A Concurrent Real-Time White Paper

2881 Gateway Drive
Pompano Beach, FL 33069
(954) 974-1700
www.concurrent-rt.com

# Real-Time Performance During CUDA™

*A Demonstration and Analysis of RedHawk™ CUDA RT Optimizations*

By: Concurrent Real-Time
Linux® Development Team

## Overview

There are many challenges to creating a real-time Linux distribution that provides guaranteed low process-dispatch latencies and minimal process run-time jitter. Concurrent Real Time's RedHawk Linux distribution meets and exceeds these challenges, providing a *hard* real-time environment on many qualified hardware configurations, even in the presence of a heavy system load.

However, there are additional challenges faced when guaranteeing real-time performance of processes while CUDA applications are simultaneously running on the system. The proprietary CUDA driver supplied by NVIDIA® frequently makes demands upon kernel resources that can dramatically impact real-time performance.

This paper discusses a demonstration application developed by Concurrent to illustrate that RedHawk Linux kernel optimizations allow hard real-time performance guarantees to be preserved even while demanding CUDA applications are running. The test results will show how RedHawk performance compares to CentOS performance running the same application. The design and implementation details of the demonstration application are also discussed in this paper.

## Demonstration

This demonstration features two selectable real-time test modes:

1. Jitter Mode: measure and graph the run-time jitter of a real-time process
2. PDL Mode: measure and graph the process-dispatch latency of a real-time process

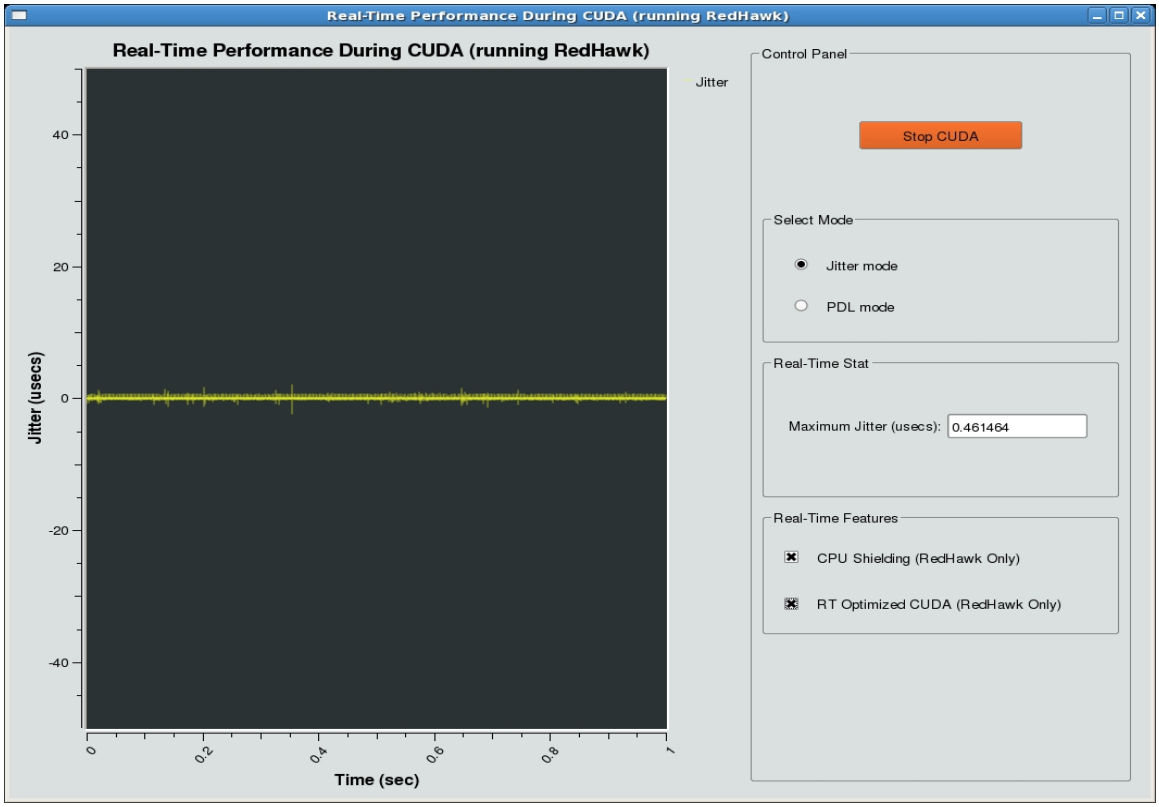
While the demonstration is running, it is possible to switch between these different modes at any time. Switching modes will reset the currently displayed graph and any statistics that are being tracked and displayed along with the graph.

On RedHawk Linux, it is also possible to enable and disable two RedHawk real-time features while the demonstration is running: CPU Shielding and RT CUDA Optimization. Changing the state of the individual real-time features will also reset the currently displayed graph and any statistics that are being tracked and displayed along with the graph. The graph and statistics are reset in order to clearly illustrate the effect that the specific feature change has made upon the real-time environment of the system running the demonstration.

Note that all of the screenshots in this paper were captured running the demonstration on the same hardware platform. The first two screenshots were captured with version 5.4.7 of the RedHawk Linux distribution installed, and the final screenshot was captured with version 5.4 of the CentOS Project's *Community Enterprise Operating System* Linux distribution installed — also known as CentOS Linux.

Both installed Linux distributions are using version 256.53 of the NVIDIA Graphics driver and version 3.1 of the NVIDIA CUDA SDK. The RedHawk Linux distribution pre-installs this software, while the CentOS Linux distribution requires this software to be manually installed.

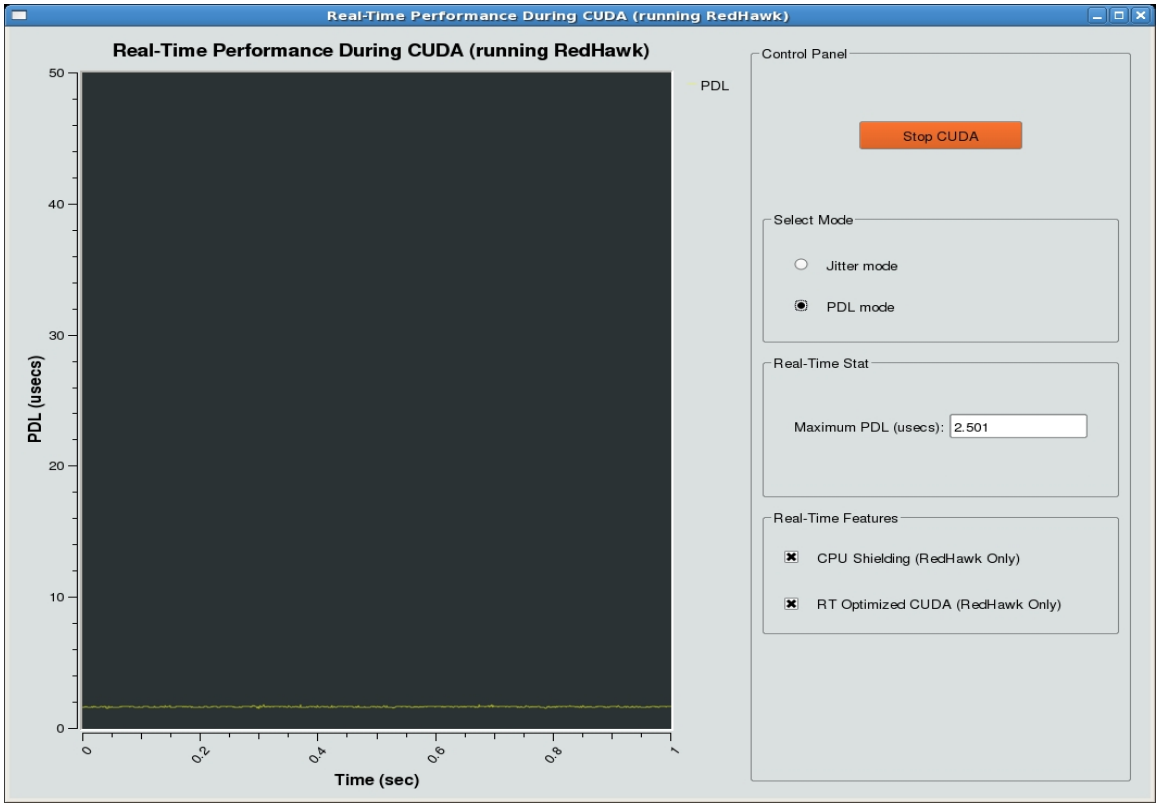
Below is a screenshot of the demonstration running in the Jitter Mode with RedHawk Linux installed:




You will notice several things in this screenshot:

- The CUDA load has been started.
- The real-time process is being continuously measured for jitter.
- The **CPU Shielding** feature of RedHawk Linux has been enabled.
- The **RT Optimized CUDA** feature of RedHawk Linux has been enabled to dramatically reduce the impact of CUDA operations on the system's real-time performance.
- Both the graph and the **Maximum Jitter** reported clearly demonstrate that RedHawk Linux is capable of running a real-time process very deterministically, even in the presence of CUDA.

Below is a screenshot of the demonstration running in the PDL Mode with RedHawk Linux installed:




You will notice several things in this screenshot:

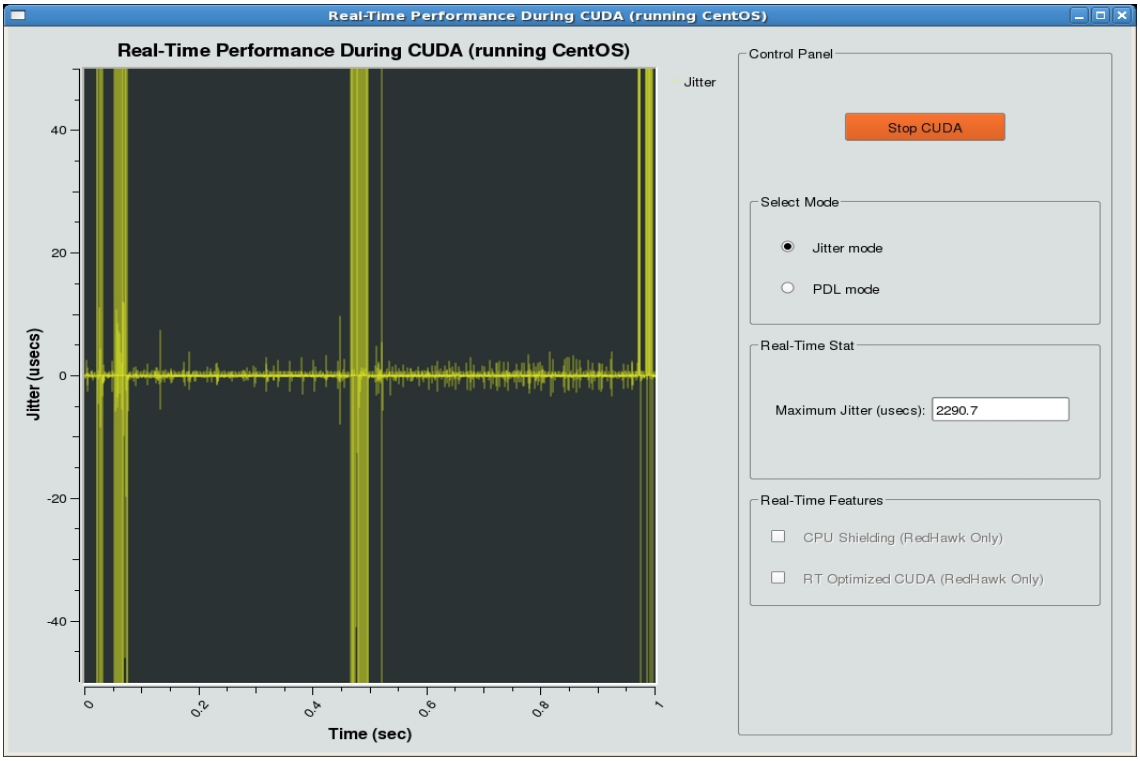
- The CUDA load has been started.
- The real-time process is being continuously measured for process-dispatch latency.
- The **CPU Shielding** feature of RedHawk Linux has been enabled.
- The **RT Optimized CUDA** feature of RedHawk Linux has been enabled to dramatically reduce the impact of CUDA operations on the system's real-time performance.
- Both the graph and the **Maximum PDL** reported clearly demonstrate that RedHawk Linux is capable of context-switching to a real-time process after an interrupt with very low latency.

When the demonstration is first run, the CUDA load is not started and none of the real-time optimization features are enabled. This allows the real-time performance of the system to be viewed and analyzed before and after the CUDA load is started.

Without the CUDA load running, CPU shielding alone is satisfactory to achieve excellent real-time performance. However, once the CUDA load is started, even CPU shielding cannot prevent the real-time process from being impacted by the large demands placed on the kernel by CUDA operations.

At this point, the **RT Optimized CUDA** feature can be enabled. Enabling this feature will protect the real-time process from the impact of the CUDA load on the system and restore the system's real-time performance.

For comparison, below is a screenshot of the Jitter Mode running with CentOS Linux installed:




You will notice several things in this screenshot:

- The CUDA load has been started.
- The **CPU Shielding** feature of RedHawk Linux is not available under CentOS Linux.
- The **RT Optimized CUDA** feature of RedHawk Linux is not available under CentOS Linux.
- Both the graph and the **Maximum Jitter** reported clearly demonstrate that CentOS Linux is *not* able to run a real-time process deterministically, especially in the presence of CUDA.

# Implementation

The above screenshots send a simple and clear message: the RedHawk kernel has been heavily optimized by Concurrent to provide an exceptional real-time environment for processes, even in the presence of a rigorous CUDA load. There are multiple processes in this demonstration that are running simultaneously to present this clear picture. The following paragraphs will provide details on these processes.

## NVIDIA Load

For both modes of this demonstration, the CUDA load is always the same: the reduction example CUDA application supplied in the freely available NVIDIA CUDA SDK is run in a loop to simulate the demands of an actual CUDA application on the system. This application performs a standard array reduction operation and is a good example of a generic CUDA program that takes full advantage of the CUDA high-

performance architecture. Array reductions are commonly used in statistics, simulation, automation, image processing and many other computationally intensive arenas.

This example application combines several standard CUDA operations:

- A large amount of GPU memory is allocated for the application to use.
- Several CUDA threads are running simultaneously on the GPU hardware and all are making concurrent use of large buffers of the allocated shared memory.
- Each GPU core runs its own thread group to reduce a different portion of the array in parallel.
- The thread groups need to continuously communicate partial results to each other.

Note again that this is a standard CUDA sample application and it has not been modified by Concurrent for this demonstration. It is also important to understand that the RedHawk Linux distribution does not change or enhance the proprietary CUDA SDK library code supplied by NVIDIA. Thus, the CUDA load of this demonstration is exactly the same regardless of which Linux distribution is used.

### Jitter Mode

In the Jitter Mode of this demonstration, the RTC is programmed to generate 2048 interrupts-per-second, and a real-time process blocks waiting for the RTC to generate each interrupt. Once an interrupt occurs, the real-time process unblocks, reads and saves the current TSC time value, and then blocks again waiting for the next interrupt. This behavior continues until the demonstration is stopped.

The difference between consecutive TSC time reads in an ideal world would be $1/2048^{th}$ of a second (approximately 500 microseconds). The difference between consecutive TSC reads, also known as the *TSC delta*, should be constant in an environment where the overhead is purely deterministic.

The graph in this demonstration mode displays the jitter between successive TSC deltas. The closer this graph is to zero, the more deterministic the system is behaving. This mode also tracks and displays the absolute value of the largest jitter measured since the demonstration started.

### PDL Mode

The PDL Mode of this demonstration uses a standard method for accurately approximating process-dispatch latency: a low-priority real-time process loops reading and saving the TSC time value, while a high-priority real-time process blocks for a microsecond waiting for a timer interrupt. Once the timer interrupt occurs, the high-priority process preempts the low-priority process, reads the current TSC time value, calculates the difference between the current and last saved TSC time value, and then blocks again allowing the low-priority process to resume. This behavior continues until the demonstration is stopped.

The difference between the current and last saved TSC time value closely approximates the interrupt process-dispatch latency of the high-priority real-time process, and these approximations are continuously graphed in this mode. This mode also tracks and displays the longest process-dispatch latency measured since the demonstration started.

## About Concurrent Real-Time

Concurrent Real-Time is one of the industry's foremost providers of high-performance real-time computer systems, solutions, and software for commercial and government markets, focusing on areas that include hardware-in-the-loop and man-in-the-loop simulation, data acquisition, industrial systems and software. Operating worldwide, Concurrent provides sales and support from offices throughout North America, Europe, Asia and Australia. For more information, please visit Concurrent Real-Time Linux Solutions at http://www.concurrent-rt.com or call (954) 974-1700.