UKUUG

news@UK

The Newsletter of UKUUG, the UK's Unix and Open Systems Users Group
Published electronically at http://www.ukuug.org/newsletter/

Volume 19, Number 2 **ISSN 0965-9412** **June 2010**

Contents

From the Secretariat

Jane Morrison

The Spring Conference that held was held towards the end of March in Manchester was well attended and successful. A description of the event can be found elsewhere in this issue.

The 3 days of Perl tutorials in April (Beginners, Intermediate and Advanced) also attracted a good number of delegates and our thanks go to Dave Cross for working with us again.

Since then we have been busy organising a full schedule of events for the rest of the year, including a Regular Expressions tutorial by Damian Conway (Wednesday 11th August in London), OpenTech 2010 (Saturday 11th September in London) and the UKUUG un-conference (Saturday 16th October in Birmingham). We are again pleased to be working in collaboration with Josette Garcia at O'Reilly to bring you various tutorials. Look out for further announcements.

The next Spring Conference will be held in Leeds in March 2011. The call for papers is enclosed with this mailing.

I am very pleased to confirm that Novell have renewed their Gold Sponsoring membership with UKUUG for a further 12 months. These sponsoring memberships do greatly assist us and allow us to keep event fees as low as possible.

The UKUUG Annual General Meeting will be held this year on Thursday 23rd September. The agenda and further details will be sent to you automatically during August. If you are interested in joining Council, please let me know.

Don't forget that HMRC allows UKUUG members to claim their subscription amounts for tax allowance purposes. See further details on our web site.

The next Newsletter will appear in September and has a copy date of 20th August. Submissions from members are very welcome.

Chairman's Report

Paul Waring

Recent and Upcoming Conferences

Our annual Spring conference in Manchester was a successful event as always, with a packed programme covering topics across all aspects of system administration, over a hundred delegates, and several opportunities to sample the varied cuisine on offer in the city. Thank you to all the speakers who gave up their time to discuss their latest work with us, and the numerous volunteers who helped the event run smoothly. Preparations for next year's Spring conference are already well underway, and the call for papers is included with this newsletter. I hope you will be able to join us in Leeds for another enjoyable three day tutorial and conference programme.

Looking forward to the rest of the year, we are planning to run our first one-day unconference in the autumn in Birmingham. This event will take place on a Saturday so that delegates can travel to the venue and back in one day and without having to book time off work. More details can be found within this issue, and I would like to encourage people to attend and get involved in an event which will be more relaxed and informal than our usual conferences.

Get Involved

UKUUG exists to serve its members and we are always on the lookout for people who are keen to get involved in any capacity, whether that be through volunteering to help with organising events, writing newsletter articles, or entirely new activities which haven't been tried before. We are particularly keen to get members involved in running next year's Spring conference, so if you would like to help out in any capacity please do get in touch.

Regular Expressions Tutorial with Damian Conway

This tutorial entitled *Understanding Regular Expressions (properly)* will be held on Wednesday 11th August 2010 at the Imperial Hotel, Russell Square, London WC1B 5BB, starting at 09:30 and ending at about 17:00.

Damian Conway is a well-known trainer and sought after speaker as well as the author of numerous well known software modules including: `Parse::RecDescnet` (a sophisticated parsing tool), `Class::Contract` (design-by-contract programming in Perl), `Lingua::EN::Inflect` (rule-based English transformations for text generation), `Class::Multimethods` (multiple dispatch polymorphism), `Test::Autoformat` (intelligent automatic reformatting of plaintext), `Switch` (perl's missing case statement), `NEXT` (resumptive method dispatch), `Filter::Simple` (perl-based source code manipulation), `Quantum::Superpositions` (auto-parallelization of serial code using a quantum mechanical metaphor), and `Lingua::Romana::perligata` (programming in Latin).

A well-known member of the international Perl community, Damian was the winner of the 1998, 1999 and 2000 Larry Wall Awards for Practical Utility. He currently runs an international IT training company - Thoughtstream – which provides programmer training throughout Europe, North America and Australia.

Most of his time is spent working with Larry Wall on the design of the new Perl 6 programming language and producing explanatory documents exploring Larry's design decisions.

Readers may be interested to read an interview with Damian Conway which recently appeared at **http://www.oreillygmt.co.uk/2010/05/damian-conway-on-perl-and-its-future.html**

To book a place on this tutorial, please go to **http://www.ukuug.org/events/regexps/**.

OpenTech 2010

Sam Smith

OpenTech 2010 will be held in London on 11th September 2010. Organised as in previous years by "UKUUG and friends", it is an informal, low cost, one-day conference on slightly different approaches to technology, politics and justice.

We hope to cover the following general areas and ideas:

- Justice: Civil, Social, Environmental, Criminal, Legal, other
- Community engagement
- Democracy 2.0
- Mashups, open data and uses
- Future of media and distribution
- Disaster politics and technology
- Highlights, lowlights and lessons learnt
- Mesofacts and long term thinking: big problems with massive opportunities

Talks and sessions have already been arranged on the following topics:

- Giving the Enlightenment Another Five Hundred Years
- Rewired State
- Where now for Open Video? from visionon.tv

- What's next after the General Election?
- mySociety and Project Fosbury
- For The Win: Games as a political space
- data.gov.uk, London DataStore
- Mozilla Drumbeat: Building Cool Stuff To Keep The Net Open
- Modern Perl
- Open Data & The Rewards of Failure
- People Power in Your Pocket

More details can be found at `http://www.ukuug.org/events/opentech2010/`.

Announcing FLOSS UK, 16th October 2010

John Pinner

Our first 'unconference' will take place on Saturday 16th October at the Birmingham and Midland Institute.

What is FLOSS UK 2010?

This is our first unconference, and is being held in collaboration with local FLOSS groups. It's a bit of a rebranding exercise, with the FLOSS UK 'brand' emphasising our involvement in 'Free, Libre and Open Source Software' rather than the narrower confines of traditional Unix implied by the UKUUG name.

What is the Birmingham and Midland Institute?

The BMI was founded in 1854 by Act of Parliament for "the Diffusion and Advancement of Science, Literature and Art amongst all Classes of Persons resident in Birmingham and the Midland Counties", and will give us a number of rooms of all sizes in which to hold the unconference. It's in the centre of the city, not far from the Conservatoire where Spring 2008 and Summer 2009 were held, and so is convenient for rail and coach stations as well as hotels at all price levels. It has a cafeteria, and is within a short walking distance of pubs, restaurants and takeaway food outlets. See `http://bmi.org.uk`.

What is an Unconference?

An unconference is a conference where what happens is organised by the delegates on the day. The event organisers have to arrange something, the main one being a venue, but the rest is down to the delegates. So all the hassle of talk submissions, review and scheduling is taken away. Typically:

- at the start of the day everyone gets up in turn and says who they are, what their interests are and what they'd like to do
- based on this people write proposals on PostIt notes and stick these on a board
- A moderator may read out the proposals in turn to gauge interest, and if sufficient the proposal will be put on a scheduling board
- Delegates may adjust the schedule to avoid clashes, etc.

The unconference starts...

Experience shows that the unconference format results in high quality sessions focussed on what delegstes want.

How will FLOSS UK 2010 work?

Because of the numbers we expect, following the above procedure at the beginning may be too lengthy, so we propose to invite groups to set up their own unconferences within the main event, so far we have:

- Perl: Birmingham Perlmongers
- PHP: WM PHP Group
- Python: PyCon UK Society
- Ruby: West Midlands Ruby User Group
- Linux: local LUGs

(and we're open to offers from other FLOSS groups!)

Groups will be able to make preliminary arrangements on the FLOSS UK wiki: `http://flossuk.org`.

We will be providing the basic infrastructure only – venue, wiki, wifi internet connectivity, projectors and screens, everything else is up to the delegates!

How much will it cost me?
We are aiming to keep the cost of attending this event down to the bare minimum. Everyone will be able to afford to come, and as it is a one-day event you will not need to incur accommodation costs..

USENIX events

We have received notice of the following USENIX events:

USENIX Federated Conferences Week

Events include:

- USENIX ATC '10: 2010 USENIX Annual Technical Conference Wednesday-Friday, June 23-June 25, 2010
- WebApps '10: USENIX Conference on Web Application Development Wednesday-Thursday, June 23-24, 2010
- WOSN 2010: 3rd Workshop on Online Social Networks Tuesday, June 22, 2010
- HotCloud '10: 2nd USENIX Workshop on Hot Topics in Cloud Computing Tuesday, June 22, 2010
- HotStorage '10: 2nd Workshop on Hot Topics in Storage and File Systems Tuesday, June 22, 2010
- FAST-OS Workshop Tuesday, June 22, 2010
- Tutorial on Google's Go: A Systems Programming Language Wednesday, June 23, 2010
- 1st USENIX Cloud Virtualization Summit Thursday, June 24, 2010
- Configuration Management Summit Thursday, June 24, 2010
- Tutorial on Configuration Management Solutions with Cfengine 3 Friday, June 25, 2010

Autumn 2010

- OSDI '10: 9th USENIX Symposium on Operating Systems Design and Implementation, October 4-6, 2010, Vancouver, BC, Canada
- LISA '10: 24th Large Installation System Administration Conference November 7-12, 2010, San Jose, CA

Next year

- FAST '11: 9th USENIX Conference on File and Storage Technologies February 15-18, 2011, San Jose, CA
- HotOS XIII: 13th Workshop on Hot Topics in Operating Systems May 8-10, 2011, Napa, CA

ODF Turns Five

ODF Alliance

A Look Back on the Journey to a Mature Open Standard Document Format

May 1, 2010, marks the fifth anniversary of the OpenDocument Format (OpenDocument v1.0 specification) approval as an OASIS Standard. Although the full history of ODF dates back well before this five year period as a recognized industry standard, this date marks the unofficial beginning of a campaign for document freedom that many people probably didn't expect would be so durable, or so significant.

ODF was created on the principles that interoperability and innovation were paramount, and that these are based on open standards. Not coincidentally, ODF's creation coincided with the growing support of open ICT architectures, which grew from the Web model where the standardization of HTML, an open, royalty-free standard, enabled the Web to be an open platform that enabled much innovation on top of it. The key was interoperability, or the ability of multiple parties to communicate electronically, without the need that they all run the same application software or operating system. Also critical to the development of ODF was the introduction of OpenOffice.org, the open source office suite that first implemented the format, and the rise of XML as a widely-supported foundational standard for describing structured data.

In the pre-ODF era, office productivity application innovation had all but halted. The ".doc" emerged as a de-facto – but proprietary – format, but it only worked well with one vendor's applications on one vendor's operating system. At that time, people – more importantly governments – didn't have much control of their own documents and there was little choice of applications.

However, over the last five years, ODF has led directly to the current environment where governments around the world increasingly understand and appreciate the value and utility of open formats, and more governments have made open standards-based solutions an essential feature of their eGovernment strategies and open ICT architectures. As a result, ODF is today an integral part of the global open movement.

Five years, twenty seven governments and myriad applications later, ODF is very well established – albeit continually a work in progress like any worthy format. Not only has ODF exceeded even the high expectations of many initial supporters, it can said without question that ODF has permanently altered the evolution of document formats for the better. Of course, the next five years are likely to be even more important, as more governments continue to implement their open frameworks. While we cannot gaze into a crystal ball to predict the next five years, following is a summary of ODF's accomplishments – including a time line of these milestones – of the last five years. Congratulations, and happy birthday, ODF:

Standardization – After the initial approval of ODF 1.0 in OASIS on May 1st 2005, ODF was submitted to ISO/IEC JTC1 for approval, where it was unanimously approved as an International Standard in May 2006. February 2007 saw the approval of OASIS ODF 1.1, bringing several accessibility improvements. ODF 1.2, with RDF semantic web metadata, a detailed spreadsheet formula language and other features is expected to be approved by OASIS later in 2010.

Government Adoptions – ODF has become widely recognized and increasingly accepted by public sector authorities who increasingly understand and appreciate the value and utility of open formats, while more governments make open standards-based solutions an essential feature of their eGovernment strategies. As a result, ODF is today an integral part of the global open movement. To date, Twenty seven governments (national and provincial) have adopted pro-ODF open standards policies. These adoptions have occurred through laws, executive decisions, interoperability frameworks, or policy initiatives requiring or recommending the use of ODF between and among government agencies and in their interactions with the public.

ODF Application Support – ODF is widely implemented and available on all major platforms,including traditional desktop editors, as well as web and mobile applications. At this time, all leading office suites support ODF, as do an increasing number of mobile devices. The rise of ODF and demand for open formats has coincided – not coincidentally in our view – with a proliferation of new office productivity applications not seen since the PC was first introduced in the 1980s.

Spring Conference report

Roger Whittaker

The 2010 Spring Conference and Tutorial was held at the end of March at Manchester Conference Centre.

The Tutorial took place on Tuesday 23rd March, and was an all-day session on SCons given by Russel Winder. Scons is a software construction tool: a highly flexible and configurable replacement for the traditional Make utility and the associated tools autoconf/automake. Before getting into the technical details of the real life usage of SCons, Russel motivated the discussion in a very interesting way by comparing the various build systems that are available, and pointing out that what is really needed is a domain specific language for this purpose. He outlined the reasons why, rather than re-inventing the wheel, choosing an already existing language is the best policy. In the case of SCons, that language is Python, and a SCons build script is "live" Python code, rather than simply a configuration file. We went on to use SCons to write build scripts for various example projects.

On Tuesday evening, antibodyMX very generously treated everyone who had been at the tutorial, and all those who had arrived during the afternoon to *free beer* at the Lass O'Gowrie pub close to the conference centre, after which many delegates went in search of a curry.

The conference proper started on Wednesday morning with Simon Wardley of Canonical and his talk on Cloud Computing entitled *Situation Normal – Everything must change*. This talk was both amusing and very interesting: by looking at historical parallels with other industries, and although the speaker managed to fit into his one-hour slot such ideas as Schumpeter's Creative Destruction, componentisation, the Jevons paradox and the Red Queen hypothesis, one came out of the session enlightened rather than confused, and with a much better understanding of Cloud Computing in its real context.

Sas Mihindu of the University of Salford then spoke on *Security for the Virtual Datacenter*.

For most of the rest of the conference there were two tracks. As always, there were times when I would have liked to be in two rooms at once. So what follows is a record of what I saw rather than of everything that happened at the conference.

Matt Trout spoke very enthusiastically on *21st Century Systems Perl* and gave some examples of the work that has been done recently on such modules as `Catalyst`, `DBIx::Class` and `Moose`. He talked about the move towards a standard Perl platform, and also mentioned `cpan2dist` and `autodie`.

Simon Wilkinson of Edinburgh University spoke on *Building an infrastructure for OS deployment*, a talk that was mostly about the problems of moving from CVS to git, and integrating the results with other tools such as gerrit, RT, LCFG and Prometheus.

After lunch, Theirry Carrez of Canonical described the technical aspects of the Ubuntu Enterprise Cloud, and the use of Eucalyptus in deploying instances into a private or hybrid cloud. In passing he touched on certain problems of working with the Eucalyptus code base, which though open source, is developed less openly than the bulk of the code developed by Canonical and the Ubuntu community.

Robert Watson of Cambrideg University and the FreeBSD project spoke on recent improvements to FreeBSD Jails. In this container-style virtualisation method, there is a single kernel and very little

overhead involved (about 500kB of memory per jail) in running many jails concurrently, each of which has its own network stack and global objects. This technology is now available in FreeBSD 8-STABLE and 9-CURRENT through the kernel configuration option `VIMAGE`.

The last two hours of Wednesday afternoon offered one track on the current state of Linux with Donald Forbes and John Pinner running back-to-back sessions on *The State of Linux installs* and a *Distro Roundup*. These were relatively informal sessions that produced some good discussion about the differences in approach and usability of different Linux distributions.

The conference dinner took place on Wednesday evening in a large upper room at a local Chinese restaurant.

On Thursday morning there were two talks which shared an interest in energy reduction. Mike Banahan described the work he does with schools and others through his company the Cutter Project, providing a thin client system delivering Linux or Windows desktops via Sunray devices. His demonstration of the user session following the user's smartcard was impressive, as were the claims for energy and administrative savings as compared to the traditional "PC on every desk" approach.

Jason Meers' talk was entitled *Targeting Desktop Virtualisation* and his main focus was on the use of thin clients for energy saving. He produced some interesting calculations of the huge amounts of energy and money that companies could save if they moved to a thin client model, with an estimate of 91% of energy used in running IT currently being wasted, and examples of a hospital saving £1.3 million for 1000 PCs over 5 years, and a school saving £64 per system over 3 years, even after capital costs have been taken into account. There is an article by Jason on Green IT elsewhere in this Newsletter.

I felt I had to attend the next talk in the main lecture room, simply to find out what its title meant. Patrick Debois and Julian Simpson's presentation was entitled *Hudson hit my Puppet with a Cucumber* and turned out to be a description of the integration of the Puppet system administration tool, the Cucumber domain specific language (based on Ruby) for test and behaviour driven development, and the Hudson tool for continuous building and testing of software projects and change integration.

John Collins spoke about his GNUBatch scheduler, a highly configurable replacement for `at` and `cron`. GNUBatch allows for the use of variables and arbitrarily complex conditions to programatically decide whether a job needs to be run. It has ncurses and GUI interfaces for setting up the jobs, and has recently been accepted as an official project by the GNU project.

Matt Barringer of Novell spoke about SUSE Studio, a web-based method of building custom software appliances (virtual machines, disk images and CD or DVD distributions) based on the openSUSE or SUSE Linux Enterprise distributions, with any additional custom packages desired.

A session of lightning talks followed. Two stand out as particularly memorable: Paul Waring's defence of PHP, and Russel Winder's talk on the multicore revolution in which he warned us that developments in processor design will require parallel processing algorithms to be used routinely and the operating systems that we currently know and love to be replaced. (For more in this general area, see the interview with Timothy Roscoe elsewhere in this Newsletter.)

This was a very successful conference. The venue and accommodation were good. Thanks are due to Paul Waring for the efficient organisation, particularly when a number of talks had to be arranged and re-arranged at short notice because more than one speaker had to cancel at a late stage, which had no effect on the quality of the conference because Paul worked hard to find good replacements.

Evening talk by Eng Lim Goh

Roger Whittaker

UKUUG was fortunate to be able to co-host (with GLLUG) an evening talk in mid-May by Eng Lim Goh, the CTO of Silicon Graphics International (SGI). This was held in a lecture room at University College, London and was well attended (the lecture room was full).

SGI specialise in very large high performance computing systems running Linux. These use multiple blades connected together with SGI's interconnect to form a single system image, a fact that Dr Goh demonstrated by logging in remotely to one such system and typing standard commands, including `free` which showed that the system had 5.2 TB of memory, and `cat /proc/cpuinfo` which showed 2048 cores.

Dr Goh referred to this system (and the larger ones that SGI produce with up to 16 TB of memory and 4096 cores) to the amusement of the audience as "just a PC", but such a description is accurate given that it presents a single system image and is installed with a standard Linux. He described the care with which SGI has fed back kernel modifications to the mainstream kernel which allow the use of its hardware while not compromising performance on less unusual hardware: good communications between the SGI team and the kernel community have meant that this has been a relatively straightforward process.

Some examples and demonstrations of the kind of applications that run on these supercomputers included McClaren Racing's design of Formula One cars, the design of the Speedo LZR Swim Suit, modelling of particle creation in black holes, and a computer model of the human heart which, even on this hardware requires two weeks runtime to calculate the motion of the heart during a single heart beat.

This was a most impressive talk, and the audience's appreciation and amazement were obvious at various points during the presentation. Thanks are due to John Hearns for arranging the talk and inviting UKUUG to co-host the event.

Lawrence Latif of the Inquirer was present and wrote an article on the talk:
http://www.theinquirer.net/inquirer/feature/1649635/sgi-advances-linux-hpc

---

Oggcamp – Liverpool, 1st and 2nd May

Roger Whittaker

During the first weekend in May, I attended the Oggcamp event at Liverpool, organised by the presenters of the Ubuntu UK Podcast and Linux Outlaws.

The event was excellent: it was organised on barcamp / unconference lines, but with good forward planning and a strong team of helpers.

Saturday started with an excellent keynote by Simon Phipps. He spoke mostly about digital freedom issues including surveillance, DRM, ACTA and the Digital Economy Bill. He urged people to make election candidates aware of the issues and why they matter.

For most of the time there were talks in more than one track. I attended the following during the rest of Saturday.

Nathan Dumont: *It's a Snake*, which was a talk on using Python for sysadmin tasks. Mark Johnson: a talk on SVG, including using of CSS and javascript with SVG. A talk on JPEG Forensics by Freakyclown. This was extremely interesting, with examples of how to prove that images have been doctored and how to obtain much more information than you might think possible from the EXIF metadata.

Then there was a panel discussion, with Dan Lynch, Adrian Bradshaw, Brad Pierce, Samantha Baer,

Simon Phipps and Chris Proctor. This was followed by a long raffle prize draw with some very worthwhile prizes.

On Sunday I attended the following talks.

Neil Wallace on GUI development, describing mostly how he decided to write his own dental practice software (openMolar) after problems with his supplier. He wrote the software in Python using PyQt4. Philip Herron on Crules, a new dynamic language that he is working on. Alan Bell on Vote Geek. Unfortunately I only caught the end of this, but graphics and underlying calculations on the site here are extremely interesting, with all possible outcomes in terms of popular votes illustrated on a triangular plot, with the region corresponding to a hung parliament clearly shown (some parts of that region corresponding to a potential political crisis: e.g. if Labour were to be third in the popular vote but first in number of seats.) Mark Johnson on *Software Patents explained to my Granddad.* He used the analogy of what would happen if chess moves were patentable: it was a persuasive analogy at first sight but possibly broke down slightly under discussion afterwards. Lorna Jane Mitchell on Subversion versus distributed version control systems.

Andy Stafford-Clark gave an MQTT workshop followed by his talk on *The Twittering House*, about the home automation and energy monitoring systems that he has set up so that he can monitor and control his house through the web, twitter and text messages. His work has been quite widely covered by the media. Particularly interesting and amusing were the stories of the mousetraps that report when they have caught a mouse, and the twitter feed that he set up to monitor the movement of the ferry that he takes to go to work (the ferry company ignored him when he tried to tell them about it but later embedded it in their web page without telling him, at which point he got their attention by playing an April fool on them).

Julian Todd spoke on *The Democracy supply chain* talking about some of the excellent work done by MySociety and related organisations, mentioning amongst other things the freedom of information request for a copy of the government spending COINS database, and the Where Does MY Money Go website.

Open Source and Free Deduplication

Peter Baer Galvin

Overview

There is certainly a lot of industry-wide interest in deduplication. Companies like Data Domain (now purchased by EMC) were founded on the premise that companies are willing to add complexity (e.g., appliances) in exchange for reducing the number of blocks used to store their data. For instance, deduplication seems to be a perfect addition to a backup facility. Consider the power of a device that can be a backup target: as it receives blocks of a backup stream, it throws out blocks it has previously stored, replacing that block with a pointer to the duplicate block.

A quick logic exercise of analyzing the types of data that are being backed up should convince you that there is quite a lot of duplication in general (operating system images, binaries, and repeated backups of the same user and application data) and that there is quite a huge potential for savings of disk space via deduplication. Virtual machine images are very deduplicatable, for example, while user and application files are less so. But even when data is not intrinsically duplicated, from the time it is created through its life-cycle there may end up being many, many copies of it. Consider that deduplication can be used as part of a business continuance (disaster recovery) scenario, in which the deduplicated on-disk backup is replicated to a second site. Only sending a given block once can be quite a savings in network bandwidth, as well as the obvious savings of only needing enough storage at the second site to hold one copy of each block.

It's an established pattern in IT that a new feature implemented first by a startup as part of a separate product goes on to become a standard component of other companies' products. That pattern certainly

seems true of Sun's implementation of deduplication as part of ZFS, included in an open source and free OpenSolaris distribution. The announcement of the integration of deduplication into ZFS and details of the implementation are available in a blog post by Jeff Bonwick, Sun's lead engineer on the project [1]. I would expect to see deduplication, just like snapshots, thin provisioning, compression, replication, and myriad other features, becoming a component of many storage devices. Thus, even if you are not interested in ZFS deduplication, you may be interested in how deduplication works and what problems it can solve.

How It Works

Deduplication works by "thumb-printing," in which an entity (either a file or a block, typically) is checksummed, resulting in a hash value. Hashing is very effective, providing in almost all cases a unique value for a unique entity. If the values match, the entities are probably the same, and the new entity is not stored; rather, a pointer is stored pointing to the already stored matching entity.

The checksumming occurs at one of two possible times, depending on the implementation. The checksum analysis is overhead, taking up CPU cycles and I/O cycles as an inbound block is checksummed, and that result is checked against the checksums of all other blocks currently stored. For that reason and others, some implementations perform deduplication in postprocessing. That is, they store all entities on write request, and then later compare the entities and remove duplicates. That is how NetApp deduplicates on their filers.

Alternately, the deduplication can occur at the time of writing, which is how Data Domain and ZFS deduplication works. This case takes a performance penalty at write time, but does not use up as much space as the post-processing method.

ZFS deduplication, as with other features of ZFS such as compression, only works on data written after the specific feature is enabled. If a lot of data already exists in a ZFS pool, there is no native way to have that deduplicated. Any new data will be deduplicated rather than written, but for the existing data to be deduplicated, that data would need to be copied to another pool (for example) or replicated to a ZFS file system with enabled deduplication.

In ZFS, once deduplication is enabled, the ZFS variable dedupratio shows how much effect deduplication is having on data in a ZFS pool. ZFS has file system checksumming enabled by default. Deduplication uses checksumming too, and enables a "stronger" checksum for the file system when enabled. ("Stronger" means less likely to have a hash collision. See Bonwick's blog for more details.) By default it uses sha256. As mentioned above, hashing almost always results in matches only when the hashed entities exactly match. But there is a long way between "almost" and "always." Hashes can have collisions in which hashes of two non-matching entities have the same values. In those cases, there could be corruption as one entity is thrown out and replaced by a pointer to the other entity, even though the entities are not the same. See the discussion below about the ZFS deduplication "verify" option for details on how to solve this problem within ZFS.

Getting to the Right Bits

Deduplication was integrated into OpenSolaris build 128. That takes a little explanation. Solaris is Sun's current commercial operating system. OpenSolaris has two flavors – the semiannual supportable release and the frequently updated developer release. The current supportable release is called 2009.06 and is available for download [2]. Also at that location is the SXCE latest build. That distribution is more like Solaris 10 – a big ol' DVD including all the bits of all the packages. OpenSolaris is the acknowledged future of Solaris, including a new package manager (more like Linux) and a live-CD image that can be booted for exploration and installed as the core release. To that core more packages can be added via the package manager.

For this example I started by downloading the 2009.06 OpenSolaris distribution. I then clicked on the desktop install icon to install OpenSolaris to my hard drive (in this case inside VMware Fusion on Mac OS X, but it can be installed anywhere good OSes can live). My system is now rebooted into 2009.06. The good news is that 2009.06 is a valid release to run for production use. You can pay for support on it, and important security fixes and patches are made available to those with a support contract. The

bad news is that deduplication is not available in that release. Rather, we need to point my installation of OpenSolaris at a package repository that contains the latest OpenSolaris developer release. Note that the developer release is not supported, and performing these next steps on OpenSolaris 2009.06 makes your system unsupported by Sun. But until an official OpenSolaris distribution ships that includes the deduplication code, this is the only way to get ZFS deduplication.

```
host1$ pfexec pkg set-publisher -O http://pkg.opensolaris.org/dev opensolaris.org
Refreshing catalog
Refreshing catalog 1/1 opensolaris.org
Caching catalogs ...
```

Now we tell OpenSolaris to update itself, creating a new boot environment in which the current packages are replaced by any newer packages:

```
Refreshing catalog
Refreshing catalog 1/1 opensolaris.org
Creating Plan ...
DOWNLOAD          PKGS    FILES     XFER (MB)
entire            0/690   0/21250   0.0/449.4
SUNW1394          1/690   1/21250   0.0/449.4
...

A clone of opensolaris-1 exists and has been updated and
activated. On the next boot the Boot Environment opensolaris-2
will be mounted on /.  Reboot when ready to switch to this updated
BE. You should review the release notes posted at
http://opensolaris.org/os/project/indiana/resources/relnotes/200906/x86/
before rebooting.
```

A few hundred megabytes of downloads later, OpenSolaris adds a new grub (on x86) boot entry as the default boot environment, pointing at the updated version. A reboot to that new environment brings up the latest OpenSolaris developer distribution, in this case build 129:

```
host1$ cat /etc/release
    OpenSolaris Development snv_129 X86
  Copyright 2009 Sun Microsystems, Inc. All Rights Reserved.
  Use is subject to license terms.
    Assembled 04 December 2009
```

At this point, ZFS deduplication is available in this system.

```
host1$ zfs get dedup rpool
NAME     PROPERTY   VALUE   SOURCE
rpool    dedup      off     default
```

Testing Deduplication

Now that we have the deduplication bits of OpenSolaris, let's try using them:

```
host1$ pfexec zfs set dedup=on
rpool cannot set property for 'rpool': pool and or dataset must be
upgraded to set this property or value
```

Hmm, the on-disk ZFS format is from the 2009.06 release. We need to upgrade it to gain access to the deduplication feature.

```
host1$ zpool upgrade
This system is currently running ZFS pool version 22.

The following pools are out of date and can be upgraded. After being upgraded,
these pools will no longer be accessible by older software versions.

VER      POOL
---      ------------
14       rpool
Use zpool upgrade -v for a list of available versions and
```

```
their associated features.

host1$ zpool upgrade -v
This system is currently running ZFS pool version 22.
The following versions are supported:
VER DESCRIPTION
--- --------------------------------------------------------
1   Initial ZFS version
2   Ditto blocks (replicated metadata)
3   Hot spares and double parity RAID-Z
4   zpool history
5   Compression using the gzip algorithm
6   bootfs pool property
7   Separate intent log devices
8   Delegated administration
9   refquota and refreservation properties
10  Cache devices
11  Improved scrub performance
12  Snapshot properties
13  snapused property
14  passthrough-x aclinherit
15  user/group space accounting
16  stmf property support
17  Triple-parity RAID-Z
18  Snapshot user holds
19  Log device removal
20  Compression using zle (zero-length encoding)
21  Deduplication
22  Received properties

For more information on a particular version, including supported releases, see
http://www.opensolaris.org/os/community/zfs/version/N,
where N is the version number.

host1$ pfexec zpool upgrade -a
This system is currently running ZFS pool version 22.
Successfully upgraded 'rpool'
```

Now we are ready to start using deduplication.

```
host1$ zfs get dedup rpool
NAME     PROPERTY        VALUE   SOURCE
rpool    dedup           off     default

host1$ pfexec zfs set dedup=on rpool
host1$ zfs get dedup rpool
NAME     PROPERTY        VALUE   SOURCE
rpool    dedup           on      local
host1$ zpool list rpool
NAME     SIZE    ALLOC   FREE    CAP     DEDUP   HEALTH  ALTROOT
rpool    19.9G   10.7G   9.19G   53%     1.00x   ONLINE  -
```

To test out the space savings of deduplication, let's start with a fresh zpool. I added another virtual disk to my OpenSolaris virtual machine. Now let's make a pool, turn on deduplication, copy the same file there multiple times, and observe the result:

```
host1$ pfexec zpool list
NAME     SIZE    ALLOC   FREE    CAP     DEDUP   HEALTH  ALTROOT
rpool    19.9G   10.8G   9.08G   54%     1.05x   ONLINE  -

host1$ pfexec zpool create test c7d1

host1$ zpool list
NAME     SIZE    ALLOC   FREE    CAP     DEDUP   HEALTH  ALTROOT
rpool    19.9G   10.8G   9.08G   54%     1.05x   ONLINE  -
test     19.9G   95.5K   19.9G   0%      1.00x   ONLINE  -

host1$ zfs get dedup test
```

```
NAME     PROPERTY    VALUE    SOURCE
test     dedup       off      default

host1$ pfexec zfs set dedup=on test

host1$ zfs get dedup test
NAME     PROPERTY    VALUE   SOURCE
test     dedup       on      local

host1$ df -kh /test
Filesystem  Size  Used  Avail   Use%  Mounted on
test        20G   21K   20G     1%    /test

host1$ ls -l /kernel/genunix
-rwxr-xr-x 1 root sys 3358912 2009-12-18 14:37 /kernel/genunix

host1$ pfexec cp /kernel/genunix /test/file1
host1$ pfexec cp /kernel/genunix /test/file2
host1$ pfexec cp /kernel/genunix /test/file3
host1$ pfexec cp /kernel/genunix /test/file4
host1$ pfexec cp /kernel/genunix /test/file5

host1$ df -kh /test
Filesystem       Size    Used    Avail   Use%    Mounted on
test             20G     14M     20G     1%      /test

host1$ zpool list test
NAME     SIZE    ALLOC   FREE    CAP     DEDUP   HEALTH  ALTROOT
test     19.9G   3.43M   19.9G   0%      4.00x   ONLINE  -
```

So, a file approximately 3MB in size and copied five times to a deduplicated ZFS pool seemingly takes up 14MB but in reality only uses 3.43MB (this space use must include the file, but also deduplication data structures and other metadata).

Also, according to PSARC (architecture plan) 557, deduplication also applies to replication, so in essence a deduplicated stream is used when replicating data [3]. Let's take a look. Fortunately, I have another (virtual) OpenSolaris system to use as a target of the replication (which we will call host2):

```
host2$ pfexec zpool create test c7d1

host2$ pfexec zfs set dedup=on test

host2$ zfs list test
NAME     USED    AVAIL   REFER   MOUNTPOINT
test     73.5K   19.6G   21K     /test
```

Now I take a snapshot on host1 (as that is the entity that can be replicated) and send it to host2:

```
host1$ pfexec zfs snapshot test@dedup1
host1$ pfexec zfs send -D test@dedup1 | ssh host2 pfexec /usr/sbin/zfs receive
-v test/backup@dedup1
Password:
receiving full stream of test@dedup1 into test/backup@dedup1
received 3.30MB stream in 1 seconds (3.30MB/sec)
```

On the receiving end, we find:

```
host2$ zfs list test
NAME     USED    AVAIL   REFER   MOUNTPOINT
test     16.4M   19.6G   21K     /test

host2$ zpool list test
NAME     SIZE    ALLOC   FREE    CAP     DEDUP   HEALTH  ALTROOT
test     19.9G   3.48M   19.9G   0%      5.00x   ONLINE  -
```

Sure enough,
3MB were sent as part of the replication, and although the receiving system thinks it has

16MB of date, it only has
3.4MB.

Unfortunately, the current `zfs send -D` functionality is only a subset of what is really needed. With `-D`, within that send, a given block is only sent once (and thus deduplicated). However, if additional duplicate blocks are written, executing the same `zfs send -D` again would send the same set of blocks again. There is no knowledge by ZFS of whether a block already exists at the destination of the send. If there was such knowledge, then `zfs send` would only transmit a given block once to a given target. In that case ZFS could become an even better replacement for backup tape: a ZFS system in production replicating to a ZFS system at a DR site, only sending blocks that the DR site has not seen before. Hopefully, such functionality is in the ZFS development pipeline.

Let's try that final experiment. First I'll create more copies of the file, then create another snapshot and send it to host2:

```
host1$ pfexec cp /kernel/genunix /test/file6
host1$ pfexec cp /kernel/genunix /test/file7
host1$ pfexec cp /kernel/genunix /test/file8
host1$ pfexec cp /kernel/genunix /test/file9

host1$ df -kh /test
Filesystem       Size    Used    Avail   Use%    Mounted on
test     20G     30M     20G     1%      /test

host1$ zpool list test
NAME     SIZE    ALLOC   FREE    CAP     DEDUP   HEALTH  ALTROOT
test     19.9G   3.45M   19.9G   0%      9.00x   ONLINE  -

host1$ pfexec zfs snapshot test@dedup2

host1$ pfexec zfs send -D test@dedup2 | ssh host2 pfexec /usr/sbin/zfs receive
-v test/backup2@dedup2
Password:
receiving full stream of test@dedup2 into test/backup2@dedup2
received 3.34MB stream in 1 seconds (3.34MB/sec)
```

Note that, even though host2 already had all the blocks it needed, one copy of the file was sent again because the sending host has no knowledge of what the receiving host already has stored. On the receiving side:

```
host2$ df -kh /test/backup
Filesystem       Size    Used    Avail   Use%  Mounted on
test/backup      20G     17M     20G     1%      /test/backup

host2$ df -kh /test/backup2
Filesystem       Size    Used    Avail   Use%  Mounted on
test/backup2     20G     30M     20G     1%    /test/backup2

host2$ zpool list test
NAME     SIZE    ALLOC   FREE    CAP   DEDUP   HEALTH   ALTROOT
test     19.9G   3.46M   19.9G   0%    14.00x  ONLINE   -
```

Even though host2 was sent the extraneous copy of the file, it discarded it, leaving it to store only one copy of the file.

Additional Analysis

No hash algorithm is perfect, in that two blocks only have the same hash if they are exactly the same. There is a very small chance that two blocks could have matching hashes even if they are not identical. By default ZFS trusts the hash values and will declare a block to be a duplicate if the hash matches. To increase safety you can set ZFS to do a byte-by-byte comparison of two blocks if the hashes match, to ensure that the blocks are identical before declaring them to be duplicates.

```
$ pfexec zfs set dedup=verify rpool
```

Of course this will negatively affect performance, using more CPU time per duplicate block.

On another performance note, the jury is still out on the performance impact of deduplication in ZFS. Theoretically, the increased overhead of checking for an existing matching hash whenever a block is about to be written may be counterbalanced by the saved write I/Os when there is a duplicate block that need not be written. But, in fact, it is too early to tell what the net result will be.

Deduplication can cause a bit of confusion about exactly what is using how much space. For example, the results of du can be grossly wrong if the data in the directories has been well deduplicated. Only `zpool list` is dedupe aware at this point. `df` and even other ZFS commands are not aware of deduplication and will not provide use information taking deduplication into account.

Conclusion

As it stands, ZFS deduplication is a powerful new feature. Once integrated into production-ready operating system releases and appliances, it could provide a breakthrough in low-cost data reduction and management. I plan to track that progress here, so stay tuned. For more details on the current state of ZFS deduplication, including bugs, features, and performance, please see the ZFS wiki [4].

Tidbits

As of this writing, Oracle has just acquired Sun Microsystems. Likely this will mean long-term changes with respect to which of Sun's products come to market and how Sun customers continue on as Oracle/Sun customers. At first blush (and first announcement), however, there seem to be very few changes for Sun customers. There were no massive layoff announcements (as some analysts had predicted), and so far, very little change in product direction. SPARC and x86 servers, storage arrays, Java, and Solaris all appear to have bright futures, as Oracle not only continues those products but increases the R&D budgets for most of them. At least in the immediate shadow of the merger, all seems to be well in Sun's product portfolio and direction. For more details on Sun under Oracle, including replays of the Oracle presentations about the purchase, have a look at **http://www.oracle.com/us/sun/**

References

[1] **http://blogs.sun.com/bonwick/entry/zfs_dedup**

[2] **http://www.opensolaris.com/get/index.jsp**

[3] **http://arc.opensolaris.org/caselog/PSARC/2009/557/20091013_lori.alt**

[4] **http://hub.opensolaris.org/bin/view/Community+Group+zfs/dedup**

The Green IT Agenda

Jason Meers

The green IT agenda

Information Technology has planted itself firmly at the heart of all modern businesses and has become inseparable. From the moment you wake up, until the moment you go to sleep billions of transistors, sensors, control systems, software applications and business rules shape your day. Gone are the days where a mechanic would diagnose a problem with your car by opening the car bonnet or hood to remove the distributor cap, or a bank manager would decide whether to grant a loan based on his or her local knowledge of an individual or their business. These manual processes have all been replaced by automated systems running almost entirely in tiny wafers of silicon and germanium. Some would say for the better, some would say for the worse. It depends if you are sat waiting for a breakdown truck because you don't have a laptop with an RS232 interface cable and a copy of the engine management diagnostic software, or the ability to reason with a live human being who can see through the credit scoring, risk weighting and loan-to-value ratios sent through from head office. The pessimist might

argue that we have taken a step back, whilst the optimist might argue that we have made progress, but few would now want to be without the luxuries we have been afforded in the modern era such as mobile phones, satellite navigation, online banking, Internet shopping and the ability to carry around 40,000 MP3s and a blockbuster movie in a device no bigger than a pack of cards. For some people the electronic, virtual world is far more "real" than the real world. Do you see yourself as "The immortal Dark Prince of Azeroth, raider of the Burning Legions", or "Keith the Regional Accounts Director of Firetop Mountain" who gets anxious when his Blackberry doesn't receive any emails for 20 minutes? Either way technology has become interwoven into our lives, even if you don't regard yourself as a geek.

The evolution of energy efficiency

Who would have thought ten years ago that energy efficiency and carbon reduction would be a big enough issue to justify the United Nations holding a conference about climate change? Just as manufacturers were starting to attach "Energy Star" logos to flat screen monitors to persuade us to replace our old CRT versions with less power-hungry models, Intel engineers were busy developing a range of portable office heaters they would later name the "Pentium 4". Nobody had seemed to notice how much our power requirements for completing the same day to day tasks had gone up. During the 90s it was much cooler to have your own personal paper clip to ask you if you were writing a letter ever time you started a new document than to worry about polar bears, ice caps and the fact that we might all look like David Dickinson from Bargain Hunt by 2020. So how did we get here? In 1993 the original Pentium processor consumed roughly 9 to 12 Watts of power, followed by the Pentium II in 1997 consuming roughly 16 to 43 Watts and in 1999 the Pentium III processor which ran between 16 and 42 Watts (depending on processor speed and manufacturing process). In 2000 and 2001 it all starts to go horribly wrong and we see the "Pentium 4" and the dual-core "Pentium 4 D" using up to 115 Watts and 130 Watts respectively. Once you add on all of the other associated chips on the motherboard, graphics cards, network cards, sound cards, memory and hard-drives the situation starts to look really grim. Based on this Pentium 4 model, each PC could be using up to 200 Watts of power, for up to 168 hours a week, which by modern figures would be creating roughly 992.41 Kg of carbon per year, and using 1747kWh of electricity per year, which in turn would cost £244.61 per year (assuming 14p per kWh and 568g carbon produced per kWh, as per UK L2 building regulations calculations). In addition to this for every 3 Watts of energy/heat produced and average air conditioning system will draw another 1Watt of electricity to remove it, giving us another 330kg of carbon and 582kWh of additional electricity at an additional cost of £81.53 per year.

What have we achieved so far

The good news is that we have the Intel Atom, the Via Nano, AMD's Cool-n-Quiet, Intel's SpeedStep and ACPI power management. The mistakes made with Intel's "Netburst" architecture which left the door open for AMD's faster, more efficient processors a few years ago have led to improvements in multi-core processors, processor efficiency and power saving technologies which allow modern processors to turn off or power-down sections of the processor when not in use. Just as Intel's "Netburst" architecture became the "Net-bust" architecture and was abandoned in 2008. "Moore's Law" about cramming more and more components into the same space ran out of steam when atoms and electrons refused to be squashed any further, just as the IT industry had started to loose its appetite for single-core processors that get increasingly hotter and power-hungry as speeds increase. You could argue that the physical limits to how far you can push a single piece of silicon came to a dead-end roughly five years before people realised it was the wrong road to be taking anyway. Faster is sooo "last year", out is the new up.

The carrot and the stick

So why would I want to go green anyway? Apart from the obvious savings in electricity and cooling re-using and re-purposing existing equipment is much more environmentally friendly than buying new low-energy computers. Recent figures from computer recycling charities show that 75% of all the energy a device will consume over its lifetime occurs during the actual manufacturing process. And the stick might be initially introduced to you as a CRC "Carbon Reduction Commitment" if you are

a public or private sector organisation that is metered half-hourly as 20,000 organisations currently are, or exceeded 6,000 megawatt hours of energy consumption during 2008 as 5,000 organisations currently are(including all subsidiaries). The fines, or penalties if you prefer to call them will run into millions of pounds for most of the organisations caught up in this legislation in 2010, however other legislation such as the FIT or "Feed In Tariff" will actually reward organisations who reduce their dependence on "dirty" energy, in favour of renewable energy sources such as solar and wind.

What is the business case for green IT?

- Reduced cost
- Reduced complexity
- Reduced Management overhead and server sprawl
- Extending the useful working life of existing equipment
- Taking the carrot
- Avoiding the stick

Where do we need to be in two years

- Re-using existing equipment instead of scrapping it
- Increasing the use of Terminal Services, Citrix/Xen and VDI
- Building our own internal clouds to consolidate compute and storage
- Developing legislation to encourage remote working
- Developing legislation to clarify the position on data security in the cloud

Where do we need to be in five years

- Investing in low power thin clients based on Atom, Nano and ARM processors
- Extending the cloud across other business processes or creating a hybrid
- Making part-time or full-time remote working the norm to reduce CO2 from cars
- Developing legislation to bring disenfranchised workers back into the economy
- Bringing into law clear legislation to offer a carrot for introducing Green-IT

Where do we need to be in ten years

- Treating the IT services as just another utility like electricity, gas and water
- Introducing the stick for organisations that do not show corporate responsibility
- Teaching our children how to be business solution architects, not computer users
- Growing up and taking responsibility for our excessive lifestyles (One Planet Living)

The real issues moving into the next decade

In one sentence: Resistance to change and self preservation. In the cold light of day the truth might simply be that less people need to be involved in the technical "nuts-and-bolts" of IT as cloud computing Everything-as-a-Service becomes more prevalent, and the softer skills such as business strategy, businesses process and business project management become more prominent. The love affair with "My Computer" may also be short lived as people demand ubiquitous access from everywhere, from anything. Our children may laugh at us when we tell them we had a quad core 3Ghz PC on each desk even though we could only type with two fingers at a time. The future of the computer or "Mobile Internet Device" may actually lie in the hands of the ARM processors found currently in many embedded and mobile devices. These ARM processors sport amazing battery life and power-consumption statistics and can already run Windows CE, Linux and some of the newer Internet-centric operating systems being developed. We may have to learn new skill-sets, and we may have to embrace new

technologies and hardware devices if we are to remain in a job. The good news here is for the many part-time workers, mothers with small children, physically disabled and other disenfranchised workers who need not be excluded from a new era built on cloud computing and home working.

Summary

In 500 BC the Greek philosopher Heraclitus wrote "Nothing endures but change", which in a twist of irony was then edited by Issac Asimov on Heraclitus's wiki to read: "The only constant is change, continuing change, inevitable change, that is the dominant factor in society today. No sensible decision can be made any longer without taking into account not only the world as it is, but the world as it will be."

This article was previously published in ***NCC IT Advisor magazine*** *(Spring 2010).*

HPC for Free

Andy Thomas

Have you ever been on a DLR train late at night as it slowly rumbles its way through the sprawling Canary Wharf area of London's Docklands? Or indeed, the financial district of any large city; gazing through the windows as the glass and steel towers glide by, revealing floor upon floor of desks, chairs, filing cabinets, potted plants – and hundreds of desktop computers. Stretching as far as the eye can see, their standby lights winking slowly in the gathering gloom, have you ever wondered about those computers? Has it ever struck you that they are idle for most of their lives, used only when the office lemmings come in and sit down to their daily grind? And has it ever occurred to you that right here is an enormous waste of raw computing power, when the same company might be spending huge sums of money running a traditional HPC cluster in their data centre?

This scenario is repeated in medium and large companies and organisations all over the world but it is surprisingly easy and very inexpensive to harness the untapped computing power of those idle desktop PCs to create a part-time HPC when the offices close. All it needs is a file server and some open source software; what if you could shut down all those desktop PCs and reboot them over the network into a diskless version of UNIX or Linux from a file server... Voila! you have created an instant HPC!

In this, the first of a two-part article, we look at how one internationally-respected mathematics research centre created its 'SCAN' HPC facility in this way and in the second part, to be published in the next issue, detailed implementation details and a FreeBSD diskless boot how-to will show you how to set up HPC facilities using your own or your employer's PCs and a fileserver for little or no cost.

In the beginning

When the Department of Mathematics at Imperial College London started looking at setting up HPC facilities for their research programmes in 2003, things looked decidedly grim. Faced with no data centre of their own, a very restricted budget (like most educational institutions) and a desire to start using a working facility as soon as possible, the shiny new Beowulf clusters emerging in the early 2000's were simply beyond their reach. But they did however have one asset – several large rooms containing new Windows PCs used exclusively for teaching. It was while discussing the HPC problem with a research mathematician, Dr Gunnar Pruessner, over an after-work drink one Friday evening that the solution suddenly hit us both, almost simultaneously – why not boot those Windows PCs into a diskless Linux during the night when the teaching PC rooms closed, using a simple remote server providing DHCP and TFTP network boot services as well as a nfs fileserver for the HPC user home directories? And the local hard disk on each PC would remain untouched – for reasons more political than technical, any interference with the Windows installations or the addition of other operating systems on the hard disks was not an option. Why had no-one thought of this before?

Early trials

Things moved very quickly – by the following Monday a working demonstration using an old PC

running SuSE Linux 7.3 as the boot server and another old PC fitted with a 3Com 3C905 network card with an on-board Lanworks boot ROM, as the client, linked to the server via an isolated private network (actually, a cross-over Ethernet patch lead), proved that this could be done. But we had a problem – the new PCs in the department had on-board network interfaces, and PXE had replaced the optional boot ROMS of the PCI bus network cards. After some unsuccessful attempts to boot a Linux image via PXE, Gunnar suggested using FreeBSD instead – as a UNIX 'power user', Gunnar had always used FreeBSD as his desktop operating system of choice and was very familiar with it so this seemed a natural solution. And FreeBSD 4.6 worked, literally out-of-the-box!

All thoughts of implementing the HPC on Linux were now abandoned – support for PXE in Linux at that time was poor whereas FreeBSD 'had it all', giving us a working system for very little effort and we were keen to press on and create a HPC facility ready for productive use. All the open source applications and packages of interest to us could be built for FreeBSD, the FreeBSD ports system is certainly easier to use than the much more involved process demanded by other *n*xs of building individual support libraries, include files, etc and then the packages themselves, and the FreeBSD kernel could be compiled with Linux ABI (Application Binary Support) so that commercial packages like Matlab, Mathemetica, Maple, etc could in future be run on this HPC.

Setting the BIOS of the desktop PCs to attempt to boot from the network via PXE first and then if unsuccessful, boot Windows from the local hard disk was all that was required; the FreeBSD server's dhcpd server was configured to provide a network boot image via its TFTP server as soon as the remote PC's network connection was established after being powered up or rebooted from Windows and gaining a DHCP lease from the server. On booting FreeBSD, the PC would then mount additional filesystems from the server such as the user home directories in `/export/home`, applications under `/usr/local` and additional scratch disks as `/bitbucket1`, `/bitbucket2`, etc. Bearing in mind the requirement that the PC local hard disk must not be touched, the swap disks were also mounted from the server via NFS which had the interesting possibility that we could in future implement a 'suspend-to-swap' scheme, where the machine state at shutdown is preserved in a named swap file, which is then opened by the same machine the next time the HPC cluster starts up, allowing the node to continue from where it stopped during the previous session.

The next challenge was to integrate this isolated client/server proof-of-concept demonstration into the college's existing network infrastructure, which is based on a central DHCP server that gets its network parameters for each requesting client from a central hosts database. This ensures, among other things, that only systems with MAC addresses known to the host database will get a DHCP lease and hence a functioning network connection. It also implies that only one DHCP server can exist in such an environment – we could not simply connect our local FreeBSD DHCP server to the network and expect things to work as they did in the test setup.

In collaboration with the networks department, the central database/DHCP system was enhanced to allow the DHCP server to request a network boot image from a remote server for PCs which were defined in the hosts database as belonging to a special class aptly named 'Beowulf'. Simply adding the new PCs in the Maths department to the 'Beowulf' class caused them to be fed with a FreeBSD netboot image immediately after obtaining a DHCP lease and unless the ESC key was pressed promptly, they would then go on to boot FreeBSD instead of Windows from the local hard disk. We were now able to boot Windows PCs on normal network connections in the teaching rooms into FreeBSD.

Going live, and some interesting problems

Tests with a few PCs booting into FreeBSD were successful and it was not long before we were booting all the PCs in a teaching room after it closed to students for the day and running them as a parallel compute cluster during the night. At about this time, the acronym SCAN meaning 'Super Computer At Night' (borrowed from the book 'Using MPI', published by the MIT Press) was adopted to describe the fledgling HPC. Word soon got around that we had an HPC of sorts and researchers started to use the experimental facility, writing programs in C and Fortran using both the standard MPI libraries and where such high level facilities were not needed, plain and fast TCP/IP sockets (an idea proposed and implemented by Gunnar).

Interest from the department grew and it was soon decided to make this a permanent service, available every weekday night, at weekends, during the Christmas and Easter holiday periods when the college was closed and during the long summer vacation after the students had gone home. This meant automating the system so that the PCs would shut down Windows when the teaching room closed, reboot into FreeBSD over the network and then early the following morning, shutdown FreeBSD and reboot back into Windows before the rooms re-opened. We also had to make sure that if someone rebooted Windows during a class, the system would not come back up running FreeBSD!

We had some problems here – when booting a PC at a time when the SCAN was not scheduled to be available, with the TFTP server disabled, some brands of PCs (e.g. Dell GX400) would give up trying to boot from PXE after 1 minute and boot Windows normally from the hard disk but other makes (HP d530) would try for up to 10 minutes unless the ESC key was pressed by a user. The PXE time-out was configurable in the BIOS on Dell PCs but was fixed on HP PCs and could not be changed and was rather disconcerting for Windows users. After trying a variety of solutions involving multiple network interfaces and virtual IP addresses on the server, enabling/disabling the TFTP daemon at different times, etc the solution finally adopted was to enable the TFTP server permanently but switch the TFTP root to serve either a FreeBSD boot image or alternatively, when Windows only booting was required, a PXELINUX boot loader which was configured to boot the PC system immediately from its local hard disk.

Using PXELINUX in this way puts control of the PCs firmly into the hands of the boot server – when any PC is powered on or reboots, it always accesses the FreeBSD server's TFTP server first and then boots whatever operating system is scheduled to be available at that time of day. This has some useful benefits - a few of the PCs at that time were dual-boot Windows XP and Red Hat Enterprise Linux 5 systems which defaulted to booting Windows from the hard disk but by using PXELINUX with a different configuration file, we could remotely reconfigure the systems to boot into Linux instead of Windows. Other operating systems could also be booted over the network from the comfort of the sysadmin's chair without having to visit the PCs or make any changes to their configuration.

Another problem we ran into was when 80 systems suddenly lost their network connections in the middle of the night – their DHCP leases had expired and not been renewed. The same thing happened the following night, and then again the night after that. It took some all-night stays in the college to get to the bottom of this but the cause was simple; the day before this started happening, the college's network department had changed the switches feeding the teaching room networks for a different make and model. These switches were doing something that no switch should normally do – decrementing a packet's TTL (time to live). As with most other BSD-derived DHCP clients, FreeBSD's DHCP request, discovery and renewal packets had very short initial TTL values such that by the time they reached the central DHCP server via the old switches, they had a TTL of one and this was reduced to zero with the new switches! As a result, the DHCP server was discarding the renewal requests on arrival as the packets had expired; interestingly, the only reason these systems got a DHCP lease in the first place was because the PXE-generated DHCP request had a much longer TTL and when FreeBSD started to boot, it detected that the network interface was already up with a valid IP address so skipped requesting another DHCP lease. This was eventually fixed by getting the switches reconfigured.

Network switches continued to cause problems from time to time – some switches would crash or lock up if too many systems were rebooted out of Windows and attempted to load FreeBSD at the same time, and this sometimes knocked completely unrelated IT equipment off the network! So we decided to stagger the reboots by shutting Windows down in groups of five machines at a time with a one minute interval before the next group shut down, and so on. This neatly side-stepped the issue of investigating a motley collection of switches of varying firmware revisions, etc that were not ours to touch and it also reduced the load on the network boot server.

Going into production, the nightly SCAN schedule

With the teething problems ironed out, the SCAN service finally went live with an automated schedule as follows:

- the TFTP root directory on the FreeBSD boot server is a symbolic link to either the FreeBSD

network boot image or the PXELINUX boot loader; it normally points to PXELINUX during the daytime to prevent the PCs from accidentally being booted into FreeBSD.

- 10 minutes before the teaching room closes for the night, a cron job changes the TFTP root symlink to point the TFTP server root at the FreeBSD boot image
- a DOS batch script pops up a message 5 minutes before the room closes to warn Windows users that the room is closing and that the PCs would shortly be rebooting – the script is executed using the 'at' command (the Windows equivalent of cron)
- at closing time, another batch script on the PCs shuts down the first group of 5 PCs and reboots into FreeBSD
- one minute later, the next group of 5 PCs shut down and reboot
- and so on until all the PCs in that room have rebooted. Finally, the TFTP root symlink is switched back to point to PXELINUX 30 minutes after the room closes so as to make sure that other Windows PCs in different rooms that are not yet closed will not come back up running FreeBSD if someone should reboot them.

There are several rooms – one is part of the departmental library and closes at 6pm while others are kept open for student use until 8pm, 10pm and 11pm respectively, so this reboot sequence is repeated for each room at its respective closing time.

The following morning, if it is a weekday, a cron job running on each PC will execute at 5.50am and shut the system down and reboot out of FreeBSD and back into Windows. (This early shutdown is to ensure that various Windows updates, and software upgrades have sufficient time to complete before the teaching rooms re-open to students at 8am). Some rooms are closed at weekends so systems in those rooms continue to run FreeBSD until 5.50am the following Monday and various other schedules can be configured to cater for holiday periods and long vacations.

Performance, performance and yet more performance

From the start, tests using the MPI Pallas benchmarking suite were very encouraging even on a standard 100 Mbit/s Ethernet network. At a machine level, it is possible to use MFS (memory file system) to hold computation results locally and only write data out to the user's filesystem on the remote NFS server shortly before the HPC closes down in the morning. This coupled with only one (or in later years, two or four) CPU cores on each memory bus promises a high throughput. The system is particularly suited to asynchronous parallel computation, for example Monte Carlo simulations, and the Imperial SCAN with just 79 active PC nodes has comfortably out-performed a multi-million dollar Cray T3E.

In 2008, by way of an experiment, we temporarily included into the SCAN a HP blade cluster comprising ten BL460c G1 blades, each fitted with two quad-core Xeon processors clocked at 2.83 GHz and 16 GB of memory, and housed in a HP BladeSystems 7000c rackmount chassis. But comparisons of the performance of this very expensive blade system with the distributed desktop PC-based SCAN were inconclusive – the blade system with its short gigabit network connections to a dedicated local switch performed some tasks a little quicker than the distributed system but others actually ran slower and this was confirmed by Pallas benchmark tests. Overall, the performance was similar – but the distributed PC platform was infinitely cheaper!

Advantages of the diskless approach and scalability

Setting up large scale computation facilities in this way has some tremendous advantages; one is that it is very easy to control, maintain and upgrade since there is only one operating system image that is used by all of the client PCs. (The hard disks in the Windows PCs remain untouched and are not used in any way while the systems are running FreeBSD). So instead of having to make changes to, upgrade or even rebuild each machine individually, this work is carried out on the image that will be served by the network boot server only and the next time the client nodes boot from this, they too will be running the new or changed image. It is of course possible to customise the image and the start-up scripts to some extent so that machines in different rooms can load a different configuration at boot time, for

example. And in the current implementation, much of the booted PCs live filesystem, using the union filesystem, is hosted on the boot server which makes it easy to make immediate 'hot' changes to the operating system that is running on all of the client PCs, tune the kernel while it is running, add new user accounts, etc without having to reboot to load a new operating system image.

But the real beauty of the system is the almost infinite scalability and ease with which more PCs (or nodes) can be added to the SCAN; anything from a single computer to many thousands can be added simply by enabling booting from the network in their BIOS and adding them to the 'Beowulf' DHCP server group and the system will operate over different network subnets. Currently the system operates with a core group of 79 nodes always being available in the Maths department but in tests carried out in collaboration with the departments of Physics and Chemistry using their teaching PC cluster rooms, as many as 200 systems have been in simultaneous operation across three departments.

Today

Several years on, this has become a standard 'production' resource, proving to be extremely reliable and requiring very little maintenance. Standing the test of time, the basic ideas have remained unchanged since 2003 – the PCs have since gone through several generations of renewal, from 450 MHz Pentium 2 systems with 128 MB of memory to dual and quad-core Pentium 4 machines with 4 GB of memory and we now have a minimum of 230 CPU cores available every night. The infrastructure has evolved too – the bootable image is now the 64-bit version of FreeBSD release 7.1, the boot server is a 64-bit AMD machine while the user home directories have been moved to a separate 32-bit Pentium 4 machine with server motherboard, SCSI controller and several large disks, running FreeBSD 8.0. Network speed has increased from 100 Mbit/s to 1 Gbit/s and all of the PCs and the servers have gigabit Ethernet interfaces.

And there has even been a little bit of 'feature creep' – in an ICT environment where Windows is the predominant operating system with MacOS and Red Hat Linux available as alternatives, some people are scared of FreeBSD so the boot server now hosts a MySQL database and apache server to support a web interface for user account management and for scheduling the dates and times when the HPC service starts and stops.

But to this day, the HPC has cost the Maths department nothing – the PCs are provided anyway from central funding for teaching purposes, there has always been a policy of keeping them powered on at all times to extend the life of power supplies and hard disks, and a 'reuse and recycle' ethic combined with donations from various members of Maths staff of PC components, disks and other parts has breathed new life into old PC tower cases and kept the two servers providing the infrastructure up to date.

In part two, we take a technical look at how this was done and show just how easy it is to implement this.

The second part of this article will appear in the next Newsletter.

The Barrelfish Multikernel: an interview with Timothy Roscoe

Rik Farrow

Increasing CPU performance with faster clock speeds and ever more complex hardware for pipelining and memory access has hit the brick walls of power and bandwidth. Multicore CPUs provide the way forward but also present obstacles to using existing operating systems design as they scale upwards. Barrelfish represents an experimental operating system design where early versions run faster than Linux on the same hardware, with a design that should scale well to systems with many cores and even different CPU architectures.

Barrelfish explores the design of a multikernel operating system, one designed to run non-shared copies of key kernel data structures. Popular current operating systems, such as Windows and Linux, use

a single, shared operating system image even when running on multiple-core CPUs as well as on motherboard designs with multiple CPUs. These monolithic kernels rely on cache coherency to protect shared data. Multikernels each have their own copy of key data structures and use message passing to maintain the correctness of each copy.

In their SOSP 2009 paper [1], Baumann et al. describe their experiences in building and benchmarking Barrelfish on a variety of Intel and AMD systems ranging from four to 32 cores. When these systems run Linux or Windows, they rely on cache coherency mechanisms to maintain a single image of the operating system. This is not the same thing as locking, which is used to protect changes to data elements which themselves consist of data structures, such as linked lists, that must be changed atomically. In monolithic kernels, a change to a data element must be visible to all CPUs, and this consistency gets triggered when a CPU attempts to read or write this data in its own cache. Cache consistency mechanisms prevent the completion of this read or write if the cache line is invalid, and also mean that execution may be paused until the operation is complete.

In a multikernel, each CPU core runs its own kernel and maintains its own data structures. When a kernel needs to make a change to a data structure (e.g., memory page tables) that must be coordinated with kernels running on other cores, it sends messages to the other kernels.

I asked Timothy Roscoe of the Systems Group at ETH Zurich if he could answer a few questions about Barrelfish, working in a manner similar to Barrelfish, using asynchronous messaging. Before I begin the interview, Mothy wanted me to point out that the development of Barrelfish involves a very large team of people, and he is just one person among many working on this very complex project. You can learn more about this team by visiting the Barrelfish Web site `http://www.barrelfish.org/`.

Farrow: Barrelfish maintains separate kernel state, and this seems to me to be one of the key differentiators from monolithic kernels.

Roscoe: Actually, this is not quite, but nearly, true: monolithic kernels started with a single shared copy of kernel state, and to a limited extent they have started to replicate or partition this state to reduce memory contention on multiprocessors. Solaris is probably the most advanced version of this. The model, however, remains one of a single image managing the whole machine, with the replication and/or partitioning of kernel state as an optimization.

In a multikernel, this is the other way around. No kernel state at all is shared between cores by default, and so consistency must be maintained by explicitly sending messages between cores, as in a distributed system. The model is one of replicated or partitioned data which is accessed the same way as one would access replicas in a distributed system. In particular, depending on the consistency requirements, changing some OS state may be a two-phase operation: a core requests a change and, at some point in the future, gets confirmation back that every other core has agreed to it, or, alternatively, that it conflicted with some other proposed change and so didn't happen.

In principle, we could share kernel data between cores in Barrelfish, and this might be a good idea when the cores are closely coupled, such as when they share an L2 or L3 cache or are actually threads on the same core. We also intend to do this at some point, but the key idea is that the model is of replicated data, with sharing as a transparent optimization. In traditional kernels it's the other way around.

Farrow: Barrelfish has a small CPU driver that runs with privilege, and a larger monitor process that handles many of the tasks found in a monolithic operating system. Barrelfish is not a microkernel, as microkernels share a single operating system image, like much larger monolithic kernels. Barrelfish does seem to share some characteristics of microkernels, such as running device drivers as services, right?

Roscoe: You're right that every core in Barrelfish runs its own CPU driver, which shares no memory with any other core. Also, every core has its own monitor process, which has authority (via capabilities) to perform a number of privileged operations. Most of the functionality you would expect to find in a UNIX kernel is either in driver processes or servers (as you would expect in a microkernel) or the distributed network of monitor processes.

Farrow: The SOSP paper talks about a system knowledge base (SKB) that gets built at boot time using probes of ACPI tables, the PCI bus, CPUID data, and measurement of message passing latency. Could you explain the importance of the SKB in Barrelfish?

Roscoe: The SKB does two things. First, it represents as much knowledge as possible about the hardware in a subset of first-order logic – it's a Constraint Logic Programming system at the moment. This, as you say, is populated using resource discovery and online measurements. Second, because it's a reasoning engine, the OS and applications can query it by issuing constrained optimization queries.

This is very different from Linux, Solaris, or Windows: traditional OSes often make some information about hardware (such as NUMA zones) available, but they often over-abstract them, the format of the information is ad hoc, and they provide no clean ways to reason about it (resulting in a lot of non-portable complex heuristic code). The SKB is not a magic bullet, but it drastically simplifies writing OS and application code that needs to understand the machine, and it means that clients can use whatever abstractions of the hardware are best for them, rather than what the OS designer thought useful.

We currently build on ARM, x86_64, x86_32, and Beehive processors. We're currently also porting to Intel's recently announced SCC (Single-chip Cloud Computer), which is a somewhat unconventional variant of x86_32.

One interesting feature of Barrelfish is that you don't really "port" the OS to a different architecture; rather, you add support for an additional CPU driver. Since CPU drivers and monitors only communicate via messages, Barrelfish will in principle happily boot on a machine with a mixture of different processors.

Farrow: While reading the paper, I found myself getting confused when you discussed how a thread or process gets scheduled. Could you explain how this occurs in Barrelfish?

Roscoe: Well, here's one way to explain this: Barrelfish has a somewhat different view of a "process" from a monolithic OS, inasmuch as it has a concept of a process at all. It's probably better to think of Barrelfish as dealing with "applications" and "dispatchers."

Since an application should, in general, be able to run on multiple cores, and Barrelfish views the machine as a distributed system, it follows that an application also, at some level, is structured as a distributed system of discrete components which run on different cores and communicate with each other via messages.

Each of these "components," the representative of the application on the core, so to speak, is called a "dispatcher." Unlike a UNIX process (or thread), dispatchers don't migrate – they are tied to cores. When they are descheduled by the CPU driver for the core, their context is saved (as in UNIX), but when they are rescheduled, this is done by upcalling the dispatcher rather than resuming the context. This is what Psyche and Scheduler Activations did, to first approximation (and K42, which is what we took the term "dispatcher" from, and Nemesis, and a few other such systems).

Farrow: So how do you support a traditional, multi-threaded, shared-memory application like OpenMP, for example?

Roscoe: Well, first of all, each dispatcher has, in principle, its own virtual address space, since each core has a different MMU. For a shared-memory application, clearly these address spaces should be synchronized across the dispatchers that form the application so that they all look the same, whereupon the cache coherence hardware will do the rest of the work for us. We can achieve this either by messages or by sharing page tables directly, but in both cases some synchronization between dispatchers is always required when mappings change.

As an application programmer, you don't need to see this; the dispatcher library handles it. Incidentally, the dispatcher library also handles the application's page faults – another idea we borrowed from Nemesis and Exokernel.

Application threads are also managed by the dispatchers. As long as a thread remains on a single core,

it is scheduled and context-switched by the dispatcher on that core (which, incidentally, is a much nicer way to implement a user-level threads package than using signals over UNIX). Note that the CPU driver doesn't know anything about threads, it just upcalls the dispatcher that handles these for the application, so lots of different thread models are possible.

To migrate threads between cores (and hence between dispatchers), one dispatcher has to hand off the thread to another. Since the memory holding the thread state is shared, this isn't too difficult. It's simply a question of making sure that at most one dispatcher thinks it owns the thread control block at a time. The dispatchers can either do this with spinlocks or by sending messages.

Farrow: Why should a multikernel work better than a monolithic kernel on manycore systems? In your paper, you do show better performance than a Linux kernel when running the same parallel tasks, but you also point out that the current Barrelfish implementation is much simpler/less functional than the current Linux kernel.

Roscoe: Our basic argument is to look at the trends in hardware and try to guess (and/or influence) where things are going to be in 10 years.

The main difference between a multikernel like Barrelfish and a monolithic OS like Linux, Windows, or Solaris is how it treats cache-coherent shared memory. In monolithic kernels, it's a basic foundation of how the system works: the kernel is a shared-memory multi-threaded program. A multikernel is designed to work without cache-coherence, or indeed without shared memory at all, by using explicit messages instead.

There are four reasons why this might be important:

First, cache-coherent shared memory can be slower than messages, even on machines today. Accessing and modifying a shared data structure involves moving cache lines around the machine, and this takes hundreds of machine cycles per line. Alternatively, you could encode your operation (what you want to be done to the data structure) in a compact form as a message, and send it to the core that has the data in cache. If the message is much smaller than the data you need to touch, and the message can be sent efficiently, this is going to be fast.

"Fast" might mean lower latency, but more important is that cores are generally stalled waiting for a cache line to arrive. If instead you send messages, you can do useful work while waiting for the reply to come back. As a result, the instruction throughput of the machine as a whole is much higher, and the load on the system interconnect is much lower – there's just less data flying around.

Ironically, in Barrelfish on today's hardware, we mostly use cache-coherent shared memory to implement our message passing. It's really the only mechanism you've got on an x86 multiprocessor, aside from inter-processor interrupts, which are really expensive. Even so, we can send a 64-byte message from one core to another with a cost of only two interconnect transactions (a cache invalidate and a cache fill), which is still much more efficient than modifying more than three or four cache lines of a shared data structure.

The second reason is that cache-coherent shared memory can be too hard to program. This sounds counterintuitive – it exists in theory to make things easier. It's not about shared-memory threads vs. messages per se either, which is an old debate that's still running. The real problem is that hardware is now changing too fast, faster than system software can keep up.

It's a bit tricky, but ultimately not too hard to write a correct parallel program for a shared-memory multiprocessor, and an OS is to a large extent a somewhat special case of this. What's much harder, as the scientific computing folks will tell you, is to get good performance and scaling out of it. The usual approach is to specialize and optimize the layout of data structures, etc., to suit what you know about the hardware. It's a skilled business, and particularly skilled for OS kernel developers.

The problem is that as hardware gets increasingly diverse, as is happening right now, you can't do this for general mass-market machines, as they're all too different in performance characteristics. Worse, new architectures with new performance tradeoffs are coming out all the time, and it's taking longer

and longer for OS developers, whether in Microsoft or in the Linux community, to come out with optimizations like per-core locks or read-copyupdate – there's simply too much OS refactoring involved every time.

With an OS built around inter-core message passing rather than shared data structures, you at least have a much better separation between the code responsible for OS correctness (the bit that initiates operations on the replicated data) and that responsible for making it fast (picking the right consistency algorithm, the per-core data layout, and the message passing implementation). We'd like to think this makes the OS code more agile as new hardware comes down the pipe.

The third reason is that cache-coherent shared memory doesn't always help, particularly when sharing data and code between very different processors. We're beginning to see machines with heterogeneous cores, and from the roadmaps this looks set to continue. You're going to want to optimize data structures for particular architectures or cache systems, and a one-size-fitsall shared format for the whole machine isn't going to be very efficient. The natural approach is to replicate the data where necessary, store it in a format appropriate to each core where a replica resides, and keep the replicas in sync using messages – essentially what we do in Barrelfish.

The fourth reason is that cache-coherent shared memory doesn't always exist. Even a high-end PC these days is an asymmetric, non-shared memory multiprocessor: GPUs, programmable NICs, etc., are largely ignored by modern operating systems and are hidden behind device interfaces, firmware blobs, or, at best, somewhat primitive access methods like CUDA.

We argue that it's the job of the OS to manage all the processors on a machine, and Barrelfish is an OS designed to be able to do that, regardless of how these programmable devices can communicate with each other or the so-called "main" processors.

It's not even clear that "main" CPUs will be cache-coherent in the future. Research chips like the Intel SCC are not coherent, although they do have interesting support for inter-core message passing. I'm not sure there's any consensus among the architects as to whether hardware cache-coherence is going to remain worth the transistor budget, but there's a good chance it won't, particularly if there is system software whose performance simply doesn't need it.

Barrelfish is first and foremost a feasibility study for this – knowing what we now do about how to build distributed systems, message passing, programming tools, knowledge representation and inference, etc., we can build an OS for today's and tomorrow's hardware which is at least competitive with current performance on a highly engineered traditional OS and which can scale out more effectively and more easily in the future. If a handful of researchers can do that, it sounds like a result.

References

[1] Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schüpbach, and Akhilesh Singhania, "The Multikernel: A New OS Architecture for Scalable Multicore Systems," Proceedings of the 22nd ACM Symposium on OS Principles, Big Sky, MT, USA, October 2009: `http://www.barrelfish.org/barrelfish_sosp09.pdf`.

N.B.: The Barrelfish team also includes researchers Jan Rellermeyer, Richard Black, Orion Hodson, Ankush Gupta, Raffaele Sandrini, Dario Simone, Animesh Trivedi, Gustavo Alonso, and Tom Anderson.

Originally published in ***;login: The USENIX Magazine,****, volume 35, number 1, April 2010, (Berkeley, CA: USENIX Association, 2010). Reprinted by kind permission.*

Developing Large Web Applications

Kyle Loudon

O'Reilly Media
ISBN: 978-0-596-80302-5
304pp.
£ 26.99
Published: March 2010

reviewed by Paul Waring

This book claims to be 'the book for you' if 'you're ready to build a finely crafted large site'. As someone who runs a business-critical site which has to stay up and is getting increasing complex in terms of functionality, this sounds like the book for me.

The first chapter is rather brief and runs through the tenets of large web applications, although with no mention of what the authors consider 'large' to mean. The second chapter is entitled 'Object Oreintation', which seems rather odd as I would expect most programmers to be using object oreintation, and thus not need a basic introduction to the difference between private, public and protected. Programmers not using an object oreintated language don't have access to this functionality, so this chapter is wasted on them. The dreaded UML class diagram also rears its head, together with a page of skeleton PHP classes which don't seem to serve any particular purpose.

Hoping that the first two chapters were just meant to lay the groundwork and bring beginners up to speed, I swiftly moved on. However, I was disappointed to find that chapter 3 spends twenty three pages discussing 'large-scale HTML', which seems to boil down to indenting code, using CSS instead of tables, not using deprecated tags and get to grips with RDF. Chapter 4 gives CSS the same treatment, with pages of code which have no relevance to large scale development. Onto chapter 5 and we're now learning the basics of JavaScript, which is all well and good but there's little detail on how this applies to large applications.

Moving on to chapter 6 we get an introduction to "data managers" and how to abstract an SQL data source, which seems to be a reimplementation of the existing frameworks which already perform this function. Chapter 7 is about "large scale PHP" and is padded out with plenty of code, which breaks the basic rule of separating business logic from HTML by embedding markup into the PHP functions. Chapter 8 gives a similar treatment to AJAX.

It's not until chapter 9 that we finally get to the topic of performance, something which really matters when you are building or maintaining large web applications. This chapter is actually useful and relevant, but you would do better by plumping for 'High Performance Web Sites' (O'Reilly) which covers these topics in much greater detail. The final chapter discusses the architecture of a project, which seems to boil down to which directories you store your files in.

All in all, I found this book to be a disappointing read. There are too many code samples, many of which are trivial and do not provide a good illustration of the concepts under discussion. Most of the chapters seem to be aimed at complete beginners, who are unlikely to end up developing a large scale system until they have obtained some experience working on smaller projects. If you have never built a large web application before, are starting from scratch and plan to use PHP exclusively, then there may be some useful information in this book, otherwise there is little to recommend it.

Building Web Reputation Systems

Randy Farmer and Bryce Glass

Yahoo! Press
ISBN: 978-0-596-15979-5
336pp.
£ 30.99
Published: March 2010

reviewed by Paul Waring

Building a reputation system can be a fiendishly difficult task. How do you allow new users to build up trust, whilst still being able to partipate from the offset? How do you ensure that 'trusted' users are rewarded for their efforts, without opening the system to abuse? Fortunately, help is at hand with this focussed and niche book from two employees of Yahoo!.

The first chapter introduces the concpept of reputation systems, with some useful case studies and examples. Refreshingly, no mention is made of software at this point, which allows the reader to understand the topic properly before getting bogged down with implementation details. The authors move on to discuss the various aspects of reputation, such as who is making a claim and what is the claim about? This sometimes seems like an obvious question, but by breaking a reputation statement down into its constituent parts, the authors make you think much more deeply about what a claim (e.g. 'Person X enjoyed film Y') actually means.

The next few chapters concentrate on reputation in more detail, looking at aspects such as the difference between a qualitative claim (e.g. 'this restaurant was excellent') and a quantitive one (e.g. a 5 star rating). They also cover some of the pitfalls you may encounter with reputation systems, such as the problem of trying to assign an overall reputation to something which has only received a small number of ratings. Common reputation methods are also covered, working up from the most simple models to more complex ones which require a certain amount of thought and planning.

Later chapters cover interesting topics such as how reputation should be displayed (arguably there is no pointing in earning reputation if no one knows about it) as well as how to use it to rank results and weed out undesirable content. Not until chapter 9 do we hear any serious discussion about integration, testing and tuning, together with an interesting warning about the dangers of excessive tuning of reputation systems. A detailed case study of an existing large-scale reputation system (Yahoo! Answers) rounds off the book nicely, although an appendix on reputation frameworks is available for those who want to probe deeper into the technical aspects.

Overall, whilst this book might seem like a niche topic, reputation systems are something which need to be considered by anyone working on a site which allows users, registered or anonymous, to contribute and rate content and each other, from eBay sales to forum posts. If you fall into this category, this book is well worth a read as it covers all the aspects of building a reputation system, yet does so in an abstract way which does not rely on any one technology or language. Even if you are not in a position where dealing with reputation systems is part of your everyday work, the book is worth a read simply for the interesting and in-depth analysis of how these systems can work, and why the final rating of an object might not mean what you think it does.

Alternative DNS Servers

Jan-Piet Mens

UIT Cambridge
ISBN: 978-0-9544529-9-5
694pp.
£ 29.50
Published: October 2008

reviewed by John Collins

This book is an exhaustive treatment of the Domain Name System and various DNS servers.

DNS is how you get from a text domain name such as the part following the `@` in an email address or the bit between `http://` and the first single `/` if any of a web address, to the actual IP address of a machine somewhere in the world which you are trying to talk to (and in some cases the reverse).

The most commonly-used and completely comprehensive DNS Server is BIND and for most UNIX or Linux users this will probably be the only one they may have met. However there are several competitors, and variations on those competitors to take data via LDAP or various database engines.

I confess I had only heard of one of the alternatives to BIND when I picked up this book. However I learnt about various others, their pros and cons, their relative performance and much other data.

The book is in three sections. In the first section DNS is fully described, including some of its history. In the second and largest section the various servers are presented in microscopic detail apart from the Microsoft proprietary DNS servers which are an overview – the author steers the reader onto open-source alternatives. In the third section, "Operational Issues" some practical issues are discussed, together with some performance tests.

The book concludes with some comprehensive appendices on various topics notably LDAP and also some Perl DNS Servers – in case after all that you want to write your own.

The detail this book goes into is incredible. It documents features of BIND which are only sketchily covered in manuals I had read and previously had to guess at their usage (wrongly in a few cases). Further reading suggestions point the reader to other books on various topics.

I'm sure this book will be very helpful to anyone wanting to install a DNS server and make the best us of it in their environment. Perhaps BIND may not be the best solution for them and they may select something else. They may or may not want a Caching server. They may want to take information from a database. They may even want to "roll their own". Whatever they do, I am sure that this book will be a considerable help.

HTML & CSS: The Good Parts

Ben Henick

O'Reilly Media
ISBN: 978-0-596-15760-9
352pp.
£ 26.99
Published: February 2010

reviewed by Gavin Inglis

Spring 2010 seems a strange time to publish a new book about HTML. But this is no beginner's tutorial dragging out the same tired old code. This book is a lecture from a veteran: a masterclass, if you like.

HTML & CSS: The Good Parts is aimed at experienced web professionals and serious amateurs. Part philosophy, part history, and part advanced tutorial, it considers how web publishing got to where it is

and how we might best take advantage of the situation. Some readers might balk at phrases like "One can read markup like music", but the core of this book is serious, practical, considered advice on how to build and maintain a website in a chaotic and vulnerable technical environment.

Chapter four is fairly representative of the work as a whole. Entitled "Developing a Healthy Relationship with Standards", it is a discussion of the standards that are out there, why they came about, and a detailed 80/20 rule for "standards-friendliness". One interesting suggestion is that a web developer's most valuable asset is not her technical or artistic chops, but instead her skill at interacting positively with the rest of her team.

Philosophically Henick's approach has a lot in common with the CSS Zen Garden project. His Four Habits of Effective Stylists are simplicity, flexibility, consistency and maintaining a sense of place. The Zen garden metaphor, "demonstrating tangible harmony and precision in spite of evolving surroundings and Nature's unpredictability" does seem to be a good fit to the dynamic environment that web designers are forced to inhabit.

This one's not just for the hippies and the samurai though; there are plenty of specifics with example code. The various components of the HTML/CSS toolkit are considered thoughtfully: there are chapters on lists, links, colours, backgrounds, forms and even – gasp – tables. Only for data, naturally. The image chapter introduces some basic image processing and optimising. Many readers may not have considered the typography on their web site and there are a healthy thirty-four pages devoted to this.

Chapter six is an in-depth study of how CSS properties are actually implemented: box models, floats, clears and their interaction with each other and the cascade. Following through the discussion of two and three column designs is a valuable exercise for anybody who's ever scratched their head at why layout behaves the way it does. The chapter ends on an artistic note, with a discussion of proportion: grid systems, the rule of thirds and the golden ratio.

Given its launch date, one might expect this book to be a primer for HTML 5. It's not. Instead there are a few references to the likely changes at the end of the relevant chapters. In the meantime, the author's advice is "Avoid inappropriate elements, lean hard on the universal attributes, and wait breathlessly for HTML5 to catch on."

Chapter fourteen is saved for the end, and it's the HTML ghost train, taking a scary route through browser wars, template fragility, frames and "Invalid Markup for Stupid Reasons". Beyond a break in the track lies "The Awful Parts", including the likes of IE user interfacea manipulation, non-linked underlining, and of course, `<blink>`.

If all you want is a general book on HTML with copious code fragments and browser specific details, this is not the book for you. However if you want to elevate your coding and perhaps gain a little inner clarity, *HTML & CSS: The Good Parts* is money well spent.

RESTful Web Services Cookbook

Subbu Allamaraju

Yahoo! Press
ISBN: 978-0-596-80168-7
320pp.
£ 30.99
Published: March 2010

reviewed by Mike Fowler

Representational State Transfer (REST) is a software architectural model based on the unified interface presented by HTTP. REST applications follow the conventional client-server architecture of the web and benefit from all the other software surrounding the web, in particular caching software such as Squid.

The book is laid out in the standard O'Reilly cookbook format with each recipe consisting of a problem statement, a solution and a discussion of the solution. The recipes are grouped into 14 chapters ranging from the basic uses of the uniform interface, resource and representation design through to the more complex conditional requests and security.

One criticism commonly laid against REST is the idea that REST is only useful for Create, Read, Update and Delete (CRUD) applications. The author dedicates an entire chapter to discussing a number of ways of using REST to solve activities that clearly do not full into CRUD. Examples include copying, merging, moving and undoing. This section of the book was by far the most useful and after working through a couple of the recipes you quickly discover that CRUD really is only the beginning of a RESTful service.

The recipe examples were the most disappointing aspect of this book. All examples in the book are simple HTTP dumps detailing the headers and content bodies where appropriate. After working through a few recipes I found the examples no longer helped, the information provided in the main text was sufficient for me. Although the author details at the very start why they have chosen not to provide language specific examples, I felt that a handful may have been useful. An example detailing how you might provide a RESTful service in Java and an example on how to consume a RESTful service from Perl would certainly ease the transition from the raw HTTP examples into the concrete code that would be used.

On the whole I found this book quite useful and would recommend it to people who are just starting out in RESTful web service development. Advanced developers would be advised to look for material dedicated to their chosen platform and development environment.

Book discounts for members

Please note that the prices quoted in book reviews in the Newsletter are always the full published price.

UKUUG members will be interested to know that O'Reilly offer a 35% discount off list price to UKUUG members. Details of how to order and benefit from this saving are at
http://www.ukuug.org/books/oreilly/

The book *Alternative DNS Servers* reviewed above, published by UIT, can be obtained by UKUUG members at the special price of £15 (published price £29.50) by mailing **ukuug-offer@uit.co.uk** before the end of July 2010.

Contributors

John Collins is a UKUUG Council member and runs Xi Software Ltd which specialises in System Management tools for UNIX and Linux systems. Xi Software is based in Welwyn Garden City and has been in existence for nearly 21 years.

Mike Fowler is the Senior Systems Engineer of a UK based technology driven utility company. A firm open-source advocate, he drove the company from Microsoft based solutions to complete Linux/PostgreSQL stacks. He is a major contributor to YAWL, an open source BPM/Workflow system. When not melded with a keyboard, he spends his time playing as many different stringed instruments as he can.

Peter Baer Galvin is the chief technologist for Corporate Technologies, a premier systems integrator and VAR **www.cptech.com**. Before that, Peter was the systems manager for Brown University's Computer Science Department. He has written articles and columns for many publications and is co-author of the Operating Systems Concepts and Applied Operating Systems Concepts textbooks. As

a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide. Peter twitters as `PeterGalvin` and blogs at **`http://www.galvin.info`**

Gavin Inglis works in Technical Infrastructure at the EDINA National Data Centre in Edinburgh. He is a teacher, photographer and musician who has recently discovered the joys of spreadsheets as a recreational tool.

Jason Meers is a Linux developer and Green IT consultant for **`www.oneplanetcomputing.com`**. He specialises in low-carbon computing and the re-deployment of legacy PCs, laptops and thin clients into centrally managed, secure desktops for use with the new generation of Microsoft, Citrix and VMware desktop virtualisation servers.

Jane Morrison is Company Secretary and Administrator for UKUUG, and manages the UKUUG office at the Manor House in Buntingford. She has been involved with UKUUG administration since 1987. In addition to UKUUG, Jane is Company Secretary for a trade association (Fibreoptic Industry Association) that she also runs from the Manor House office.

Timothy Roscoe is part of the ETH Zürich Computer Science Department's Systems Group. His main research areas are operating systems, distributed systems, and networking, with some critical theory on the side.

Andy Thomas is a UNIX/Linux systems administrator working for Dijit New Media and for Imperial College London and as a freelancer. Having started with Linux when it first appeared in the early 1990's, he now enjoys working with a variety of UNIX and Linux distributions and has a particular interest in high availability systems and parallel compute clusters.

Paul Waring is chairman of UKUUG and is currently the Technical Manager for an insurance intermediary.

Roger Whittaker works for Novell Technical Services at Bracknell supporting major Linux accounts in the UK. He is also the UKUUG Newsletter Editor, and co-author of three successive versions of a SUSE book published by Wiley.

Correction

Dr Owen Le Blanc has written to point out that contrary to the assertion in Peter Salus's article *The Advent of Linux* which appeared in the UKUUG December 2009 Newsletter, Bruce Perens was not involved in any way with with MCC Linux.

Contacts

Paul Waring
UKUUG Chairman
Manchester

John M Collins
Council member
Welwyn Garden City

Phil Hands
Council member
London

Holger Kraus
Council member
Leicester

Niall Mansfield
Council member
Cambridge

John Pinner
Council member
Sutton Coldfield

Howard Thomson
Treasurer; Council member
Ashford, Middlesex

Jane Morrison
UKUUG Secretariat
PO Box 37
Buntingford
Herts
SG9 9UQ
Tel: 01763 273475
Fax: 01763 273255
office@ukuug.org

Roger Whittaker
Newsletter Editor
London

Alain Williams
UKUUG System Administrator
Watford

Sam Smith
Events and Website
Manchester