CS533
Modeling and Performance Evaluation of Network and Computer Systems

Monitors

(Chapter 7)

Monitors

That which is monitored improves. – Source unknown

- 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

Example: gprof

% time	cumulative seconds	self seconds	calls	self us/call	total us/call	name
83.67	0.41	0.41	10	41000.00	49000.00	runSim
12.24	0.47	0.06	708202	0.08	0.08	slip
4.08	0.49	0.02	708202	0.03	0.11	speed
0.00	0.49	0.00	708199	0.00	0.00	position
0.00	0.49	0.00	50	0.00	0.00	GetFlag
0.00	0.49	0.00	10	0.00	0.00	setup
0.00	0.49	0.00	1	0.00	0.00	gettime

- 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)

```
04:58:53.680001 cs.WPI.EDU.59457 > saagar.wpi.edu.ssh: P 193:241(48) ack 256 win
 27512 <nop,nop,timestamp 51273481 430361043> (DF)
04:58:53.680610 saagar.wpi.edu.ssh > cs.WPI.EDU.59457: P 256:304(48) ack 241 win
 10336 <nop,nop,timestamp 430361101 51273481> (DF) [tos 0x10]
04:58:53.680977 cs.WPI.EDU.59457 > saagar.wpi.edu.ssh: . ack 304 win 27512 <nop,
nop,timestamp 51273481 430361101> (DF)
04:58:53.691672 saagar.wpi.edu.wizard > ns.WPI.EDU.domain:  6143+ A? sprobe.cs.w
ashington.edu. (42) (DF) [tos 0x10]
04:58:53.692187 saagar.wpi.edu.ssh > cs.WPI.EDU.59457: P 304:512(208) ack 241 wi
n 10336 <nop,nop,timestamp 430361103 51273481> (DF) [tos 0x10]
04:58:53.692436 ns.WPI.EDU.domain > saagar.wpi.edu.wizard:  6143 2/6/3 CNAME[|do
main] (DF)
04:58:53.692905 cs.WPI.EDU.59457 > saagar.wpi.edu.ssh: . ack 512 win 27512 <nop,
nop,timestamp 51273482 430361103> (DF)
04:58:53.693022 saagar.wpi.edu.11032 > wicse.cs.washington.edu.http: S 637950672
:637950672(0) win 5840 <mss 1460,sackOK,timestamp 430361103 0,nop,wscale 0> (DF)
 [tos 0x8]
04:58:53.693193 saagar.wpi.edu.ssh > cs.WPI.EDU.59457: P 512:624(112) ack 241 wi
n 10336 <nop,nop,timestamp 430361103 51273482> (DF) [tos 0x10]
04:58:53.693615 cs.WPI.EDU.59457 > saagar.wpi.edu.ssh: . ack 624 win 27512 <nop,
nop,timestamp 51273482 430361103> (DF)
```

- *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

(Picture here that cannot convert to PDF)

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

Issues in Software Monitor Design – 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

Issues in Software Monitor Design – Misc

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

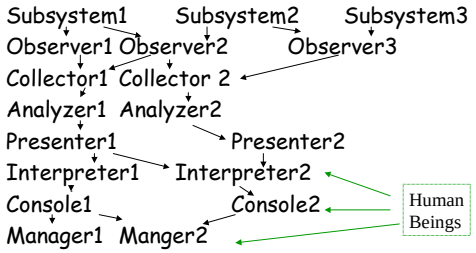
Monitoring Distributed Systems

- 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

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

19

Components of a Distributed Systems Monitor



20

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

21

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

22

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

23

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

24

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