Scientific Programming 13 (2005) 173–175
IOS Press

Book review

High Performance Linux Clusters by A. Joseph D. Sloan, O'Reilly & Associates Inc., Sebastopol, CA, USA, 2004. ISBN: 0-596-00570-9

High Performance Linux Clusters describes how to provide substantial computational resources for large-scale computing. The approach is to use low-cost or freely available components arranged as a computational cluster. The book has 350 pages and includes a Preface, an Index, and five Parts divided into 17 Chapters and an Appendix. The Parts introduce the idea of cluster computing, provide a quick-start overview, describe how to configure clusters for particular needs, and tell how to write message-passing applications. The fifth Part is the Appendix, which includes copious references, both in-print and on-the-network.

The book's goal provides guidance when building the most effective distributed computing resource given tight budgetary constraints. These days, this means using hardware based on Intel chips (or clones), Ethernet, and software based on the Linux operating system. An analogy is a couple just starting a family, whose budget requires the investment of "sweat equity" in order to own a dream house. So, this book's goal is to help the scientist, engineer or economist whose program takes too long to run on the available computer, and who has only a limited budget to invest. The book's advice: invest "sweat equity" to achieve the goal of reducing the time-to-solution via building a cluster. So how to build a cluster? This book provides check lists to assist with both the planning and the building of a cluster.

The book covers hardware and network issues, but the main emphasis is a step-by-step guide to selecting, obtaining, configuring, installing, and finally, using, the software which makes a LAN into a cluster. The book promises to guide the reader through the whole process. It won't delve deeply into any particular step but rather provides a strategic overview of the process. Follow the plan of the book, so it promises, and the result will be a working, useful cluster. Supercomputing-on-a-budget may be had for those who merely have the need, but not the resources, for jobs that would otherwise require "big iron".

The Preface introduces the plan of the book. The plan is to use freely available software (alas, the hardware must be found somehow). The software should be downloaded from the Internet. The author has done the reconnoitering, and has found out which software to get. The hardware may be recycled computers, or might be purchased new. How to keep the costs low is certainly discussed. Then one needs the applications software, which probably means doing some actual programming by using a message-passing library.

Chapter One starts with the question, what is a cluster? If a department has its computers on a Local Area Network, is that already a cluster? Sadly, no, a LAN is not a cluster. There's a further layer of logical connectivity needed to go from LAN to cluster. That transition is what this book describes. Don't think that "grid" is the latest buzzword for cluster; grids have a layer of abstraction beyond that present with clusters. With grids, the user is not expected to know the name or address of the eventual resource provider – the names are replaced by descriptions, the services are "virtualized" by a layer of software providing an abstraction beyond clustering. With cluster computing, the users will know the names of the computers providing the services. The limitations of parallel computing, specifically expected speed-ups and the expected scaling as problem size increases are discussed.

Chapter Two prompts one to state clearly the goals for building a cluster. What question is one trying to answer and how is the cluster expected to help get the answer? I especially like this chapter. It prompts the asking of a series of questions pertinent to deciding on how to proceed. The first step in any engineering project (and building a cluster certainly qualifies as engineering!) is to state clear goals. The author examines the likely implications of the hardware and software choices available. The idea of cluster kits is presented, both download based and CD-ROM based. Next, there is a discussion of benchmarking the initial results to establish a baseline in anticipation of measuring the results of software configuration changes or hardware additions.

ISSN 1058-9244/05/$17.00

Chapter Three discusses the variables in the decisions regarding hardware. Hand-me-down hardware is cheaper, but may constrain the capability of the resulting cluster. So how to reduce the costs of new hardware? The compute nodes will not continually need keyboards and monitors, so one can eliminate them. But what to do if the BIOS will not boot without a keyboard? The author discusses solutions, in this case a switch to connect one keyboard-and-monitor pair to any of several computers. And how to build the network? Will good old Ethernet suffice? It's cheaper than a dedicated high-performance interconnect, and suitable for many purposes.

Chapter Four discusses the variables in the decisions that are to be taken regarding the software. The author's bias in favor of Open Source software appears to be well placed. Everything needed for fully functioning clusters can be downloaded for free on the Internet. Indeed, one question facing the cluster maker is the choice of alternatives. Linux itself and the attendant configuration and security issues are the subject here. For example, the trades involved with having a single front-end connection to outside networks are discussed. A dedicated front-end allows much easier communications within the cluster itself. For example, some packages used are easier, or are only possible, to configure if the secure shell is present.

Chapter Five discusses openMosix. Many virtual memory systems have a set of addresses for the user space of a process, and a distinct set for the system addresses of a process. openMosix is a set of patches to the kernel which allows the user addresses to be moved to a different node within the cluster. System calls are still processed on the originating node. One difficulty is that there is little control over which process is executed on which node. The openMosix system is supposed to balance the load automatically. Nevertheless, as processes make system calls, there is communication from the node to which user addresses have been moved to the system space on the originating node.

The hardware choice is almost certainly Intel, or a clone. The software choice is Linux, perhaps Red Hat although other Linux sources will do. So the biggest decision remaining is how to establish the cluster software itself. The author presents this as a choice between OSCAR and ROCKS. These are two schemes for selecting and installing cluster software. ROCKS includes the installation of Red Hat Linux, OSCAR does not.

Chapter Six discusses OSCAR. OSCAR is the *Open Source Cluster Application Resources* package. Basically, OSCAR provides the administrator with a graphic interface to guide the choice of included software. It downloads the individual packages and installs them. It ensures that they have mutually compatible, and useful if not optimal, configurations. The author remarks that OSCAR delays, rather than permanently reduces, the need to eventually learn cluster administration. The key to OSCAR is the *opd*, the OSCAR Package Downloader. This is the software which brings the packages to be included on the cluster to the system. OSCAR is supported by the Open Cluster Group, which in turn is supported by Dell, IBM, Intel, NCSA, and ORNL. The chapter describes the installation and configuration process.

Chapter Seven discusses ROCKS. ROCKS is from NPACI. The most significant difference between ROCKS and OSCAR is that ROCKS installs Red Hat Linux automatically, so the administrator doesn't have to prepare the nodes. ROCKS allows additional software to be selected in the form of *rolls* (get it? ROCKS and rolls). One of the rolls is the Intel compilers, C/C++ and Fortran. A license is required from Intel to use them. Again, the chapter describes the installation and configuration process.

Chapter Eight discusses cloning systems, that is, automating the process of duplicating system software. Note that a system cannot be duplicated completely in that it would give the clone the same network parameters as the original system. So the cloning has to be "almost" with just enough differences to allow the network to work. Use of *Kickstart. g4u* (ghost for Unix), and *SystemImager* are discussed.

Chapter Nine discusses programming software. The discussion of programming languages is the only section of the book I found not to be helpful. First, Fortran (note the lowercase) means Fortran 2003. The Gnu Compiler Collection has a Fortran 95 component, to be merged into the GCC main distribution with the 4.0.0 release. (I have both the g95 fork and the mainline gfortran versions of Gnu Fortran on my Linux computer today.) Even if considering only free compilers, a discussion of Fortran 77 is as out-of-date as a discussion of K&R C.

The most unhelpful suggestion is that the programmer learns a new language while simultaneously learning message passing! However, I feel the bigger issue has been missed. This is, after all, *High-Performance* Clusters. The missing issue is that of Gnu Compiler Collection (GCC) compilers versus commercial compilers. A homogeneous cluster needs only a single-seat for the compiler, perhaps on the front-end node, or on

a dedicated compile server-node. An earlier chapter mentioned the Intel compilers, which generate more efficient code on Intel hardware than the GCC compilers, as commercial compilers do generally. If one obtains a modest increase in execution rate, a commercial compiler is well worth the cost. If one can qualify for the Intel non-commercial license, the Intel compilers are free (but read Intel's definition of "non-commercial" carefully, it's more strict than you might think!). In any case, the Academic prices of several compiler vendors are modest. This means one may have high performance compilers for both C/C++ and Fortran for less than the costs of a single node. No matter how fond one is of the GCC, or of free software in general, getting several nodes worth of execution for the costs of a single node must be a good buy. The discussion of high performance computing will be helped by a reference to Dowd & Severance's excellent book **High Performance Computing**; unfortunately out-of-print, from the same publisher.

But that's enough, let's move forward. Some version of MPI will most likely be the message-passing library used on a cluster. The next chapter describes the installation of LAM/MPI and MPICH. The discussion then progresses to debuggers. The section on HDF5 states that there is no Fortran binding, which is wrong. Simply go to the website mentioned in the book to get the Fortran binding. I'm especially pleased to see the section on SPRNG, parallel random-number generators may well be needed with cluster computing. It's good to see that the needs of these users are not overlooked.

Chapter Ten discusses software for managing a cluster. Now that there are many nodes to maintain, and the administrator must repeat every maintenance item on each node. That is merely a tedious task (perhaps to be delegated to a student) if there are only 8 nodes or 16 nodes. When there are hundreds of nodes, it's an impossible task. So there's software to automate distribution of configuration files throughout the cluster. This chapter discusses the C3 and Ganglia packages.

Now that the cluster is a well-administered, users will want to run programs on it. Unless the supported user community is so small that there's little chance of one job interfering with another there must be scheduling software. A small group of supported users will benefit from restart, monitoring and notification services, a larger group will demand these services. The subject of Chapter Eleven is OpenPBS and the Maui scheduler.

After actual jobs start running, input/output becomes a concern. So Chapter Twelve discusses parallel file systems. The emphasis is on PVFS, the Parallel Virtual File System.

Part IV, which comprises Chapters 13, 14, 15, 16, 17, is an all too brief introduction to MPI-1. A beginner needs a more thorough discussion. But the basics of message passing are introduced, along with debugging, timing and profiling. For more than just getting started, one will want to read further. Fortunately, several good books are available; some are mentioned in the Appendix.

Now the trip from plans to computing is complete. As Eisenhower said, "Plans are worthless, but planning is essential." Does this book have the answer to all questions? No, of course not and it doesn't make that claim. But it does provide a strategic plan, it tells one what questions to ask, and where to look for answers. The cluster administrator will want to read the instructions accompanying each package anyway; as packages are being updated. The instructions that a colleague has used recently might not apply to the version downloaded today. What this book does best is to organize the questions, so the prospective cluster administrator may be able to anticipate the answers with some prescience. While the plans may or may not be worthless, the planning remains essential.

If I had to assemble a cluster for a research group, I would certainly want this book at hand, before, during, and after the effort. This book will help with the planning of a cluster, and with the actual assembling of it. I would use this book to write check lists for each step, starting with an overall check list. Next I would follow each chapter to develop a more detailed check list to implement each step of the overall check list. The book is small enough for other individuals on the team to read it. Thus informed, they will be able to make helpful contributions to the process, each individual may anticipate to the needs of the jobs. With this book, an individual or team may be able to quickly converge on the best use of limited resources in the form of cluster computing. And that was the goal.

Dan Nagle
Purple Sage Software Inc.
USA