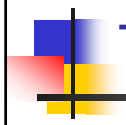
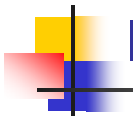


The Linux Trace Tool

By
Sushant Rewaskar

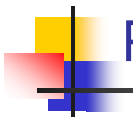
Outline

- Introduction
- Related tools
- Architecture of the tool
- Overhead
- Usage
- Demo



Introduction

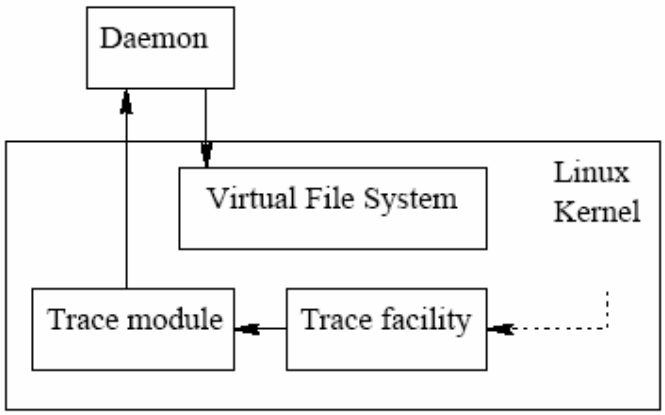
- Official website
  - http://www.opersys.com/LTT/index.html
- LTT captures all required information to reconstruct a systems behavior later
- LTT is open source
- It has low overheads



Related tools

- Analysis of individual application
  - DCPI, Morph, path profiler, Quantify and GProf
- Inbuilt tools in Linux
  - PS, Top
- Tracing certain key events
  - WindView, DejaView, Hyper kernel trace
  - SimOS

Architecture of the tool



Data Collection: kernel trace facility

- Provides a unique entry point for all other kernel tracing facilities.
- Events are passed to the trace module.
- Allows the trace module to configure the tracing parameters.

Data collection: Kernel instrumentation

Event type	*Event subtype*	*Event detail*
Trace start	None	Trace module specific
System call entry	None	System call ID and instruction pointer
System call exit	None	None
Trap entry	None	Trap ID and instruction pointer
Trap exit	None	None
Interrupt entry	None	Interrupt ID
Interrupt exit	None	None
Scheduling change	None	Incoming task, outgoing task and outgoing task's state
Kernel timer	None	None
Bottom half	None	Bottom half ID

Data collection: Kernel instrumentation

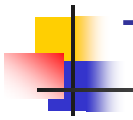
Process	Create kernel thread	Thread start address and PID
	Fork	PID of created process
	Exit	None
	Wait	PID waited on
	Signal	Signal ID and destination PID
	Wakeup	Process PID and state before wakeup
File system	Buffer wait start	None
	Buffer wait end	None
	Exec	File name
	Open	File name and descriptor
	Close	File descriptor
	Read	File descriptor and quantity read
	Write	File descriptor and quantity written
	Seek	File descriptor and offset
	Ioctl	File descriptor and command
	Select	File descriptor and timeout
	Poll	File descriptor and timeout

Data collection: Kernel instrumentation

Timer	Expired	None
	Set itimer	Type and time
	Set timeout	Time
Memory	Page allocate	Size order
	Page free	Size order
	Swap in	Page address
	Swap out	Page address
	Page wait start	None
	Page wait end	None

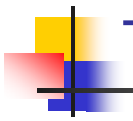
Data collection: Kernel instrumentation

Socket communication	Socket call	Call ID and socket ID
	Socket create	Socket type and ID of created socket
	Socket send	Type of socket and quantity sent
	Socket receive	Type of socket and quantity received
Inter-process communication	System V IPC call	Call ID and entity ID
	Message queue create	Message queue ID and creation flags
	Semaphore create	Semaphore ID and creation flags
	Shared memory create	Shared memory ID and creation flags
Network	Incoming packet	Protocol type
	Outgoing packet	Protocol type



Data Collection: Trace module

- Retrieves additional information about the event (time stamp, CPU-ID)
- Filters events based on the configuration.
- Records events in trace buffer.
- Signal daemon when buffers are filled

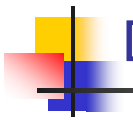


Data collection: Trace Daemon

- Provides user with a number of option to control the tracing process
  - e.g. duration of trace
- Configures the trace module according to user options
- Reads the '/proc' directory to record the initial state of the system before tracing started.
- Commits the buffer when it receives "buffer full" signal.

Data Collection: Analysis tool

- Uses raw traces and system state(initial state obtained form the proc file) to reconstruct system behavior
- Extract system wide and per process statistics
- Provides user with a graphical and text system behavior reconstruction
- Provides a graphical interface to easily browse and analyze traces



Data Collection Overhead

- Tested the programs with following 6 configurations
  - Original 2.2.13 kernel. (base configuration)
  - Modified kernel. All events are ignored.
  - Events are logged by the module but daemon is not running.
  - Daemon is running but it is not writing data to file
  - Events are logged ,reported and written
  - Events are recorded only for core kernel events.

Data Collection Overhead

Conf.	*Compile*	*Archive*	*Compress*	*Desktop*
1	240.2	357.4	141.2	246.8
2	240.8	358.0	141.4	249.2
Δ	0.25 %	0.17 %	0.14 %	0.97 %
3	243.6	359.1	141.1	252.2
Δ	1.42 %	0.48 %	-0.07 %	2.19 %
4	245.7	358.1	141.6	252.9
Δ	2.29 %	0.20 %	0.28 %	2.47 %
5	246.9	365.5	141.4	258.3
Δ	2.79 %	2.27 %	0.14 %	4.66 %
6	246.3	363.6	141.6	252.9
Δ	2.54 %	1.74 %	0.28 %	2.47 %

Data Collection Overhead

- The overhead in term of disk space is depends on the type of tracing
  - For a graphical interface the average is around 0.5Mb/sec
  - For text interface it's around 0.1Mb/sec.

Workstation characterization

- Applications where monitored for 30 seconds
  - X server, Netscape, start office, x11amp and a script running ps every 5 seconds

Workstation characterization

Application	*ps*	*User*	*Running*	*Wait I/O*
X server	8.6	12.79	15.28	0.04
netscape	3.25	15.43	17.17	1.85
staroffice	2.7	4.73	5.52	2.60
x11amp	1.15	1.62	2.19	0
ps script	0.43	0.04	0.08	0

Small server characterization

- Applications where monitored for 30 seconds
  - X servers, ftp daemon, KDE window manager kwm and a script running ps every 5 seconds

Small server characterization

Application	*ps*	*User*	*Running*	*Wait I/O*
X server	7.2	0.48	3.45	0
ftp #1	13.1	0.29	10.92	0.26
ftp #2	10.8	0.32	11.77	0.63
kwm	1.63	1.33	21.2	0
ps script	3.6	0.38	2.25	0

Trace example

Syscall entry	(678777)	1021	SYSCALL : write
File system	(678779)	1021	WRITE : 3
File system	(679107)	1021	START I/O WAIT
Sched change	(679151)	0	IN : 0; OUT : 1021
	...		
IRQ entry	(691806)	0	IRQ : 14, IN-KERNEL
Process	(691818)	0	WAKEUP PID : 1021
IRQ exit	(691823)	0	
Sched change	(691824)	1021	IN : 1021; OUT : 0
File system	(691826)	1021	START I/O WAIT
File system	(691827)	1021	END I/O WAIT
Syscall exit	(691936)	1021	
Syscall entry	(691941)	1021	SYSCALL : sigreturn

Tracing the System

- tracedaemon -ts *T*./out.trace ./out.proc
  - T is the number of seconds you want to trace the system.
- Other options are
  - **-e : traces on the specified events . few of the events are listed below**
    - **"START" - Trace start**
    - **"SYS_ENTRY" - System call entry**
    - **"SYS_EXIT" - System call exit**
  - **-c Trace CPU-ID. Only events occurring on CPU-ID.**
  - **-P Trace PID. Only include process number PID.**
  - **-g Trace GID. Only include processes part of GID.**
  - **-u Trace UID. Only include processes belonging to UID.**

Data Visualization

- Use command “tracevisualizer”
- Icon legend

Icon	Corresponding event
	Bottom half
	I/O end
	I/O start
	IRQ
	Kernel Timer
	Schedule change
	System call
	Trap

Demo