ABSTRACT:

Communications networks are very different from other kinds of computing applications. Networks require both very high reliability and very high performance. Not only must they process large volumes of data at high speed, but they must do so while limiting their downtime to minutes per year. Network failures can have huge economic consequences and can even result in the loss of life. Additionally, in this post-9/11 world, failure also has a national security dimension.

This technical whitepaper introduces MontaVista Linux Carrier Grade Edition—what it is, how it evolved, and how it provides a foundation for developing and deploying carrier-grade applications like those described above.
# Table of Contents

Introduction ................................................................................................................................................................... 5

   Intended Audience .................................................................................................................................................... 5

   Organization .............................................................................................................................................................. 6

Overview and Background ............................................................................................................................................ 6

   Carrier Grade Requirements ...................................................................................................................................... 7

       Linux Foundation Carrier Grade Linux Workgroup ................................................................................................. 7

       Service Availability Forum ................................................................................................................................... 8

Architecture and Features .......................................................................................................................................... 8

Standards Compliance ............................................................................................................................................. 10

Open Source Advantages ........................................................................................................................................ 10

Supported Hardware Architectures .......................................................................................................................... 11

CGE Features in Detail ................................................................................................................................................ 11

   Serviceability ........................................................................................................................................................... 11

       Microstate Accounting ........................................................................................................................................ 12

       LVM ................................................................................................................................................................. 12

       Kernel Crash Dumps ......................................................................................................................................... 13

       Flight Recorder ................................................................................................................................................ 13

       Live Application Coredump ............................................................................................................................... 13

       Field-Safe Application Debugger .................................................................................................................... 13

       Runtime Application Patcher ................................................................................................................................... 14

       Forced Unmount ............................................................................................................................................... 14

       Block Device Removal ................................................................................................................................... 14

   Performance ............................................................................................................................................................ 15
<table>
<thead>
<tr>
<th>Feature</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-Time Kernel and Scheduling</td>
<td>15</td>
</tr>
<tr>
<td>Timers</td>
<td>15</td>
</tr>
<tr>
<td>Fast Application Loading</td>
<td>16</td>
</tr>
<tr>
<td>Fast Boot Times</td>
<td>16</td>
</tr>
<tr>
<td>Latency Measurement Tools</td>
<td>16</td>
</tr>
<tr>
<td>Symmetric Multiprocessing (SMP)</td>
<td>17</td>
</tr>
<tr>
<td>Clustering</td>
<td>17</td>
</tr>
<tr>
<td>oprofile</td>
<td>17</td>
</tr>
<tr>
<td>Redundancy</td>
<td>17</td>
</tr>
<tr>
<td>RAID</td>
<td>18</td>
</tr>
<tr>
<td>Redundant Ethernet</td>
<td>18</td>
</tr>
<tr>
<td>Clustering</td>
<td>18</td>
</tr>
<tr>
<td>Network Replicated Storage</td>
<td>19</td>
</tr>
<tr>
<td>Networking</td>
<td>19</td>
</tr>
<tr>
<td>Virtual Local Area Networks</td>
<td>19</td>
</tr>
<tr>
<td>Virtual Routing and Forwarding (VRF)</td>
<td>19</td>
</tr>
<tr>
<td>Intermediate Queuing (IMQ)</td>
<td>20</td>
</tr>
<tr>
<td>IPSec</td>
<td>20</td>
</tr>
<tr>
<td>Reliability</td>
<td>20</td>
</tr>
<tr>
<td>Fast, Journaling file systems</td>
<td>20</td>
</tr>
<tr>
<td>Resource Monitoring</td>
<td>21</td>
</tr>
<tr>
<td>Memory Error Detection and Correction (EDAC)</td>
<td>21</td>
</tr>
<tr>
<td>openhpi</td>
<td>21</td>
</tr>
<tr>
<td>OpenIPMI</td>
<td>21</td>
</tr>
<tr>
<td>Code Reviews and QA</td>
<td>22</td>
</tr>
</tbody>
</table>
Introduction

Communications networks are very different from other kinds of computing applications. Networks require both very high reliability and very high performance. Not only must they process large volumes of data at high speed, but they must do so while limiting their downtime to minutes per year. Network failures can have huge economic consequences and can even result in the loss of life. Additionally, in this post-9/11 world, failure also has a national security dimension.

One example is enough to illustrate the potential impact network failures can have. In 1991, a packet-switched network failed. This failure caused hundreds of millions of dollars in financial losses, knocked out air traffic control in the New York City region for over 8 hours, and disrupted 85,000 travelers.1

Building communication networks that are reliable and perform well is economically and technically challenging: the physics, the finances, and the computing are often state-of-the-art.

As one would expect, communication networks comprise both hardware and software. The proportion of network functionality being implemented in software continually increasing.

Typical networking applications are gateways, bridges, routers, signaling servers, and management servers. Since these kinds of applications are sophisticated, complex and very expensive to design, it makes economic sense to provide a common set of software services that these applications can use as a foundation. In other words, they need an operating system that matches their sophisticated and complex needs.

MontaVista Linux Carrier Grade Edition is such an operating system.

This technical whitepaper introduces MontaVista Linux Carrier Grade Edition—what it is, how it evolved, and how it provides a foundation for developing and deploying carrier-grade applications like those described above.

Intended Audience

The intended audience for this white paper includes:

Technical managers who want to know more about the capabilities of carrier-grade operating systems so they can make better managerial decisions

Software developers who want an overview of the components of a carrier-grade operating system and their capabilities so they can decide what they must build and what is already available

System engineers who need to understand what reliability and fault-tolerance features are available to them.

You should have a basic understanding of the architecture of UNIX or Linux, but detailed knowledge of the internals of the Linux operating system are not necessary to benefit from reading this whitepaper.

1 Economic Impacts of Infrastructure Failures, Report to the President’s Commission on Critical Infrastructure Protection, Booz, Allen & Hamilton, 1997
Organization

This whitepaper is organized into sections that convey a complete picture of MontaVista Linux Carrier Grade Edition (CGE).

The Overview and Background section introduces CGE from a historical and scope perspective, and indicates what requirements drive the product’s capability and what markets it addresses.

The Architecture and Features section describes CGE’s key components and features, as well as how it relates to Linux software available in the open source community.

The Standards Compliance section describes the relevant industry standards bodies and consortiums, and why these entities’ efforts are crucial to the industry.

The CGE Features in Detail section provides a comprehensive view of the capabilities and utility for these capabilities in the product.

The Application and Kernel Development section describes the software development productivity tools provided by the MontaVista DevRocket IDE and how these tools enhance time-to-market.

The CGE Ecosystem section describes the hardware and software alliances among CGE users and vendors that aim to provide a complete solution for communications network equipment providers.

The Example Deployment section shows how one company is using CGE and related technology to deliver differentiated products to their customers.

The final section is a glossary of CGE related terms.

Overview and Background

Carrier Grade Linux had its origins in 2001 when the Open Systems Development Labs (OSDL), now part of the Linux Foundation, formed the Carrier Grade Linux (CGL) Working Group to advance Linux in the communications infrastructure space.

The CGL Working Group started as a small, core group of communications network equipment providers (NEPs), major hardware platform providers, semiconductor manufacturers, and Linux distributors working to define, collect and prioritize industry requirements that accomplish the enhancement of standard Linux for the demanding carrier environment. The idea was to have the key stakeholders who wanted to see Linux succeed in the communications network infrastructure market, work together to drive innovation and adoption into the network.

Now membership has expanded to include most of the leading NEPs, more Linux distributors and hardware equipment suppliers and even a major service provider who has come to see CGL as a critical building block for bringing reliable new services to market. As of late 2009, the CGL Working Group published version 4.0 of the CGL specification.

MontaVista Linux Carrier Grade Edition (CGE) is a complete, standard, COTS (commercial off-the-shelf) Linux distribution that has been extensively enhanced to provide the reliability, availability, and serviceability required by carrier grade applications. CGE is derived from the standard set of multiple open source software repositories; MontaVista then adds CGL enhancements from a variety of sources. Most of the CGL specification requirements point to one or more open source projects. Others are still being developed and introduced to the open source community over time, ultimately with the goal of inclusion into mainstream distributions such as the standard kernel
from kernel.org, various open-source projects, and other distributions. MontaVista compliments the distribution with custom developed features, leading edge development tools, technical support, and professional services.

Telecommunications and data communications applications use CGE in communications network infrastructure solutions. CGE is targeted for edge and core telecommunications including applications for the converging IP and voice networks such as soft switches, SGSN (Serving GPRS Support Node) and GGSN (Gateway GPRS Support Node), RNC (Radio Network Controller), Voice over IP (VoIP) gateways, and many other applications. CGE provides protocols LTE and 4G specifications require. It is ideal for deployment in any carrier-grade environment.

**Carrier Grade Requirements**

Communication network equipment providers have some very specific and unique requirements. These requirements fall into three major categories:

- Reliability
- Availability
- Serviceability

Vendors have historically used these as the primary terms used to define the requirements for communication network infrastructure.

**Linux Foundation Carrier Grade Linux Workgroup**

The Linux Foundation CGL Workgroup has broadened and added depth to these categories, specifying the following categories in the Carrier Grade Linux Requirements Documents.²

- Availability
- Clustering
- Serviceability
- Performance
- Standards
- Hardware
- Security
- Serviceability

The specification further classifies each requirement by priority. Carrier-grade systems claiming compliance against the specification must implement priority 1 requirements. Priority 2 requirements are desirable, and priority 3 requirements are currently considered not important, but may become so in the future.

Many of these requirements are general system requirements that have to do with the Linux kernel, the core libraries, and tools, and are commonly found in the popular Linux distributions. Many of the requirements apply primarily to the carrier environment, but many also apply to general purpose systems.

²[http://www.linuxfoundation.org/collaborate/workgroups/cgl](http://www.linuxfoundation.org/collaborate/workgroups/cgl)
Service Availability Forum

The Service Availability Forum (SAForum) develops standards for middleware in highly available systems. Several CGL requirements reference SAForum specifications for application availability, hardware availability, and hardware management. Details can be found at the SAForum website.³

The SAForum has two main sets of specifications. The Hardware Platform Interface (HPI) defines a rich set of APIs for discovering, monitoring and controlling the hardware in a carrier grade system. Developers may use this to find out what hardware is in a system, detect hardware errors such as temperature problems, fan failures and power supply problems, and control hardware power and hot swap.

The SAForum also produces the Application Interface Specification (AIS). This defines a set of APIs for clustering software. Developers may use AIS for application configuration, monitoring, failover, inter-process communication, locking, and a host of other services. Services provided by AIS include:

- Availability Management Framework
- Checkpoint Service
- Cluster Membership Service
- Event Service
- Information Management Model
- Lock Service
- Log Service
- Message Service
- Naming Service
- Notification Service
- Security Service
- Timer Service
- Software Management Framework
- Platform Management Service

These services work together to provide a rich set of clustering APIs that are tightly integrated. In addition, the Platform Management Service integrates HPI into the AIS model so that hardware states can be more easily integrated into the system-wide management done by AIS.

Architecture and Features

The architecture of MontaVista Linux Carrier Grade Edition (CGE) is based on a combination of the MontaVista Linux kernel, open source projects championed by MontaVista and others, and leading edge technology extensions from MontaVista.

CGE is based on the 2.6 Linux kernel, with a number of additional features provided over and above those in vanilla Linux.

³ [http://www.saforum.org](http://www.saforum.org)
These extra, high value features include:

**Serviceability Features** – MontaVista Field Safe Application Debugger, Runtime Application Patcher, kernel crash dumps, flight recorder, live application coredump, microstate accounting, resource monitoring

**Performance Features** – real-time kernel with breakable locks for improved latency, interrupt and preemption latency measuring tools, and application loading and locking

**Redundancy Features** – Ethernet bonding, application heartbeating and failover, multi-hosted RAID, forced unmount, block device removal, and DRBD

**Networking Features** – VRF, IMQ

**Security Features** – IPSec, SELinux

**High Availability Hardware** Support – IPMI and SAForum HPI, with support for ATCA, including hot swap management

**Standards Compatibility** – PICMG xTCA, Linux Foundation CGL 4.0, LSB 3.0, IPv6 (including mobile IPv6), SA Forum, ANSI and POSIX

**Development Tools** – gcc toolchain, gdb, kdb and kgdb kernel and driver debuggers, and a complete, Eclipse-based integrated development environment (IDE), as well as memory leak checkers, profilers, the Linux Trace Toolkit, and more.

In addition to these features, MontaVista adds board and architecture support for existing kernel features and backports significant feature functions from future kernels. These features include high resolution timers (HRT), kexec, kdump, EDAC, full NAPI support, full VLAN support, NBD, TIPC, and OCFS2.

All work done by MontaVista goes through at least a peer review process. Work destined immediately for mainstream also gets reviewed in open source as part of the mainstream process.
Standards Compliance

Historically, communications and data service network providers built solutions on proprietary platforms (both hardware and software) to meet availability, reliability, performance, and response-time requirements. These platforms were closed systems, expensive to develop and maintain, and lacked support for third-party software and emerging standards.

Today, the industry is turning away from proprietary platforms towards open, standards-based building blocks for hardware, operating systems, and high availability middleware. This approach offers a number of advantages: cost effectiveness, rapid support for new architectures, new services, and increased bandwidth, while supporting the requirements for high availability, scalability, security, reliability, and maintainability.

One of the unique things about the carrier grade market is that standards bodies and industry consortia drive carrier grade requirements. MontaVista CGE meets the following standards:

**LSB, POSIX, ANSI C** – MontaVista CGE, like standard Linux, complies with the Linux Standards Base 3.0 [corey – yes, I missed this. 4.0 is current, but we are 3.0 compliant][dean – isn't LSB 3.0 now?] and supports the POSIX and ANSI C APIs

**Linux Foundation Carrier Grade Linux Spec** – Full compliance with the Linux Foundation 4.0 CGL Specification with all of the priority 1 (must-have) items in the CGL 4.0 specification and many of the other priority 2 and lower features.

**IPv6 and Mobile IPv6** – MontaVista CGE has been certified compliant to IPv6; it includes the USAGI IPv6 and Mobile IPv6 patches and significant backports from future kernels for support of various features.

**Service Availability Forum AIS** – MontaVista Software became the first commercial software vendor to support the Application Interface Specification (AIS) v1.0 through the openais project. MontaVista still provides openais and also provides a more complete package of features with the OpenSAF package. See http://www.openais.org/doku.php and http://www.opensaf.org.

**Service Availability Forum HPI** – MontaVista packagesopenhpi for supporting a standard API for maintaining hardware. openhpi has support for many platforms, including ATCA. See http://www.openhpi.org/

**Advanced TCA** – CGE has been qualified and integrated on many Advanced TCA system platforms with more vendors being added all the time. Advanced Telecommunications Computing Architecture (ATCA) is the standard architecture based on the next-generation, highly advanced telecommunications specifications developed by the PCI Industrial Computer Manufacturers Group (PICMG).

**OpenIPMI** – MontaVista CGE architects maintain this important open source project for a common implementation of the IPMI standard hardware management interfaces.

Open Source Advantages

Open source solutions have several advantages over closed, proprietary solutions. First, customers are not at the mercy of a single vendor. They are free to walk away if their vendor changes terms or stops supporting their product. Customers have a variety of alternatives for obtaining support.

Second, the source code may be used as a learning tool, as an insurance policy, or to extend and/or modify the application.
Third, open source helps customers develop faster—and thus get to market faster—because they have a bigger toolbox to work with. There is less need to re-invent the wheel because one can build on the innovative work of members of the open source community.

Fourth, open source software is often better software. Bugs are fixed, performance bottlenecks are eliminated, and new techniques or algorithms supplant existing ones, all with rapidity unknown in the proprietary world.

Last, open source software has a lower total cost of ownership because of the simple fact that improvements can be maintained within the open source community.

MontaVista Software has been a consistent proponent and contributor to open source software including the O(1) scheduler (now mainstream in the 2.6 kernel), preemptible kernel (also in 2.6 kernel), high resolution timers, “openipmi” hardware management, “openais” clustering support, forced unmount support for file systems, additional real time enhancements to Linux, board support for many boards, and many, many bug fixes. In a recent Linux Foundation report that detailed the top contributors to the Linux kernel, MontaVista ranked in the top 15 corporate contributors and was the only embedded Linux company in the top 30.

**Supported Hardware Architectures**

MontaVista CGE supports many different architectures: the Intel IA-32 and IA-64, PowerPC architectures from IBM and Freescale, and MIPS architectures from Cavium and RMI. In conjunction with these leading semiconductor manufacturers and the dozens of board and systems manufacturers within their ecosystem, a robust list of popular platforms is constantly being expanded, including new processor architectures.

**CGE Features in Detail**

The next sections describe the specific features that make MontaVista Linux CGE the industry’s most widely deployed Linux for carrier-grade applications—not just another Linux distribution.

Note that some features apply to more than one section. For instance, RAID provides redundant hardware support and can increase performance. The features generally only appear in one section, and references are made to other sections they may support.

**Serviceability**

Serviceability is the ability to maintain a running system, a key requirement of high availability systems. This includes the ability to fix hardware and software problems, upgrade to add new capacity and services, and diagnose

---

5 To see a list of all boards supported by MontaVista Linux CGE, visit our boards page at [http://www.mvista.com/boards.php?edition=CGE&v=All&a0=1](http://www.mvista.com/boards.php?edition=CGE&v=All&a0=1)
problems in the field while running and providing active service. A lack of such ability can mean loss of redundancy and exposure to single points of failure during maintenance, with a resulting loss of availability.

MontaVista CGE provides a number of maintainability features to meet this field maintenance challenge.

**Microstate Accounting**

The standard Linux kernel provides statistical measurement of CPU usage by an application. When a timer tick goes off, the kernel accounts the entire kernel tick's worth of time to the application that is running when the tick occurs. This is an efficient way for gross level measurement of CPU usage, but it is not accurate enough to use as a measure of total CPU usage or for short-term CPU measurements.

If a system is using statistical measurement of CPU based upon time ticks to know when it is in overload, it is possible to mis-measure the CPU used by a large amount. Timer-driven applications, in particular, tend to run right after the timer tick and finish before the next timer tick. Thus, the standard statistical measurement would not know the system was using any significant amount of CPU until the system was near 100% utilization.

In many systems, the application can measure how long a particular threads runs. For instance, the application designer can know that a particular event that occurs should not take any longer than 10ms of CPU time to handle. An itimer based upon CPU usage can be started to measure this and provide a signal if the event takes too long. That way, if a thread goes into an infinite loop or consumes too much CPU resources, the application can quickly recover from the problem. But with statistical measurements, a thread can often be assigned far more CPU than it actually has used, resulting in false signals.

Microstate accounting solves these problems by tying into all the task switch locations and providing accurate measurements of how much CPU a task uses. This is an open-source project[^4], but MontaVista CGE provides significant enhancements by tying the microstate accounting into the standard kernel accounting and itimers.

Microstate accounting also provides access to where a particular application spends its time, waiting on paging, waiting on locks, etc. This can be useful to profile an application and solve performance problems.

**LVM**

The Logical Volume Manager (LVM) and RAID cooperate to give MontaVista CGE highly available storage, and high performance storage.

The Logical Volume Manager (LVM) allows the size of logical volumes to be changed on a running system without service interruption. New storage can be brought online and added to an LVM file system while the file system is active. LVM file systems can also be shrunk and storage removed from volume groups, which is useful as a prelude to hot swapping out and replacing a failing disk drive. LVM also allows snapshots of active running file systems to be taken for backup purposes.

RAID (level 0), otherwise known as striping, provides significantly optimized access to data on the disk.

MontaVista supports the latest versions of LVM and RAID level 0. For the high availability aspects of RAID, see the RAID section.

Kernel Crash Dumps

CGE provides two basic mechanisms for taking kernel coredump: LKCD and kdump.

LKCD provides customers with the ability to save a core dump of the kernel when the kernel panics. This allows kernel errors to be debugged after the fact; the system can store the core dump, immediately restart, and continue operation. This improves availability while still providing the serviceability of debugging kernel errors.

LKCD allows core dumps to be stored in one of two places: a remote network server or a local disk. With a remote network server, a server on the same subnet can receive the core dump from a target over an ethernet network. The target dumps the information to that server. With disk, the core dump is stored on a raw disk partition (often a swap partition) and the core dump is recovered from the disk partition at boot time.

The kdump feature causes a kernel to jump to a new kernel in a reserved memory area when the system panics. So a new kernel is running and the old kernel's memory is still intact. A program running in the new kernel can extract the memory for the old kernel and store it any way it likes.

No matter what core dump mechanism or destination is used, the image can be debugged with a core dump analysis program that allows the user to examine kernel data structures, running tasks, and other useful operations. MontaVista Software provides the host tool “crash” for such analysis.

Flight Recorder

When a kernel crashes, the crash dump may not provide enough information to resolve the problem. When debugging difficult problems, a trace of what the kernel was doing before the crash can be very helpful.

Flight recorder combines Linux Trace Toolkit with kernel crash dumps. A circular buffer holding an LTT trace is kept running in the kernel. This buffer can be extracted from a kernel coredump and then analyzed with the standard LTT tools. See the section on LTT for more details.

Live Application Coredump

Sometimes an application fails, but not in a way that keeps it from providing service. For instance, an application may be providing an incorrect result periodically. A coredump of the application could be useful to view the data and help diagnose a problem, but that would normally require killing the application.

Live application coredump provides a way to take a coredump of an application without stopping it for a significant amount of time. The application is stopped just long enough to clone the process and mark the pages copy on write, the application is then allowed to run. This time depends on the amount of memory, but is generally on the order of milliseconds. The coredump is then taken from the cloned process. If the application writes to a page, that page is copied on demand so the coredump is accurate.

Live application coredump can also override the coredump limit and scheduling settings for an application, so you can take a application coredump even if the limit was set to zero, and you can lower the scheduling priority for the coredump to reduce the affect on the running application.

This can result in some memory pressure on the system, so this is best done in off-peak hours.

Field-Safe Application Debugger

Standard Linux debuggers (such as gdb) are not suitable for examining running applications on deployed systems. Such debuggers typically take over signal handling, require other (foreground and background) execution threads in
multi-threaded programs to be suspended, and halt the application completely when they encounter a breakpoint. Even when just tracing execution, these debuggers are intrusive, slowing down the application causing degradation of service.

The MontaVista field-safe application debugger (FSAD) overcomes these problems, allowing system maintainers to debug applications running in the field non-intrusively. The maintainer sets tracepoints in the code to gather data during system operation. A tracepoint may activate a user-supplied routine to provide additional information desired, but neither the application nor any of its threads are suspended. This architecture allows the application to run unhindered, with minimal intrusion on the operation of the running system. FSAD has many safety checks to keep an errant user operation from causing problems in the application; this is to keep it safe from accidental errors during debugging.

The FSAD can also debug multi-threaded applications, and can even debug both parent and child execution after the parent calls fork().

**Runtime Application Patcher**

Applications in deployed systems must be available even during upgrades. Normal upgrade procedures migrate the application and/or the services it provides to a backup system, stop the application, upgrade it, start the upgraded application, and migrate the services back. This process exposes the system to single points of failure and service degradation during the upgrade process, which sometimes may take hours.

The MontaVista run-time application patcher (RAP) provides a facility to patch or upgrade applications in place, while they are running and providing services, allowing for a minimum of service interruption. RAP performs this patching procedure by providing a control process which loads the patch into memory, suspends the application at a safe point (where the flow of execution is not in a subroutine that is being replaced), patches the routines being replaced with the new ones that are loaded into memory, and resumes application execution. The switchover procedure occurs in a matter of milliseconds, providing continuous availability even during system upgrades.

An additional benefit is that patches are removable, allowing applications to be reverted to previous states in cases where field operation uncovers problems not evident in laboratory testing.

**Forced Unmount**

Highly available systems must sometimes “unmount” file systems to perform system maintenance. Standard Linux does not allow the unmounting of a file system if there are files or directories in the file system that are currently in use. Moreover, in the case of NFS, if the host serving the file system is no longer available, the timeouts to unmount the file system can run into minutes even if there are no open files.

In order to allow the system maintainer more flexibility, especially when an orderly shutdown is not possible, CGE allows the root user to force the unmounting of file systems (including NFS). Programs that reference an open file descriptor or traverse a file path on an unmounted file system receive an error (ENXIO). When unmounting a file system, the forced unmount algorithm first recursively closes all files open on that file system, driving the open count on all inodes to 0. Subsequent calls to close() for file descriptors referencing the unmounted file system complete without errors.

**Block Device Removal**

Highly available systems must also sometimes remove block devices (normally disks) to do system maintenance and repair.

CGE allows the root user to force the removal of SCSI and FibreChannel block devices. Programs that have a removed block device open will receive an error (EBADF) the next time they reference that device. The algorithm
for block device removal calls the forced unmount feature mentioned above if there are any file systems open on the device.

If the device to be removed is a RAID member, the RAID member will be hot swap removed from the RAID device. This results in better hot swap performance as compared to simply removing the device, since it needs no RAID error recovery phase (with potential SCSI/IDE I/O recovery as well).

**Performance**

Performance is a key attribute of carrier grade systems and as such, MontaVista has added new features or modified existing features to enhance standard Linux and make it suitable for carrier grade applications. This includes two different types of performance: throughput and real-time.

**Real-Time Kernel and Scheduling**

In addition to the standard O(1) scheduler that comes with Linux (note that MontaVista helped author the O(1) scheduler) and the desktop level of preemption that comes with Linux, MontaVista CGE has a real-time, fully preemptive kernel that comes from the Linux real-time project. This provides additional latency guarantees that cannot be met by the normal kernel.

Note that though a real-time kernel helps a system meet tight latency guarantees, just having a real-time kernel does not provide magic dust that makes a system real-time. Meeting real-time requirements requires engineering the entire system, including proper thread and task design, proper assignment of priorities to all tasks, threads, and interrupts, and careful analysis of flows of execution to make sure that the portions of the system that have tight latency requirements can meet their schedules. But without a real-time kernel, meeting hard real-time requirements is impossible, so a real-time kernel provides the base upon which a real-time system can be built.

Also note that real-time kernels tend to run slower than non-real-time kernels. The methods required to meet real-time deadlines are a tradeoff between throughput performance and latency performance. In general, the higher the throughput the higher the worst-case latencies. In general, the desktop level of preemption provides the best compromise between throughput and latency performance for most carrier applications. Applications that care only about throughput without regard to latency should use a non-preemptible kernel.

**Timers**

Linux provides timers with POSIX semantics and microsecond resolution. Such higher resolution timers provide better control of system events and devices, and improve system efficiency by eliminating busy-wait loops and allowing for finer grained timeouts.

Time in standard Linux is kept in terms of jiffies. A jiffy is the period of the system clock, usually 10ms, although it can be as small as 1ms on some architectures and is configurable on others. For time resolution of less than 2 jiffies, an alternative to standard Linux timers is required. MontaVista provides user-accessible timers with POSIX 1003.25 semantics. The standard clocks (CLOCK_MONOTONIC and CLOCK_REALTIME) provided by every POSIX compliant system have been augmented in CGE by two additional clocks (CLOCK_MONOTONIC_HR and CLOCK_REALTIME_HR) with microsecond resolution, as allowed for in the standard. These two time bases provide the user with access to the system high resolution timers.

MontaVista CGE provides enhancements to systems that do not properly implement high-resolution timers in the mainstream kernel, and tests that high-resolution timers are working properly on all supported systems.
Fast Application Loading

Real-time systems often require not only rapid response to events, but dependable (low variation) latencies, or minimal jitter. One of the characteristics of demand paged systems such as Linux is that the TEXT (code) pages of the application are brought into memory one at a time from a disk or other non-volatile image when they are first accessed. Thus, a running program with its working set of pages already resident in memory will run most efficiently, with a minimum of page faults and associated latencies.

But, during application startup, many pages are accessed in quick succession because they need to be brought into RAM one at a time from disk. This causes unpredictable delays when starting an application for the first time.

CGE solves this startup performance problem by allowing the user to specify 100% loading of an executable at startup, which brings the entire executable into memory at once. Subsequently, the (sometimes large) executable can take advantage of paging (swapping) to disk if memory pressure requires more pages.

Developers can also specify locking the entire executable into memory upon load, allowing for maximum determinacy during execution.

MontaVista CGE provides two different methods to provide this locking. The first comes from the stock Linux kernel using mlockall() to lock all an applications memory. This has some notable disadvantages because it requires root access and it locks all stack memory, which can consume a large amount of memory that is mostly unused.

MontaVista also authored a modification to glibc to either mlock() memory as it is loaded, or to read memory into buffers instead of mapping it. Using mlock() requires root access, but does not lock the stacks into memory. The user can size their stacks and lock them as necessary. The read loading does not require root access, but the memory loaded can be paged out. It is still useful in a system without a paging disk, as the memory will be locked without the program having to have root access.

Fast Boot Times

It is often desirable, in a high availability environment, to reboot quickly to reduce downtime. It may be necessary to reboot due to a system failure or “on demand” because of some general condition driven by management policy. But many systems, when they reboot, by default run time-consuming power-on-self-tests (POST) and clear all memory. The requirement here is to bypass the POST sequence.

The standard Linux kernel provides kexec to allow a kernel to directly boot another kernel and thus bypass the firmware portion of a boot and reduce reboot time. Though this comes with the standard Linux kernel, it requires significant board-specific work that is often not in the kernel. It also does not interoperate well with some other features like real-time. MontaVista adds kexec support for all supported platforms and make sure all features work together.

Latency Measurement Tools

CGE provides interrupt and preemption latency measuring subsystems that can be configured into the kernel at build-time. These subsystems collect information during execution that allows the developer to tune applications, drivers, and kernel to provide the desired real-time responsiveness by pinpointing problems early on. Though not exactly a performance feature, this allows developers to tune a system more easily for maximum real-time performance.

The latency measurement subsystems work by tracking the places in the kernel and drivers where preemption or interrupts are disabled, and then matching these with the places where they re-enabled. The longest 10 regions for disabling each kind of latency are recorded and associated with the source lines that cause the enabling or disabling, and can be read via the /proc interface.
The latency measurement tools measure all lock regions, not just sampling periodically, and so they are not suitable for deployed systems due to their overhead. Reading the latency measurements resets the counters so that a developer can take statistics for various subsystems and loads.

**Symmetric Multiprocessing (SMP)**

One way of increasing the workload of a system is to increase the number of processors operating in parallel. All CGE supported architectures have some systems that support SMP.

SMP improves performance by providing more than one processor running in the same memory. This works well for applications that do not communicate, or for applications that are tightly coupled and require high-speed communication. However, SMP systems tend to not work as well for multiple applications that use a lot of main memory bandwidth, or for applications that require large amounts of exclusive access to the same areas of memory. The contention for memory resources tends to degrade the performance.

Note that SMP kernels do degrade real-time performance due to the requirement for inter-processor communication. This is unavoidable, so to meet the tightest real-time requirements, SMP should not be used.

**Clustering**

Clusters are loosely connected systems that cooperate closely. Clusters have both advantages and disadvantages compared to SMP. Clusters can scale to larger systems than SMP systems, they are cheaper, and they are inherently more reliable.

Clustered applications tend to be more difficult to design and maintain, though. Since the application must be divided up between different systems, this division must be done in a way that does not result in too many messages having to be sent, or in parts of the application becoming bottlenecks. Applications where the parts require little or no intercommunication tend to perform very well in clustered environments. The stronger the requirement for intercommunication, the poorer they perform.

**oprofile**

When trying to improve the performance of a system, sometimes looking at a big view can be helpful, and the ability to get a more detailed analysis of the overall operation of a program can give ideas on how to do this. oprofile does statistical measurement of the kernel and of userland processes, taking counts of where the system was executing when its timer goes off. Since the timer is generally asynchronous to the system's clock, this gives a pretty good analysis even for timer-driven loads.

Though oprofile does come in the kernel, support for architectures and boards can be missing. MontaVista makes sure oprofile works on all supported platform.

**Redundancy**

MontaVista CGE supports a number of redundant subsystems, including redundant Ethernet, RAID disk mirroring, and multi-hosted SCSI.
MontaVista Linux CGE provides a RAID level 1 (mirroring) subsystem that allows for the automatic duplication of disk data. This results in better availability of carrier grade application data, since if a disk fails, the RAID subsystem can carry on with the remaining disk while a new disk is brought in to repair the broken one.

The software driver joins two or more standard partitions on separate block storage devices. The block storage device could be a SCSI or IDE disk drive. Once joined, the software automatically mirrors data written to both disks. If a disk fails, data will be read from the surviving disk.

CGE includes a version of the lilo package utility that properly boots from RAID 1 volumes, even when an active disk has failed. CGE also provides the tools to manage a RAID 1 subsystem, and supports a multi-hosted SCSI bus.

**Redundant Ethernet**

Redundant Ethernet is a term that refers to a driver (the bonding driver) that permits redundant LAN connections to be grouped together to provide for link failover, link throughput aggregation, or both. All links in a bond have the same Ethernet and MAC addresses.

Link failover refers to the process by which a failure of one component of a redundant link can be masked by having its traffic routed to another link.

The bonding driver accomplishes this by continuously monitoring all links available to it, and removing any links from service that are no longer operative. With the proper sort of switch, the redundant links can have their bandwidth aggregated to provide better throughput, and the failover operation can be obtained with any class of switch.

**Clustering**

Though handled in the performance section, clustering also gets a mention here because in addition to performance, clustering can be used to provide redundancy for applications. Using industry standard SA Forum APIs and open...
source application failover clustering technology, failures of any one component are masked from the application, providing enhanced levels of availability.

The OpenSAF package in MontaVista CGE provides a rich set of SA Forum AIS features for providing application redundancy. The openais package also provides a smaller set of AIS features and is still supported.

**Network Replicated Storage**

Sometimes you need to store data on a filesystem in a fault-tolerant manner. MontaVista CGE offers a couple of ways to accomplish this.

OCFS2 is a clustered filesystem from Oracle. It is used with Oracle databases, but it can be used stand-alone as a clustered filesystem. This means that the database be stored on multiple disks in the cluster; if one of the systems fails then another one can take over.

As you might expect, OCFS2 takes some work to set up properly. If you have simpler needs, then DRBD allows two networked systems to replicate their block devices. This acts much like RAID-1, except each disk runs on a separate system and the software uses the network to sync the devices. DRBD operates in an active/standby mode. Setup is fairly easy, and DRBD provides good redundancy.

**Networking**

**Virtual Local Area Networks**

Virtual local area networks (VLANs) are a broadcast domain made up of an arbitrary set of LAN segments, generally irrespective of physical location.

VLANs make it possible to have Ethernet networks with a geographical extent that is larger than the limit imposed by the need to detect collisions. Recall that Ethernet is a CSMA/CD (Carrier Sense, Multiple Access/Collision Detection) technology: a node must be able to detect that another node is already transmitting. This need to detect collisions imposes a physical limit to the size of an Ethernet segment, unless a VLAN is used.

VLANs require a bridge (switch) that understands VLANs and are managed so that various ports can be made members of the various VLANs. Hardware and firmware routes packets to and from ports with the correct VLAN membership.

The most commonly used method for implementing inter-switch communication of VLAN information across a backbone is frame tagging. CGE incorporates the infrastructure and hardware independent driver for VLAN support and a few driver examples are available with VLAN implemented. VLAN support can be very hardware specific because different ethernet devices handle the extra bytes needed to implement tagging in different ways. Therefore, every device and device driver combination may need work to implement VLAN.

MontaVista adds VLAN support for all drivers that need it, and tests VLAN support, as many Linux drivers do not support it.

**Virtual Routing and Forwarding (VRF)**

In many systems, a service provider may want to provide multiple services on the same machine, where each service can have its own set of routing tables and interfaces that are isolated from other services on the same machine. Standard Linux cannot do this—all interfaces appear in the network space of all applications.
To solve this problem, MontaVista has authored VRF. It provides multiple virtualized networks, including applications, routing tables and interfaces, on a single piece of hardware, where each VRF may have its own network namespace and servers. A VRF contains a set of network interfaces, routing tables, fib rules, hash tables and sockets, which are logically separate, although running the same kernel and application codes.

In addition, an iptables match for VRF has been added to the netfilter/iptables infrastructure to allow selective application of filtering and translation rules.

Intermediate Queuing (IMQ)

In generic Linux, netfilter primarily works on egress (outgoing) traffic; ingress (incoming) traffic has limited function. MontaVista has added IMQ, which adds the ability to use general netfilter functions on ingress traffic as well as egress traffic. This lets users perform the same packet filtering and modification on incoming traffic, before the packets are routed, that can be done on outgoing traffic after the packets are routed.

IPSec

More critical carrier traffic is moving to the internet, as this happens protecting signaling traffic becomes critical. MontaVista CGE has support for IPSec with IKEv2, allowing SCTP to run over the top of IPSec, allowing critical signaling traffic to be protected.

Reliability

The Linux kernel is known for its reliability. MontaVista works hard to keep the kernel reliable even with the added features. Also, improving reliability requires more than just making sure the software works well. Most hardware faults and many software faults can be predicted and handled before they become a critical issue.

Fast, Journaling file systems

MontaVista CGE includes several high performance journaling file systems:

- EXT3, the extended file system, version 3, an extension to the standard Linux file system, EXT2.
- XFS, a high performance, journaling file system from SGI
- JFS, a 64-bit journaling file system from IBM

Journaling file systems augment normal file systems with a journal of pending write operations. Only when a write has successfully completed will the corresponding journal entry be deleted.

The advantage of journaling file systems comes at boot time, after a system crash. Even if an update is only partially completed when a system crashes, a journal entry for that update still exists, and can be applied immediately when the system reboots.

With some data sets approaching 500 GB, rebuilding the file metadata can take several hours. But journals remove the need to traverse the entire disk to rebuild the on-disk data structures to make the metadata consistent and accurate.

This has a number of advantages, improving boot time and improving reliability by being able to keep data consistent on the disk even during a system crash.
Resource Monitoring

Resource monitoring allows users to monitor kernel events such as packets received/sent, memory, processes, etc. Statistics can be kept on these kernel events, such as high and low watermarks for system resources. Events and statistics can have thresholds to generate messages when thresholds are exceeded. CGE provides RMON for monitoring various resources. These monitors are compiled into the kernel since they monitor kernel resources.

RMON monitors kernel resources in real-time using counts and gauges. It can monitor:

- Ethernet statistics, either by interface or in aggregate
- Free memory pages
- Number of processes
- Number of zombies
- Number of processes created
- System load

User-space resources may also be defined and monitored from either user-space or from the kernel. CGE comes with an API for controlling and using RMON.

Developers may tie these monitors into the system monitoring framework for a system to detect software problems before they cause an outage. They may use RMON to predict out-of-memory conditions, conditions where too many processes are running on the system, packet overloads, and CPU overloads.

Memory Error Detection and Correction (EDAC)

Carrier grade systems generally have hardware that can detect memory errors and often correct single-bit errors in memory. However, this hardware needs to be monitored so that errors, even corrected errors, can be reported. This way, a faulty memory chip can be detected and replaced before it fails totally.

Standard Linux provides an EDAC framework, but many systems do not have a driver for it. MontaVista adds EDAC drivers to all supported systems that have capable hardware.

openhpi

The openhpi package provides SAForum HPI APIs for monitoring the hardware on a system. This allows system developers to monitor temperatures, fan speeds, power supply voltages, and a host of other types of hardware information provided through IPMI, SNMP, and proprietary interfaces.

Openhpi also supports hot-swap and other maintenance on ATCA systems.

OpenIPMI

OpenIPMI is open source software that interfaces to the Intelligent Platform Management Interface (IPMI) standard that is sponsored by Dell, HP, Intel Corporation and NEC Corporation. In IPMI, each system has a small controller that remains available all the time, even when the main CPUs are not running an operating system. This enables hardware maintenance even if a system has failed or has not yet booted. These controllers usually have a local interface that can be accessed from the main CPU, but often have a network interface that allows access via Ethernet.

OpenIPMI consists of a set of device drivers for directly accessing IPMI controllers and a library that can use the device driver or a network interface to communicate with IPMI controllers. IPMI provides the same types of
information as SAForum HPI, but specifies a messaging interface instead of an API. openhpi has a plugin that uses the OpenIPMI library, providing a consistent SAForum API to an IPMI interface.

In addition to the HPI-type capabilities, OpenIPMI implements other IPMI capabilities such as providing a serial-port interface over the network through the IPMI controller and providing methods to modify the IPMI controller's configuration information.

**Code Reviews and QA**

In addition to features driving reliability, good engineering processes also improve reliability. Code reviews are a standard part of a high quality software engineering process. MontaVista engineers review code brought in from open source in their respective areas of expertise for potential problems. Code developed internally by MontaVista also undergoes peer review. And, of course, any patch to the mainstream Linux kernel gets extensive peer review by the Linux community.

Code for other critical operating system components, like the compilers and base libraries, also goes through similar review in open source, both with partners and inside MontaVista. Less used packages tend to go through less review, but tend to have communities of users that use the tools, report bugs and supply fixes for the software.

In addition, MontaVista develops tests and uses tests supplied as part of the components to verify that the function of each component works properly and continues to work properly as MontaVista fixes and enhances their products. MontaVista has automated much of this testing and continues to automate more of the testing to decrease the time required to test the product, allow more continuous testing, and free QA engineers to develop more tests.

**Application and Kernel Development**

A carrier grade operating system platform brings tremendous benefits for the entire service life of a deployed system, but, as they say, “getting there can be half the battle”. Getting the system developed and debugged for deployment in time for the market window is essential.

The quality and depth of the development tools available have a big impact on an organization’s ability to design, implement and test an application quickly and efficiently.

CGE comes with a complete suite of application development tools:

- Eclipse-based DevRocket Integrated development environment (IDE)
- Configuration tools
- Command-line tools
- Associated utilities

Developers benefit from this wide array of tools because they comprehensively address each phase of development.

**DevRocket**

MontaVista DevRocket is the Integrated Development Environment that supports MontaVista Linux Platform and Application development. DevRocket delivers a set of tools designed to streamline and automate common
embedded Linux development and analysis tasks. Based on standard Eclipse plugins, DevRocket increases developer productivity by simplifying the complex development tasks of embedded Linux. MontaVista DevRocket is a fully graphical integrated development environment (IDE), providing the tools and functionality customers need to develop and deploy system software and applications built using MontaVista Linux operating system technology. DevRocket for CGE is ready to run on various host operating systems and supports integration with third-party Eclipse-based development components.

DevRocket is a comprehensive IDE. It can create a variety of different kinds of projects (kernel, application, file system image), edit and browse code with syntax awareness and highlighting, and check files in and out of CVS and other source code control systems. It supports both cross and native tools, individual and team development, and it is adaptable to a wide variety of different targets.

DevRocket can analyze and optimize code to meet performance or resource consumption goals. It traces and analyzes systems events, allowing developers to learn more about their system. It identifies memory leaks, further optimizing the application’s memory footprint.

Since DevRocket is based on Eclipse, an open source IDE, it is easily extensible, either by the developer or by acquiring specialized plugins. It is capable of growing to meet new or expanded needs.
Linux Trace Toolkit and Dynamic Probes

Dynamic kernel probing is a technology that allows the kernel to be probed (data structures examined, break points established) without the requirement that debugging code be compiled with each module. This allows carrier grade developers to observe system behavior easily and quickly at the debugger level without having to compile the modules under observation with debug capabilities beforehand, or indeed even having to decide which modules may need to be probed.

The dynamic probe kernel option needs to be compiled into your system so that the probe driver is available to insert and respond to probes placed by the system developer or maintainer.

Figure 4 - Output of the Linux Trace Toolkit displayed in DevRocket.
Target Configuration

As part of DevRocket, MontaVista features a built-in, full-featured kernel and filesystem configuration facility with an RPM database plug-in tool for tracking and querying the system for which packages are installed.

The Platform Image Builder allows the user to build a complete set of packages to put on a target's filesystem. It automatically handles dependencies, and it allows the user to select individual files to include or exclude, in addition to handling entire packages.

![Figure 5 - Target image screenshot.](image_url)
Platform Image Builder also gives a graph-based view of the dependencies of a filesystem. Using this tool the user can create and maintain a target's filesystem easily.

Figure 6 - View of the filesystem dependencies

**Project Wizards**

Project wizards automate routine tasks, such as library creation and file importing, allowing developers to focus on differentiating their application. An easy-to-use wizard automates the creation of ready-to-deploy platform images—the bootable OS and application software that run on the target device. This wizard enables developers to rapidly build software stacks including customized kernel and device-specific system and application components, while meeting footprint and performance objectives, saving valuable development time.

**GNU Toolchain**

CGE comes with a complete set of GNU compilation tools, the industry standard. The GNU tools comprise a C/C++ compiler, linker, loader, and debugger, plus assorted utilities like make and cvs. These tools support cross-development as well as hosted development.
The CGE Ecosystem

Our customers rely on MontaVista to provide the very best in Linux operating system software and services. Customers also need to bring in commercial hardware platforms, advanced networking protocols, database and data management software, element management, middleware, and other runtime and software development offerings.

MontaVista Software is committed to supporting and growing this tremendous ecosystem and working with our customers to ensure that solution stacks are available from our alliance partners and us.

Many of the alliance partners who tend to focus on MontaVista Linux CGE also work within the industry standards efforts as members of the Service Availability Forum, Linux Foundation CGL and PICMG. These relationships foster higher levels of collaboration and often better integration and leverage of each other’s capabilities.

A Sample Deployment

NEC Corporation is a leading provider of communication network equipment and has been leading a drive in telecommunications towards open standards platforms based on AdvancedTCA.

The first product based on the new NEC platform was the Packet Core Node, with Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN), which can handle large-volume IP-packet transmission applications such as video streaming and Global Positioning System (GPS) in 3rd generation (3G) services.

When it was released in 2003, this new platform was the world’s first working model of an Advanced-TCA based commercial product. On the ATCA architecture, the new platform also employs Carrier-Grade Linux and NEC’s middleware. NEC’s middleware boasts high Carrier-Grade reliability and capabilities, backed by its long-standing experience and know-how in public network systems. NEC has delivered a new SGSN and GGSN product for commercial deployment.

In order to meet customer needs over the next few years, NEC believed it needed a platform that increased throughput, optimized hardware size, and assured high reliability at the OS level. NEC chose an Intel processor-based ATCA platform powered by MontaVista Linux Carrier Grade Edition (CGE) as the solution they would rely on.

As a result, the new platform was completed in a shorter development period and with reduced development costs by utilizing this open software/hardware platform. The new SGSN and GGSN network elements realize a maximum of 6.2 Gbps throughput, offering a variety of mobile internet services. The deployed systems also met the stringent uptime requirements of the carriers it was delivered to.

“We selected this new platform that is based on open standards to shorten the development time and to reduce total costs... Having selected this new platform, we are convinced that we can quickly respond to the demands for low cost and high performance from the mobile phone market, which is expanding rapidly.”

-Mr. Takatoshi Numasato, senior general manager, Mobile Network Operation Unit, NEC
Summary: The MontaVista Linux CGE Advantage

The adoption of Linux in the communications network infrastructure space is very evident. MontaVista Software is the clear leader in this movement, in terms of focus and implementation against the industry benchmarks of the relevant standards, adoption by the Tier 1 equipment providers, as well as in deployment into the carrier networks.

MontaVista Linux CGE is proven in deployed networks where the required service availability is being achieved. Over the course of pioneering the introduction of Linux into these networks since 2002, MontaVista Software has broadened it expertise and the hardened reliability of the CGE distribution to a level not easily matched by newcomers only now entering this challenging arena.

This expertise and reliability, combined with the ever expanding ecosystem, the powerful DevRocket IDE, and the world class support & services provided by MontaVista Software, makes MontaVista Linux CGE the premier Carrier Grade Linux and the clear choice for Linux in any demanding, high availability application environment.
# Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>bonding</td>
<td>When multiple network interfaces share the same IP and MAC addresses, they are said to be bonded. Traffic is agnostic about which interface to use.</td>
</tr>
<tr>
<td>cluster</td>
<td>A set of systems connected over a network that provide a common function.</td>
</tr>
<tr>
<td>failover</td>
<td>The process of moving active processing of a function from one device to another.</td>
</tr>
<tr>
<td>filesystem</td>
<td>A method of organizing data, generally on a disk, in a way that allows the data to be separated and found easily.</td>
</tr>
<tr>
<td>heartbeat</td>
<td>A message between systems so that the receiving system knows the transmitting system is still operational.</td>
</tr>
<tr>
<td>hot swap</td>
<td>The ability to add and remove hardware while a system is powered and operational without rebooting the operating system.</td>
</tr>
<tr>
<td>jiffy</td>
<td>A jiffy is the period of the Linux system clock. Most timers operate using jiffies. See also timer tick.</td>
</tr>
<tr>
<td>latency</td>
<td>The amount of time between when an event occurs to when the handling of the event occurs. For instance, the time between an interrupt and when the interrupt handler runs is called interrupt latency. A system may have many types of latency, such as scheduling latency, network latency, etc.</td>
</tr>
<tr>
<td>oops</td>
<td>An oops is a Linux kernel message, usually the result of a panic.</td>
</tr>
<tr>
<td>open source</td>
<td>Open source applications permit unencumbered redistribution of the application’s source code.</td>
</tr>
<tr>
<td>panic</td>
<td>A panic is a general-purpose method for halting the kernel when an internal error is detected. A panic creates a dump of registers and the stack contents that help in debugging. It can be used to trigger a kernel core dump.</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Array of Independent Disks. A set of disks combined together to make it look like a single device in order to increase performance, available space, and/or availability of storage.</td>
</tr>
<tr>
<td>real-time</td>
<td>A property of a system that provides guarantees of timing. Generally, this is measured in worst case latency of an event occurring to the handling of the event.</td>
</tr>
<tr>
<td>routing</td>
<td>The operation of handling sending a network packet to the next step in getting to its destination.</td>
</tr>
<tr>
<td>swapping</td>
<td>Moving program memory to/from a disk. Program memory is not generally loaded from disk completely at startup, instead, the program is loaded as it tries to execute portions that have not been loaded. Also, if the system runs low on memory, it may move a program's memory onto a disk and reuse that memory for another purpose. When the program attempts to use that memory again, it will be retrieved from the disk.</td>
</tr>
<tr>
<td>timer tick</td>
<td>A periodic interrupt used by the kernel to drive timers.</td>
</tr>
</tbody>
</table>