

Success Stories

Success reports from people using mod_perl

Last modified Mon Nov 23 03:31:07 2009 GMT

Table of Contents:

- 1. Performance raised from 1.5 banner per second to over 20 banners per second, 10 million banners a week without a problem
- 2. Allakhazam's Magical Realm
- 3. BBC
- 4. BSat
- 5. Internal Call Center Database
- 6. Computer Aided Teaching system at Mathematics Department at the University of Western Australia
- 7. ColbyChem: a free web server for ISIS/Host
- 8. dslreports.com: million pageviews per day
- 9. EDDS Tax Management System for Canadian CCRA
- 10. mod_perl deployment at EToys
- 11. iAgora - Study, Travel, Work Abroad - Connecting Internationals
- 12. Performance increase of around 1100% compared to ASP
- 13. moviesdatabase.com or imdb.com
- 14. mod_perl 2.0 drives isoldmyhouse.com
- 15. Christmas Lights Webcam
- 16. Gay personals system
- 17. Running email through mod_perl 2.0
- 18. Mod_perl Uber Alles
- 19. Forced to improve the quality
- 20. Rent.com runs mod_perl
- 21. Students astronomy site
- 22. Non-web use for Apache/mod_perl: SMS app
- 23. mod_perl 2.0
- 24. Move from ActiveWare PerlScript on IIS4 to Apache and modperl improved performance by factor 60

- 25. mod_perl contact management system for Fortune-500 pharmaceutical giant
- 26. Major Teachers Union Launches New Site on mod_perl
- 27. Mod_perl 2 used to generate aggregated search results and contextually targeted advertisements.
- 28. Microsoft Network, 1 million hits per week through modperl
- 29. Large real-time stock exchange game
- 30. News agency uses mod_perl for their online system with over 6.5 million stories
- 31. bivio Investment Club Accounting, Taxes, and more
- 32. DateZilla.com
- 33. mod_perl 2.0 at the biggest Japanese employment site
- 34. modperl usage in financial institutions
- 35. Klassno.by a belarussian auction site
- 36. Modperl at the world's largest discount commodities trading firm.
- 37. 277M page views in Jan 2002
- 38. Red Hat's use of mod_perl
- 39. TERMIUMplus trilingual database

If you have a success story to share please submit it to the modperl mailing list. Please include the following information (using plain text, no html please):

URL:
Title:
Contact Person:
Traffic: (hits/day/month/whatever)
Success Story:

1 Performance raised from 1.5 banner per second to over 20 banners per second, 10 million banners a week without a problem

1 Performance raised from 1.5 banner per second to over 20 banners per second, 10 million banners a week without a problem

1.1 Marshall Dudley <mdudley (at) EXECONN.COM> exclaimed:

- Date: Fri, 6 Mar 1998 10:30:10 -0500

Lincoln Stein wrote:

```
>
> Hi All,
>
> I'm looking for more mod_perl success stories like the one that Jeff
> posted the other day. They will be used for vignettes in an
> introductory chapter of the book that Doug and I are writing. If you
> have a story you'd like to share (particularly one in which mod_perl
> "defeats" one of its competitors) could you mail it to me or post it
> to the list? For the vignettes we need some sort of identifying
> information, either along the lines of "a major Southwestern
> University" or "Kulturbox company of Berlin, Germany".
>
> Jeff, do you mind us using your story and identifying Texas A&M
> directly?
>
> Lincoln
```

You may not want to touch this one, but adultad.com contracted me to fix their adult banner exchange to where it could throw more than 1.5 banners a second. I put it under mod_perl, and it now tops out at slightly over 20 banners per second. It is now throwing approximately 10 Million banners a week solid without a problem. The banner exchange (both banner throwing/logging and click-thru redirection/logging) is running 100% under mod_perl.

Marshall

2 Allakhazam's Magical Realm

2.1 Andy Sharp <asharp <at> nector.com> exclaimed:

- Date: Wed Nov 07 21:20:11 2001
- Traffic: 1,800,000 Unique Page Loads per day
- URL: <http://everquest.allakhazam.com>, <http://camelot.allakhazam.com>, <http://eqbeastuary.allakhazam.com>.

Almost everything on the site runs in mod_perl. We have 4 systems running the site, one static server (PIII 450, Linux, Apache/mod_proxy). Two database servers (Dual P800, FreeBSD, Mysql) which are replicated, and the one mod_perl server (PIII 800, FreeBSD, Apache/mod_perl). The idea to use the proxy server to intercept any requests for text or images which was not dynamic came directly from the mod_perl guide (<http://perl.apache.org/docs/1.0/guide/>).

It's been a rough ride sometimes, as I've been in the process of learning the guts of Apache and more about perl than I ever thought I'd need to know. Since the site first started, I've migrated from a Module based system, to Apache::Registry (I wasn't writing good enough perl for the module based system to work well), and more recently have been migrating high volume scripts back to the Module/Handler based system.

That's been the true benefit of mod_perl in developing this site. It's been a learning process as we roll out a new application or area of the site, watching our hit load go up and up, and then spending hours looking for performance bottlenecks in code which was never intended to run as often as it does.

mod_perl gives us an incredibly fast development time. Sometimes, the speed of development does mean that lower quality code creeps into the production environment, but it allows us (me) to get things done which would take much much longer in another application environment. Perls "there are many ways to do it" extends into mod_perl, meaning that I can try something new quickly, and come back later to optimize it.

Among the features we have on the site:

Application layer security, based on a custom written Session tracking system. A recursively threaded forum system on every page, this system accounts for the bulk of the page views. It's also real time in terms of both comments being added, and ratings to the messages propagating through. User uploaded data through out the site, we allow players to track their characters, add meta information to database entries. Detailed web based administration system based on the Application security layer.

The speed of development of perl, coupled with the rich resources of CPAN, and the incredible power of mod_perl have made this site possible.

Running the same site in other technologies would have been possible, but would either require more hardware, or more time to develop.

3 BBC

3.1 Mark Hewis <mark.hewis (at) bbc.co.uk> exclaimed:

- Traffic: 500 req/sec
- URL: <http://bbc.co.uk/>

We have been running ModPerl to deliver dynamic applications as part of bbc.co.uk for the past 4 years. At present we have well over 50 applications (and growing) using Modperl and based on the Apache::Registry framework. These include community, message boards, games, voting, quizzes, searches and listings applications. It serves a sustained 500 application requests per second using a distributed architecture.

As we have adopted Apache 2.* on our head-end servers we will next be looking at how Modperl 2.* can help us in our application layer.

4 BSat

4.1 Mike Fletcher <lemur1 (at) MINDSPRING.COM> exclaimed:

- Date: Fri, 6 Mar 1998 13:01:58 -0500

At my former employer (Aaaahh . . . Sorry, just feels good to say that :), I rewrote a commercial interface to a defect tracking system. The original product was a bunch of Bourne shell scripts that all sourced one humongous configuration script. It took on the order of 10-12 seconds to return some pages (and some of those weren't even excuting any queries against the defect database) on a mostly idle SS20. Under mod_perl, that dropped to approximately 2-4 seconds for everything but really large queries (i.e. everything in the db).

5 Internal Call Center Database

5.1 Steven Lembark <lembark (at) wrkhors.com> exclaimed:

- Date: Sat, 15 Dec 2001 05:19:49 -0600

The URL is on an internal LAN for a company whose name I cannot use. The site gets up to a few hundred hits per second supporting a telephone call center database. My company was asked to develop a web front end onto a TB data warehouse. The existing system (carefully crafted in C) was so slow people couldn't get their work done (e.g., 45-minute query times). We re-did the back end and slapped an interface on it using mod_perl.

The first time the users saw it they asked for a "Stop" button like the existing system had so they could abort long-running queries. Then we went over where to put it with me running queries. They gave up on the idea because the data was returned too fast for them to hit a button.

Through 4+ weeks of User Acceptance Testing ("UAT") they asked for a few dozen changes in the reports. Few of them took longer than 20 minutes to implement. In several cases they got annoyed that the company email took longer to deliver the fix notice than make the change.

Using Perl we were also able to handle the database manglement software for tablespace and table creation, web site auth. and reporting code and most of the ETL process management code in one language. That also saved us quite a bit of work.

6 Computer Aided Teaching system at Mathematics Department at the University of Western Australia

6.1 Kevin Judd <kevin (at) MATHS.UWA.EDU.AU> exclaimed:

- Date: Mon, 9 Mar 1998 09:41:44 +0800
- URL: <http://CalMaeth.maths.uwa.edu.au>

At the Mathematics Department at the University of Western Australia I have a web-based computer aided teaching system using mod_perl. The students have individual weekly assignments in calculus, statistics, linear algebra with diagnostics and assessment built in. The system relieves academic staff of the burden of assignment marking and provides more personal interaction with students. The system requires database management and connection to a computer algebra engine. The transfer from a slow/unreliable/Macintosh/Hypercard/Mathematica system to a fast/reliable/web system took a couple of months and I had never programmed in perl before. The whole excersize was amazingly painless and it was entirely mod_perl's doing.

<http://CalMaeth.maths.uwa.edu.au>

Kevin

7 ColbyChem: a free web server for ISIS/Host

7.1 jwkuehne (at) colby.edu (John Kuehne) exclaimed:

- Date: Wed, 13 May 1998 10:23:31 -0400 (EDT)

Dear mod_perl gang,

The following is somewhat late in the "success story" thread of a few months ago, but I think there might be some interest for the database crowd. Below is a brief summary of a talk that I gave at a meeting in Philadelphia last week. Sponsored by Molecular Designs Limited (MDL), the meeting was attended by several hundred representatives of industry and government, and was concerned with the problems related to large molecular and reaction databases, and their use in combinatorial chemistry, drug discovery, etc. (These are databases consisting of molecular structures and their models, and reactions. A database user can pose an sql in the language of chemistry - molecular structures drawn with ISIS/Draw or ChemDraw - to find data that have substructure similarity, conformationally flexible similarity, reaction similarity, and much more. The structures, models, and reactions are displayed using MDL's chime plugin, itself based on RASMOL, which renders 'live' 3-D drawings that can be rotated and displayed in a number of ways from within the web page.)

Last November, Dr. Shattuck proposed that we build a reaction database of reaction mechanisms studied by Dr. Mundy and his colleagues, using MDL's reaction database software. Furthermore, it was his idea that we make this a web project open to all. Our first idea was to buy a license for MDL's ChemScape server, which links NetScape Enterprise server to MDL's database library. Unfortunately, the upgrade from our current MDL license to include ChemScape server was too expensive, not to mention NetScape Enterprise server.

I started working on a web server based on Apache and mod_perl that would act as a gateway to MDL's database software.

Although MDL's database server protocol is not public, they do provide a command line interface called hostcli, which has most of the functionality of the proprietary server. The use of hostcli is restricted to one machine, but within that machine one may run any number of hostcli processes.

ColbyChem, the project that I presented at the meeting, makes use of hostcli by opening it on a pseudoterminal for each database user. The novel aspect of ColbyChem is its use of the integrated Apache/perl server running in

single user (-X) mode for each database user.

Because perl is embedded in Apache, dynamic variables are retained between calls to the server children. Certain Apache packages use this to open a persistent database connection to industry standard databases such as Oracle, but this is not an option with proprietary interfaces, such as MDL's.

In order to adapt this to the idea of opening hostcli on a pty for each user, I run a dedicated Apache/perl daemon for each user, in single-mode (-X), on a separate port. That way, each Apache daemon caches the perl program and retains dynamic variables between calls. In essence, it becomes a new

7.1 jwkuehne (at) colby.edu (John Kuehne) exclaimed:

application, composed of Apache and perl, running under my program. The effect is similar to an X client. The browser is like the X server.

Entrance to ColbyChem is through a dedicated login daemon running on port 9000. Upon receiving a valid login name, the daemon forks an Apache/perl daemon on a port specified in a password-like file, and transfers the browser to this new port. Authentication, which is very important here, is carried out entirely on this new daemon. The user supplies a password. ColbyChem encrypts it and compares with the encrypted password assigned to the user. If successful, ColbyChem forks and execs hostcli on the pty. It then records the IP number and sends back a cookie for secondary authentication upon browser reconnect. The cookie is different for each session, is not based only on an easily guessed system parameters like time or checksums, and does not reveal, to within the limitations of crypt(), the original or encrypted password. My solution for the cookie is to take the password, which is secret, and permute it using rand() seeded by time. The permuted cleartext password is then encrypted and sent back as the cookie. Thus, even if one knew the permutation order and cookie, it would still be impossible to recover the original password.

ColbyChem presents side-by-side frames. The left frame contains a query builder and controls for hit-list logic and display. The right frame displays the data indented in the natural hierarchy of the database. Models, structures, and reactions are displayed using MDL's chime plugin.

Essentially, ColbyChem is nothing more than a graphical front-end for hostcli, written in 1200 lines of perl. The heart of ColbyChem is two routines, each a page of code. The first routine, rd2perl, translates an export file from hostcli into a perl data structure that has the hierarchy of the original database, i.e. it imports the database into perl. The second routine

recursively descends the branches of this structure until it reaches the tips, whereupon it prints out the data indented to reflect the database hierarchy.

MDL has just delivered an Oracle interface to its molecular and reaction databases. This opens the possibility of using established packages for persistent database connections that offer the flexibility of ChemScape server from within Apache/perl, without the novel hack of running dedicated daemons on separate ports for each user.

John Kuehne, Ph.D.
Information Technology Services
Colby College
4200 Mayflower Hill Drive
Waterville ME 04901

jwkuehne@colby.edu
207-872-3652

8 dslreports.com: million pageviews per day

8.1 Justin <jb (at) dslreports.com> exclaimed:

- Date: Fri, 15 Mar 2002 13:41:24 -0500
- Traffic: a million pageviews per day
- URL: <http://dslreports.com>

`http://dslreports.com` pages are entirely modperl, I do almost a million pageviews per day, and each page is from 20 to 200 sql statements.. and entirely produced by modperl, not templates or anything. most pages are produced in less than 200ms.. all the 150 internal forums as well (10000 posted messages per day and over 2 million messages online and searchable) are modperl. Everything is modperl :)

I use two modperl servers, one sql server, and one frontend (mod proxy) server.. all of them are dual cpu 500-1mhz cpus and from 1-4gig of memory..

This is all coded and administered now by me, part-time, while I do other stuff (like run the business).. I doubt it could be produced by any other tool, at least not without way more support staff and equipment.

regards
-Justin

9 EDDS Tax Management System for Canadian CCRA

9.1 Jay Lawrence <jay (at) lawrence.net> exclaimed:

- Date: Tue, 08 Jan 2002 14:08:27 -0800 (PST)

There are few things more sure in life than death and taxes. Ok, well I can think of one more - tax forms!

The Canada Customs and Revenue Agency (CCRA - our Federal tax collection agency - like the infamous IRS) has a collection of approximate 10,000 forms, guides and other publications that require management and control.

For the past 6 or 7 years these forms were managed using a proprietary database software that was costly to maintain and difficult to extend. As well the system was housed on aging SPARC processors. In order to meet on going and changing business requirements the system would need to be upgraded or replaced. It turns out that by using mod_perl, Linux and MySQL plus some contracting time the entire system was replaced for the cost of 1 years operation costs.

A customized document management system was created to meet the unique business requirements of the forms management group at CCRA. This includes document versioning and multiple document formats for each document name. The filing and classification methods are continuously evolving and so the addition and decomission of some metadata fields is necessary.

New documents are created by either starting a new version of an existing form or document or creating a new document header record. Then each document format, PDF, MS Word, Form Flow, etc., is uploaded using the file upload feature and the libapreq module to decode the uploaded files.

Since security is a concern - we cannot place access to this document collection on the internet. Instead we must report out files and then use rsync to move our data to a staging server. By using Perl we have been able to change from a weekly reporting cycle to a daily reporting cycle. As well, by using Perl we have been able to fix some really nasty decisions that were made 6 or 7 years ago when publishing to the web was an unknown process to most. Finally, by dumping the old software CCRA and its clients were able to chuck out all those modems and go via the web.

Perl is practical for extracting and reporting - the turnaround time and cost effectiveness of this project is a testimony to that claim!

10 mod_perl deployment at EToys

10.1 Perrin Harkins <perrin (at) elem.com> exclaimed:

- Traffic: 2.5 million+ page views/hour during Christmas 2000

In 1999, I joined the development team at the rapidly growing on-line toy retailer eToys.com. At the time, the site was running on a pretty standard platform of Perl CGI and a MySQL database. Traffic was increasing, and the servers were already straining under the load.

Our major task was to figure out how to get this system to scale large enough to handle the expected Christmas traffic. The toy business is all about seasonality, and the difference between the peak selling season and the rest of the year is enormous. The site had barely survived the previous Christmas, and the MySQL database didn't look like it could scale much further.

There was no time for a significant re-write of the existing code, so we looked to mod_perl's CGI acceleration capabilities to get us through. Using the Apache::PerlRun module and the persistent database connections provided by Apache::DBI, we were able to do a basic port to mod_perl and Oracle in time for Christmas, and combined with some new hardware we were ready to face the Christmas rush.

The peak traffic lasted for eight weeks, most of which were spent frantically fixing things or nervously waiting for something else to break. Nevertheless, we made it through. During that time we collected the following statistics:

- * 60 - 70,000 sessions/hour
- * 800,000 page views/hour
- * 7,000 orders/hour

According to Media Metrix, we were the third most heavily trafficked e-commerce site, right behind eBay and Amazon.

It was clear that we would need to do a re-design for 2000. We had reached the limits of the current system and needed to tackle some of the harder problems that we had been holding off on. Using mod_perl and a variety of other open source tools, we rebuilt the site to use a modular object-oriented design, improving flexibility as well as performance (see the Tutorials section for more information). Our capacity planning for Christmas 2000 was for three times the traffic

of the previous peak. That's what we tested to, and that's about what we got:

- * 200,000+ sessions/hour
- * 2.5 million+ page views/hour
- * 20,000+ orders/hour

See the Tutorials section for more information on how this feat was accomplished!

11 iAgora - Study, Travel, Work Abroad - Connecting Internationals

11.1 Roger Espel Lima <roger (at) iagora.net> exclaimed:

- Date: Fri, 16 Nov 2001 17:58:05 +0100
- Traffic: several million hits / month
- URL: <http://www.iagora.com/>

iAgora was started in mid-1998, as a community site for internationally minded people. After investigating the major existing web development systems, we chose to go with Linux, Apache and mod_perl. Three years later, we're very happy with this choice.

At iAgora we are constantly adding features and sections to our site, and refining the ones we have. For us it was very important to have a flexible platform, that would give us complete freedom in organizing our code, and customizing how the pages are generated.

We have found the combination of Linux, Apache and mod_perl to be:

* cost-effective

There are no software licences to pay, the programs are easy enough to install and configure, and many free support and middleware modules can be obtained from CPAN.

* stable

The running servers have had very few crashes, and generally not needed much maintenance. We have also found it very useful to be able to administer the servers remotely.

* flexible

Since mod_perl lets perl access low-level hooks within Apache, it is possible to have complete control over any aspect of its operation.

For instance, we found it easy and convenient to create virtual URLs, where some path elements were matched to database queries rather than directories on disk, while still basically serving an HTML file.

* adapted for large site creation

Mod_perl gives us complete control over how HTML and perl code interface to each other. By using a templating to the fullest extent, we minimize the amount of duplication both in HTML and perl. This also lets us have common navigation and design accross the

whole site, while separately maintaining the various form-based applications that make the site.

Contact Person:

- * Technical: Roger Espel Llima <roger@iagora.net>
- * Business: Philippe Negre <philippe@iagora.net>

12 Performance increase of around 1100% compared to ASP

12.1 Abigaël Duesberg <abi (at) idl-net.com> exclaimed:

- Date: Tue, 10 Nov 1998 23:16:31 +0100

Hi,

I saw that there were requests for success stories, so here is ours. We had to create 21 websites that basically had the same textual content (but different ads+clickthroughs, different designs, different access rights, etc...), that needed to sometimes remain unseen and act as gateways to other sites, and sometimes show up, with changing content and links according to user rights. Also, it had to answer search engine bots with different content using yet another database of robot user agents, as well as (coupled with LWP stuff) try to relate automatic posting to search engine databases to bots that came visiting (I know this isn't really good, but then, food is sometimes more important, :-() and to optimise meta tags, resubmission, etc...

It's all done in mod_perl, and in three days time it served a bit more than 4 million mod_perl hits, and submitted 180.000 forms to search engines. Everything's running on a 300mhz x86, with 128megs of ram. As a comparison, the early development tests were done using CGI on the same PC, and ASP on a more powerful one running IIS. We also tried using java servlets but the results were so desperate that I will not mention them here in respect for those people that use them. Given the time it took either for the CGI to be finished, or for the ASP to connect to it's SQL Server 6.5 to yield the right results or send the right page, we had been planning to buy 5 other PCs to get the job done with those solutions. Our benchmarks run with about 15.000 iterations of a series of calls to the servers that were under no other load show that ASP is hardly faster than CGI when database access is used (and then you have to take into account the fact that the ASP PC was fairly stronger, (I don't remember the CPU but it had 512megs of ram), but that mod_perl induces a performance increase of around 1100% !!! Also, it seems to be using less resources (though I haven't tested that fully), or using them for so short time lapses that one doesn't even notice.

The mod_perl development of the whole project was done by one person in less than three weeks (stress-testing included) , and it is running flawlessly.

I am looking for something stronger, but all that comes to mind is a deeply heart-felt "Thanks !".

Abigaël Duesberg
ASP - Lotus - LiveWire - Perl - Java

13 moviesdatabase.com or imdb.com

14 mod_perl 2.0 drives isoldmyhouse.com

14.1 "Chris Faust" <cfaust (at) doyougot.com> exclaimed:

- Date: Wed, 11 Jun 2003 11:21:16 -0400
- Traffic: 2+ million hits per day - 550,000 page views per day on average
- URL: <http://www.isoldmyhouse.com>

As consultants we were hired to repair, revamp and rebuild an online classifieds site in which a lot of cost and effort was placed in promoting the site and generating traffic but the site itself was based on a 3rd party product that simply could not handle the half million hits a day the site was getting.

Without a lot of effort the decision was made to build a custom solution from the ground up using Perl and Apache under Linux.

After completing the project and having some difficult issues with the current ISP we moved the entire site to an ISP that we have had a long term relationship with and who provides us with everything one would need to properly maintain such a project.

Little did we know that the second we moved to our new ISP it was like opening up the flood gates (long story relating to other ISP), overnight this CGI driven site went from a half million hits a day to a million and with it came a number of problems, a lot of which were unfixable without adding more hardware - there was simply far too much traffic coming through during the peak times of the day.

Having spent a week doing everything we could, optimizing everything possible it was clear that at best, we may be able to gain enough to just keep our heads above water.

Reluctantly we knew we had no choice but to give mod_perl a try, we really didn't think it was going to make that much of a difference but every little bit counted at this point.

We knew that it was going to be very difficult to setup apache and especially convert our code over - I mean after all I've heard as many stories of nightmare conversions as success stories.

After about the first week of pouring through the documentation and experimenting on our development server, I realized HOW WRONG I WAS..

Once we understood what was expected, conversion of the current code was less painful and a lot more interesting to do than some of the phone calls or meetings that led up to getting the contract for the project itself.

Once everything was done we could see instantly the improvement on our dev server, what we didn't know nor what we were prepared for was what would happen once this was running in production, I mean sure it was fast when there is only 2 of us on the machine, so was the old site.

14.1 "Chris Faust" <cfaust (at) doyougot.com> exclaimed:

What we saw after going live was one of those moments when you are just blown away, where you are sitting there saying "I see it but I just don't believe it".

At our best estimate we gained more than a 300% performance increase, during peak hours we were seeing load times of 20 - 30, processing going defunct etc. etc. prior to mod_perl.

Since the day we went live we haven't seen the machines even sweat, even the DB machine was impacted by the change in a positive way.

We are currently up over 2 million hits a day, the 1 million hits gained since going live with mod_perl has resulted in practically nothing (everything is still saying "Give me More!!!")

We'd like to think it was easy moving to mod_perl because we are such awesome coders, but of course the truth is it's due to the awesome documentation at <http://perl.apache.org>, the fantastic support of mod_perl in all those perl modules we have all come to depend on, the invaluable mailing lists and mailing list archives, and what I personally think is the coolest thing of all, Stas Bekman who never left me or anyone else I've seen on the mailing list hanging for any answer.

We have just completed a re-design of the site and have been up and running under Apache 2 and mod_perl 2 for about 6 months now with as few problems as anyone could ever hope to have.

Mod_perl is clearly the solution for high traffic sites, however because of our experience with mod_perl we have since done everything in it, from the simplest of form mailers to complex sites because in my eyes there is no reason not to do things the best possible way the first time around!

Thanks to Everyone on the Mod_perl Team

15 Christmas Lights Webcam

15.1 alek (at) komar.org exclaimed:

- Date: Fri May 27 13:21:18 CDT 2005
- URL: http://www.komar.org/cgi-bin/xmas_webcam

Alek Komarnitsky (<http://www.komar.org/>) used `mod_perl` for his christmas lights webcam that got worldwide attention as bazillions of surfers not only viewed his thousands of holiday lights, but turned them on and off via the Internet. In 2002 and 2003, he had used regular CGI/Perl, which choked under heavy load such as being Slashdotted. So for 2004, he converted to `mod_perl` and ApacheBench testing showed a 5 fold increase in the number of sustainable connections/second - "why didn't I do this earlier!" While Alek finally had to disclose that it was all a fun little hoax (<http://www.komar.org/xmas/hoax/>), the use of `mod_perl` was real and enabled a single 2.4 Xeon running Linux/Apache to continue providing dynamically generated content (complete with images that "appeared" to be changing) despite a ton of interest (<http://www.komar.org/faq/slashdot-effect/2004-christmas/>). While he doesn't know if Santa uses `mod_perl`, it was certainly helpful in Alek's attempt to provide some christmas cheer (http://www.komar.org/xmas/email_santa/2004-christmas/) to people around the world.

16 Gay personals system

16.1 Michael Bacarella <mbac (at) netgraft.com> exclaimed:

- Date: Sun, 6 Jan 2002 19:01:59 -0600
- URL: <http://m4m4sex.com>

`http://m4m4sex.com/` is a gay personals system that targets local gay communities in a number of cities. The site is written exclusively in `mod_perl` and uses `mysql` as a backend. The only purpose of the site is to ensure that people find sex partners as efficiently as possible.

I'm impressed with how well `mod_perl` and `mysql` have held up under considerable load as more cities were added and as more users joined. It also continued to please as we re-worked the site to allow users to switch between languages (4 at this point) on-the-fly.

It all happens on vanilla x86 hardware. Everything from user signups, account maintenance, online chat, secure payment options, renewing subscriptions, reports, etc.

Hooray for `mod_perl`!

17 Running email through mod_perl 2.0

17.1 Ken Simpson <ksimpson (at) ghpbjymdczr.mailchannels.com> exclaimed:

- Date: Mon, 6 Dec 2004 14:15:25 -0800
- Traffic: Low (in development)
- URL: <http://www.mailchannels.com/opensource/>

We have been using `mod_perl` successfully for several months now as a flexible email proxy -- we just wrapped `Net::Server::Mail` and with a few additional hacks and it worked. Matt Sergeant did the same thing with `qpsmtpd` and I have heard that the performance results were initially very promising (<http://msgs.securepoint.com/cgi-bin/get/qmail0411/120/1/1/1.html>).

More details of our hack (patches etc.) are at <http://www.mailchannels.com/opensource> and <http://search.cpan.org/dist/Apache-SMTP/lib/Apache/SMTP.pm>.

IMHO, using `mod_perl` as a general application server is a great idea. For us there really was no other viable alternative. We looked at POE, Sendmail's milter API, `Net::Server` and of course `qpsmtpd` but the reliability, portability, and scalability of Apache was what caused us to go through the effort of making our bits work on `mod_perl`.

To configure a mail server, it's just a matter of adding a `VirtualHost` section to the Apache configuration et voila. And as packages such as `mod_throttle` move over to Apache 2, we will gain the wonderment of a solid resource management tool for mail traffic. Joy!

Regards,
Ken

18 Mod_perl Uber Alles

18.1 Christopher A. Thompson <x4 (at) ROCKETMAIL.COM> exclaimed:

- Date: Wed, 15 Apr 1998 09:34:19 -0700

I have put up a site that's a true testament to mod_perl's power. (He said humbly).

<http://openscape.org> now contains the new site that I've been writing over the last 2 weeks.

The site is generated 100% dynamically by my module Obelisk.pm. Apache 1.2.6 and mod_perl 1.10 are used, and the module is inserted to run on <Location />. MySQL and DBD::MySQL provide the back end object store.

I keep all text, news items, and the like in the SQL database. at request time, the module takes the following steps.

```
$method = $r->method;  
$loc = $r->uri;
```

\$loc is then parsed out. Depending on the "page" requested the module generates a page based on several SQL calls, and prints the result back out. I pass args on to the subrequests this way too, such as <http://openscape.org/rnews/12> will read news item 12. It's all handled in the URL parsing. For the forms handling when you post a news item, I use CGI_Lite to grab things off POST. (If \$method is POST), since Apache::cant grab POST by default. I plan to implement my own POST handler, I just haven't gotten around to it.

You can post comments on news items, and those will be generated dynamically too. (a-la slashdot.org if you're familiar).

The amazing part of all this is twofold. First, it's all done in 427 lines of perl and 6 SQL tables. Slashdot is 2500 lines of code. Second, while I don't have any definitive numbers, this looks like it's going to scale very large. I've thrown a few large parallel requests at it (just simple LWP gets, in many parallel processes) and it doesn't seem to slow down. This box is just a P5/166 with 64megs RAM and Linux 2.0.31.

This all occurs with no CGI.pm, no Apache::Registry, no on disk content but the Obelisk.pm. I am so spoiled by this method that I don't think I can go back. I'm writing a Doc on the process and I'll have it up soon. I know I'm not the first person to do this, but the process

doesn't seem to be exceedingly documented. Oh, and Obelisk will be GPL'ed as soon as I gather it into a form that's fit for human consumption.

Thanks Doug and crew for mod_perl.

-Chris

===

```
-----  
Chris Thompson | I do not wish it to be misconstrued that  
ct@x4.net      | at no time was I not in total  
ct@cthompson.org | Disagreement --Anonymous  
-----
```

19 Forced to improve the quality

19.1 modus (at) PR.ES.TO exclaimed:

- Date: Fri, 6 Mar 1998 10:03:41 -0800

At the risk of this becoming a giant mod_perl lovefest, I'll second that. I've learned more about perl & apache in my dozen months or so of mod_perl than in my many years of work with apache & perl. mod_perl has definately forced me to improve the quality of my perl coding manyfold & taught me more than I ever thought I wanted to know about Apache.

On Fri, Mar 06, 1998 at 06:53:36PM +0100, Eric Cholet wrote:

```
> We've a mod_perl web site that allows subscribers to view news stories and
> news photographs from a major news agency. All content is received via a
> satellite link and users can view it in real time, as well as search
> through a huge archive database.
```

```
>
```

```
> What I like about mod_perl is its "double" reward: not only is it fast and
> efficient, but it has been an enlightening experience working with such an
> elegant tool and reading this list.
```

```
>
```

```
> ----
```

```
> Eric CHOLET - LOGILUNE
```

```
> email: cholet@logilune.com
```

```
> I am Pentium of Borg. Division is Futile. You will be approximated.
```

```
--
```

```
Patrick Michael Kane
```

```
<modus@pr.es.to>
```

20 Rent.com runs mod_perl

20.1 Eric Hammond <ehammond (at) rent.com> exclaimed:

- Date: Fri, 14 Dec 2001 14:27:41 -0800
- URL: <http://www.rent.com/>

Rent.com is a dynamic, database driven web site built on mod_perl.
Initial development took 3 months to replace an NT/IIS/ASP
implementation.

21 Students astronomy site

21.1 Smelly Belly <smiley (at) SEDS.ORG> exclaimed:

- Date: Fri, 6 Mar 1998 12:07:59 -0700

I run a web site for approximately 1200 students of introductory astronomy here at the U of Arizona. The server is an old Sun Sparc 1 and we use lots of perl CGI's to connect to a database on the backend and create custom pages. Before mod_perl, the site was unacceptable slow. Now, with the scripts re-written to use mod_perl, the dynamically created pages load faster than regular HTML files.

Mr. Guy Smiley

--

e-mail: (smiley at seds dot org)

website: (double u double u double u dot seds dot org slash tilde smiley)

phone: (five two zero three two one one nine six four)

--

"I root for a big comet or asteroid as a way of cleansing the planet."

George Carlin

22 Non-web use for Apache/mod_perl: SMS app

22.1 Bas A.Schulte <bschulte (at) zeelandnet.nl> exclaimed:

- Date: Fri, 22 Mar 2002 14:01:28 +0100

Preface

This is a story about how about I've used a combination of perl, Apache and mod_perl to create a component-based service architecture that implements a platform for building SMS applications. By reusing capabilities offered by Apache/mod_perl I saved a lot of time developing the system. The strong OO features of perl that I used enabled me to build a very flexible system as well to cope with future requirements. We had the platform in place in about 6 weeks, starting with absolutely nothing: no hardware, no development environment, no technology choices made beforehand.

Introduction

The purpose of the system to be developed was to provide a server platform on top of which arbitrary SMS (Short Message Service) applications can be developed quickly. It should be built using a stable and scalable architecture with room for future enhancements such as integrated billing and reporting options.

An SMS application can be characterized by subscribers sending text-based commands to the platform and have the platform dispatch to the right application instance. The application instance handles the command, executing whatever application-logic defined by that particular application, and usually generate one or more responses. It should also be possible that the platform initiates messages to subscribers as a result of a request sent by another subscriber as well as be able to generate messages based on timers

There also was a requirement to have the framework publish application-specific data in XML to allow customers to display this data on other media channels such as a website.

Connecting the platform to external entities for the transmission and reception of SMS messages such as SMSC's (SMS Centers distribute SMS messages to and from mobile subscribers) and SMS Gateways (smart front-end to one or more SMSC's unifying the method to reach subscribers from multiple telecom operators) should be flexible enough to be able to "plug-in" different protocols such as HTTP/SMTP/CIMD/SMPP as needed.

Component architecture

Early on in the project I decided to go for a distributed component architecture. Individual components should be deployable on multiple physical machines. This offers the required scalability and the ability to define a convenient security scheme by running components on segments of a network with differing outside visibility requirements.

As I started modelling this "world", I ended up with the following

components:

1. Application server

Within this application server, multiple instances of multiple SMS application instances should be running. The actual application-logic is running within this component. This component provides two external services:

- handleMessage(CommandRequest)

This service takes an instance of a CommandRequest object and runs the command in the appropriate application instance.

- handleTimer(Timer)

This services handles expiry of a timer set by the application-logic of an SMS application.

- getView

This service allows a client to retrieve application-defined views in XML.

2. Timer service

A persistent service that maintains timers set by application instances within the game application server and invokes the

handleTimer service of the game application services upon expiry of a timer.

External service offered:

- setTimer(Timer)

3. Virtual SMS gateway (VSMSC)

This component handles communication with the outside world (the external entities such as SMSC's and SMS gateways). This component is split up in 2 subcomponents, one that handles input from mobile subscribers and one that handles output to mobile subscribers. Each subcomponent provides one service:

- handleMessage(Message)

The input component receives requests from the outside world using pluggable subcomponents that handle protocol details, the output component transmits requests to the outside world using pluggable subcomponents that handle protocol details.

4. XML Views service

This component offers an HTTP interface to retrieve

application-specific views in XML. It uses customer-specific XSLT stylesheets to transform the XML data. This component is largely based on Matt Sergeant's AxKit. AxKit allow the source of your "document" to be delivered by your own provider class by subclassing off of AxKit::Provider. My provider class talks to the application server's getView service while AxKit performs its miracles with all kinds of transformation options.

Components Figure 1 System components

Apache/mod_perl as a component container

When thinking about how to implement all this I was tempted to look into doing it with some J2EE-thingy. However, there was this time-constraint as well as a constraint on available programmer-hands: I had one freelance programmer for 20 days and I had to arrange the whole physical part (get the hardware, a co-location site etc.). Then it struck me that this application server really looked like a vanilla regular mod_perl web application: receive request from user, process, send back reply. No html though, but Message objects that could be serialized/deserialized from text strings. There were of course some differences: the reply is not sent back inline (i.e. upon reception of a request via SMS, you can't "reply"; you have to create a new message and send that to the originator of the request) and there also was the timer service: I can't make Apache/mod_perl do work without having it received a user-initiated request.

The good thing was I've been doing Apache/mod_perl for some years now so I knew beforehand I could create a schedule acceptable from the business point of view that was also feasible based on experience with the technology.

So, for each component except the timer service, I defined separate Apache/mod_perl instances, one for the application server, one for the SMS output component, one for the SMS input component and one for the XML Views component.

Each instance defines a URL for each service that the component running in the instance provides.

Component communication

I took a shortcut here. I wanted to go for SOAP here as it seems a natural fit. It will allow me to move components to other languages (management and marketing still seems hung up on java) fairly easy. My personal experiences with SOAP on earlier projects weren't too good and I just couldn't fit playing with SOAP into my schedule. So I took my old friends LWP::UserAgent, HTTP::Request and Storable to handle this part (perl object instance -> Storable freeze -> HTTP post -> Storable thaw -> perl object instance).

The good thing is that this actually is a minor part of the whole system and I know I can put SOAP in easily when the need arises.

"Breaking the chain"

I did make one mistake in the beginning: all service calls were synchronous. The initial HTTP request would not return until after the whole chain of execution was done. With possibly long running actions in the server component, this was not good. I had to find a way to execute the actual code *after* closing the connection to the client. Luckily, Apache/mod_perl came to the rescue. It allows you to set a callback that executes after the HTTP responses are sent back to the client and after it closes the TCP/IP connection.

Result

We had the platform in place in about 6 weeks, starting with absolutely nothing: no hardware, no development environment, no technology choices made beforehand. Based on former experience, the decision to go with a LAMP architecture (Linux, Apache, MySQL, Perl) running on fairly cheap intel boxen was made quickly. MySQL was, and is, not on my wishlist, but the whole battle of moving Oracle in would have been both a time as well as a money killer, either of which we didn't have a lot of at the time.

Aside from having one production SMS application (a mobile SMS game), I've done a prototype SMS application on this platform to check if it really is easy to create new apps. It took me about 4 hours to implement a "SMS unix commandline" application: I can login to the application server using SMS, send Unix commands with my mobile phone and receive their output (make sure your command doesn't generate more than 160 characters though). The application also maintains state such as the working directory I'm in at any given time.

Performance is 'good enough' with the platform running on 2 fairly cheap Intel boxen, it handles 40 to 60 incoming request per second. As I haven't spent one second on optimization yet (anyone know the command to create an index in MySQL?), that number is fine for me. I did put 1 gigabyte in each machine though as the Apache child

processes eat up quite some memory.

Future enhancements and considerations

SOAP

I really want SOAP. It just seems to make sense to do so: it was invented for doing stuff like this and I like the concept of WSDL. It allows you to define the interface in an XML file so clients "know" what type of parameters the service needs as well as the return parameter types.

SOAP will also allow new components that are not perl. SOAP is available in a lot of languages and integration of the various SOAP implementations is getting better every day (see here).

Framework for service-based architecture

I'd like to extract the code that handles the communication between the components in the current system and create a generic framework that allows one to easily create an Apache/mod_perl-based components container. The available services would be registered in httpd.conf and there should be a service-discovery mechanism. On the client side, I'm thinking about something that makes it easy to create client-side stubs. Stay tuned...

Apache/mod_perl 2.0

This looks very promising to create generic components containers. It is very easy to create non-HTTP based services with Apache 2.0 with mod_perl's 2.0 support for writing protocol modules in perl. Also, the various multi-process models (most notably threading) available in Apache 2.0 should result in better performance or at least more choices as far as the process model is concerned.

Lamp

I'm still a little unsure about LAMP. Can we move to relatively cheap hardware and a free OS when we were used to (very) expensive HP, Sun or IBM hardware and get away with it? Personal experience and what I've read from others seems to indicate we can. Experience will tell, and if it breaks, moving the platform to either of the above three should be a no-brainer. We live in interesting times.

23 mod_perl 2.0

23.1 JT Smith ~ Plain Black exclaimed:

- Date: Sun, 18 Dec 2005
- URL: <http://www.spreadwebgui.com>

We've built WebGUI (www.spreadwebgui.com), our web application framework and content management system, on top of mod_perl. Before we began, in 2001, and recently when we started our next gen system we evaluated many different platforms on which to build WebGUI. Nothing even comes close to mod_perl for flexibility and power. Developing in Perl also means we're able prototype and deploy new features faster than we could in any other language.

Today there are more than 15,000 web sites that run on WebGUI, over half of which are either already running on mod_perl 2.0, or are in the process of transitioning to it. We're so confident about mod_perl 2.0's new capabilities and reliability that starting with WebGUI 6.8 we're requiring our users to use mod_perl 2.0 going forward.

24 Move from ActiveWare PerlScript on IIS4 to Apache and modperl improved performance by factor 60

24.1 Jeff Baker <jeff (at) GODZILLA.TAMU.EDU> exclaimed:

- Date: Tue, 3 Mar 1998 21:13:06 -0600

I'd like to share my recent success story. Over the last four days, students living on campus here at Texas A&M University have had to go through what is called "contract renewal," where they indicate whether or not they will continue to live on campus in the coming academic year. In the past, this has all been done very tediously with scantron forms and human-eye error correction. This year, the system was moved to the web. The code was user-proofed to prevent the usual mistakes, with the addition of some fancy authentication and session tracking mechanisms.

The system was originally written using ActiveWare PerlScript on IIS 4.0, but when I was done, it was glaringly obvious that it was far too slow. In only 14 days, we ported the code to Apache and mod_perl, with the same NT platform underneath. The performance (transactions/sec) was more than 60 times better!!!

The system went online Friday night, and in the course of its 4-day run, it served 400,000 documents, the bulk of which were generated on the fly. Ten thousand people used the system, and all went without a hitch.

Here's to mod_perl!
Jeffrey

25 mod_perl contact management system for Fortune-500 pharmaceutical giant

25.1 Rick Mangi <rmangi (at) TGIX.COM> exclaimed:

- Date: Fri, 6 Mar 1998 12:14:49 -0500

I have 2 success stories to share:

1. I'm finishing a web-based mod_perl/javascript (client side) contact management system with heavy Apache::DBI and Registry use. This system is for a "fortune-500 pharmaceutical (sp?) giant". It is replacing an unmanageable (their description) Lotus Domino application.

2. In production, a mod_perl server for gathering web traffic statistics for an up and coming web traffic reporting company. The mod_perl enhanced server gathers data from thousands of client and server based proxies around the world. Data is stored in Oracle using Apache::DBI. This replaced a poorly designed PHP server (poor choice using php in this scenario imho).

Rick

--

Rick Mangi	Tel: (212) 972-2030
Thaumatargix, Inc.	Fax: (212) 972-2003
317 Madison Avenue, Suite 1615	rmangi@tgix.com
New York, NY 10017	http://www.tgix.com
thau'ma-tur-gy, n. the working of miracles	
"Perl is a state of mind as much as it is a language grammar"	

26 Major Teachers Union Launches New Site on mod_perl

26.1 Perrin Harkins <perrin (at) elem.com> exclaimed:

- Date: Thur, 27 Jan 2005 14:27:41 -0800
- URL: <http://www.uft.org/>

The New York-based United Federation of Teachers, a 150,000-member teachers union for New York City's teachers in the city's five boroughs and 1,300 schools, has moved their site from a proprietary Windows-based system to a new system built on mod_perl and other open source technologies.

The site uses the (mod_perl-based) Krang content management system (<http://krang.sf.net/>) for publishing, enabling a wider range of people to contribute content with a consistent look-and-feel and a workflow system to handle the review process. The new site also features integration with the union's existing AS/400 systems, and further integration work is underway to bring more resources on-line for a wider audience.

My company, Plus Three (<http://plusthree.com/>) did the technical work on this project. Plus Three specializes in helping large non-profit and political organizations achieve their goals with open source software.

An article about the project was published in eWeek here:
<http://www.eweek.com/article2/0,1759,1754012,00.asp>

27 Mod_perl 2 used to generate aggregated search results and contextually targeted advertisements.

27.1 "Richard F. Rebel" <rrebel (at) whenu.com> exclaimed:

- Date: Mon, 23 Feb 2004 15:10:31 -0500
- Traffic: 12Million per day
- URL: <http://bidtxt.whenu.com/Search?templ=12%26kw=fitness%26num=3>

The two applications listed above are part of a suite of applications that provide bid text results to around 30 million client software installations around the globe. Roughly 12 million dynamically generated pages are created daily and the volume is ever increasing. We expect to increase to 50 million pages per day over the next 3 months.

Bid text results are provided by third parties via HTTP. For every impression, a connection to a remote HTTP server is initiated with query arguments derived from the initial request. Results are returned in XML format and normalized to match our internal DTD. The applications then perform selection based on various criteria, search and replacements. An XSL transform with a template is executed to create the resulting page which is then returned to the requesting client. The entire process usually executes in less than .3 seconds per request.

Due to the nature of the type of application and it's dependency on remote HTTP servers providing data, the wily nature of the internet, and the sheer volume of clients and requests, extensive use of timeouts and other protective measures is required.

With the exception of the use of XML::Sablotron, the entire application is written in perl running in mod_perl2 which is running in Apache2 with the worker MPM.

Mod_perl has proven to be reliable, fast, and easy to develop for, plus mod_perl also has an active and supportive user base.

Interesting tidbits:

Hardware: 5 Penguin Computing Relion 140, 2x2.8ghz Xeon, 2GB RAM
OS: Linux 2.4 kernel
Software: Apache2+mod_perl2, XML::Sablotron (libsablot)
Capacity: 2000 dyn pages per second (peak).

28 Microsoft Network, 1 million hits per week through modperl

28.1 Vivek Khera <khera (at) KCILINK.COM> exclaimed:

- Date: Fri, 6 Mar 1998 10:34:32 -0500

>>>> "LS" == Lincoln Stein <lstein@CSHL.ORG> writes:

LS> I'm looking for more mod_perl success stories like the one that Jeff
LS> posted the other day. They will be used for vignettes in an

The Microsoft Network promotion running to increase subscribership
located at <http://winamillion.msn.com/> is run on mod_perl. The
contest ends at the end of the month, so check it out before then ;-)

Anyhow, the system is currently pounding nearly 10 million hits per
week to the web pages, of which about 1 million go through mod_perl.
Each of those accesses runs through on average 3 SQL queries to a
MySQL database and 2 references to DB_File databases.

There is no way in heck it would have run without mod_perl. By the
way, this is using Squid in accelerator mode, as I described in the
tuning docs. Squid handles about 93% of the content (the static and
mostly static stuff).

v.

--

```

=====
Vivek Khera, Ph.D.                Khera Communications, Inc.
Internet: khera@kciLink.com       Rockville, MD          +1-301-258-8292
PGP/MIME spoken here             http://www.kciLink.com/home/khera/

```

29 Large real-time stock exchange game

29.1 Sven Neuhaus <Sven.Neuhaus (at) de.uu.net> exclaimed:

- Date: Fri, 05 Jun 1998 16:13:18 +0200
- URL: <http://www.wmboerse.de>

Hello,

another mod_perl success story:

Have a look at www.wmboerse.de - it's a german real-time stock exchange simulation game for the soccer world championship. Participation is free and there are some nice prizes to be won.

The technology used is Apache, mod_perl, DBI and DB::Adabas. The project is sponsored by Sun Microsystems (they are supplying a Sun Ultra Enterprise 450 with 3 CPUs @ 300Mhz and 1GByte RAM at the moment), UUNET Germany (bandwidth) and Software AG (Adabas-D database).

The server is a real beast. It's amazingly fast. The game is running since Sunday. At the moment, there are 2344 players, 183 of them have been active in the last 10 minutes. We are expecting a large increase in players as soon as national television reports about the game.

The load is at 0.80, there are 123 processes, still 400MB RAM free (we plugged in 512 MB today, previously the box had 512MB). We will increase the maximum number of child processes if we get close to the current limit (100).

Here's some data from the Apache status page:
Server uptime: 4 hours 10 minutes 58 seconds
Total accesses: 254671 - Total Traffic: 902.9 MB (!)
CPU Usage: u27.68 s10.98 cu2.03 cs.63 - .274% CPU load
16.9 requests/sec - 61.4 kB/second - 3717 B/request
18 requests currently being processed, 14 idle servers

Anyway, grab a browser and have a look. The project is a great success so far, and it couldn't have been done this easily and quickly without mod_perl and the other great free software out there.

Thanks and enjoy!

-Sven Neuhaus

30 News agency uses mod_perl for their online system with over 6.5 million stories

30.1 Eric Cholet <cholet (at) LOGILUNE.COM> exclaimed:

- Date: Fri, 14 May 1999 10:06:24 +0200
- URL: <http://www.afp-direct.com>

`http://www.afp-direct.com` hosts the Agence France-Presse's online database of news stories and photographs. Agence France-Presse is the world's third largest news agency. The online database, available through subscription, contains over 6.5 million stories and photographs in a full-text searchable database. The web site makes the most of `mod_perl` and its array of modules such as persistent connections to back-end servers and custom authentication.

31 bivio Investment Club Accounting, Taxes, and more

31.1 Rob Nagler <info <at> bivio.net> exclaimed:

- Date: Wed Nov 07 22:24:48 2001
- Traffic: 50,000 pages/day
- URL: <http://www.bivio.com>

bivio.com is a web-delivered application written entirely in perl which provides complete accounting, tax preparation, automatic downloads of broker transactions, message boards, file sharing, email aliases, and more. Apache/mod_perl on Linux has functioned incredibly reliably with +99% uptime.

Our declarative MVCF application framework (250 perl classes) is available under the Artistic License from <http://www.bivio.net> This includes a demo application <http://petshop.bivio.net> which is a more concise implementation of J2EE's Blueprint Architecture.

32 DateZilla.com

32.1 Dr. David R. Baird <dave (at) zerofive.co.uk> exclaimed:

- Date: 26 Nov 2007 11:13:24 -0500
- URL: <http://www.datezilla.com>

DateZilla.com is a directory of dating, matchmaking and matrimonials services and resources. The site is built on mod_perl and HTML::Mason, with a MySQL database. The speed of mod_perl and MySQL, combined with the sophisticated caching available in Mason, make the site very fast despite running on a bargain basement server. The management backend (content, ad serving, affiliate management, user stats) is also built on the mod_perl/Mason/MySQL combo, and allows a large site to be maintained by a small team from the comfort of their laptops.

33 mod_perl 2.0 at the biggest Japanese employment site

33.1 Batara Kesuma <bkesuma (at) ml.gaijinweb.com> exclaimed:

- Date: Wed, 11 Jun 2003 15:20:45 +0900
- Traffic: 4 million pageviews / month
- URL: <http://www.find-job.net>

Find Job! <http://www.find-job.net> is the biggest computer and Internet related employment site in Japan, and it is running on mod_perl completely. We have around 75,000 registered users and 10,000 registered companies. Our pageviews are around 4 million per month. Our whole system was running under plain CGI written in Perl, until recently we changed it to mod_perl ModPerl::PerlRun with HTML::Template templating system. On some scripts we saw up to 400% of speed improvement, and total CPU load average went down around 50%. The change from mod_cgi to mod_perl ModPerl::PerlRun itself was painless at all.

Right now we are working on our scripts, to make it mod_perl ModPerl::Registry compatible. We are going to run our site on mod_perl ModPerl::Registry once it is finished, and hope to see some more speed improvement.

34 modperl usage in financial institutions

34.1 Marcin Kasperski <Marcin.Kasperski <at> acn.waw.pl> exclaimed:

- Date: 04 Dec 2002 22:52:29 +0100
- URL: <http://www.inteligo.pl>

I am more than happy being now able to add the new nice reference. Please, patch my English where necessary...

Polish internet bank named Inteligo (<http://www.inteligo.pl>) recently migrated its transactional web service (the application used by the bank clients to make different kinds of payment orders, check account balances, etc) from a complicated Java-based solution to a modperl application. The application implements a web frontend to the business services implemented by the main bank system and accessed via the bank middleware. It is worth mentioning that the application constitutes the main access channel for the bank clients.

After a few days of production use the application is perceived to be much faster and lighter than the one previously used.

Two words of warning:

- inteligo 'informational' website (the pages visible under www.inteligo.pl) still use PHP and probably will continue to,
- don't treat this as easy 'perl is faster than Java' claim, there was a lot of design and programming work behind the new application...

Being a person who suggested using this technology and worked in a core development team I can admit that modperl fulfilled my performance expectations and allowed us to develop a complicated application fairly quickly.

Thanks to all the people who developed this nice piece of software and its documentation and to everyone who answered my and my colleagues questions during the project.

35 Klassno.by a belarussian auction site

35.1 Aleksandr Guidrevitch <alex (at) tera.by> exclaimed:

- Date: Sun Jul 23 15:37:06 PDT 2006
- URL: <http://klassno.by/>

Klassno.by is a belarussian auction site, written in mod_perl. It is designed primarily for the users from Belarus, and takes into account local specifics. It uses fulltext morphological analysis for indexing and searching.

In the hope of a brilliant future with tens queries per second and thousands of active lots, mod_perl was selected as a development platform in the very beginning. It uses Class::DBI and Template Toolkit, self-made MVC framework, and is written in OO Perl. It is running Apache 1.3/mod_perl, mysql5 with triggers and views where possible, and nginx (<http://sysoev.ru/nginx>) as a frontend http server.

36 Modperl at the world's largest discount commodities trading firm.

36.1 B. W. Fitzpatrick <fitz (at) onShore.com> exclaimed:

- Date: Fri, 6 Mar 1998 16:58:39 -0600

30000 customers looking at live quotes, dynamic charts and news.
"[...] More importantly, mod_perl allowed us to work the webserver and code around our design--not the other way around."

> I'm looking for more mod_perl success stories like the one that Jeff
> posted the other day. They will be used for vignettes in an
> introductory chapter of the book that Doug and I are writing. If you
> have a story you'd like to share (particularly one in which mod_perl
> "defeats" one of its competitors) could you mail it to me or post it
> to the list? For the vignettes we need some sort of identifying
> information, either along the lines of "a major Southwestern
> University" or "Kulturbox company of Berlin, Germany".

We just completed a website for Lind-Waldock & Co.
(<http://www.lind-waldock.com/>), the world's largest discount commodities trading firm. The site is to be used by their customers (>30,000) for live and delayed quotes, dynamic charts, and news pertaining to the futures industry, as well as access to their online order entry system. The site will take quite a beating once all of their customers transition to it from Lind's previous Windows application--plenty of live and delayed data is auto-refreshed.

Scenario: Client needed to develop a website that could authenticate off their existing customer database, and many links needed to be dynamically generated to reflect the level of service that the customer subscribed to (this info also kept in the database). The customer area had to be SSL enabled, fast, and support a slew of Perl scripts that the quote vendor had already written. And of course, they needed the whole thing yesterday.

They already had Netscape Enterprise Server and we investigated some NSAPI solutions but were terribly disappointed with what Netscape had to offer. We did some tests and decided to run with Stronghold and mod_perl. We wrote less than 10 lines of code to get the site authenticating off the database using Apache_DBI and just a few more to handle the dynamic URL generation.

We began analysis on Dec 1, and delivered the completed site on Mar 4--with 2 weeks off for Christmas, no less! Two days after release, the site is averaging about 3 requests a second--and that is certain

to grow exponentially as more customers make the switch from the old Windows application.

More importantly, mod_perl allowed us to work the webserver and code around our design--not the other way around.

-Fitz

Brian W. Fitzpatrick

fitz@onShore.com

<http://www.onShore.com/>

37 277M page views in Jan 2002

37.1 Jan Willamowius <jan (at) mobile.de> exclaimed:

- Date: Fri, 01 Mar 2002 12:17:08 +0100
- Traffic: in January 2002: 277 Mio page views, 10 Mio visits
- URL: <http://www.mobile.de>

All pages are dynamically created by a Linux cluster running Apache with mod_perl from a MySQL database (even though some pages look static).

151 Mio banner ads served from a small Linux cluster with Apache + mod_perl connecting to a MySQL database

mobile.de is one of the biggest online carmarkets in Germany. Its services are aimed at both professional car dealers and private buyers and sellers. Under the company's URL - www.mobile.de - individuals can offer and search for cars free of charge. Professional dealers pay a fee of EUR 101.24 per month. The fee entitles each dealer to list up to 250 vehicles in the database.

--Jan

38 Red Hat's use of mod_perl

38.1 Chip Turner <cturner (at) redhat.com> exclaimed:

- Date: 29 Nov 2002 23:30:24 -0500
- URL: <http://www.redhat.com/>

Red Hat's main website, www.redhat.com, is a mod_perl site (mainly Apache::ASP). Also, the Red Hat Network's website, rhn.redhat.com, is pure mod_perl, using a custom templating system (which I'll release one day, as soon as I find time..). It's actually a fairly complicated use of mod_perl for a web application, totalling around 60k lines of perl.

39 TERMIUMplus trilingual database

39.1 Jay Lawrence <jay (at) lawrence.net> exclaimed:

- Date: Tue, 08 Jan 2002 13:55:07 -0800 (PST)
- URL: <http://www.termium.com/>

TERMIUMplus (www.termium.com) is a trilingual application that allows translators and terminologists to search a collection of 1.5 million entries in English, French and Spanish. The system is freely available to any employee of the Canadian Federal government as well as by subscription to individuals and organizations outside. The terms and the user interface are both trilingual.

mod_perl plays an integral role in the success of this system. Because the server experiences significant amounts of traffic during the middle of the day efficient request handling is of paramount concern. It is not uncommon to be servicing over 100 concurrent requests at 2pm. Not only does the system perform very well but it is also very stable. I don't think our httpd's have ever crashed - and almost all requests are in the sub-second response range.

If great performance and stability were not enough - mod_perl (Perl) - has allowed us to provide a very easy to use and enjoyable interface to our database servers. The servers are actually on NT running a proprietary database software package. The database software is very good at performing both full text and exact term searches of the term data. However, the software interface to the database engines is weak and unusable at best. By using Perl to talk to the database server's HTTP interface we were able to extract the desired results data and then use Perl's power to reformat the results into something pleasing and tailored to the user's preferences. Because each record has over 100 fields and each field can have a number of sub components - I don't think the job would be doable in any other language than Perl! In addition to reformatting the output of the database we also employ some processing of search terms. This processing is unique to our data collection but helps increase recall by eliminating stopwords such as "a", "an", "le", "les", etc.

In addition to the fancy user interface TERMIUMplus also offers a server-to-server term translation service. This allows other search engines to offer on-the-fly term translation as part of their service. An excellent feature when dealing with a bi or tri-lingual document corpus. You are welcome to see this yourself by visiting:

<http://strategis.ic.gc.ca/engdoc/search.html>

Check on Bilingual search and try a word such as "turbofan". As a note, I am not aware of what software the Strategis search system was built with.

The entire system runs on a dual processor Sun 250 with 2G of RAM (We discovered how important lots of RAM is for this level of concurrent user activity) for the front end of the request processing. For the database queries we have 2 quad Xeon NT boxes which we divide between Extranet and Internet traffic. We will be replacing the Sun 250 with a

39.1 Jay Lawrence <jay (at) lawrence.net> exclaimed:

quad processor Sun 450 with 8G of RAM.

In addition to mod_perl we use MySQL as our user sessions database and intend to start replacing many functions of our proprietary back end database with functions developed using mod_perl and MySQL. Linux is our front-line development system and CVS is our versioning management system. We use CVS to then move our work on to a Sun staging system for pre-release testing and then finally rsync to push final code on to production servers. All of our code runs as well on Linux as it does on Solaris - with no modifications other than compile time options for the major packages of the application.

I feel that using mod_perl to build TERMIUMplus has allowed for the construction of a high quality service which is capable of handling a significant user load. It is very rare (never?) that we experienced any major problems with the Apache, mod_perl, and Perl portion of our system. Most of our operational difficulties are coming from our vendor supplied software at the database backend where daily server problems are experienced.

Software costs aside I wouldn't build this application using anything but mod_perl, Apache and MySQL!

Table of Contents:

Success Stories	1
Performance raised from 1.5 banner per second to over 20 banners per second, 10 million banners a week without a problem	4
1 Performance raised from 1.5 banner per second to over 20 banners per second, 10 million banners a week without a problem	4
1.1 Marshall Dudley <mdudley (at) EXECONN.COM> exclaimed:	5
Allakhazam's Magical Realm	6
2 Allakhazam's Magical Realm	6
2.1 Andy Sharp <asharp <at> nector.com> exclaimed:	7
BBC	8
3 BBC	8
3.1 Mark Hewis <mark.hewis (at) bbc.co.uk> exclaimed:	9
BSat	10
4 BSat	10
4.1 Mike Fletcher <lemur1 (at) MINDSPRING.COM> exclaimed:	11
Internal Call Center Database	12
5 Internal Call Center Database	12
5.1 Steven Lembark <lembark (at) wrkhors.com> exclaimed:	13
Computer Aided Teaching system at Mathematics Department at the University of Western Australia	14
6 Computer Aided Teaching system at Mathematics Department at the University of Western Australia	14
6.1 Kevin Judd <kevin (at) MATHS.UWA.EDU.AU> exclaimed:	15
ColbyChem: a free web server for ISIS/Host	16
7 ColbyChem: a free web server for ISIS/Host	16
7.1 jwkuehne (at) colby.edu (John Kuehne) exclaimed:	17
dslreports.com: million pageviews per day	19
8 dslreports.com: million pageviews per day	19
8.1 Justin <jb (at) dslreports.com> exclaimed:	20
EDDS Tax Management System for Canadian CCRA	21
9 EDDS Tax Management System for Canadian CCRA	21
9.1 Jay Lawrence <jay (at) lawrence.net> exclaimed:	22
mod_perl deployment at EToys	23
10 mod_perl deployment at EToys	23
10.1 Perrin Harkins <perrin (at) elem.com> exclaimed:	24
iAgora - Study, Travel, Work Abroad - Connecting Internationals	25
11 iAgora - Study, Travel, Work Abroad - Connecting Internationals	25
11.1 Roger Espel Lima <roger (at) iagora.net> exclaimed:	26
Performance increase of around 1100% compared to ASP	28
12 Performance increase of around 1100% compared to ASP	28
12.1 Abigaël Duesberg <abi (at) idl-net.com> exclaimed:	29
moviesdatabase.com or imdb.com	30
13 moviesdatabase.com or imdb.com	30
13.1 Rob Hartill <robh (at) IMDB.COM> exclaimed:	31

mod_perl 2.0 drives isoldmyhouse.com	32
14 mod_perl 2.0 drives isoldmyhouse.com	32
14.1 "Chris Faust" <cfaust (at) doyougot.com> exclaimed:	33
Christmas Lights Webcam	35
15 Christmas Lights Webcam	35
15.1 alek (at) komar.org exclaimed:	36
Gay personals system	37
16 Gay personals system	37
16.1 Michael Bacarella <mbac (at) netgraft.com> exclaimed:	38
Running email through mod_perl 2.0	39
17 Running email through mod_perl 2.0	39
17.1 Ken Simpson <ksimpson (at) ghpbjymdczr.mailchannels.com> exclaimed:	40
Mod_perl Uber Alles	41
18 Mod_perl Uber Alles	41
18.1 Christopher A. Thompson <x4 (at) ROCKETMAIL.COM> exclaimed:	42
Forced to improve the quality	44
19 Forced to improve the quality	44
19.1 modus (at) PR.ES.TO exclaimed:	45
Rent.com runs mod_perl	46
20 Rent.com runs mod_perl	46
20.1 Eric Hammond <ehammond (at) rent.com> exclaimed:	47
Students astronomy site	48
21 Students astronomy site	48
21.1 Smelly Belly <smiley (at) SEDS.ORG> exclaimed:	49
Non-web use for Apache/mod_perl: SMS app	50
22 Non-web use for Apache/mod_perl: SMS app	50
22.1 Bas A.Schulte <bschulte (at) zeelandnet.nl> exclaimed:	51
mod_perl 2.0	56
23 mod_perl 2.0	56
23.1 JT Smith ~ Plain Black exclaimed:	57
Move from ActiveWare PerlScript on IIS4 to Apache and modperl improved performance by factor 60	58
24 Move from ActiveWare PerlScript on IIS4 to Apache and modperl improved performance by factor 60	58
24.1 Jeff Baker <jeff (at) GODZILLA.TAMU.EDU> exclaimed:	59
mod_perl contact management system for Fortune-500 pharmaceutical giant	60
25 mod_perl contact management system for Fortune-500 pharmaceutical giant	60
25.1 Rick Mangi <rmangi (at) TGIX.COM> exclaimed:	61
Major Teachers Union Launches New Site on mod_perl	62
26 Major Teachers Union Launches New Site on mod_perl	62
26.1 Perrin Harkins <perrin (at) elem.com> exclaimed:	63
Mod_perl 2 used to generate aggregated search results and contextually targeted advertisements.	64
27 Mod_perl 2 used to generate aggregated search results and contextually targeted advertisements.	64
27.1 "Richard F. Rebel" <rrebel (at) whenu.com> exclaimed:	65
Microsoft Network, 1 million hits per week through modperl	66
28 Microsoft Network, 1 million hits per week through modperl	66
28.1 Vivek Khera <khera (at) KCILINK.COM> exclaimed:	67

Large real-time stock exchange game	68
29 Large real-time stock exchange game	68
29.1 Sven Neuhaus <Sven.Neuhaus (at) de.uu.net> exclaimed:	69
News agency uses mod_perl for their online system with over 6.5 million stories	70
30 News agency uses mod_perl for their online system with over 6.5 million stories	70
30.1 Eric Cholet <cholet (at) LOGILUNE.COM> exclaimed:	71
bivio Investment Club Accounting, Taxes, and more	72
31 bivio Investment Club Accounting, Taxes, and more	72
31.1 Rob Nagler <info <at> bivio.net> exclaimed:	73
DateZilla.com	74
32 DateZilla.com	74
32.1 Dr. David R. Baird <dave (at) zerofive.co.uk> exclaimed:	75
mod_perl 2.0 at the biggest Japanese employment site	76
33 mod_perl 2.0 at the biggest Japanese employment site	76
33.1 Batara Kesuma <bkesuma (at) ml.gaijinweb.com> exclaimed:	77
modperl usage in financial institutions	78
34 modperl usage in financial institutions	78
34.1 Marcin Kasperski <Marcin.Kasperski <at> acn.waw.pl> exclaimed:	79
Klassno.by a belarussian auction site	80
35 Klassno.by a belarussian auction site	80
35.1 Aleksandr Guidrevitch <alex (at) tera.by> exclaimed:	81
Modperl at the world's largest discount commodities trading firm.	82
36 Modperl at the world's largest discount commodities trading firm.	82
36.1 B. W. Fitzpatrick <fitz (at) onShore.com> exclaimed:	83
277M page views in Jan 2002	84
37 277M page views in Jan 2002	84
37.1 Jan Willamowius <jan (at) mobile.de> exclaimed:	85
Red Hat's use of mod_perl	86
38 Red Hat's use of mod_perl	86
38.1 Chip Turner <cturner (at) redhat.com> exclaimed:	87
TERMIUMplus trilingual database	88
39 TERMIUMplus trilingual database	88
39.1 Jay Lawrence <jay (at) lawrence.net> exclaimed:	89