The Linux Enterprise Cluster

Karl Kopper, No Starch Press, 2005, \$49.95

ISBN: 1593270364

Have you ever wanted to build a highly available cluster? This book shows you how to do it, using commodity hardware and free software. Author Karl Kopper has worked with distributed computing environments on a number of platforms; he uses the cluster he successfully deployed at a

food distribution company as the basis for the examples in the book.

The book begins with some short introductory chapters outlining the properties of a Linux enterprise cluster. The most important of these is that the nodes within a cluster should not know they are part of a cluster. From this, it follows that an unmodified operating system can be used on a cluster node, and that the failure of one node will not impact the other nodes.

The remainder of the book is split into four parts. The first, "Cluster Resources," contains chapters that are pertinent for any Linux system. They deal with starting services (using `/etc/inittab` and System V init scripts), packet handling (iptables, ipchains, and route), and kernel compilation. The author provides many command-line examples and block diagrams, and makes comments where appropriate relating to 2.4 and 2.6 series kernels.

Part 2, "High Availability," begins by illustrating how to make ssh and rsync work. The step-by-step examples are invaluable. It is a pity that there are a couple of errors therein, and I'm hoping that some notes relating to these errors will soon find their way onto the book's errata page at the publisher's Web site.

Part 3, "Cluster Theory and Practice," outlines the differences between Linux virtual server clusters that use network address translation and those that use direct routing. Complete recipes are given for building clusters of each type (using director failover in the second instance).

The final part, "Maintenance and Monitoring," demonstrates how you can use the Mon package to generate alerts based on simple network management protocol status responses, and the Ganglia package to collect and display node performance data.

Overall, the book contains most of what you'll need

to build an enterprise cluster. You can, of course, buy a Linux cluster off the shelf from companies such as Orion Multisystems, but you will still need a book like this to help you successfully administer it and exploit its capabilities.—*G. K. Jenkins*

Working Effectively with Legacy Code

Michael Feathers, Prentice Hall PTR, 2004, \$44.99

ISBN: 0131177052

For most people, legacy code is "difficult-to-change code that we do not understand." In practice, Michael Feathers translates this definition into "code without tests." If we had automated unit tests for the code we have to modify, we could change the code quickly and verifiably. Even if we did

not fully understand it, test failures would immediately warn us about potential problems in our modifications.

The problem is that it is not easy to get tests in place. Many designs are so convoluted that system dependencies make individual components almost impossible to test separately. Fortunately, Feathers provides a handful of useful techniques that can help make maintenance nightmares disappear.

The core of this book deals with practical techniques for changing software. It describes typical problems software maintainers must face, using an original FAQ format. This format makes chapter titles lengthy and rather whimsical, but it also converts this book into a useful guide to have at hand when dealing with legacy code.

The book provides techniques for changing the way that code is organized to make builds (hence, testing) easier. You learn how to add features to code being tested using test-driven development and programming by difference. Examples, mostly in Java and C++, show you how to get tests into place. Easy-to-understand effect-analysis techniques lead you to the "pinch points" where you should test the behavior of your legacy code. Simple techniques help you uncover and communicate the architecture of large systems. Most of all, the practical advice in this book is given without forgetting the rationale behind your decisions, such as when to sacrifice object encapsulation to improve unit testability and, more importantly, your understanding.—*Fernando Berzal*

Reprinted from *Computing Reviews*, © 2005 ACM, <http://www.reviews.com>