
Preface

mod_perl is an Apache module that builds the power of the Perl programming language directly into the Apache web server. With mod_perl, CGI scripts run as much as 50 times faster, and you can integrate databases with the server, write Apache modules in Perl, embed Perl code directly into Apache configuration files, and even use Perl in server-side includes. With mod_perl, Apache is not only a web server, it is a complete programming platform.

Getting mod_perl running is easy. Tweaking mod_perl and Apache for the best performance and reliability is much more difficult. This book is about building mod_perl, using it, programming with it, and optimizing it.

What You Need to Know

To use this book effectively, you need to be familiar with the day-to-day details of running a web server, and you need to know the Perl programming language. We expect that you have already programmed in the Perl programming language. Having written simple CGI scripts and having experience with setting up the Apache web server are definite pluses. Knowing your way around the operating system is another plus, but not a requirement.

Most examples in the book were done on the Linux operating system, but the examples and discussions should apply equally well to other operating systems, especially other Unix flavors. There is a dedicated section on installing mod_perl on Windows machines in Chapter 2.

Who This Book Is For

This book is not solely about mod_perl web development. It covers two main topics: server administration and programming under mod_perl.

At first, you might think these two topics are unrelated. But in real life, the programmer often needs to know how to build and maintain the server, and the administrator ends up doing much of the programming and tweaking himself.

In this book, administrators will learn:

- How to build and configure the server, with emphasis on improving server performance while keeping memory usage low.
- How to make sure the server is operating nonstop and, in case of malfunction, how to get it back online in no time.
- How to maximize performance by using multiple servers and additional tools such as proxies.
- How to choose the right machine and components. Often the most expensive machine isn't much faster than a cheaper one with more carefully chosen components.
- How to allow users to run custom scripts on a `mod_perl` server.

As for programmers, the good news is that you can be a capable `mod_perl` programmer while knowing very little about it. But most of us don't want to stop at being simply capable: we want to develop code that's robust, scalable, and blindingly fast. Here's a taste of the sort of things we cover in this book:

- In CGI, it's often hard to find what's wrong with a CGI script that returns a non-descriptive error message to your browser. You can try the `error_log` file, but with a complex script you have to use the `-d` switch and call the Perl debugger, which can be difficult for CGI scripts that can't be run from the shell. In Chapter 22, we'll show you how you can run the script in debug mode and control it.
- Alas, `mod_perl` is picky about coding style—for example, it doesn't like it when you forget to close a file after opening it. But if you ask nicely, it might enter a special mode where it will clean up for you. In Chapter 6, we'll show you how to keep `mod_perl` happy and keep the `error_log` file small.
- As you may already know, `mod_perl` is very fast. But with a little effort you can make it even faster. The idea is simple: the more memory (RAM) you have, the more requests you will be able to serve. However, you may be able to serve more requests using the same amount of RAM, thanks to memory sharing. For more information, see Chapter 10.
- With `mod_perl`, you never need to reinvent the wheel. If you need a so-called “shelf solution,” this book includes quite a few copy-and-paste scenarios to inspire you.
- Many programmers use `mod_perl` in conjunction with databases. We start with the simplest and most basic databases (flat files), continue to Database Management (DBM) implementations, and finally do an in-depth study of relational databases with SQL.

Of course, there's lots more, as you can tell from just the sheer size and weight of the book. This book is filled with gems of information that, taken together, provide a wealth of information on how to work effectively with `mod_perl`.

How This Book Is Organized

This book has four parts:

Part I: mod_perl Administration

Part I of this book focuses on the administration of `mod_perl`: getting it installed, configuring `mod_perl` and your web site as a whole, performing upgrades, and doing maintenance.

Chapter 1, *Introducing CGI and mod_perl*

Chapter 2, *Getting Started Fast*

Chapter 3, *Installing mod_perl*

Chapter 4, *mod_perl Configuration*

Chapter 5, *Web Server Control, Monitoring, Upgrade, and Maintenance*

Chapter 6, *Coding with mod_perl in Mind*

Part II: mod_perl Performance

Part II of the book is about how to use `mod_perl` to its fullest: it covers choosing a hardware platform, writing code, configuring the operating system, and configuring the Apache/`mod_perl` server itself.

Chapter 7, *Identifying Your Performance Problems*

Chapter 8, *Choosing a Platform for the Best Performance*

Chapter 9, *Essential Tools for Performance Tuning*

Chapter 10, *Improving Performance with Shared Memory and Proper Forking*

Chapter 11, *Tuning Performance by Tweaking Apache's Configuration*

Chapter 12, *Server Setup Strategies*

Chapter 13, *TMTOWTDI: Convenience and Habit Versus Performance*

Chapter 14, *Defensive Measures for Performance Enhancement*

Chapter 15, *Improving Performance Through Build Options*

Chapter 16, *HTTP Headers for Optimal Performance*

Part III: Databases and mod_perl

Part III tackles how to integrate databases with `mod_perl` in the most effective and efficient manner.

Chapter 17, *Databases Overview*

Chapter 18, *mod_perl Data-Sharing Techniques*

Chapter 19, *DBM and mod_perl*

Chapter 20, *Relational Databases and mod_perl*

Part IV: Debugging and Troubleshooting

Part IV of the book discusses how to uncover errors in mod_perl code and how to correct them.

Chapter 21, *Error Handling and Debugging*

Chapter 22, *Troubleshooting mod_perl*

Chapter 23, *Getting Help and Online Resources*

Part V, mod_perl 2.0

Part V covers the aspects of the new mod_perl 2.0.

Chapter 24, *mod_perl 2.0: Installation and Configuration*

Chapter 25, *Programming for mod_perl 2.0*

This book also contains the following useful appendixes:

Appendix A, *mod_perl Recipes*

Appendix B, *Apache Perl Modules*

Appendix C, *ISPs Providing mod_perl Services*

Appendix D, *The Template Toolkit*

Appendix E, *The AxKit XML Application Server*

Appendix F, *HTTP Status Codes*

Reference Sections

At the end of almost every chapter in this book, we include lists of resources that give further detail on relevant topics. The references are usually either URLs or book references. Unfortunately, URLs tend to change or disappear over time, so if you read this book some time after it has been published and some of the URLs aren't valid anymore, try to use a search engine such as Google to find the updated link. If you still can't find the listed resource, try to look it up in the Internet archive: <http://www.archive.org/>.

Many chapters refer to the Request For Comments documents (RFCs), which are mirrored by hundreds of Internet sites all around the world and are easy to find. A good starting point is <http://www.rfc-editor.org/>.

Filesystem Conventions

Throughout the book, unless mentioned otherwise, we assume that all the sources are downloaded and built in the directory `/home/stas/src/`. If you follow the same convention, you need only to replace *stas* with your username.

As you will learn in Chapter 12, most `mod_perl` users run one plain Apache server and one `mod_perl`-enabled Apache server on the same machine. We usually install these into the directories `/home/httpd/httpd_docs` and `/home/httpd/httpd_perl`, respectively.

Apache and Perl Versions

We have used `mod_perl` 1.26 and Apache 1.3.24 in most of the examples in this book. You should be able to reproduce all the presented examples with these or later versions of `mod_perl` and Apache.

We have tested all the examples with Perl 5.6.1. However, most of the examples should work the same under all Perl versions between 5.005_03 and 5.8.0.

At the time of this writing, Apache 2.0 is very young and `mod_perl` 2.0 is still in development. See Part V for information on `mod_perl` 2.0. While much of this book should apply to both `mod_perl` 1.x and `mod_perl` 2.0, the code has been tested only on `mod_perl` 1.26.

Typographic Conventions

The following typographic conventions are used in this book:

Italic

Used for filenames, command names, directory names, and Unix utilities. It is also used for email addresses, URLs, and new terms where they are defined.

Constant Width

Used for code examples and for function, method, variable, and module names.

Command Interpreter Program (Shell) Conventions

When you type a command and press the Enter key to execute this command, it's usually interpreted by some kind of command interpreter program, known as a *shell*. In this book we will use this term when we refer to a command interpreter program.

If you are running your web server on some Unix flavor, it is likely that you are using the C-style shell (e.g., *csh* or *tcsh*) or the Bourne-style shell (e.g., *sh*, *ksh*, or *bash*) for starting programs from the command line. In most examples in this book, it doesn't matter which shell program is used. In places where a different syntax should be used for different shell programs, we will say so.

The following command-line conventions are used in this book:

```
panic% command
```

`panic%` is a shell prompt when you are logged on as a non-*root* user, usually yourself.

`panic#` command

`panic#` is a shell prompt when you are logged on as *root*. It implies that you have to become a *root* user to run the command. One of the ways to switch to *root* mode is to execute the *su* utility and supply the *root* user password.

Installing Perl Modules

`mod_perl` and all the various Perl modules and helper utilities mentioned in this book are available via FTP and HTTP from any of the sites on the Comprehensive Perl Archive Network (CPAN) at <http://cpan.org/>. This is a list of several hundred public FTP and HTTP sites that mirror each others' contents on a regular basis.

You can search for and install Perl modules in two ways:

- Manually, by going to <http://search.cpan.org/>, finding the module, then downloading, building, and installing it. You can also browse the modules by categories or authors at <http://cpan.org/>.
- Automatically, by using Andreas Koenig's CPAN shell or (on MS Windows systems) the Perl Package Manager (PPM). These tools allow you to search for available modules and install them with a single command.

Manual Installation

When you download a module manually, it's best to find the one closest to you. You can find a list of CPAN mirrors at <http://mirror.cpan.org/>.

You can download the source packages with your browser, or, if you know the URL of the package, you can use any command tool to do that for you. In this book, we usually use the *lwp-download* perl script (which is bundled with the `libwww-perl` package, by Gisle Aas) as a client. You can use any other utility to download the files from the Internet.

Once you've downloaded the Perl module you want, you'll need to build and install it. Some modules are 100% Perl and can just be copied to the Perl library directory. Others contain some components written in C and need to be compiled.

Let's download the CPAN shell package, which we will use shortly:

```
panic% lwp-download http://www.cpan.org/authors/id/ANDK/CPAN-1.60.tar.gz
Saving to 'CPAN-1.60.tar.gz'...
115 KB received in 2 seconds (56.3 KB/sec)
```

Prerequisites Needed to Install Perl Modules on Windows

While Unix operating systems include standard utilities such as *tar*, *gzip*, and *make*, Windows systems don't. For this reason, you will have to go through some extra steps to ensure that you can install modules from the CPAN under Windows.

We assume here that you are using the ActivePerl distribution from ActiveState.

The first utility needed is *make*. On Windows, such a utility (called *nmake*) is distributed by Microsoft for free. You can download a self-extracting archive from <ftp://ftp.microsoft.com/Softlib/MSLFILES/nmake15.exe>. When you run this executable, you will have three files: *readme.txt*, *nmake.err*, and *nmake.exe*. Copy these files into a directory in your PATH,* such as C:\Windows\System, C:\Windows, or even C:\Perl\bin. You will now be able to replace any use of *make* in the examples in this book with *nmake*.

Some examples, and the use of *CPAN.pm*, also require command-line utilities such as *tar* or *gzip*. There are a number of projects that have ported such tools to Windows—for example, GnuWin32 (<http://gnuwin32.sourceforge.net/>) and UnixUtils (<http://unxutils.sourceforge.net/>). These toolkits allow you to use standard Unix utilities from your Windows command line.

Another option is Cygwin (<http://www.cygwin.com/>), which puts a Unix layer on top of Windows. This allows you to use many Unix-specific applications, but these must run from the Cygwin shell. If you use Cygwin, you should use the normal Unix steps discussed in this book, not any Windows-specific ones.

There is another downside of Windows: compilation tools aren't included. This means that some modules that use C extensions (e.g., *mod_perl*) can't be installed in the normal way, and you have to get precompiled distributions of them. In such cases, it is a good idea to follow the PPM instructions given later in this Preface, which should allow you to install binary versions of some of the modules discussed here.

Building a Perl Module

Building a Perl module and installing it is simple and usually painless. Perl modules are distributed as *gzipped tar* archives. You can unpack them like this:

```
panic% gunzip -c CPAN-1.60.tar.gz | tar xvf -
CPAN-1.60/
CPAN-1.60/lib/
CPAN-1.60/lib/CPAN/
CPAN-1.60/lib/CPAN/Nox.pm
CPAN-1.60/lib/CPAN/Admin.pm
CPAN-1.60/lib/CPAN/FirstTime.pm
CPAN-1.60/lib/Bundle/
```

* To see your PATH, run `echo %PATH%` from the command line.

```
CPAN-1.60/lib/Bundle/CPAN.pm
CPAN-1.60/lib/CPAN.pm
CPAN-1.60/TODO
CPAN-1.60/ChangeLog
CPAN-1.60/t/
CPAN-1.60/t/loadme.t
CPAN-1.60/t/vcmp.t
CPAN-1.60/MANIFEST
CPAN-1.60/Makefile.PL
CPAN-1.60/cpan
CPAN-1.60/README
```

Or, if you are using a GNU *tar* utility, you can unpack the package in one command:

```
panic% tar zxvf CPAN-1.59.tgz
```

Once the archive has been unpacked, you'll have to enter the newly created directory and issue the *perl Makefile.PL*, *make*, *make test*, and *make install* commands. Together, these will build, test, and install the module:

```
panic% cd CPAN-1.60
panic% perl Makefile.PL
Checking if your kit is complete...
Looks good
Writing Makefile for CPAN

panic% make
cp lib/CPAN/Nox.pm blib/lib/CPAN/Nox.pm
cp lib/Bundle/CPAN.pm blib/lib/Bundle/CPAN.pm
cp lib/CPAN/Admin.pm blib/lib/CPAN/Admin.pm
cp lib/CPAN.pm blib/lib/CPAN.pm
cp lib/CPAN/FirstTime.pm blib/lib/CPAN/FirstTime.pm
cp cpan blib/script/cpan
/usr/bin/perl -I/usr/lib/perl5/5.6.1/i386-linux
               -I/usr/lib/perl5/5.6.1 -MExtUtils::MakeMaker
               -e "MY->fixin(shift)" blib/script/cpan
Manifesting blib/man3/CPAN::Nox.3
Manifesting blib/man3/Bundle::CPAN.3
Manifesting blib/man3/CPAN::Admin.3
Manifesting blib/man3/CPAN.3
Manifesting blib/man3/CPAN::FirstTime.3

panic% make test
PERL_DL_NONLAZY=1 /usr/bin/perl -Iblib/arch -Iblib/lib
-I/usr/lib/perl5/5.6.1/i386-linux -I/usr/lib/perl5/5.6.1
-e 'use Test::Harness qw(&runtests $verbose);
    $verbose=0; runtests @ARGV;'
t/*.t
t/loadme.....ok
t/vcmp.....ok
All tests successful.
Files=2, Tests=31, 3 wallclock secs ( 1.22 cusr + 0.91 csys = 2.13 CPU)
```


Become *root* if you need to install the module on the whole system:

```
panic% su
<root password>

panic# make install
Installing /usr/lib/perl5/man/man3/CPAN::Nox.3
Installing /usr/lib/perl5/man/man3/Bundle::CPAN.3
Installing /usr/lib/perl5/man/man3/CPAN::Admin.3
Installing /usr/lib/perl5/man/man3/CPAN.3
Installing /usr/lib/perl5/man/man3/CPAN::FirstTime.3
Writing /usr/lib/perl5/5.6.1/i386-linux/auto/CPAN/.packlist
Appending installation info to /usr/lib/perl5/5.6.1/i386-linux/perllocal.pod
```

Using the CPAN Shell

A simpler way to do the same thing is to use Andreas Koenig's wonderful CPAN shell (recent Perl versions come bundled with this module). With it, you can download, build, and install Perl modules from a simple command-line shell. The following illustrates a typical session in which we install the `Apache::VMonitor` module:

```
panic% perl -MCPAN -e shell

cpan shell -- CPAN exploration and modules installation (v1.60)
ReadLine support enabled

cpan> install Apache::VMonitor
Running install for module Apache::VMonitor
Running make for S/ST/STAS/Apache-VMonitor-0.6.tar.gz
Fetching with LWP:
  http://cpan.org/authors/id/S/ST/STAS/Apache-VMonitor-0.6.tar.gz
Fetching with LWP:
  http://cpan.org/authors/id/S/ST/STAS/CHECKSUMS
Checksum for /root/.cpan/sources/authors/id/S/ST/STAS/Apache-VMonitor-0.6.tar.gz ok
Apache-VMonitor-0.6/
Apache-VMonitor-0.6/README
Apache-VMonitor-0.6/Makefile.PL
Apache-VMonitor-0.6/MANIFEST
Apache-VMonitor-0.6/CHANGES
Apache-VMonitor-0.6/VMonitor.pm

CPAN.pm: Going to build S/ST/STAS/Apache-VMonitor-0.6.tar.gz

Checking for Apache::Scoreboard...ok
Checking for GTop...ok
Checking for Time::HiRes...ok
Checking for mod_perl...ok
Checking if your kit is complete...
Looks good
Writing Makefile for Apache::VMonitor
cp VMonitor.pm blib/lib/Apache/VMonitor.pm
Manifesting blib/man3/Apache::VMonitor.3
  /usr/bin/make -- OK
```

```
Running make test
No tests defined for Apache::VMonitor extension.
/usr/bin/make test -- OK
Running make install
Installing /usr/lib/perl5/site_perl/5.6.1/Apache/VMonitor.pm
Installing /usr/lib/perl5/man/man3/Apache::VMonitor.3
Writing /usr/lib/perl5/site_perl/5.6.1/i386-linux/auto/Apache/VMonitor/.packlist
Appending installation info to /usr/lib/perl5/5.6.1/i386-linux/perllocal.pod
/usr/bin/make install UNINST=1 -- OK
```

```
cpan> exit
```

Notice that the CPAN shell fetches the *CHECKSUMS* file and verifies that the package hasn't been tampered with.

The latest CPAN module comes with a small utility called *cpan*, which you can use to start the CPAN shell:

```
panic% cpan

cpan shell -- CPAN exploration and modules installation (v1.60)
ReadLine support enabled
```

Using the Perl Package Manager

If you are using ActivePerl on Windows, or the Perl/Apache/mod_perl binary distribution discussed in Chapter 2, you will have access to a handy utility called *ppm*. This program installs Perl modules from archives separate from the CPAN that contain precompiled versions of certain modules.

For first-time configuration, do the following:

```
C:\> ppm
PPM interactive shell (2.1.5) - type 'help' for available commands.
PPM> set repository theoryx5 http://theoryx5.uwinnipeg.ca/cgi-bin/ppmserver?urn:/
PPMServer
PPM> set repository oi http://openinteract.sourceforge.net/ppmpackages/
PPM> set save
PPM> quit
C:\>
```

These steps will allow you to access a number of interesting packages not available from the ActiveState archive (including mod_perl). To see a list of these packages, type search in the PPM interactive shell, or visit <http://openinteract.sourceforge.net/ppmpackages/> and <http://theoryx5.uwinnipeg.ca/ppmpackages/>.

Now, when you want to install a module, issue the following commands:

```
C:\> ppm
PPM> install Some::Module
PPM> quit
C:\>
```

It's as easy as that! Alternatively, you might want to do it directly:

```
C:\> ppm install Some::Module
```

This will have the same effect.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly & Associates, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international/local)
(707) 829-0104 (fax)

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about books, conferences, Resource Centers, and the O'Reilly Network, see the O'Reilly web site at:

<http://www.oreilly.com>

The web page for this book lists errata, examples, or any additional information. You can access this page at:

<http://www.oreilly.com/catalog/pmodperl/>

This book also has a companion web site at *<http://www.modperlbook.org/>*. Here you will find all the source code for the code examples in this book. You will also find announcements, errata, supplementary examples, downloads, and links to other sources of information about Apache, Perl, and Apache module development.

Acknowledgments

Many people have contributed to this book over the long period while it was in the works.

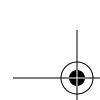
First things first. This book wouldn't exist without Doug MacEachern, creator of `mod_perl`. Doug's preliminary overview of `mod_perl` 2.0 was used as the basis of Chapters 24 and 25.

We're also greatly indebted to many people who contributed chapters or appendixes to this book. Andreas Koenig contributed Chapter 16, with helpful corrections, additions, and comments from Ask Björn Hansen, Frank D. Cringle, Mark Kennedy, Doug MacEachern, Tom Hukins, and Wham Bang. Matt Sergeant contributed Appendix E, with helpful comments from Robin Berjon. Andy Wardley contributed Appendix D.

We cannot thank enough the following reviewers, who have reviewed huge portions of the book (or the whole book) and provided good advice: Andreas Koenig, Ged Haywood, Gunther Birznies, Issac Goldstand, Mark Summerfield, Paul Wilt, Per Einar Ellefsen, Philippe M. Chiasson, and Robin Berjon. Thank you, guys. Without you, this book wouldn't be nearly as useful as it is now.

The following people also contributed much to the book: Aaron Johnson, Ask Björn Hansen, Brian Ingerson, David Landgren, Doug MacEachern, Ed Philips, Geoff Young, Pat Eyler, Perrin Harkins, Philippe Bruhat, Rafael Garcia-Suarez, Stéphane Payrard, Tatsuhiko Miyagawa, and Ken Williams. Thank you all for taking time to improve the book.

Since the book is loosely based on the `mod_perl` guide, we must acknowledge the following people who have indirectly contributed to the book by helping with the guide (about 200 names!): Aaron Johnson, Ajay Shah, Alexander Farber, Andreas J. Koenig, Andreas Piesk, Andrei A. Voropaev, Andrew Ford, Andrew McNaughton, Anthony D. Ettinger, Artur Zambrzycki, Ask Björn Hansen, Barrie Slaymaker, Bill Moseley, Boris Zentner, Brian Moseley, Carl Hansen, Chad K. Lewis, Chris Nokleberg, Chris Winters, Christof Damian, Christophe Dupre, Cliff Rayman, Craig, Daniel Bohling, Daniel Koch, Daniel W. Burke, Darren Chamberlain, Dave Hodgkinson, Dave Rolsky, David Harris, David Huggins-Daines, David Landgren, David Mitchell, DeWitt Clinton, Dean Fitz, Doug Bagley, Doug Kyle, Doug MacEachern, Drew Taylor, Ed Park, Ed Phillips, Edmund Mergl, Edwin Pratomo, Eric Cholet, Eric Strovink, Evan A. Zacks, Ewan Edwards, Frank Cringle, Frank Schoeters, Garr Updegraff, Ged Haywood, Geoff Crawshaw, Geoffrey S. Young, Gerald Richter, Gerd Knops, Glenn, Greg Cope, Greg Stark, Gunther Birznies, Hailei Dai, Henrique Pantarotto, Honza Pazdziora, Howard Jones, Hunter Monroe, Ilya Obshadko, Ime Smits, Issac Goldstand, James Furness, James G. Smith, James W. Walden, Jan Peter Hecking, Jason Bodnar, Jason Rhineland, Jauder Ho, Jay J, Jean-Louis Guenego, Jeff Chan, Jeff Rowe, Jeffrey W. Baker, Jens Heunemann, Jie Gao, Joao Fonseca, Joe Schaefer, Joe Slag, John Armstrong, John Deighan, John Hyland, John Milton, John Walker, Jon Orwant, Jonathan Peterson, Joshua Chamas, Karl Olson, Kavitha, Kees Vonk, Ken Williams, Kenny Gatdula, Kevin Murphy, Kevin Swope, Lance Cleveland, Larry Leszczynski, Leslie Mikesell, Lincoln Stein, Louis Semprini, Lupe Christoph, Mads Toftum, Marc Lehmann, Marcel Grunauer, Mark Mills, Mark Summerfield, Marko van der Puil, Marshall Dudley, Matt Sergeant, Matthew Darwin, Michael Blakeley, Michael Finke, Michael G. Schwern, Michael Hall, Michael Rendell, Michael Schout, Michele Beltrame, Mike Depot, Mike Fletcher, Mike MacKenzie, Mike Miller, Nancy Lin, Nathan Torkington, Nathan Vonnahme, Neil Conway, Nick Tonkin, Oleg Bartunov, Owen Williams, Pascal Eeftinck, Patrick, Paul Buder, Paul Cotter, Pavel Shmidt, Per Einar Ellefsen, Perrin Harkins, Peter Galbavy, Peter Haworth, Peter J. Schoenster, Peter Skov, Philip Jacob, Philip Newton, Radu Greab, Rafael Garcia-Suarez, Ralf Engelschall, Randal L. Schwartz, Randy Harmon, Randy Kobes, Rauznitz Balazs,



Rex Staples, Rich Bowen, Richard A. Wells, Richard Chen, Richard Dice, Richard More, Rick Myers, Robert Mathews, Robin Berjon, Rodger Donaldson, Ron Pero, Roy Nasser, Salve J. Nilsen, Scott Fagg, Scott Holdren, Sean Dague, Shane Nay, Stephane Benoit, Stephen Judd, Steve Fink, Steve Reppucci, Steve Willer, Surat Singh Bhati, Terry West, Thomas Klausner, Tim Bunce, Tim Noll, Todd Finney, Tom Brown, Tom Christiansen, Tom Hughes, Tom Mornini, Tuomas Salo, Tzvetan Stoyanov, Ulrich Neumerkel, Ulrich Pfeifer, Vivek Khera, Ward Vandewege, Wesley Darlington, Will Trillich, Yann Kerhervé, and Yann Ramin. Thank you all!



