



---

# *Real-Time Image Generation on Linux<sup>®</sup>*

*Ken Jackson, Vice-President CSS, Concurrent Computer Corporation*

*Chris Henderson, Senior Vice-President, MultiGen-Paradigm*

*Presented at the IMAGE 2006 Conference, Scottsdale, Arizona July 2006*

---

*This paper explores the real-time requirements of today's image generators and their associated rendering software running on a real-time Linux operating system. Synchronization methodology is scrutinized with a focus on performance goals and how it can be accomplished for large multi-channel Image Generators (IGs). Concerns on the performance of the rendering software are brought forward. The rendering software used to be a small percentage of the graphics display workload and the graphics drawing engine was always the performance throttle. Now with the availability of high-performance Scalable Link Interface (SLI) graphics boards and the release of quad-SLI graphics boards, one has to be concerned with the time associated with creating the OpenGL calls from the visual database. Synchronization of multiple drawing engines has been addressed by hardware genlock. The rendering software also needs a signal to start its process and many of the packages use vertical sync as the event to trigger their top of frame. Other hardware signals are discussed that provide a more deterministic trigger that provides a common signal for all associated channels. In addition, a real-time IG needs the tools to be able to determine the performance of its rendering software and the driver for the graphics boards. The paper investigates a trace tool that allows the user to identify numerous cross CPU interrupts, that are caused by the driver implementation for the graphics boards, and also how it can be used to identify frame overruns on the rendering software.*

---

## **1. Overview**

Image generation is in the midst of significant change. Just as the microprocessor changed the computer industry, the Graphics Processing Unit (GPU) has changed the image generation market with the advent of the PC-IG. The days of proprietary graphics engines are reaching their end and the days of inexpensive commercial off the shelf (COTS) systems are hitting their stride.

There have been many papers dedicated to how new hardware has changed the industry that stress the importance of COTS based systems over proprietary based systems. The next step is to analyze the COTS system itself to see what can be done to improve the performance of the image generator and provide the tools to monitor its performance. This will give the industry a methodology to ensure they are getting the most performance out of the selected hardware and not a situation where you can have mismatched CPU and GPU performance.

---

## ***2. Linux Operating System***

The Linux operating system provides many advantages as a platform for creating the next generation of image generators. Three of the most prominent requirements of an operating system for a real-time IG are that it is open software, there are real-time variations, and it is supported by powerful real-time development tools. Open software gives the integrator the freedom to look into the system performance of the IG and modify the constructs if required. While looking at the standard Linux distributions that are available, one notices that there are non-deterministic aspects that can impact the performance of the IG. Most of the standard Linux distributions don't have preemption enabled, use the big kernel lock for protection, and have a significant latency associated with spin lock hold times. If the frame synchronization event occurs during one of these lockouts, the start of frame can be delayed, sometimes for multiple milliseconds. Depending on the complexity of the scene that will be rendered, this delay of access to the kernel can be enough to cause a frame overrun which will cause the display to jitter.

In order to demonstrate the effect of non-deterministic performance we ran a program on two systems running Red Hat® Enterprise 4 and the other is running identical hardware configurations. One system is Concurrent's RedHawk™ release 4.1. Fig. 1 and 2 below show a comparison of two Linux distributions and the amount of variation between the runs of the test program. The program runs at 2kHz with a normal execution time of 500 microseconds and was run continuously for an hour. The RedHawk system shows a worst case jitter of 46 microseconds (Fig. 1) while the Red Hat system has a worst case jitter of over 39 milliseconds (Fig. 2). Now granted that IGs don't support a 2kHz frame rate, but it does show the amount of delay caused by the operating system and even at a 60Hz frame rate, 39 milliseconds will be noticed. The main difference between the two operating systems is the real-time extensions built into the kernel.org kernel. Most Linux distributions do

---

## 2. Linux Operating System

---

not enable kernel preemption. This means that the start of a frame could be delayed if a process is currently executing inside the kernel when the frame start time occurs.

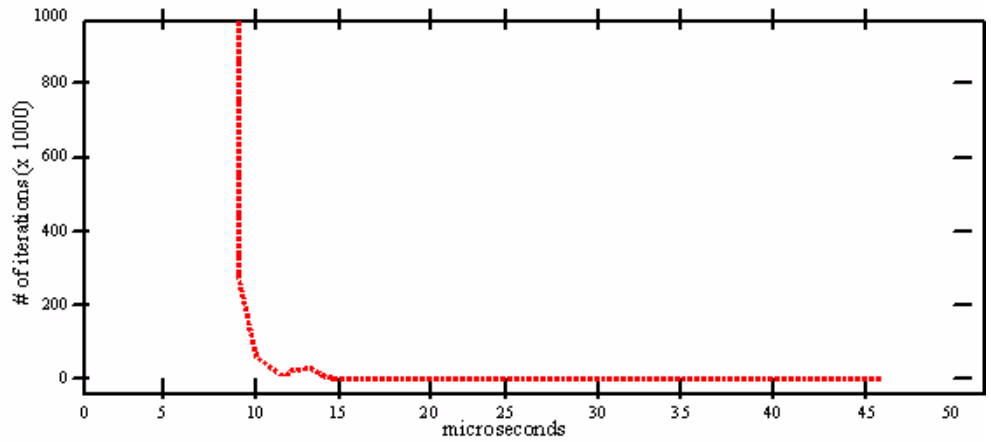


Fig. 1. Example of RedHawk jitter

---

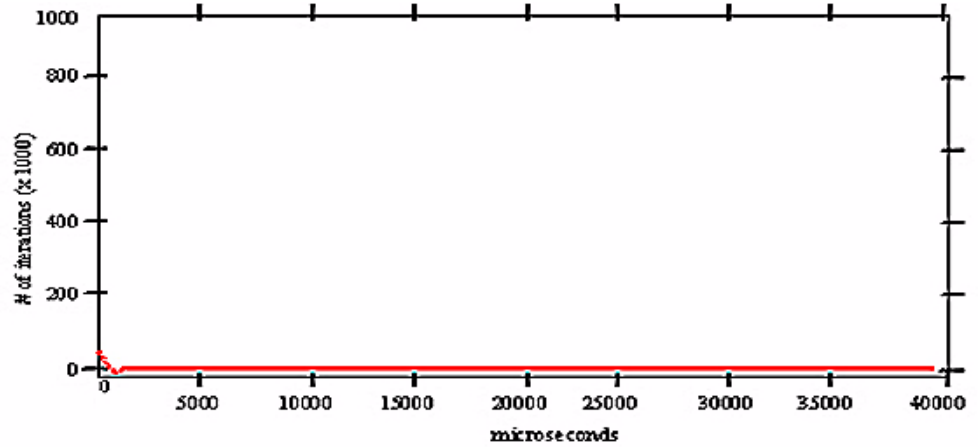


Fig. 2. Example of Red Hat jitter

---

Enabling kernel preemption significantly reduces that frame start delays of this type. However, there are still many critical sections inside the kernel where preemption can be blocked for extended periods of time. Some of the worst offenders in standard Linux code for delaying preemption occur in the disk subsystem in the code that synchronizes disk buffers to disk and in the management of file system journeying under file systems like ext3. These preemption lockouts can cause delays in frame start time that extend for as much as forty milliseconds. RedHawk not only enables kernel preemption by default, but it has also had a thorough analysis of all critical sections in the Linux kernel. Kernel algorithms in RedHawk have been modified such that preemption delays are minimized. This approach allows multiple processes to be executing in the kernel simultaneously. The only time a process may have to wait is if it is accessing the same data structure that another higher priority process is accessing.

This results in minimum variation in runtime, which gives the system designer a solid base to build his application. A system that can cause an overrun just by the overhead of the operating system can force the designer to overcompensate by using higher performance hardware to ensure that the frame time is met most of the time. The cost of this approach can be significant and it does limit the frames/second that the system can guarantee. The real-time performance of the selected operating system keeps the system running in a deterministic manner. This is then coupled with the driver for the graphics board and the rendering software to give the designer the overall system requirements to meet a specific frame rate for a database fidelity that will support a known number of polygons/frame.

Now that we have determined that we need a real-time foundation for our real-time image generator, let's focus on the application, the rendering software. The rendering software takes commands from the real-time host, converts the visual database into OpenGL commands and sends them to the driver for the graphics board for display. For this analysis we are going to use MultiGen-Paradigm's Vega Prime™ as our basis for discussion.

Vega Prime has three main threads of execution; the application thread, the cull thread, and the draw thread. The application thread interacts with the simulation host. Some of the functions may not be concerned with graphics, but are used to provide database services, such as height-above-terrain, database intersections, sensor simulation, etc. The cull thread removes scene graph elements that are not visible and the draw thread generates OpenGL commands to display the scene graph. Broadly speaking, during one IG frame,

each of the threads will be processing information received from the simulation host or generated by one of the other Vega Prime threads.

Vega Prime has several modes of scheduling and coordinating these independent threads. One scheduling mode has the draw thread generate OpenGL commands, and then issue an OpenGL swap buffer command. The draw thread's execution is then suspended until the pixels generated on a previous frame have appeared on the output display device and the electron beam of the display device has been moved to the top of the monitor. This occurs at the vertical Hz rate of the display, typically 60 Hz. This is known as vertical blank synchronization. The draw thread then continues and can tell the application thread (which had suspended itself previously) to continue to run and start processing new instructions from the simulator host.

The problem is that the IG's frame rate is now tied to the display device's vertical refresh rate. Often, one wants the IG to run at a different rate. A better synchronization methodology would be an external trigger with multiple timing sources. An external timing trigger that is separate from the vertical sync allows an IG to schedule its main frame loop independently of the Hz rate of the display device, while still scheduling at a predictable rate. An example of this external trigger is the Real-Time Control and Interrupt Module (RCIM) from Concurrent. The RCIM is a PCI board that supports multiple high performance timers and generates and accepts external interrupts as well as creates interrupts based on its timers. The RCIM can generate interrupts at very predictable intervals and deliver these interrupts to a user application running on RedHawk Linux with very low latencies. By having the IG's main execution suspend itself until the interrupts occur, highly deterministic frame scheduling can be achieved. The RCIM can also receive external interrupts and deliver them to user applications with low overhead. A simulation host could use this to control the IG's frame rate independently of the display device refresh rate. In addition, it has a built-in GPS option so visual systems in different locations can share the same time synchronization for the most realistic distributed simulation.

Based on the use of this type of signal a designer can tune the application to make the best use of the frame time available and now has the ability to make recommendations on the level of hardware needed to support a given frame rate. Specifically, instead of a single trigger generated from vertical sync, the IG can have multiple triggers to use as much time in the frame as possible. Most current designs try to leave enough head room on the processors to allow for worst case losses due to non-deterministic behavior of the system.

The final requirement is to have access to a set of development tools that can graphically display the timings of the rendering software. One such tool set that works with standard distributions from Red Hat and SUSE® are Concurrent's NightStar™ tools. Specifically, the NightTrace™ tool can track frame overruns and show which process is causing the overrun. Fig. 3 below shows several frames that are with a 60Hz frame rate. The first two are within the frame and the third represents an overrun. Using NightTrace, one can easily find what portion of the frame time is being used and by which process. This type of insight is invaluable to make sure you are getting the most from your real-time IG. Depending on the rendering load one can select the proper CPU to match the capability of the GPU.

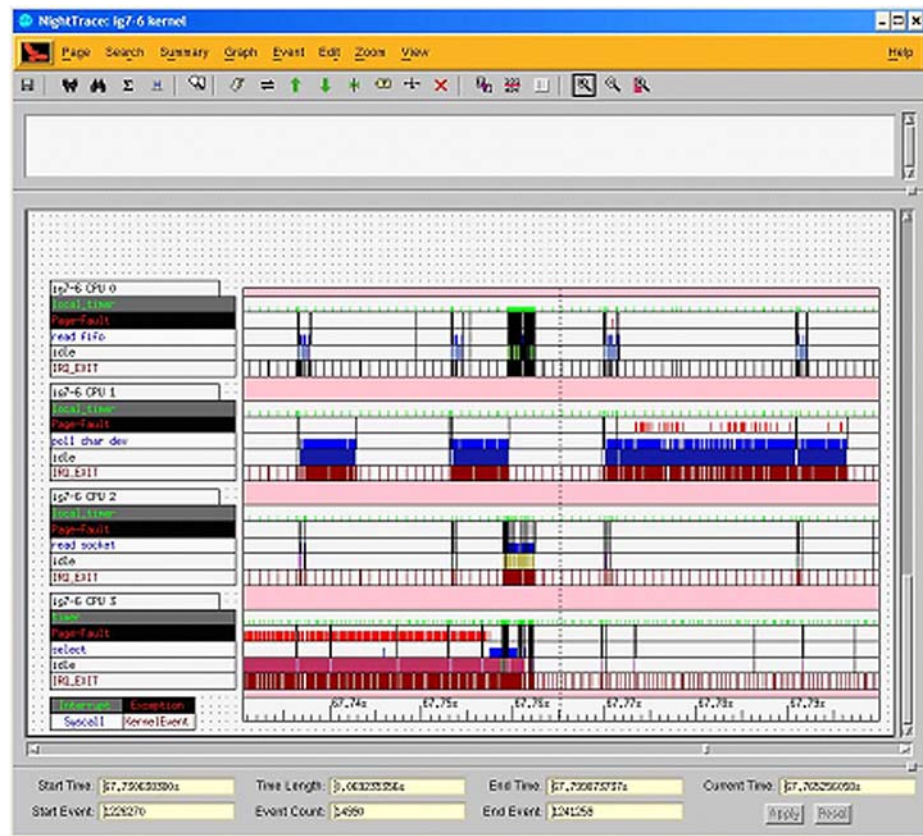


Fig. 3. Example of NightTrace analysis of Vega Prime

---

### 3. Conclusion

---

### ***3. Conclusion***

In summary, the Linux operating system is a good foundation to build image generators but a real-time Linux operating system is a great foundation. The determinism of the real-time operating system gives the image generator designer the most available performance to run the rendering software. The separate synchronization signal provides a high degree of flexibility for the IG designer to make the most of the frame time. The trace tools provide graphical representation of the frame time to show the IG designer the effects of his modifications. The combination of all these capabilities makes real-time image generation a reality.

---

### ***4. Authors' Biographies***

Chris Henderson is Senior Vice President of Engineering for MultiGen-Paradigm. MultiGen-Paradigm is the leading supplier of real-time 3D solutions for visualization, simulation, and training applications.

Mr. Henderson has over 15 years experience in real-time simulation. In his role at MultiGen-Paradigm, he oversees the development of all products and services, including the development of Creator, Creator Terrain Studio, Vega Prime and its modules and FlightIG.

Mr. Henderson holds a Master of Science in Computer Science from the University of Louisiana at Lafayette; Minor in Mathematics.

---

Kenrick R. Jackson is Vice President, Concurrent Special Systems. Concurrent is a leader in providing on-demand systems to the broadband industry and real-time computer systems for industry and government.

Mr. Jackson has more than 25 years of diverse engineering and management experience in the computer industry. In his role as Vice President of Concurrent Special Systems, Mr. Jackson oversees the worldwide operations of this dynamic company, including industry solutions, integration and professional services, custom engineering, and custom products.

Mr. Jackson received a Bachelor of Science in Mechanical Engineering, a Bachelor of Science in Electrical Engineering, and a Masters of Computer Engineering from Florida Atlantic University.

---

***Copyright***

Concurrent Computer Corporation and its logo are registered trademarks of Concurrent Computer Corporation. All other Concurrent product names are trademarks of Concurrent while all other product names are trademarks or registered trademarks of their respective owners. Linux® is used pursuant to a sublicense from the Linux Mark Institute.

© 2006 Concurrent Computer Corporation

RTLit-0042 0706