	High	Medium

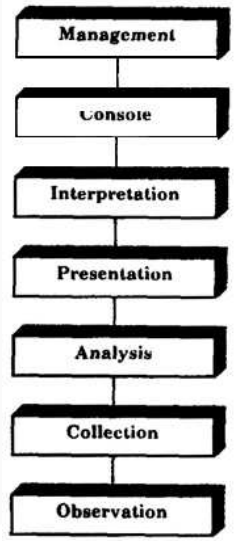
Firmware and Hybrid Monitor

- Firmware monitors are implemented by modifying the processor microcode
- Hybrid monitor : A monitor using a combination of software, hardware, or firmware
  Examples of hybrid monitor: Diamond monitor (Hughes ,1980).

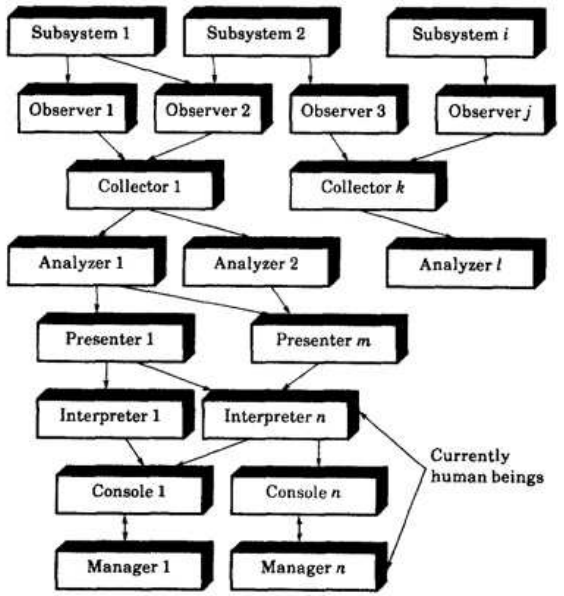
DISTRIBUTED-SYSTEM MONITORS

Layered view of a distributed-system monitor.

1. *Observation*
2. *Collection*
3. *Analysis*
4. *Presentation*
5. *Interpretation*
6. *Console*
7. *Management*



Components of a distributed-system monitor



Layer Detail

- **Observation**
  - **Implicit spying :**
  - **Explicit instrumenting**
  - **Probing**
- **Collection**
- **Analysis**
- **Presentation**
- **Interpretation**
- **Console Functions**

Observation

- **Implicit spying** is the first and least intrusive monitoring technique. It requires promiscuously observing the activity on the system bus or network link. This technique is often used to monitor local-area networks in which all stations can hear all conversations, and one station is designated to be the observer. The advantage is that there is almost no impact on the performance of the system being monitored.

  Implicit-spying observers are often accompanied by one or more *filters* that allow the monitor to decide which activities to record. Not all data observed in a system may be of interest at all times. The filters help decide whether to keep a record of an observed event or to ignore it. The filters generally consist of conditional expressions set by the monitor user. The conditions may be, for example, Boolean, arithmetic, or set membership.
- **Explicit instrumenting** requires incorporating trace points, probe points, hooks, or counters in the system. This approach causes some overhead on the system and is used to augment the data obtained from implicit observing. Each component in the system that needs to be monitored may have to be instrumented differently. However, it helps to have a standard data-naming and reporting format so that other monitor components making use of this data can get it and use it in a device-independent manner.
- **Probing** requires making “feeler” requests on the system to sense its current performance. For example, in a computer network, a specially marked packet sent to a given destination and looped back by the destination may provide information about queueing at the source, at intermediate bridges, the destination station, and back. This information is useful in determining the current load level on a path. It may also be used for diagnostic and reliability analysis.

Collection : Advertising and Soliciting

- **Advertising and Soliciting**
  - **Advertising** consists of observers sending the data on a system bus or a shared medium in such a form that all collectors can receive it. In tightly coupled systems, it is also possible to put it in a shared memory segment that can be accessed by all collectors and observers.
  - **Soliciting** requires that each collector send queries to each observer and get the data individually. The queries may be sent periodically or only on occurrence of certain events.
- The collectors may operate at one or more layers. For example, in monitoring a large network, it is possible to divide the network into several subnetworks,
- When collecting data from several observers, clock synchronization often becomes an important issue.

Three key applications of the monitor are related to the speed, accuracy, and availability of the services

- *Performance Monitoring*
- *Error Monitoring*
- *Configuration Monitoring*

Three key applications of the monitor are related to the speed, accuracy, and availability of the services

- *Performance Monitoring*
  helps to quantify the quality of service when the service is provided correctly. The users are generally interested in observing the system performance in terms of throughput, response time, and utilization of various resources. They may also be interested in seeing summary statistics of various events on the system. Depending upon the frequency of occurrence of each event and their importance, the summary statistics may consist of counts, class counts (or histograms), or a time-stamped trace.
- *Error Monitoring*
  The error is defined as the “incorrect performance.” The system appears to be operating, accepting user requests, and performing the service, but the end result is not what the user wanted. A monitor should provide the error statistics, counts, class counts, or traces in a manner similar to that described under performance statistics. The error statistics on various components of the system may be sorted to determine the unreliable parts of the system.

- *Configuration Monitoring*
  Configuration monitoring relates to the nonperformance of the system components. It allows the user to determine which components are up. A monitor can determine this by promiscuous observation of traffic on the system bus or on the broadcast network medium by polling the components or by sending a marked packet. A monitor should record system initializations and any configuration changes due to components joining or leaving the configuration. The user can get a count, class count, or a trace of these events. The class count is generally that of interevent time—the interval for which the component was up or down.

Exercises

7.1 For each of the following measurements list the type of monitor that can and cannot be used. Which type of monitor would you prefer and why?

a. Interrupt response time

b. Instruction opcode frequency

c. Program reference pattern **d.** Virtual memory reference pattern in a multiprogramming system

e. CPU time required to send one packet on a network

f. Response time for a database query

7.2 For each of the following environments, describe how you would implement a monitor to produce a program counter histogram:

a. Using a hardware monitor

b. Using a software monitor on an IBM PC with the CPU having a trace bit.

c. Using a software monitor on a TRS-80 with the CPU not having a trace bit.

7.3 Choose a computer system or subsystem. Assume that prototypes of systems you selected already exist and you have decided to measure their performance. Make a list of quantities, if any, that you could measure using a

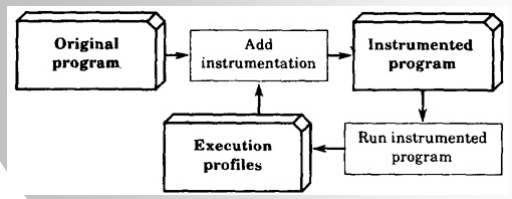
a. Software monitor

b. Hardware monitor **c.** Firmware monitor

In each case, describe how performance metrics of interest to you could be calculated using the quantities measured. Discuss how you would resolve some of the issues you would face in using or designing a monitor for your system.

PROGRAM EXECUTION MONITORS AND ACCOUNTING LOGS

- *Tracing*: To find the execution path of a program.
- *Timing*: To find the time spent in various modules of the program.
- *Tuning*: To find the most frequent or most time-consuming sections of the code.
- *Assertion Checking*: To verify the relationships assumed among the variables of a program.
- *Coverage Analysis*: To determine the adequacy of a test run.



Designing a Program Execution Monitor

1. ***Measurement Unit***: The execution monitor divides the program into smaller measurement units such as modules, subroutines, high-level language statements, or machine instructions..
2. ***Measurement Technique***: The two basic measurement techniques are tracing and sampling.
3. ***Instrumentation Mechanism***: A program has to be compiled and linked before it can be executed. The instrumentation can be added before compilation, during compilation, after compilation (before linking), or during run time.
4. ***Profile Report***: Most program monitors produce an execution profile showing a frequency and time histogram. For large programs, several summaries at different levels of hierarchy may be presented; for example, summaries by modules, and then for each module by procedures, and for each procedure by statements.

For CPU time, these resources are called **self-time** and **inherited time**, respectively. Many monitors have the ability to limit or expand (zoom in or zoom out) the amount of detail.

TECHNIQUES FOR IMPROVING PROGRAM PERFORMANCE

- code optimization
  finding the dynamic frequency of execution of various program modules
- I/O optimization
  consists of combining several I/O requests into larger records, changing the file access method, and caching or prefetching data
- paging optimization.
  consists of observing the page reference pattern and reorganizing program segments so that the paging activity is minimized.