Open source High Availability middleware liberates telecom systems developers

By Zsolt Haraszti

More than a decade ago, the telecommunications industry started migrating from in-house proprietary Operating Systems (OSs) to COTS and open source OSs such as Linux. In this article, Zsolt explores how the very same market forces are now becoming applicable to the next layers of the software stack, specifically High Availability (HA) and system management middleware.

Throughout the history of communications systems, fault tolerance and other means to building resilient platforms were the exclusive and proprietary purview of Telecommunications Equipment Manufacturers (TEMs) and, more recently, Network Equipment Providers (NEPs). After the “dark ages” of telecom in the late 1990s, telecommunications system design began to experience a renaissance with ever-increasing openness in hardware and software. Breaking with a proprietary past, TEMs and NEPs are more frequently building highly available systems from COTS hardware running open source system software. In particular, they are increasingly turning to Linux and other open source software technologies as the platform of choice for next-generation converged voice and data systems.

The move to COTS hardware and open source OSs represents the commoditization of the essential computing infrastructure. TEMs/NEPs and other Original Equipment Manufacturers (OEMs) once invested in internal resources that developed solutions from the bottom of the software stack all the way to the top, from hardware design itself to OS, to enabling middleware to systems management, to applications and service delivery. Today, such vertically integrated designs have given way to investments in adding value above standards-based and standards-compliant commodity platforms.

Open source value line

Businesses based on open source system software draw a value line (Figure 1) between the commodity base platform and the value-added technologies and services above it. Investment beneath this value line represents marginal activity to support a shared community resource (for example, Linux and Apache server) and offers outsourcing opportunities. Differentiating engineering and productization only occur above the value line. Open source software-based businesses must realize that the value line is dynamic and fluid, moving upward through the stack.
OEM building blocks
This transition exposes the seams in what TEMs, NEPs, and other OEMs once promoted as seamless vertical solutions. As such, communications equipment companies are actually more aligned with the value-added resellers and integrators of old; they construct voice and data systems by integrating and adding value to off-the-shelf building blocks. Their ongoing differentiation comes from unique in-house applications and service delivery vehicles, further differentiated by branding and support services.

Outsourcing system software began more than a decade ago, even before the rise of Linux and open source software in telecom. Starting in the late 1990s, OEMs shifted their traditional in-house investment in kernels and Real-Time Operating Systems (RTOSs) toward buying COTS versions of the same software from vendors such as Wind River and Sun Microsystems. The more recent move to Linux at the OS level represents a further step in this progression from proprietary commercial software to open source commercial software.

In fact, analyst firm Venture Development Corporation estimates that almost 25 percent of telecom software expenditures in 2006 were Linux based. Moreover, the Linux/open source portion of telecommunications spending is growing at 26 percent/year, much faster than the total market, which they cite at 19 percent CAGR.

High Availability (HA) middleware
Until recently, the area of enabling middleware for telecommunications closely resembled the legacy OS market. OEMs still spent unjustifiably large amounts of their budgets reinventing, developing, and maintaining completely proprietary software subsystems for fault management, alarms/notification, provisioning, redundancy, check pointing, and application infrastructure functions. When these key functions were outsourced, the technology came from a small number of Independent Software Vendors (ISVs) or software groups at Independent Hardware Vendors (IHVs). Both ISVs and IHVs, while implementing standards-based middleware offerings, did so with their own proprietary flavor or with outright proprietary paradigms and an agenda of locking OEMs into particular software and hardware schemes.

Open source Linux revolutionized the system software layer of communications systems and other embedded designs (to say nothing of enterprise servers, blades, and workstations). Now, the advent of open source HA middleware is radically changing the businesses of TEMs and, with the convergence of voice and data, NEPs as well.

By breaking away from the proprietary past and delivering open source middleware, ISVs are helping OEMs realize a range of benefits long promised but seldom delivered by open systems offerings.

Rapid innovation
Open source developers’ motto is “release early, release often.” By leveraging a core of dedicated developers, reaching out to larger groups of early adopters, and serving an ever-larger end-user community, open source projects can prototype, integrate, propagate, and field test in very tight cycles. Projects and programs can accrue new features faster and deploy with higher quality through cross-company cooperation and the power of networking.

Standardization
In-house middleware is typically designed to serve a single internal audience and gives industry standards short shrift. Single source commercial middleware is targeted at a wider audience, but suppliers often enhance standard interfaces and functionality and end up locking customers into their solutions. By contrast, open source middleware exists for a community whose members enhance and maintain it.
and share the need and responsibility to interoperate. Open source software in general and middleware in particular start with standards like those from the Service Availability Forum (www.saforum.org), SCOPE Alliance (www.scope-alliance.org), and Carrier Grade Linux (www.linux-foundation.org) as core requirements instead of returning to standardization after the fact as window dressing.

**Reduced risk**

OEMs have grown accustomed to very limited warranties and near zero acceptance of liability by ISVs and other suppliers. Perhaps the greatest risk of working with proprietary software vendors is the single source nature of most software products. If a key software supplier is acquired, faces IP litigation, or otherwise founders financially, OEMs can lose their license to design-in and deploy critical software components, often with no legal recourse and no technical alternatives.

By contrast, open source software, as a shared resource, distributes risk across a community of developers and users. No one entity need face any risk alone, and users have the option of contracting with a supplier to warranty and indemnify against specific risks around IP, performance, and suitability. Moreover, if the company behind a project or technology changes course or goes out of business, OEMs can find alternate sources for support and services of the same code base. OEMs can leverage project community resources and choose to be self sufficient by maintaining and supporting key software themselves.

Insurance against lock-in extends beyond OEMs and helps protect carriers and operators. The standards compliance and transparency afforded by open source highlights OEM value-add and helps ensure greater interoperability among various TEMs’ and NEPs’ offerings.

**Lower costs**

Shared development + shared support = shared risk = lower costs. It’s that simple. When technology development and support do not reside in a single corporate entity, everyone benefits from lower acquisition costs, standards-based interoperability, more support and service options, and avoiding single source lock-in.

**Time to market**

Open source middleware delivers on long-standing promises of faster time to market by accelerating evaluation, acquisition, integration, and test. Open source software by definition gives communications applications developers and integrators the opportunity to “try before you buy” with access to source code for review and prototyping. Open source software is usually less expensive than proprietary alternatives, easing justification, and has straightforward licensing terms, especially dual licensing, that simplify negotiation. OEMs’ test and quality assurance departments can leverage access to the original project team and peer communities to address software bugs and performance issues in days instead of months.

**Reducing costs and focusing on differentiation**

Popular notions of cost reduction through free and open source software center on lower acquisition costs; actual lower costs extend across the lifetime of the software and of the organization that acquires, integrates, deploys, and supports it. A decade ago, the total software content in communications systems ran from about 100,000 lines of code up to 1 million lines. That count included OS code, device interfaces, protocol stacks and daemons, systems management and fault resilience subsystems, and the differentiated code directed at a specific application. In the intervening years, all types of embedded systems have shifted from a hardware-centric design approach to one that focuses on software and services delivered, with software content doubling annually. This dramatic growth in software means that companies must staff to meet the challenge of developing, integrating, and deploying code bases more than two orders of magnitude larger than a decade ago. Economic realities, however, dictate that OEMs find themselves with smaller headcounts.

In response to this software content explosion, OEMs began to outsource software engineering, both literally and figuratively. In particular, they outsourced shared hardware support and software infrastructure by turning to COTS hardware and open source software. In doing so, they further shifted the scale of software engineering involved. For example, instead of line card applications running on 5,000 line legacy RTOS platforms and managing a handful of calls, applications built on open source Linux (with 4 million lines of code) handle thousands of calls on COTS hardware blades. And while many TEMs and NEPs can boast OS expertise, their management most often chooses to build on preintegrated and tested commercial Carrier Grade Linux platforms to mitigate risk and limit marginal investment.

The same logic now applies to enabling middleware for HA. The software content in COTS HA middleware can run from 250,000 to as high as 500,000 lines of
code. Based on standard methodologies (for example, Boehm’s Constructive Cost Model)[1], an OEM would need to invest at least 2,800 staff-months to create such code from scratch (under optimal conditions with at least 85 engineers working on schedule for more than 2.5 years). Even with the availability of open source software components to perform the same functions, OEMs should think twice about integrating and supporting such a code base wholly with their own resources. Ongoing maintenance for 250,000 lines of source code, even with minimal new code or patches, can occupy more than a dozen full-time software engineers.

Fortunately, communications systems developers no longer need to roll their own availability and fault resilience middleware, either from legacy internal code or from available open source building blocks “in the wild.” Such a high initial and ongoing investment level justifies and is driving the trend toward shared, community-based development, with options for both community and commercial productization and support.

**Open source does the heavy lifting**

Since the beginning of communication systems development, equipment manufacturers were forced to invest, often massively, in building and maintaining infrastructure software that did little to differentiate their wares or add to their bottom line. The shift to COTS hardware, standard communication protocols, and open source software platforms helped liberate TEMs and NEPs from such marginal activities by outsourcing hardware and software to shared ecosystem and community resources. The heavy lifting of providing HA, manageability, and serviceability, however, still fell squarely on the shoulders of OEMs and integrators. With the advent of off-the-shelf, open source solutions such as the OpenClovis Application Service Platform, communications systems developers can now focus on differentiated, customer-driven features and functionality.

**OpenClovis and open source**

Clovis Solutions was founded in 2002 with a mission to offer communications systems vendors key software infrastructure components. Over time, the company recognized a growing opportunity to build on its R&D team’s more than 400 years of collective communications systems expertise and expand it into a global community of developers, deployers, and partners.

In 2006, the company relaunched itself as OpenClovis and released its product source code as open source. Table 1 shows how OpenClovis Application Services complement and extend a Carrier Grade Linux or RTOS-based platform. Each of the layers and component types in the diagram engenders its own ecosystem and community.

<table>
<thead>
<tr>
<th>Community</th>
<th>Component type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICMG</td>
<td>Commercial off-the-shelf hardware can mean white box PCs with AT or ATX motherboards or a range of SBC systems, but in this context, the term most commonly indicates AdvancedTCA blades with Intel and AMD 32- and 64-bit application processors. The advantage for modular telecommunications systems lies in the widespread knowledge and support for hardware also found in desktop PCs and enterprise servers. That knowledge base is enhanced and expanded by the particular ecosystems and communities built around organizations like PICMG and the Communications Platform Trade Association (CP-TA).</td>
</tr>
<tr>
<td>The Linux Foundation</td>
<td>Carrier Grade Linux: Many of today’s rich and robust communications systems are deployed over versions of Carrier Grade Linux. The Carrier Grade Linux specification, now in its fourth version, was developed by leading hardware suppliers, TEMs, NEPs, Linux distribution vendors, and ISVs to provide direction for the evolution of the Linux kernel and base OS. Today, the carrier grade specification is maintained by the Linux Foundation with compliant implementations from a dozen software and systems suppliers and from community sources (for example, Carrier Grade Debian Linux).</td>
</tr>
<tr>
<td>Service Availability Forum</td>
<td>Service Availability Forum APIs: The SA Forum is a consortium of communications and computing companies working to develop and promote adoption of High Availability and management software interface specifications. SA Forum APIs provide standardized methods for systems designers and integrators to monitor and control systems chassis and other infrastructure hardware.</td>
</tr>
<tr>
<td>OpenClovis</td>
<td>OpenClovis Application Services provide a rich, open source middleware platform for building highly available communications applications. By presenting core SA Forum APIs and extended OpenClovis interfaces, developers and integrators can build on more than a dozen service types that include availability, component management, check pointing, mediation, fault and chassis management, and diagnostics.</td>
</tr>
</tbody>
</table>

References

[1] Using CoCoMo 81Intermediate Model, with input values of size=500000, mode1.20, rely1.49, datai.00, pcri1.15, timel.11, stor1.00, virt1.00, turn1.00, acapi1.00, aexpi.91, pcapi1.00, vepxi.00, lexpi.95, modpi.91, tool1.91, scedi1.04

**Zsolt Haraszti leads the product management team at OpenClovis, Inc. and has 14 years of experience in the data and telecommunications industry. He has worked at Ericsson IP Infrastructures, Inc. as a principal engineer and a software development team lead focusing on gigabit edge and blade router architectures, and at Ericsson Research, Stockholm, as a research scientist dealing with VoIP over wireless problems and communications networks performance analysis. His work has generated numerous patents and scientific papers. Zsolt holds MS and PhD degrees in Electrical Engineering from North Carolina State University.**