



Carrier Grade Linux 3.0: Building out and looking forward

By Bill Weinberg

In February 2005, the Open Source Development Lab (OSDL) released the latest version of the Carrier Grade Linux (CGL) Requirements Definition – version 3.0. OSDL CGL builds on the strong adoption of the 1.1 and 2.0 releases, which now form the basis for half a dozen off-the-shelf Linux platform products and dozens of deployed systems from Telecommunication Equipment Manufacturers (TEMs) and Networking Equipment Providers (NEPs).

This article by Bill Weinberg of the OSDL provides insight into current Carrier Grade Linux adoption trends, and introduces readers to key innovations in Carrier Grade Linux 3.0. It also provides useful guidance to designers presently evaluating CGL implementations.

CGL 3.0 – the next generation

CGL requirements continue to evolve in response to needs expressed by TEMs, NEPs, and their customers at carriers and operator companies. The current 24 members of the CGL committee are listed in Table 1. The CGL committee has the following subcommittees and chairpersons:

- Marketing – Bob Monkman of MontaVista
- Technical – Terence Chen of Intel
- Steering Chair – Peter Badovinatz of IBM

Their input is amplified by the hundreds of customers represented by CGL supplier ecosystem members on the CGL technical and marketing working groups. In all, some two dozen CGL initiative members collect and integrate input from across the communications industry and around the world.

CGL adoption

The Carrier Grade Linux definitions act to drive the marketplace forward. Its leading-edge requirements respond to real-world needs, inspire original solutions by Open Source developers, and drive Linux-based

platform providers to deliver compliant shrink-wrap distribution products.

Today's CGL suppliers have made great strides in offering distributions and tool kits that are CGL 1.1 or 2.0 compliant, and adoption has grown as a result. Global TEMs, NEPs, and carriers deploying CGL are shown in the sidebar. Vendors have that have shipped or announced CGL compliant implementations are shown in Table 2.

Leading edge technologies and specifications need to pause every so often to take stock of adoption trends and

10art-ni	Comverse	IBM	Nokia	Sun Microsystems
Aduva	Ericsson	Intel	Novell	TimeSys
Alcatel	Fujitsu	LynuxWorks	NTT Corporation	TurboLinux
BakBone	Hitachi	MontaVista Software	NTT Data Intellilink	Wind River
Cisco	HP	NEC	Red Hat	

Table 1

Conectiva/Mandrake	Red Hat
Hewlett Packard	TimeSys
MontaVista	TurboLinux
Novell/SuSE	Wind River

Table 2

Global TEMs, NEPs, and Carriers Deploying Carrier Grade Linux

- Agilent UK
- Alcatel
- Cisco
- Datang
- Deutsch Telecom
- Ericsson
- Fujitsu
- Huawei
- Iskratel
- Lucent
- NEC
- NTT
- Nokia Networks
- Samsung
- Siemens

response to new initiatives. For that reason, Carrier Grade Linux 3.0 is best characterized as a *technology release* for evaluation by developers and Linux distributors. Solutions based on the CGL 3.0 requirements definition are expected to arrive in the marketplace in 2006.

CGL 3.0 innovations

Carrier Grade Linux offers additions and refinements to the existing CGL 2.0 requirements set, and expands the CGL definition to cover seven distinct areas:

- Availability – 5-nines (99.999 percent) availability is targeted with no downtime allotted for system maintenance and system expansion.
- Serviceability – Enables remote management and monitoring.
- Performance – Systems must meet service deadlines in real time, and support Symmetric Multiprocessing (SMP), multithreading, and large

memory systems. Systems must also provide efficient, low latency communications.

- Standards – Compliance with standards for interoperability such as the Linux Standard Base, SA Forum Application Interface Specification (AIS), and POSIX.
- Hardware – Support for standards-based COTS hardware components, with support for hot swap and high throughput interconnect.
- Clustering – Removal of single points of failure for both hardware and software.
- Security – Provides a framework for securing both management interfaces and content streams.

Clustering

Scrutiny of real-world requirements dictated by actual TEM and NEP usage reveals that no single clustering paradigm meets the needs of all carrier class applications. Rather than attempt to invent an omnibus cluster definition, CGL 3.0

defines a set of components that comprise a CGL High Availability Cluster (HAC) as shown in Figure 1.

A CGL HAC represents a functional common denominator and is characterized as a set of at least two computing nodes that support migration by applications or workloads among those nodes. Externally established policy mechanisms determine:

- how nodes work together
- how nodes distribute loads
- which types of failover to support

The CGL HAC definition also accommodates the two divergent rationales for using clusters; to provide higher service availability (primary), or to scale compute or storage capacity (application determined).

Clustering for high availability
Broadly speaking, clustering is a key means

to achieve up-times of 5-nines or better. Availability is defined in terms of both Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR), and clustering can enhance both factors.

Any type of cluster increases MTBF by sharing the risk among multiple nodes and by eliminating single points of failure. A CGL HAC improves MTTR by supporting rapid repair and replacement methods like hot swap, and also by specifying that systems offer failover in sub-second time frames and boot/reboot in seconds.

The base configuration for a CGL HAC is a 1+1 hot standby cluster with one active node and one on standby. CGL HAC implementations can extend this basic definition to include additional active nodes (M) and/or stand-by nodes (N) as needed by an application or specified by policy.

Unlike enterprise clusters or those used in grid computing, most telecommunications applications are well served by loosely-coupled clusters without shared storage. This simpler scheme eliminates

the possibility of a failed shared component affecting the availability of the service or the system as follows:

- Eliminates single points of failure – Operators can replace or repair failing nodes without impacting service up-time.
- Supports in-place system and application upgrades – Allows maintenance of software and hardware without affecting availability of service.
- Isolates faults – Segregates failing nodes from the cluster and enables the continuation of service with the remaining healthy nodes.
- Scales Capacity – Offers increased capacity to meet varying loads or traffic.

SA Forum AIS compliance
CGL 3.0 clustering requirements build on industry-standard programming interfaces. These standard APIs include the Service Availability Forum (SA Forum) Application Interface Specification (AIS) for clustered applications. AIS is operating system independent and applies equally to proprietary and open source cluster paradigms.

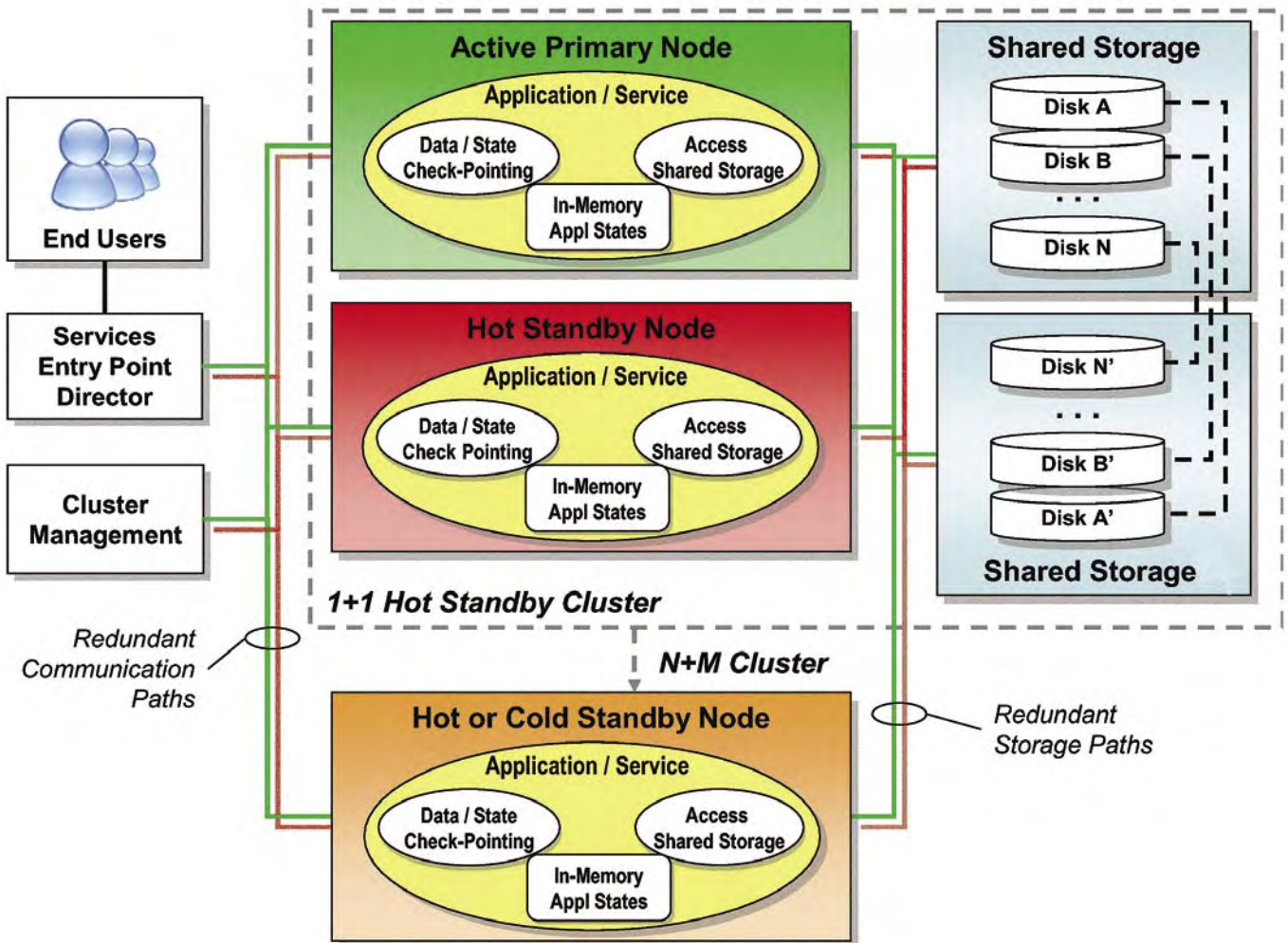


Figure 1

The SA Forum AIS specifies a family of APIs including Membership Service, Checkpoint Service, Event Service, Message Service, and Locking. AIS also specifies an Availability Management Framework (AMF) that provides resource management and the application failover policy for a cluster.

CGL 3.0 security requirements

Traditionally, telecommunication applications differed from general-purpose computing environments in several key aspects:

- telecom systems did not have many user accounts
- user accounts did not reflect individual users
- telecom systems were configured through custom user interfaces instead of logon shells

In today's IP-based world of convergent voice and data, a range of new security risks has emerged that directly impact systems built on Carrier Grade Linux:

- Use of general-purpose systems for IP-based computer telephony, PBX, voicemail, and VoIP applications
- Extensive use of in-band management interfaces
- Exposure of networks from unsecured or misconfigured IEEE 802.x wireless interfaces
- Spoofing of wireless client device IDs and BlueTooth exploits and viruses
- Cracking management systems from insecure Common Gateway Interface (CGI) interfaces

Management and control threats

According to TEMs and NEPs, the greatest threat to the communications environment comes from access to management and control interfaces by unauthorized parties. A common point of access comes from the self-provisioning of data or voice services by end users.

For example, many ISPs and VoIP service providers allow customers to create new mailboxes or route incoming calls from internal extensions to any e-mail address or telephone number in the world with just a few clicks on a web page. Facilities like these create a new set of risks:

- Unauthorized rerouting of e-mail and telephone calls by disgruntled associates or unscrupulous competitors
- Exploitation of vulnerabilities in software to jump from one security plane to another, opening new avenues for security exploits

CGL 3.0 security features

Mitigating these and other risks requires both carefully elaborated security policies and robust authentication schemes. Moreover, while many options exist for securing versions of enterprise Linux against intrusion and subversion, relatively little investment has been made to date with regard to dedicated Linux-based communications systems.

In recognition of the evolving nature of the field, the CGL working groups have elected to postpone publication of the security section of the 3.0 CGL definition until later in 2005. In the interim, member companies are solidifying their requirements for securing CGL platforms. The OSDL, for its part, has established a Security Special Interest Group (Security SIG) to generate one or more CGL security profiles and a unified architectural approach for security on communications servers.

“...the greatest threat to the communications environment comes from access to management and control interfaces by unauthorized parties.”

CGL compliance and registration

Strictly speaking, CGL is not a standard, but is rather a family of specifications that include compliance to POSIX, LSB, SA Forum, and other pre-existing and evolving standards. As such, the OSDL does not establish a compliance or conformance discipline, complete with test suites and certification regimens. Moreover, many CGL Priority 1 and Priority 2 requirements can be satisfied through multiple Linux-based mechanisms (as with clustering or real-time responsiveness).

In this context, Linux providers with a communications-directed distribution or platform that wish to use the Carrier Grade Linux brand are required to register their platform's adherence to the particulars of the CGL Requirements Definition. Furthermore, they must publish a registration document that calls out their products' implementation of each requirement and provides proof of compliance or conformance to standards implicated in the CGL definition.

CGL shopping guide

If you are embarking on a project to develop advanced voice and data products, Carrier Grade Linux platforms provide advantages in portability, reliability, performance, and development and deployment cost. However, not all Carrier Grade Linux implementations are created equal. Important CGL platform selection criteria are listed in Table 3.

Summary

Carrier Grade Linux adoption continues to expand, with equipment and services providers supporting renewed voice and data build-out with COTS hardware, and Linux and other open source software. TEMs, NEPs, carriers, and operators are building on Linux for its robustness, scalability, performance, and lower costs of acquisition and long-term ownership.

While legacy RTOS and other highly-purposed software continue to be needed for highly-specialized blades and real-time sensitive interfaces, the application domain of Carrier Grade Linux continues to expand.

The accompanying expansion of suppliers for Carrier Grade Linux solutions directly reflects the ubiquitous design-in that Free and Open Source Software (FOSS) and Linux enjoy in current and next-generation communications. Stay tuned for new Carrier Grade announcements from Linux platform providers and voice and data system suppliers throughout 2005 and 2006. **ECD**

Bill Weinberg brings more than 18 years embedded and open systems experience to his role as Open Source Architecture Specialist at the Open Source Development Labs. Bill can be contacted at bweinberg@osdl.org.



OSDL – home to Linus Torvalds, the creator of Linux – is dedicated to accelerating the growth and adoption of Linux in the enterprise. Contact the OSDL directly for membership and lab usage information.

Open Source Development Labs, Inc.

12725 SW Millikan Way • Suite 400

Beaverton, OR 97005

Tel: 503-626-2455 • Fax: 503-626-2436

Website: www.osdl.org

Selection Criteria	Selection Comments
Registration <ul style="list-style-type: none"> ✓ CGL registration 	Strictly speaking, vendors may not promote a Linux distribution or cross-development kit as Carrier Grade Linux unless they have shown that their product implements all Priority 1 Carrier Grade Linux requirements. For a listing of Registered CGL products and registration documentation, visit: www.osdl.org/lab_activities/carrier_grade_linux/registration.html
CGL Version <ul style="list-style-type: none"> ✓ CGL 2.0 or later 	With the release of the latest Carrier Grade Linux Definition, there are now three evolutionary versions of the CGL specification: 1.1, 2.0, and 3.0. A number of early adopters have been shipping 1.1 compliant platforms for over two years; six vendors have announced or shipped CGL 2.0 registered products; new CGL 3.0 product announcements are expected after the CGL 3.1 definition release later in 2005.
Linux Kernel Version <ul style="list-style-type: none"> ✓ 2.6 kernel 	CGL 1.1-based implementations were all based on versions of the 2.4 Linux kernel. At present, CGL 2.0 platforms present a mix of 2.4 and 2.6 kernel technology, with 2.4-based implementations back-porting large amounts of enabling code from a 2.6 context (such as NPPTL). Moving forward, you should look for 2.6-based CGL products.
CPU Support <ul style="list-style-type: none"> ✓ Intel IA-32 or AMD x86 ✓ Embedded CPUs 	Enterprise-oriented CGL platform products and distributions by definition target COTS hardware based on Intel IA-32 and AMD x86 architectures. More deeply embedded CGL implementations can target PowerPC Single Board Computers (SBCs). Targeted blades are based on PowerQUICC, MIPS, and Intel XScale CPU families. Note that it is common for embedded architecture implementations to support fewer CGL definition features than COTS Intel IA-32 or AMD x86 implementations.
SBC Support <ul style="list-style-type: none"> ✓ AdvancedTCA ✓ CompactPCI ✓ PC/AT 	SBC manufacturers today leverage the PC/AT core architecture and high performance Intel IA-32 or AMD x86 CPUs for COTS compatibility and cost savings. CGL builds on this core and at a minimum will boot on almost any Intel IA-32 or AMD x86 SBC in any form factor. CGL providers differentiate their platform offerings with system-specific hardware support, especially for different SBC and blade types, management capabilities (like IPMI), hot-swap and hot-plug support, and CPU and board support for application-specific blades and line cards based on PowerPC, ARM, MIPS, and other architectures.
Development Hosts <ul style="list-style-type: none"> ✓ Self-hosted ✓ Cross development 	With COTS Intel IA-32 or AMD x86 targets, the Carrier Grade device prototype itself is frequently used for self-hosted development. For larger development teams, more complex Carrier Grade applications, and more deeply embedded target CPUs, CGL suppliers should offer cross development options hosted on standard Linux, Solaris, and even Windows workstations.
Integrated Development Tools <ul style="list-style-type: none"> ✓ Eclipse ✓ CVS ✓ ClearCase 	Data and telecommunication projects are among the most complex types of embedded designs, often involving millions of lines of code, and dozens or even hundreds of developers. In the past, your organization may have looked to COTS vendors only for platforms or single tools. With next-generation CGL-based systems, your team should seek suppliers that integrate standard tools into development environments that interoperate with global development frameworks, code management tools, and project management software.
CGL Platform Value-Add <ul style="list-style-type: none"> ✓ Productization ✓ Support & services ✓ CGL registration ✓ Domain-specific value 	As with any type of open source product, CGL vendor value-add can come from Productization (integration, test, QA), Support (technical support, bug fixes, updates) or Services (customization, training). With CGL, the first value-add is definition compliance and registration of a fully-integrated CGL platform. Additional value-add comes from application-domain specific features: CPU and form-factor support.
Validation of M/W and Applications <ul style="list-style-type: none"> ✓ LAMP ✓ HA middleware ✓ IPMI ✓ VoIP 	Enterprise Linux distribution suppliers, the Free Standards Group, and the OSDL are working to make interoperability of system software, middleware, and applications as transparent as possible for all versions of Linux. However, when choosing a CGL implementation for your next project, you should work closely with your integration team, your CGL Linux supplier, and key ISVs to validate that your actual application workload is tested and fully functional. Commercial CGL providers add significant value through the technical side of their partnering programs with ISV-ware validation and assurance. If your CGL project does not build on Intel IA-32 or AMD x86 hardware, the builds from source, integration, and test will likely fall to your own team or to a third-party services company.

Table 3