Monitors

- A monitor is a tool used to observe system
  - Observe performance
  - Collect performance statistics
  - May analyze the data
  - May display results
  - May even suggest remedies

- Systems programmer may profile software
- System manager may measure resource utilization to find bottleneck
- May use to tune system
- May use to characterize workload
- May use to develop models or inputs for models

That which is monitored improves. – Source unknown

Example: gprof

<table>
<thead>
<tr>
<th>% cumulative</th>
<th>time</th>
<th>seconds</th>
<th>seconds</th>
<th>calls</th>
<th>us/call</th>
<th>us/call</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.24</td>
<td>0.47</td>
<td>0.06</td>
<td>708202</td>
<td>0.08</td>
<td>0.08</td>
<td>slip</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.49</td>
<td>0.00</td>
<td>708199</td>
<td>0.00</td>
<td>0.00</td>
<td>position</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.49</td>
<td>0.00</td>
<td>50</td>
<td>0.00</td>
<td>0.00</td>
<td>GetFlag</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.49</td>
<td>0.00</td>
<td>10</td>
<td>0.00</td>
<td>0.00</td>
<td>setup</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.49</td>
<td>0.00</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>gettimeofday</td>
<td></td>
</tr>
</tbody>
</table>

- Profile dog-mailman simulation
  - gcc with "-pg" flag
  - Adds timing "hooks" into your code
  - gprof A.out gmon.out
  - gmon.out has profile information from run
  - Also provides call graph information

Example: tcpdump (1 of 2)

- tcpdump – open source network sniffer
- tcpdump –w dump.out
tcpdump –r dump.out
* Also, ethereal and tethereal

Example: tcpdump (2 of 2)

3.8 Kbps
4.0 Kbps
6.8 Kbps

Outline

- Introduction
- Terminology
- Software Monitors
- Hardware Monitors
- Monitoring Distributed Systems
**Terminology**

- **Event** – a change in the system state.
  - Ex: context switch, seek on disk, arrival of packet
- **Trace** – log of events, with time, type, etc
- **Overhead** – most perturb system, use CPU or storage. Sometimes called artifact. Goal is to minimize artifact
- **Domain** – set of activities observable. Ex: network logs packets, bytes, types of packet
- **Input rate** – maximum frequency of events can record. Burst and sustained. Ex: tcpdump will report “missed”
- **Resolution** – coarseness of information. Ex: gprof records 0.01 seconds.
- **Input width** – number of bits recorded for each event. Input rate x width = storage required

**Monitor Classification**

- **Implementation level**
  - Software, Hardware, Firmware, Hybrid
- **Trigger mechanism**
  - Event driven – low overhead for rare event, but higher if event is frequent
  - Sampling (timer driven) – ideal for frequent event
- **Display**
  - On-line – provide data continuously. Ex: tcpdump
  - Batch – collect data for later analysis. Ex: gprof

**Outline**

- Introduction
- Terminology
- **Software Monitors**
- **Hardware Monitors**
- Monitoring Distributed Systems

**Software Monitors**

- Record several instructions per event
  - In general, only suitable for low frequency event or overhead too high
  - Overhead may be ok if timing does not need to be preserved. Ex: profiling where want relative time spent
- Lower input rates, resolutions and higher overhead than hardware
- But, higher input widths, higher recording capacities
- Easier to develop and modify

**Issues in Software Monitor Design**

- **Activation Mechanism**
  - How to trigger to collect data
  - Trap - software interrupt at appropriate points. Collect data. Like a subroutine.
    - Ex: to measure I/O trap before I/O service routine and record time, trap after, take diff
  - Trace - collect data every instruction.
    - Enormous overhead. Time insensitive.
  - Timer interrupt - fixed intervals. If sampling counter, beware of overflows

- **Buffer Size**
  - Store recorded data in memory until write to disk
  - Should be large
    - to minimize need to write frequently
  - Should be small
    - so don’t have a lot of overhead when write to disk
    - so doesn’t impact performance of system
  - So, optimal function of input rate, input width, emptying rate
**Issues in Software Monitor Design**  
- **Buffers**  
  - Usually organized in a ring  
  - Allows recording (buffer-emptying) process to proceed at a different rate than monitoring (buffer-filling) process  
  - Monitoring may be bursty  
  - Since cannot read while processes is writing, a minimum of two buffers required for concurrent access  
  - May be circular for writing so monitor overwrites last if recording process too slow  
  - May compress to reduce space, but adds overhead  

- **Misc**  
  - On/Off  
    - Most hardware monitors have on/off switch  
    - Software can have "if ... then" but still same overhead. Or can "compile out"  
      - Ex: remove "-pg" flag  
      - Ex: with #define and ifndef  
  - Priority  
    - Asynchronous, then keep low. If timing matters, need it sufficiently high so doesn't cause skew

**Outline**  
- Introduction  
- Terminology  
- Software Monitors  
- Hardware Monitors  
- Monitoring Distributed Systems

**Hardware Monitors**  
- Generally, lower overhead, higher input rate, reduced chance of introducing bugs  
- Can increment counters, compare values, record histograms of observed values ...  
- Usually, gone through several generations and testing so is robust

**Software vs. Hardware Monitor**  
- What level of detail to measure?  
  - Software more limited to system layer code (OS, device driver) or application or above  
    - Hardware may not be able to get above information  
- What is input rate? Hardware tends to be faster  
- Expertise?  
  - Good knowledge of hardware needed for hardware monitor  
  - Good knowledge of software system (programmer) needed for software monitor  
- Most hardware monitors can work with a variety of systems, but software may be system specific  
- Most hardware monitors work when there are bugs, but software monitors brittle  
- Hardware monitors more expensive

**Outline**  
- Introduction  
- Terminology  
- Software Monitors  
- Hardware Monitors  
- Monitoring Distributed Systems
Monitoring Distributed Systems

- More difficult than single computer system
- Monitor itself must be distributed
- Easiest with layered view of monitors
- May be zero+ components of each layer
- Many-to-many relationship between layers

Components of a Distributed Systems Monitor

- Management
- Console
- Interpretation
- Presentation
- Analysis
- Collection
- Observation

Observation (1 of 2)
- Concerned with data gathering
- Implicit spying - promiscuously observing the activity on the bus or network link
  - Little impact on existing system
  - Accompany with filters that can ignore some events
  - Ex: tcpdump between two IP address
- Explicit instrumentation - incorporating trace points, hooks, ... Adds overhead, but can augment implicit data
  - Ex: may have application hooks logging when data sent

Observation (2 of 2)
- Probing - making "feeler" requests to see performance
  - Ex: packet pair techniques to gauge capacity
- There is overlap between the three techniques, but often show part of system that others cannot

Collection
- Data gathering component, perhaps from several observers
  - Ex: I/O and network observer on one host could go to one collector for the system
- May have different collectors share same observers
  - Collectors can poll observers for data
  - Or observers can advertise when they have data
- Clock synchronization can be an issue
  - Usually aggregate over a large interval to account for skew

Analysis
- More sophisticated than collector
- Division of labor unclear, but usually, if fast, infrequent in observer, but if takes more processing time, put in analyzer
- Or, if it requires aggregate data, put in analyzer
  - Ex: if successful transaction rate depends upon disk error rate and network error rate then analyzer needs data from multiple observers
- General philosophy, simplify observers and push complexity to analyzers
Presentation (1 of 2)

• User interface, closely tied with monitor function
• Three key functions
• 1) Performance monitoring – helps quantify if service provided is correct
  - Throughput, response time, utilization of different components
  - Summary statistics
  - Time stamped traces

Presentation (2 of 2)

• 2) Error monitoring – incorrect performance
  - Error statistics, counts or traces
  - Maybe sort to help determine what part of system is unreliable
• 3) Configuration monitoring – non-performance of the system components
  - Tell which are up
  - Show initial configurations
  - May show only incremental configurations
  - Scope to allow zoom or whole system

Interpretation and Console

• Interpreter – uses set of rules to make judgments about state of system
  - Often need expert system to warn about faults before they occur
  - May suggest configuration changes
• Console functions – allow system manager to change system, bring up and down, allow remote diagnostics
  - Ideally, one console can get feedback and apply configuration, but some parts may be vendor specific