SpyKer

Auto-instrumented tracing of system events for rapid time-to-market

LYNUXWORKS™

Detailed information on the interaction of patterns and timings of system activities is crucial to understanding how well an application is performing. SpyKer™, the first dynamically instrumented system trace analyzer, provides this information so developers can:

- Comprehend what is going on in a system over time
- Track down elusive application bugs
- Fine-tune the performance of embedded systems

Fast, easy and non-intrusive

Traditional performance analysis tools require pre-instrumenting an application with special calls or running it under a specially instrumented kernel. SpyKer, on the other hand, enables an ordinary kernel to be auto-instrumented at runtime by dynamically patching a LynxOS or BlueCat Linux kernel binary with logging calls. No lengthy instrumentation or kernel rebooting is necessary, so development costs are minimized and debugging processes proceed faster than ever.

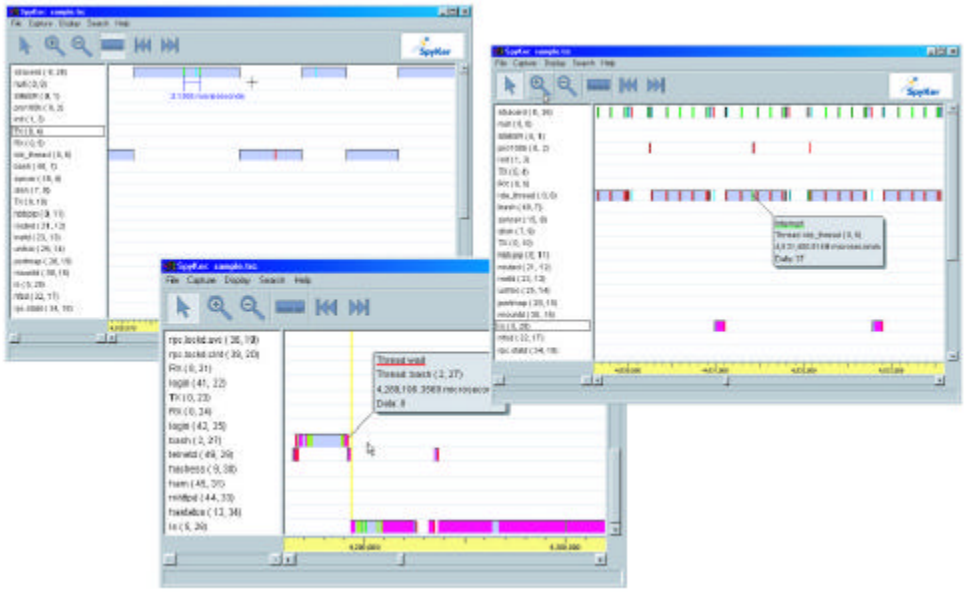
Moreover, the overhead of a SpyKer trace patch is exceedingly low, thus minimizing its impact on the target system. This enables truly objective measurements and eliminates any problem of timing invasiveness standing in the way of finding particularly evasive bugs.

Visibility into program execution

SpyKer combines its powerful auto-instrumenting facility with equally powerful front-end visualization capabilities. Developers are empowered to gather, display and interact with data concerning their application, making it easy to view normally difficult-to-understand relationships within the application.

The front-end GUI displays event data in an easy-to-comprehend fashion, including allowing multiple windows with different visualization to be displayed concurrently. A set of intuitive visualization tools enable users to:

- Invoke "popup" boxes that display information about curser-selected events



SpyKer Advantages

- Decreased development costs —fast, automated and transparent collection of timing information
- Accelerated development and debugging —easy, any-time visibility into program execution
- Enhanced product quality and reliability —accurate, actionable performance-tuning information

- Zoom in or out around areas of interest in event data
- Filter events and "jump" specific system calls or other events
- Accurately measure times between different events

Target locally or remotely

The SpyKer front-end GUI is written in Java and runs on any workstation equipped with a Java Virtual Machine. Hence the front-end can reside locally or remotely from the system to be measured (which can be targeted over a TCP/IP connection). This remote capability, combined with SpyKer's non-intrusiveness, enables SpyKer's auto-instrumentation module to be deployed in target systems for later use by developers in search of bugs and/or performance data during "real-world" sessions.

Easy event capture

A pull down menu starts the capture process on the SpyKer front-end GUI. Users can:

- Remotely configure the data collection buffers on the host system (saved to disk, network drive or ring buffer)
- Define events to be traced and time-stamped
- Set up start and/or stop trigger points around specific events

Detailed level of understanding

Ordinarily, program operations are assessed indirectly by running functional tests. The resulting metrics are then used to tune the application. With SpyKer, developers gain a more detailed level of understanding by observing low-level systems operations. The resulting data can be used to improve both the application and the functional tests that are run against it.

All events are displayed as aspects of the processes that are running, and detailed information about the processes themselves is also displayed. The list of event types that can be traced by SpyKer is both extensive and diverse. It includes system calls, interrupts, context switches, processor exceptions and many other events.

In addition to tracing the aforementioned events, developers can use SpyKer to insert trace entry points into their application or device driver code and collect event data.

Whatever the target, once data is collected, the target system is returned to its original state. In addition, SpyKer allows the users to select the events of interest to trace and then only these user-selected events are auto-instrumented to further minimize the impact of trace collection on a target system.

Options for managing trace buffers

Developers can instruct SpyKer to save trace buffers to disk, to a network mounted drive, or to continually discard the oldest trace data and use the buffer itself as the sole storage mechanism. Developers can also save the buffers in non-volatile memory if it exists in the target system. Then they can capture valuable information on the last known state of a system should it crash.

1.800.255.5969

LynuxWorks, Inc.
855 Branham Lane East
San Jose, CA 95138-1018
408·979·3900
408·979·3920 fax
www.lynuxworks.com

LynuxWorks Europe
2 Allee de la Fresnerie
78330 Fontenay Le Fleury
France
+33 1 30 85 06 00
+33 1 30 85 06 06 fax