LINUX JOURNAL™
Since 1994: The Original Magazine of the Linux Community

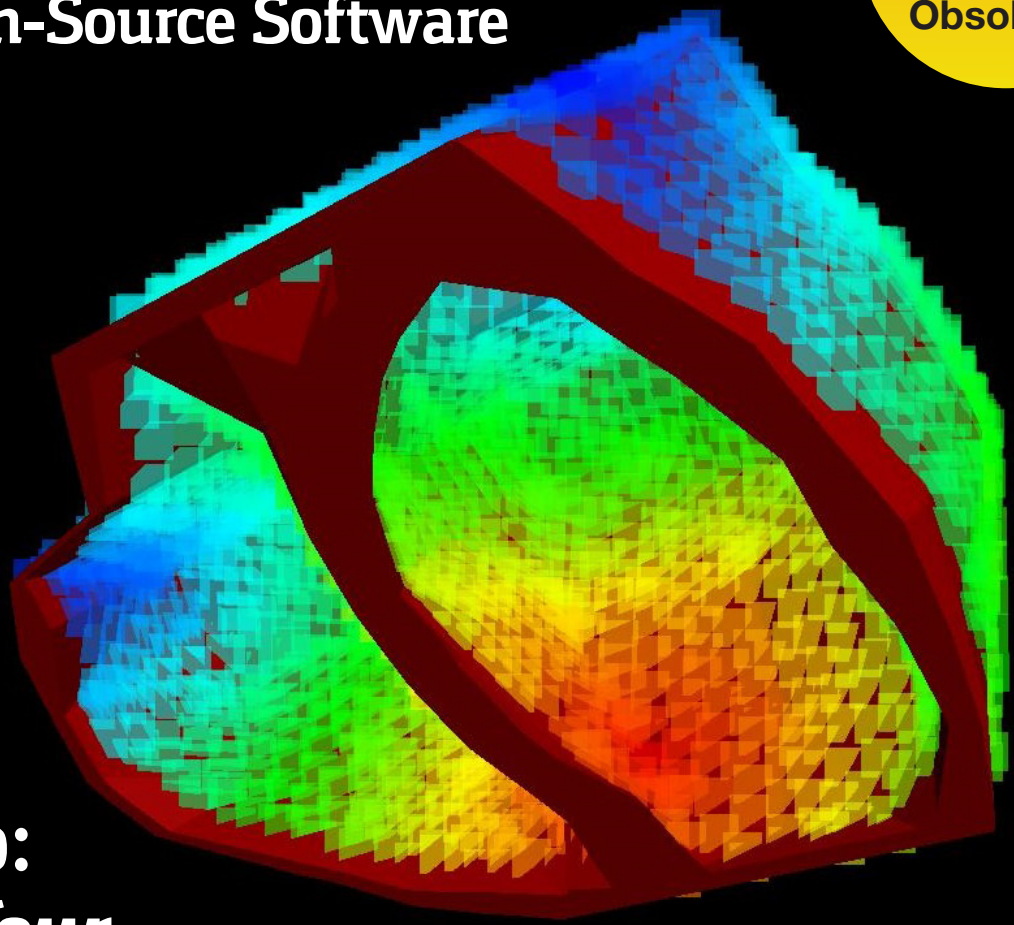
Install a Network Monitor to Find Bandwidth Hogs

Novelty Detection with Machine Learning

3D IMAGING *of* HEART ACTIVITY with Open-Source Software

MAY 2017 | ISSUE 277
http://www.linuxjournal.com

EOF: Will Anything Make Linux Obsolete?



\+

How-To: Build Your Own Cluster for Beginners

WATCH: ISSUE OVERVIEW

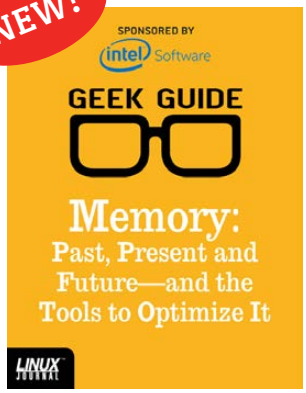
NEW!

An Architect's Guide: Linux for Enterprise IT

Author:
Sol Lederman

Sponsor:
SUSE

NEW!



Memory: Past, Present and Future—and the Tools to Optimize It

Author:
Petros Koutoupis

Sponsor:
Intel

Cloud-Scale Automation with Puppet

Author:
John S. Tonello

Sponsor:
Puppet

Why Innovative App Developers Love High-Speed OSDBMS

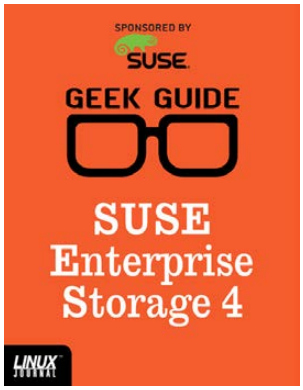
Author:
Ted Schmidt

Sponsor:
IBM

Tame the Docker Life Cycle with SUSE

Author:
John S. Tonello

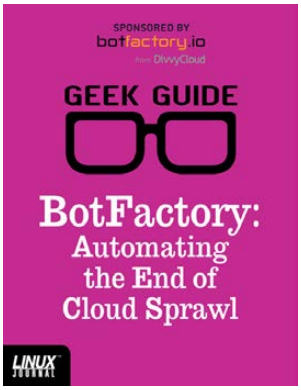
Sponsor:
SUSE



SUSE Enterprise Storage 4

Author:
Ted Schmidt

Sponsor:
SUSE



BotFactory: Automating the End of Cloud Sprawl

Author:
John S. Tonello

Sponsor:
BotFactory.io

Containers 101

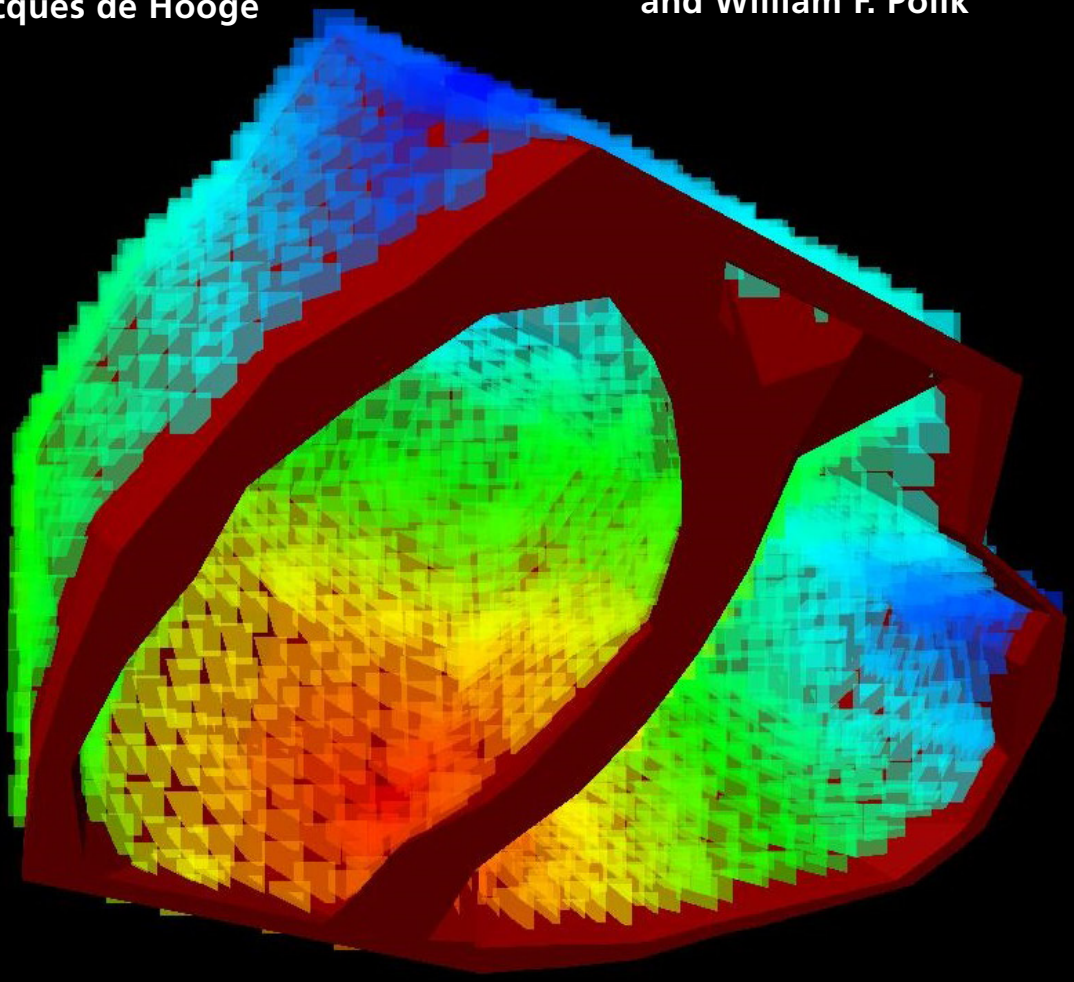
Author:
Sol Lederman

Sponsor: Puppet

CONTENTS

MAY 2017
ISSUE 277

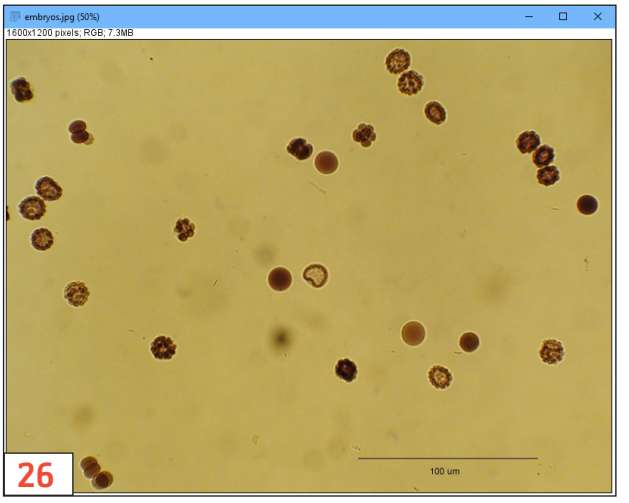
FEATURES



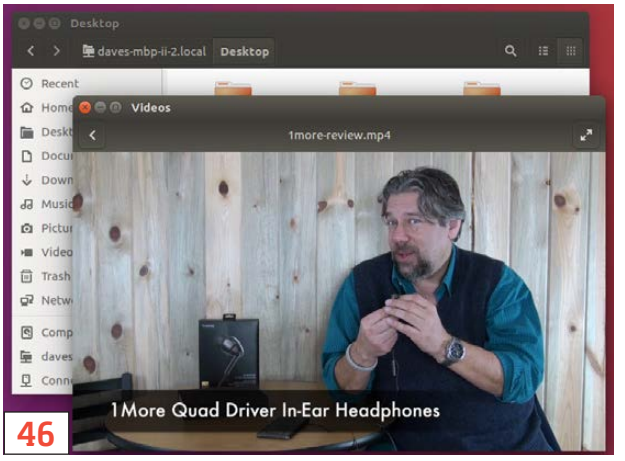
COLUMNS

IN EVERY ISSUE

22



26



46

ON THE COVER

- **3D Imaging of Heart Activity with Open-Source Software, p. 84**
- **How-To: Build Your Own Cluster for Beginners, p. 96**
- **Install a Network Monitor to Find Bandwidth Hogs, p. 64**
- **Novelty Detection with Machine Learning, p. 38**
- **EOF: Will Anything Make Linux Obsolete?, p. 112**

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., PO Box 980985, Houston, TX 77098 USA.
Subscription rate is $29.50/year. Subscriptions start with the next issue.

LINUX JOURNAL

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Associate Editor	Shawn Powers shawn@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Security Editor	Mick Bauer mick@visi.com
Hack Editor	Kyle Rankin lj@greenfly.net
Virtual Editor	Bill Childers bill.childers@linuxjournal.com

Contributing Editors

Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan • Adam Monsen

President	Carlie Fairchild publisher@linuxjournal.com
Publisher	Mark Irgang mark@linuxjournal.com
Associate Publisher	John Grogan john@linuxjournal.com
Director of Digital Experience	Katherine Druckman webmistress@linuxjournal.com
Accountant	Candy Beauchamp acct@linuxjournal.com

Linux Journal **is published by, and is a registered trade name of,**
Belltown Media, Inc.
PO Box 980985, Houston, TX 77098 USA

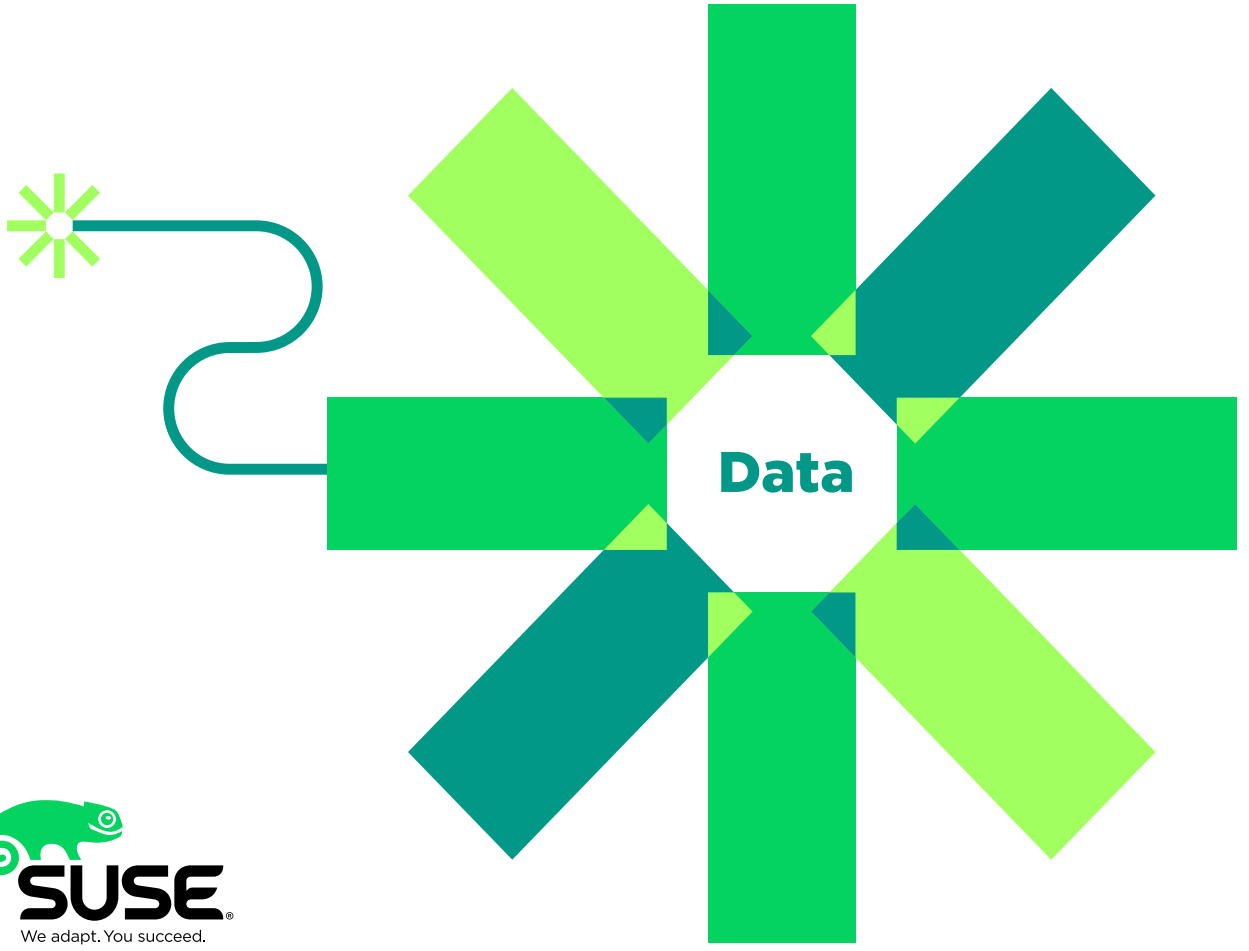
STORAGE
REDEFINED:

You cannot keep up with data explosion.

Manage data expansion with SUSE Enterprise Storage.

SUSE Enterprise Storage, the leading open source storage solution, is highly scalable and resilient, enabling high-end functionality at a fraction of the cost.

suse.com/storage



SUSE
We adapt. You succeed.

Doing Big Things

SHAWN POWERS

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via email at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

I bought a book a few years back titled, *Installing Linux on a Dead Badger* by Lucy Snyder. When I see that book on my bookshelf, it still makes me chuckle. And although a dead badger is certainly not a common operating system platform, it seems like Linux continues to end up on more and more hardware every year. Often those devices are tiny, but sometimes those devices are giant clusters of computers that crunch trillions of numbers a second. And of course, some of the tiniest installations end up having the biggest effects (smartphones, for instance). This month's issue is a reminder of just how well our favorite operating system has infiltrated our lives.

We start with Reuven M. Lerner continuing his theme of computer-based learning. Many of you probably remember the *Sesame Street* game Grover played with viewers called "One of These Things Is Not Like the Others". In a similar vein, Reuven shows how you can teach a bit of software how to play that game as it pertains to sets of data. Rather than just chopping off the highs and lows in a set of data, computer learning can determine what data is actually out of place.

Dave Taylor follows with some really awesome scripting. If you've ever spent time mindlessly browsing YouTube, you've probably liked a video so much at one point that you wanted a local copy. Or, perhaps you've

VIDEO: Shawn Powers runs through the latest issue.

wished you could strip the audio off a YouTube video and play it on your sound system. Dave describes a really cool tool for extracting audio and video from YouTube URLs. Google might not be thrilled at the idea of downloading copies of videos, but once again, Linux saves the day.

Speaking of saving the day, Kyle Rankin continues his series on systems administration. This month, he helps define the various levels and titles of administrative abilities. Ever wonder if you qualify as a junior systems administrator or a senior systems administrator? Kyle helps explain what the various titles mean, which is invaluable if you're applying for a job or planning to hire more help. Even if your interests don't lie in administration, the information is critical for anyone in IT to understand.

I did a little bit of investigating this month. Specifically, I investigated my network trying to track down some rogue traffic on my router. I learned a lot along the way, and in the end, I had a face-palm moment. Nevertheless, I figured all the network monitoring and investigative knowledge made it worthwhile. If you're uncertain what is happening on your local network, I encourage you to read my column.

Jacques De Hooge describes another way Linux and open source can help you determine what is going on, specifically what's going on in your chest. Using various open-source tools, along with a few proprietary ones, Jacques tells how Linux helps create 3D models of the heart for diagnoses. It's a perfect example of Linux quietly making the world go round, and it's also really cool to learn about!

We finish off the issue with the first part of a series on building a computer cluster using Linux. Nathan R. Vance, Michael L. Poublon and William F. Polik join forces to teach how to create an appropriate cluster from beginning to end. Whether you want to build your own cluster, or just want to learn about the technology, it's an awesome series that we're starting this month.

We also have the normal collection of *Linux Journal* goodies, including tech tips, new product announcements and UpFront oddities from around the internet. Whether you love Linux because of all the tiny places in can be installed or think it's awesome that Linux powers the internet, this issue should tickle your fancy all the same.■

RETURN TO CONTENTS

LETTERS

PREVIOUS
Current_Issue.tar.gz

NEXT
UpFront

EOF January 2017

I have been an avid and loyal *Linux Journal* subscriber for 20 years. But after reading Doc Searls' January 2017 EOF, I am thoroughly disgusted. I will not be renewing. Politics has no place in a magazine such as *LJ*. If I wanted someone's political opinion, I'd turn on MSNBC or CNN. The arrogance displayed by publishing that article is astounding.

—Doug McComber

Doc Searls replies: *Thanks for writing. And for debugging mine.*

*Under similar criticisms in the web version of the column, I wrote this (*http://www.linuxjournal.com/content/debugging-democracy#comment-3099071559*):*

> *fulp01 isn't a troll. He's a subscriber, and we value those.*
>
> *He's also right to call me to task. Judging from the almost entirely negative response this column has received so far, I was the one doing the trolling, though that wasn't my intent.*
>
> *Calling Trump a troll also distracted readers from my main point, which is that journalism is suffering in a world where a business based on surveillance is programmatically dividing people into mutually hostile echo chambers, which makes democracy suffer as well. And that, because this whole echo-system is programmatic, and to a high degree runs on Linux-based infrastructure, we (or at least some of us) are in a position to help fix it.*

I also wrote a similar response for the magazine, and something like it in my March 2017 column. Hope those help, and that you stay with us.

Open-Source Classroom—Passwords, Security Questions

Regarding Shawn Powers' "All Your Accounts Are Belong to Us" article in the February 2017 issue: good information regarding passwords and 2nd-factor authentication, etc. One thing I've done for the last ten years or so is create fictitious alternative personal data that is used for online accounts—things like birthday, mother's maiden name, high school teacher, first car and so on. I only use my real personal data when it must be done, like for banking or government stuff. I store my alternative personal information in my password manager so I don't have to remember it. This way, if any of my social media or other online accounts ever do have a breach, the data that is leaked can't be used as identification verification by a bad actor or as a pivot point to get into other accounts. The key is being consistent. Regarding password managers, since I don't trust any third party, my password manager is a strongly encrypted text file that is synced via Dropbox with a very strong high entropy password that I've committed to memory. One more thing, regarding using fingerprints and other biometrics to log on, keep in mind that you can be compelled to place your finger (or have retina scan) to unlock your device, but courts (so far) have upheld the divulging of passwords as "something you know" and hence is under 5th Amendment protection.

—Mark Dean

Shawn Powers replies: *That's all really good information, and I've considered the alternate persona thing as well. Having security questions for password recovery just seems like a bad idea, since most of the questions are fairly easy to figure out, especially if you know the person. My bank, for example, asks, "Where were you born?", "Who's your favorite singer?" and "Who's your favorite author", which are all things widely known to anyone who knows me even online.*

You're also correct about the biometrics. They're not a great way to secure a phone, but they do offer a convenience factor that in some cases I deem worth the downside. Thankfully, some apps require multiple authentication factors. Copay, my Bitcoin wallet, for instance, can be forced to require a password and biometrics. Anyway, I think the best thing we can do as an IT community is make sure we're educating folks who might not understand the significance of securing their accounts and data. Often just understanding is enough to get people to make better choices.

Politics Don't Belong in Technical Publications

I respect that this is your publication, but as a customer, I wanted to let you know my opinion and intent should the political rants continue. I don't read this publication to learn about opinions of Trump, the electoral college or the Clintons. I subscribe and pay for Linux-related news and information. Simply put, if the political overtones continue, I'll save myself the subscription cost, browse to Fox news or CNN, and not read this publication in the future.

Thank you for your consideration, and I hope to see less word count about politics and more interesting content about Linux.

—G. Powers

Doc Searls replies: *Grant, I assume you are writing in response to my "Debugging Democracy" column in the January 2017 issue—the one and only time in two decades of writing for* Linux Journal *that I've ever brought up politics (or at least that I remember).*

So you know, I've already responded to similar pushback from other

readers. Here is what I wrote in response to a reader named Mark:

> *Mark is right. I do owe readers an apology. By calling Donald Trump a troll (take a look at the Wikipedia definition of Internet troll (https://en.wikipedia.org/wiki/Internet_troll), and draw your own conclusions), I was being a troll as well. Even though trolling wasn't my intent, that has been the effect so far: every response to my January column, both here and on our website, has been as negative as Mark's, and for the same reasons.*
>
> *Opening with that remark also failed to support the main purpose of that column, which was to call for help in rescuing journalism—and real journals such as this one—from drowning in a sea of "content", way too much of which is crap routed by algorithms aimed by surveillance-gathered data into echo chambers of the like-minded. This has the effect of increasing enmity and blame toward those in echo chambers with opposing sympathies, which is worse than dangerous in democratic societies, because it tears apart the center spaces of basic agreement those societies require. You can see how this looks in* The Wall Street Journal*'s Blue Feed, Red Feed site (http://graphics.wsj.com/blue-feed-red-feed), subtitled "See Liberal Facebook and Conservative Facebook, Side by Side".*
>
> *I am sure most of the systems driving us into hostile camps are built on Linux. (Isn't everything now?) So I don't think I'm off base calling for help here.*

I believe that was published (with Mark's letter) in the March 2017 issue.

I hope this addresses your concerns. If not, let us know.

Doc Searls' Columns

I hope this note finds you well and enjoying the riches of all things Linux!

This is just a short note of appreciation for Doc Searls and his monthly *Linux Journal* columns. I suspect some readers are occasionally puzzled about the nature of those columns—they're not exactly

technical "how-to" sorts of things.

I think it's important to keep in mind the global picture, and I'm glad that *LJ* sees fit to give voice to such important issues. It would be simple to take the low road and just keep to the geek stuff. Thanks for continuing to take the high road!

—David Klann

Linux Desktop Use Case (from De Bortoli Wines, Australia)

Reaching out to Doc Searls as a recent article of his lamented the state of the Linux Desktop. I thought you might be interested in connecting with a company with Linux as the default desktop—since 2004!

<<** Warning—shameless propaganda ahead! **>>

From 2004: http://www.computerworld.com.au/article/5606/de_bortoli_wines_gets_taste_linux and http://www.google.com.au/search?hl=en&q=de+bortoli+open+standards+or+open+source+or+Linux.

—Bill

Doc Searls and Content

Regarding Doc Searls' "The Problem with 'Content'" in the March 2017 issue: nice editorial, but I think you are missing another facet of what happened to the "media" out there—namely that of partial news.

I used to read a lot of newspapers when I was younger. Over time, it became impossible to ignore a certain bias in most media outlets' reporting. And as time went on, it became more and more difficult to keep *paying* for such reporting.

One needs to look no further than the hyperventilating meltdown that the "media" has suffered under President Trump to understand that reporters have intentionally left about half of the population behind. Certainly other factors are at play, but I'm certain things wouldn't be quite as bad as they are if the "media" had not alienated half of its potential readership.

—Aki Korhonen

Doc Searls replies: *Thanks, Aki. Good points.*

I think all news is partial, in at least two meanings of the word: it's both incomplete and biased in some way. I also think the internet has utterly changed all the old media outlets by supporting countless new ones, while social media on the net has driven coverage and conversation into echo-systems that not only don't talk with each other, but distrust and dislike each other more and more, as they get fed one-sided "content", because that's what algorithms send to them. As I mentioned previously, to see this at work, check out The Wall Street Journal*'s "Blue Feed/Red Feed":* http://graphics.wsj.com/blue-feed-red-feed*. Kinda scary. In the old media world, there was common ground. In the new one, it's pretty much gone.*

I agree that big-name big-city papers and broadcasters ignored much of the country, roughly since Bush the Elder, which is why many people in

those places feel left behind. But other media came in and served those people, who not only just elected a president they prefer, but also gave the GOP a majority in both houses of Congress. Hit SCAN on your radio, even in major markets, and most of the non-sports talk you'll hear is hard right. When I'm in red states, it seems like every establishment that has TVs for patrons (even hospital waiting rooms) is playing Fox News.

At this point, however, both of the old media sides are in trouble, because the internet isn't just changing every media game; it's inviting many new ones, most of which we haven't seen yet.

Shotcut

I'd love for you to review Shotcut video editor. I recently used it to edit ~30 hours of old VHS tapes that were digitally captured into about ten hours to play in a loop on an RPi2 running Kodi as background/icebreaker for a 25th reunion party. It got people talking and reminiscing, and was a great success.

I found parts of the UI clunky, and some of the default timeline behaviors when cutting were less than helpful, but it never crashed. It did lock up once while rendering for some mysterious reason, but I just killed it and restarted it with the saved *.xml project file.

I was impressed. The best thing is it's totally self-contained—just unzip the archive, run the supplied startup script, and off you go. No dependency headaches! All Linux software should aspire to this!

—Walter B. Kulecz

Shawn Powers replies: *Thanks for the heads up. I've never tried Shotcut, but I'll give it a whirl. I'll try to write up a review as well, depending on how my experience goes.*

New F150!

To Shawn Powers: Can you post some photos of your new F150 rig in the next issue of *LJ*? (Just checking if you're making that one up.)

—David

Shawn Powers replies: *David, not only do I really have an F-150, but check out the license plate!*

Shawn's F150—check out the license plate!

WRITE *LJ* A LETTER

We love hearing from our readers. Please send us your comments and feedback via http://www.linuxjournal.com/contact.

PHOTOS

Send your Linux-related photos to ljeditor@linuxjournal.com, and we'll publish the best ones here.

RETURN TO CONTENTS

PREVIOUS
Letters

NEXT
Editors' Choice

diff -u

What's New in Kernel Development

Firmware support has become more and more difficult to maintain over time, especially as more and more features have been added. Some features aren't even about loading firmware so much as just doing something that's more easily done at the same time as loading the firmware. And, whenever the firmware API gets updated, the patch has to include updates to all user code that uses that particular programmer interface. Over time, this tends to make the patches bigger and more error-prone overall.

Luis R. Rodriguez recently proposed a new firmware API, not quite a total replacement of the existing code, but something that would at least make more sense and tolerate updates more easily. At the same time, the new code would leave open the question of certain thorny problems, such as what to do when a particular piece of firmware doesn't work. What's the fallback procedure? For this, he described the existing code as "hairy" and didn't want to touch it until various other issues could be resolved. For example, he said that the kernel's **init** code contained race conditions that would have an impact on any attempt to fix up the firmware fallback implementation, so the one would have to wait for the other.

Various folks like **Greg Kroah-Hartman** and **Bjorn Andersson** had suggestions and objections. In particular, Bjorn wanted the old firmware API to go away at some point and be fully replaced by the new interface. But, Luis said the two would have to coexist for the foreseeable future, although he did add that the old interface would become static, and all new fixes and updates would go into the new API.

Hardware acceleration involves performing certain work in hardware that was specifically built for that purpose, as opposed to doing the same work using the standard opcodes available on a general-purpose CPU. In terms of efficiency, all else being equal, specialized hardware beats the pants off general-purpose CPUs.

Binoy Jayan recently wanted to migrate some of the kernel's **crypto code** into hardware to take advantage of that speedup opportunity—specifically the **initialization vector** (IV) routines in the **dm-crypt.c** file. But, **Milan Broz** warned against moving the code out of dm-crypt.c, because it would make it harder for the crypto team to modify the key data structures in the future, if they so desired. Also, he said, some of the IV generator code was hacky and risky, and it shouldn't be considered good enough to migrate into hardware.

Ultimately, Binoy's code became more and more controversial, as folks like **Ondrej Mosnáçek** proposed completely different solutions to the problems Binoy wanted to address.

By the end of the discussion, hardware acceleration remained an option for the crypto IV routines, but there still was no agreement on the exact implementation.

The quest to access more and more memory is ongoing. **Nikita Yushchenko** recently pointed out that while **PCI** devices potentially could support up to 64-bit DMA (direct memory access) addressing, some of the PCI code, such as host bridge, had software limitations that prevented it. Nikita wanted at least to prevent PCI devices from claiming the ability to access that much memory, if it couldn't in reality.

During the course of discussion, however, and particularly with **Arnd Bergmann**, who'd written his own patch to address the issue in a different way, it turned out that Nikita wasn't entirely sure where the RAM access limitations really were. It ended up being a thorny question.

Arnd and Nikita pursued the problem together, each cursing loudly (and loudly agreeing with each other) over the horribleness of the API.

The discussion ended with only an incomplete understanding of the problem, but at least the question had been identified. The issue of how best to allow PCI devices to access 64-bit DMA addresses remains open.

The kernel boot process is one of the scariest parts of the whole kernel. Trying to support every CPU ever made, including those with hardware errors, mis-features and various other design flaws, is quite simply insane. It should be no surprise that efforts to improve the boot process tend to be highly controversial.

Trying to support the **multiboot specification**, for example, turns out to have all kinds of pitfalls. **Chao Peng** tried to do this recently, and **H. Peter Anvin** offered strenuous objection. He said:

> Multiboot has a fundamentally broken assumption, which is to do certain work for the kernel in the bootloader. This is fundamentally a bad idea, because you always want to do things in the latest step possible during the boot process, being the most upgradeable, and have the interface as narrow as possible. Therefore, using Multiboot is actively a negative step. It is declared an "Open Standard" but anything can be such declared; it really is a claim that "everything should work like Grub."

The debate was not resolved during this email thread, but typically the boot specification would need to address the kernel folks' objections before any code would be accepted.—Zack Brown

Spend Bitcoin Anywhere

I've written about Bitcoin several times during the past few years, and I still love the technology. I am a little disturbed by the amount of electricity the Bitcoin blockchain consumes using dirty power sources, but that's another discussion altogether. Although there are many places to spend Bitcoin directly, and services like Purse.io exist that allow you to spend Bitcoin at Amazon, what if you want to buy a pack of gum at the local gas station?

I recently ordered two different Bitcoin debit cards. One card is from BitPay (https://bitpay.com/card), and one is from Shift (https://www.shiftpayments.com/card). They both conceptually do the same thing, which is convert your Bitcoin into currency that can be spent anyplace that accepts debit cards. They work slightly differently in function though.

The BitPay card is a "reloadable" debit card that allows you to add US Dollars to your card. When you load the card, Bitcoin is converted at the current price, and the dollar amount is stored in your account. Once the card is loaded, Bitcoin is out of the equation, and fluctuating prices don't matter. If you want to know exactly how much money you have on your card, the BitPay card is the way to go.

In contrast, the Shift card doesn't have any money loaded onto it. Rather, the Shift card connects to a Coinbase account, and at the time of purchase, your Bitcoin is converted to US dollars. This is actually "cleaner" than the BitPay method, but the volatility of Bitcoin can mean your actual available money isn't consistent. If Bitcoin tanks, so does your buying ability with the Shift card.

Each card was $10 to buy, and neither has an ongoing fee to use. The transactions don't cost anything, and the only fees are when one of the cards is used at an ATM to get cash. Considering that you instantly can get cash from an ATM from Bitcoin, however, the small fee associated with the process isn't too difficult to accept.

If you've been avoiding digital currency because you don't have any way to spend it, I urge you to check out one or both of these cards. There are other options, but these seemed like the best deal, and I've personally used both.—Shawn Powers

Gaming Like It's 1989

It's no secret that I love classic gaming. It seems like every other month, I write about an emulation project or some online version of a 1980s classic. The system that defined my youth was the Nintendo Entertainment System, or the NES. Its chunky rectangle controller and two-button setup may seem simple today, but back then, it was revolutionary. My hands still even form to the awkward controllers automatically like they did back in middle school.

Knowing that people like me exist, and that we're now old enough to buy things, Nintendo recently released its NES Classic Edition. Although they're still absurdly hard to find, I managed to buy one. And for anyone wondering whether the tiny replica is worth the $60 (okay, I paid $80), if the NES defined your youth, I would say *yes*!

I was worried the controller wouldn't feel like the original. I read a few reviews that said they felt too light or cheaper than the old

ones. Well, I have both original controllers for my emulation machine that I wrote about a few months back and the controller that came with the NES Classic Edition, and I can say they both feel about the same. Also, although the gameplay isn't any different on the NES Classic Edition versus my emulation machine, I'm actually quite happy to pay for the "proper" device and give Nintendo money. I know ROMs are easy to find, but the only reason I download them illegitimately is that I can't buy them legally. Now, at least for 30 of the best games, I can!

I'm a hacker at heart, so although I urge you to buy the NES Classic if you're into that sort of gaming, I also want to play a few games that are not included. Thankfully, the NES Classic is super easy to hack. It's possible to hack the device to add ROMs manually, but there's also a great open-source tool called hakchi2 that will do all the heavy lifting for you. From what I can tell, it's a Windows-only program, but if you want a simple way to add a few ROMs, it's the way to go: https://github.com/ClusterM/hakchi2.

The hardest part? Finding an NES Classic Edition in stock. Good luck fellow gamers.—Shawn Powers

Image Processing on Linux

I've looked at several scientific packages in this space that generate nice graphical representations of your data and work, but I've not gone in the other direction much. So in this article, I cover a popular image processing package called ImageJ. Specifically, I am looking at Fiji (https://imagej.net/Fiji), an instance of ImageJ bundled with a set of plugins that are useful for scientific image processing.

The name Fiji is a recursive acronym, much like GNU. It stands for "Fiji Is Just ImageJ". ImageJ is a useful tool for analyzing images in scientific research—for example, you may use it for classifying tree types in a landscape from aerial photography. ImageJ can do that type categorization. It's built with a plugin architecture, and a very extensive collection of plugins is available to increase the available functionality.

The first step is to install ImageJ (or Fiji). Most distributions will have a package available for ImageJ. If you wish, you can install it that way and then install the individual plugins you need for your research. The other option is to install Fiji and get the most commonly used plugins at the same time. Unfortunately, most Linux distributions will not have a package available within their package repositories for Fiji. Luckily, however, an easy installation file is available from the main website. It's a simple zip file, containing a directory with all of the files required to run Fiji. When you first start it, you get only a small toolbar with a list of menu items (Figure 1).

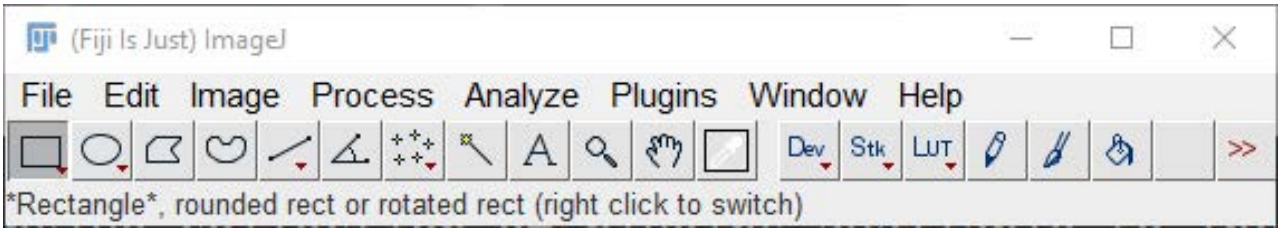


Figure 1. You get a very minimal interface when you first start Fiji.

If you don't already have some images to use as you are learning to work with ImageJ, the Fiji installation includes several sample images. Click the File→Open Samples menu item for a dropdown list of sample images (Figure 2). These samples cover many of the potential tasks you

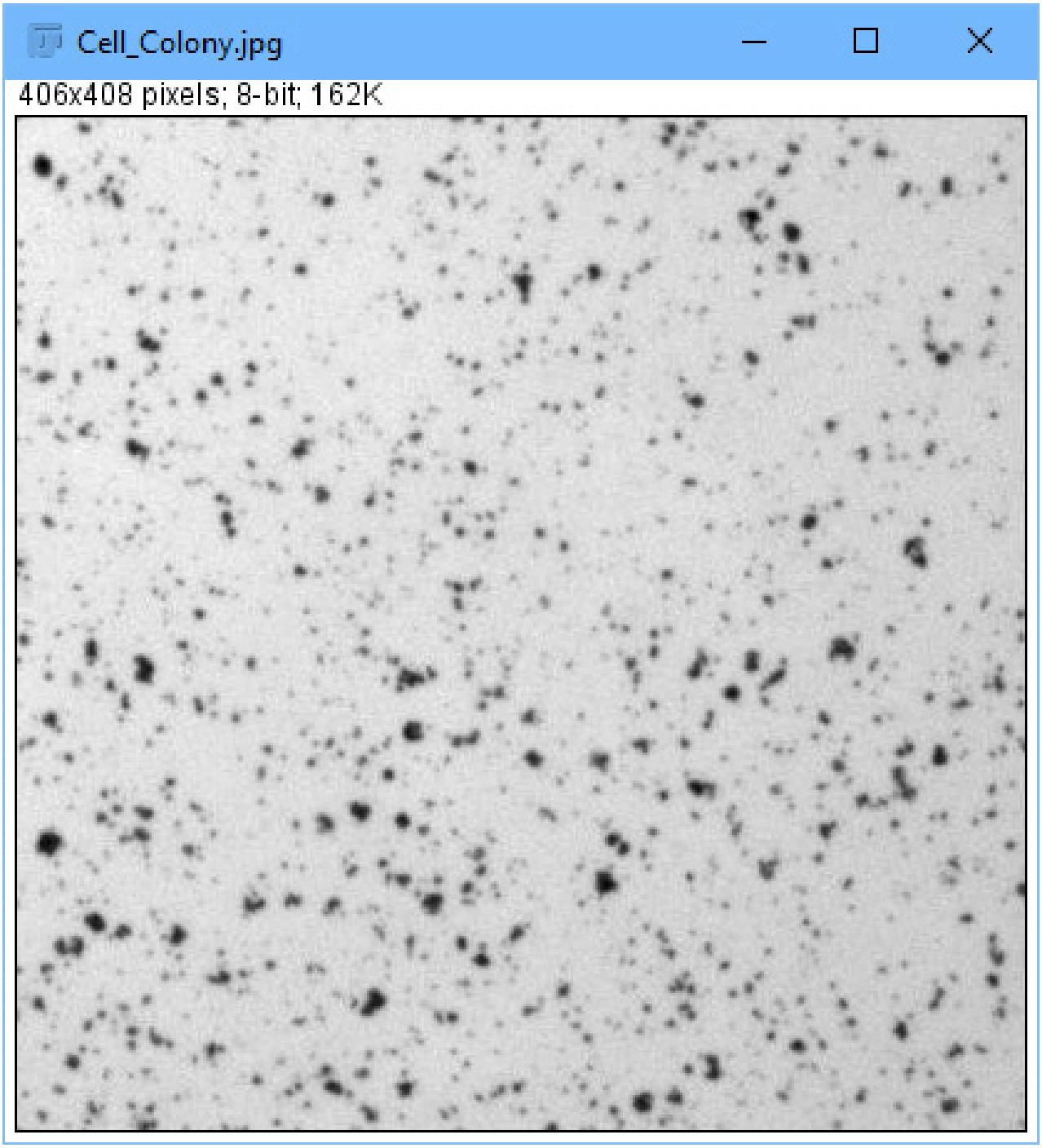


Figure 2. Several sample images are available that you can use as you learn how to work with ImageJ.

might be interested in working on.

If you installed Fiji, rather than ImageJ alone, a large set of plugins already will be installed. The first one of note is the autoupdater plugin. This plugin checks the internet for updates to ImageJ, as well as the installed plugins, each time ImageJ is started.

All of the installed plugins are available under the Plugins menu item. Once you have installed a number of plugins, this list can become a bit unwieldy, so you may want to be judicious in your plugin selection. If you want to trigger the updates manually, click the Help→Update Fiji menu item to force the check and get a list of available updates (Figure 3).

Now, what kind of work can you do with Fiji/ImageJ? One example is doing counts of objects within an image. You can load a sample by clicking File→Open Samples→Embryos.

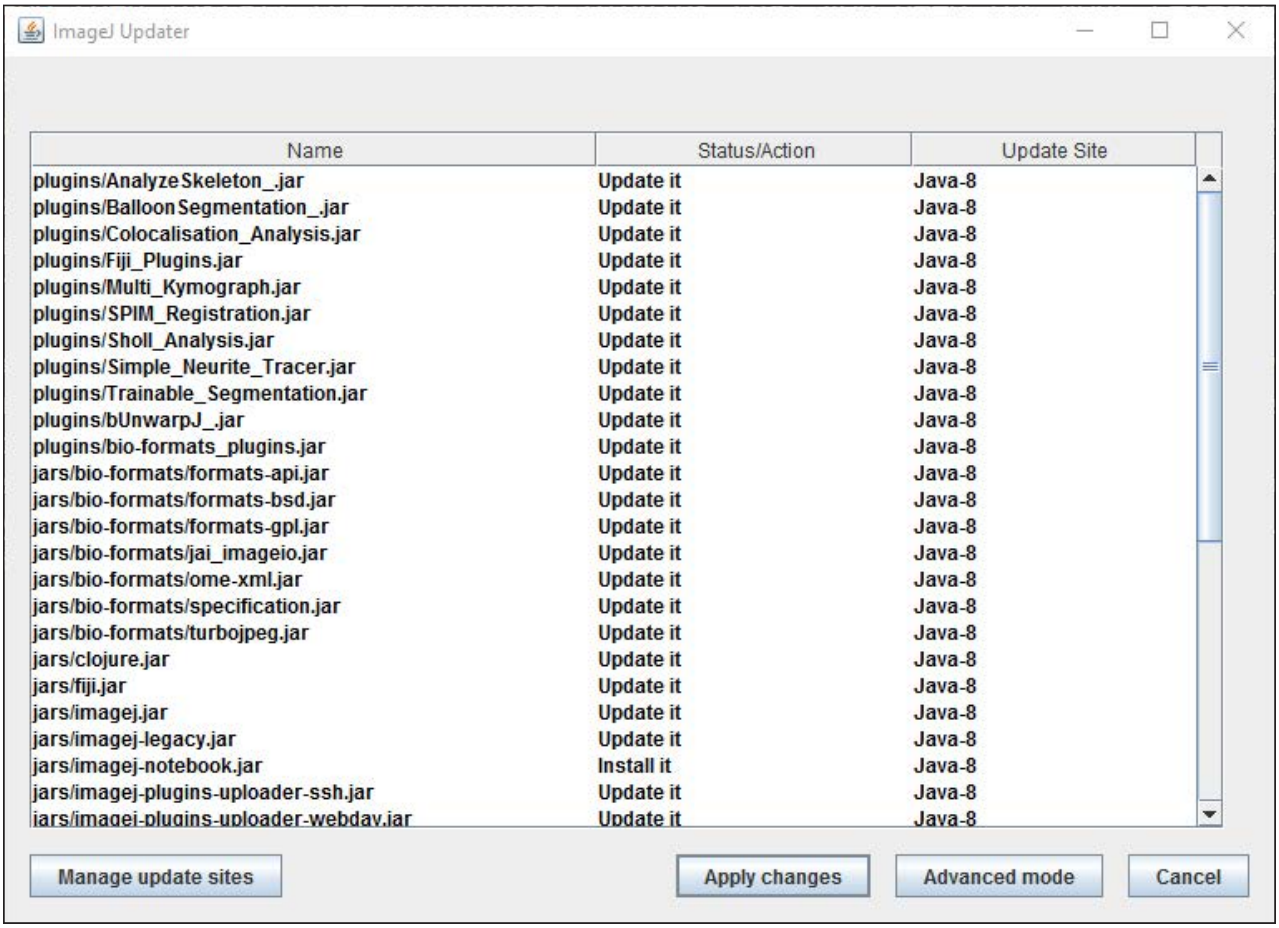


Figure 3. You can force a manual check of what updates are available.

The first step is to set a scale to the image so you can tell ImageJ how to identify objects. First, select the line button on the toolbar and draw a line over the length of the scale legend on the image. You then can select Analyze→Set Scale, and it will set the number of pixels that the scale legend occupies (Figure 5). You can set the known distance to be 100 and the units to be "um".

The next step is to simplify the information within the image. Click Image→Type→8-bit to reduce the information to an 8-bit gray-scale image. To isolate the individual objects, click Process→Binary→Make Binary to threshold the image automatically (Figure 6).

Before you can count the objects within the image, you need to remove artifacts like the scale legend. You can do that by using the

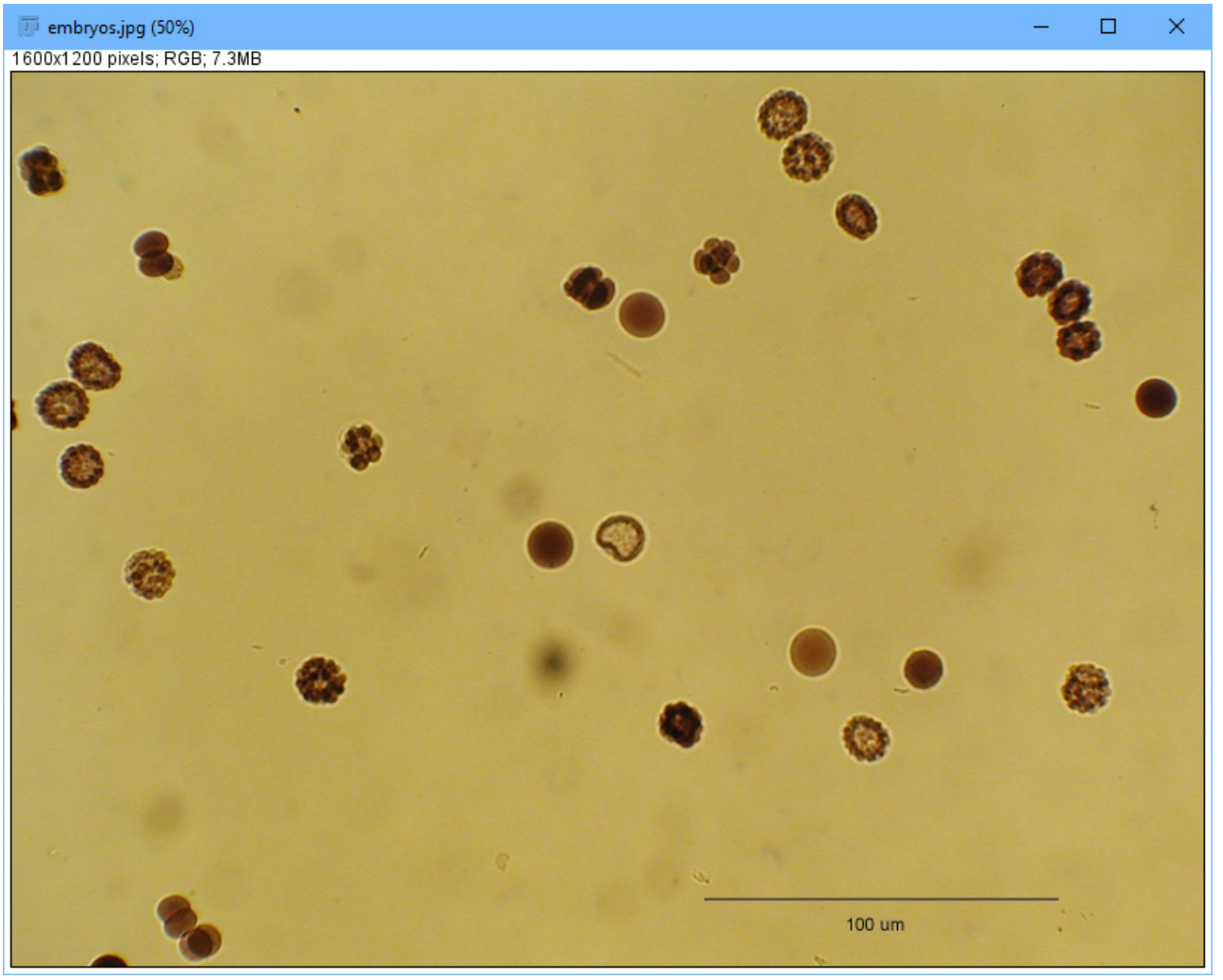


Figure 4. With ImageJ, you can count objects within an image.

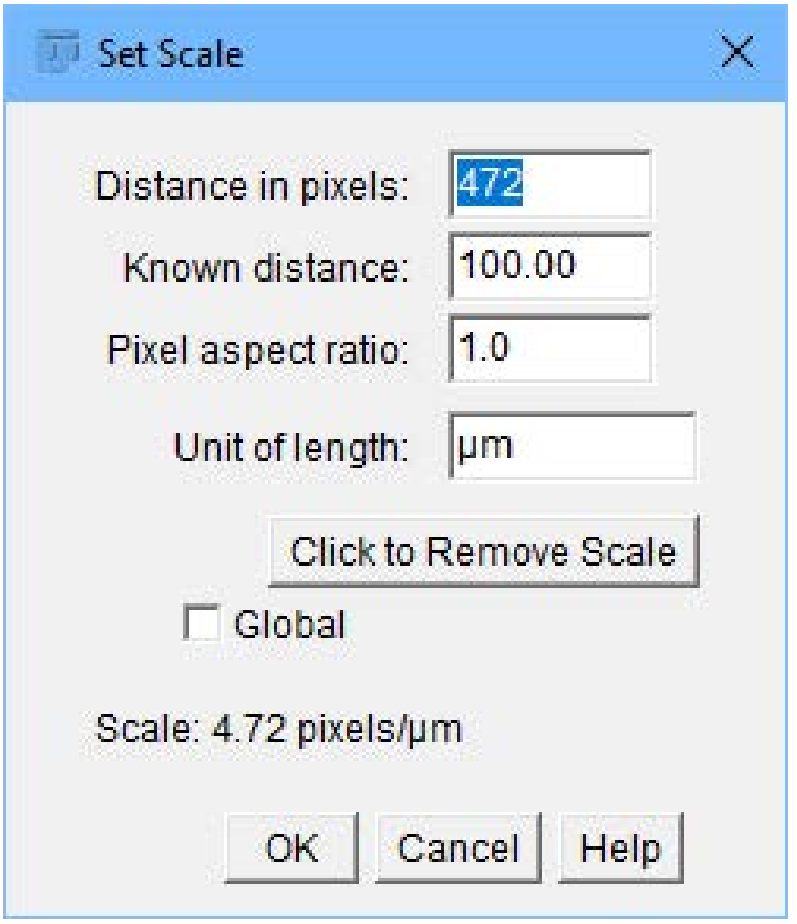


Figure 5.
For many image analysis tasks, you need to set a scale to the image.

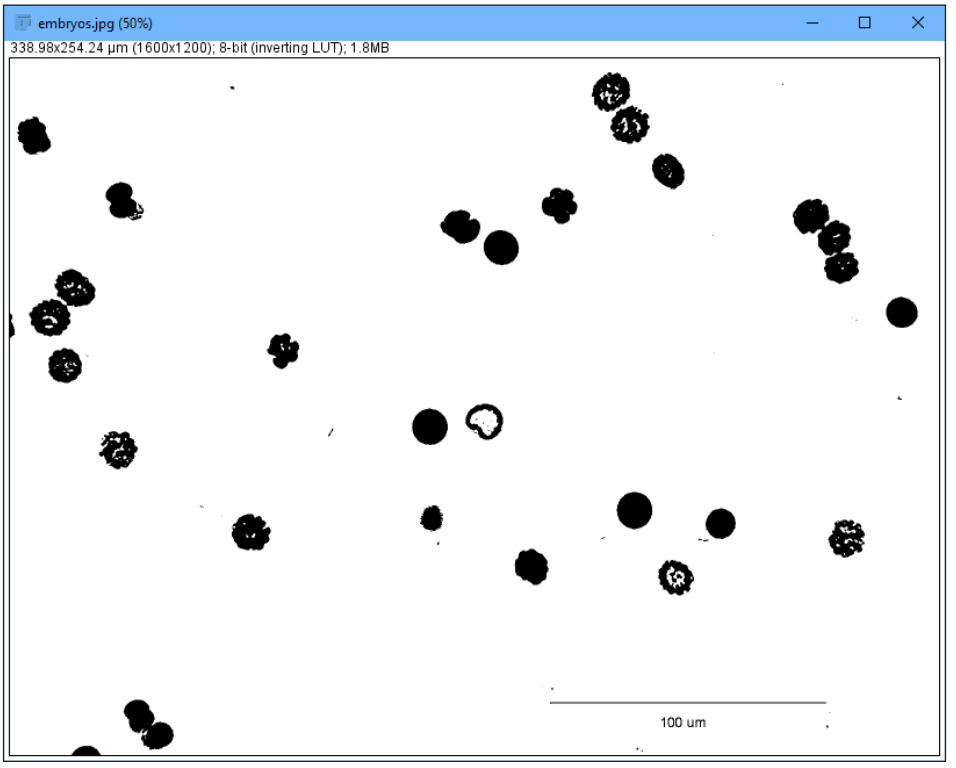


Figure 6.
There are tools to do automatic tasks like thresholding.

rectangular selection tool to select it and then click Edit→Clear. Now you can analyze the image and see what objects are there.

Making sure that there are no areas selected in the image, click Analyze→Analyze Particles to pop up a window where you can select the minimum size, what results to display and what to show in the final image (Figure 7).

Figure 8 shows an overall look at what was discovered in the summary results window. There is also a detailed results window for each individual particle.

Once you have an analysis worked out for a given image type, you often need to apply the exact same analysis to a series of images. This series may number into the thousands, so it's typically not something you will

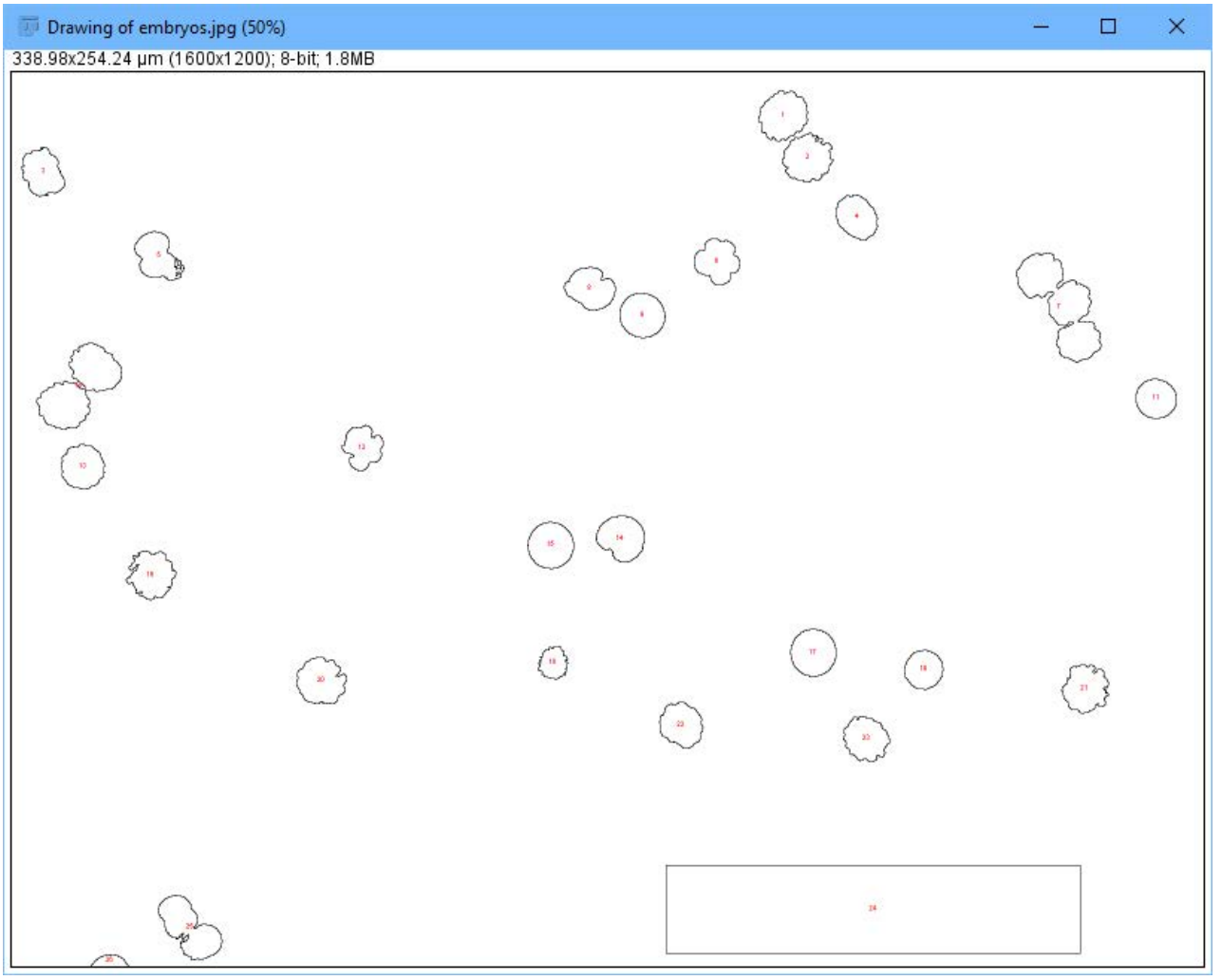


Figure 7. You can generate a reduced image with identified particles.

want to repeat manually for each image. In such cases, you can collect the required steps together into a macro so that they can be reapplied multiple times. Clicking Plugins→Macros→Record pops up a new window where all of your subsequent commands will be recorded. Once all of the steps are finished, you can save them as a macro file and rerun them on other images by clicking Plugins→Macros→Run.

If you have a very specific set of steps for your workflow, you simply can open the macro file and edit it by hand, as it is a simple text file. There is actually a complete macro language available to you to control the process that is being applied to your images more fully.

If you have a really large set of images that needs to be processed, however, this still might be too tedious for your workflow. In that case, go to Process→Batch→Macro to pop up a new window where you can set up your batch processing workflow (Figure 9).

From this window, you can select which macro file to apply, the source directory where the input images are located and the output

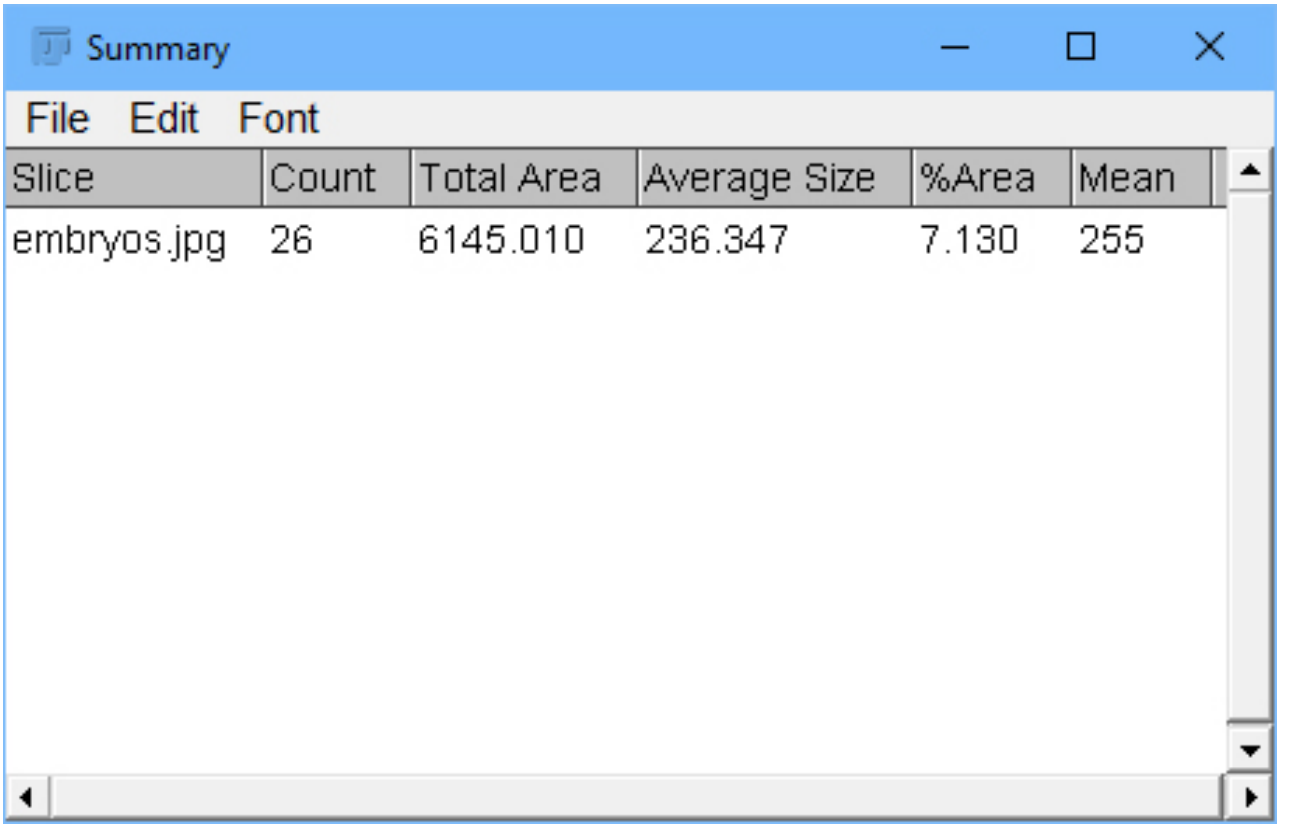


Figure 8. One of the output results includes a summary list of the particles identified.

directory where you want the output images to be written. You also can set the output file format and filter the list of images being used as input based on what the filename contains. Once everything is done, start the batch run by clicking the Process button at the bottom of the window.

If this is a workflow that will be repeated over time, you can save

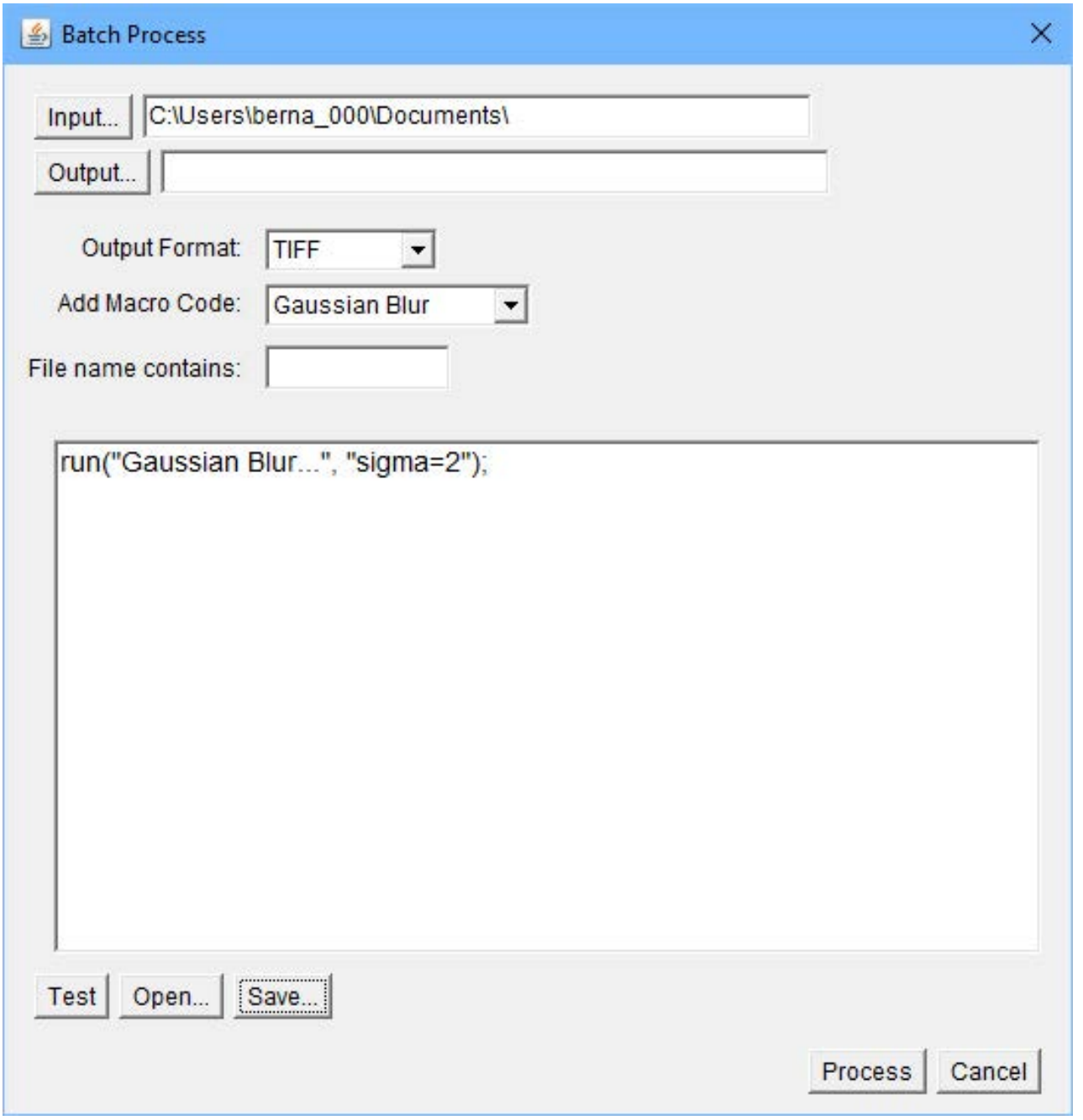


Figure 9. You can run a macro on a batch of input image files with a single command.

the batch process to a text file by clicking the Save button at the bottom of the window. You then can reload the same workflow by clicking the Open button, also at the bottom of the window. All of this functionality allows you to automate the most tedious parts of your research so you can focus on the actual science.

Considering that there are more than 500 plugins and more than 300 macros available from the main ImageJ website alone, it is an understatement that I've been able to touch on only the most basic of topics in this short article. Luckily, many domain-specific tutorials are available, along with the very good documentation for the core of ImageJ from the main project website. If you think this tool could be of use to your research, there is a wealth of information to guide you in your particular area of study.

—Joey Bernard

RETURN TO CONTENTS

THEY SAID IT

Everything that I understand, I understand only because I love.
—Leo Tolstoy

The human brain is unique in that it is the only container of which it can be said that the more you put into it, the more it will hold.
—Glenn Doman

Fear does not have any special power unless you empower it by submitting to it.
—Les Brown

Train yourself to let go of the things you fear to lose.
—George Lucas

Let us so live that when we come to die even the undertaker will be sorry.
—Mark Twain

PREVIOUS
UpFront

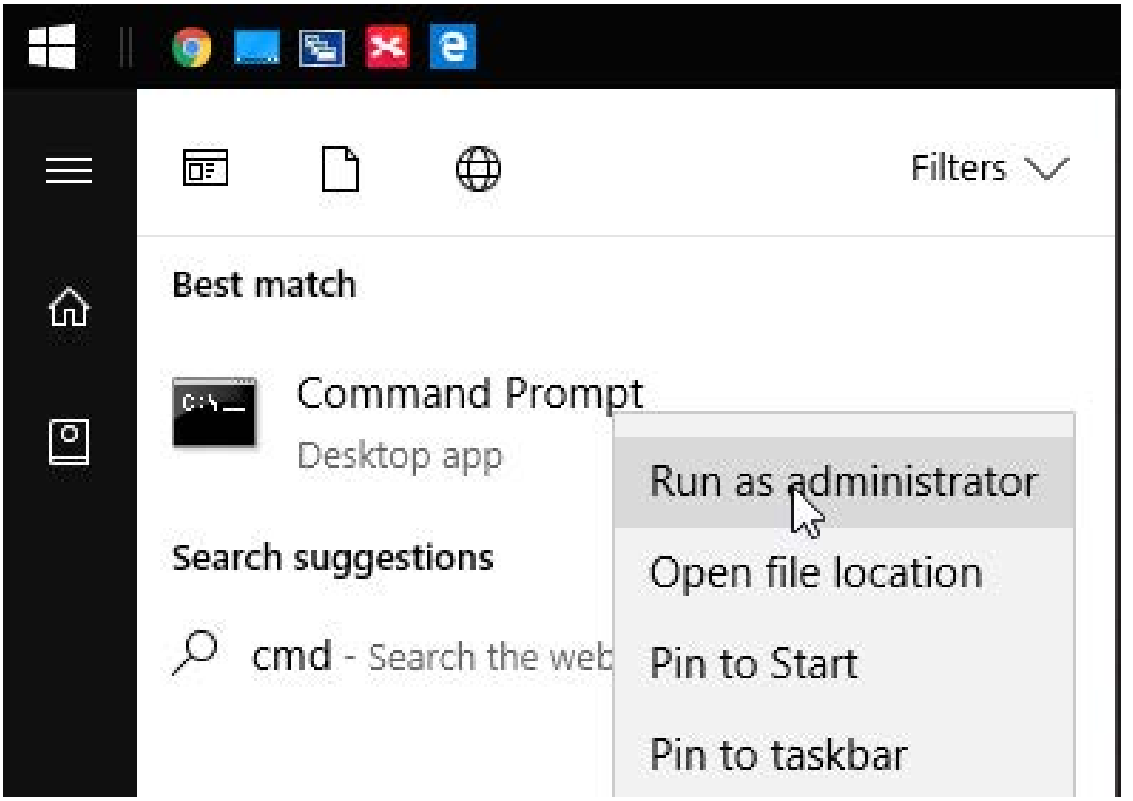
NEXT
Reuven M. Lerner's
At the Forge

Non-Linux FOSS: How to Make Windows Better? Make It Chocolatey!

Once again, my friend and fellow *Linux Journal* club member Kris Occhipinti introduced me to an awesome bit of software. This time, it's an open-source project that brings Linux-like package management to Windows! Don't get me wrong; installing software on Windows isn't difficult, but it's definitely more cumbersome than with Linux. Plus, with Chocolatey (http://chocolatey.org), you can keep your installed packages up to date as easily as you can with Linux.

There is an open-source version of Chocolatey and paid versions. With the open-source version, you can install and maintain all the community packages, which for me is plenty. Literally thousands of software packages are available to install with a simple command-line entry. And unlike Cygwin (a wonderful program as well), Chocolatey installs the same Windows applications you'd install if you downloaded the installers and went through the process on your own.

Installation on Windows can be done via the command prompt (cmd.exe) or via Powershell. If you open the command prompt as administrator (right-click, open as administrator, see screenshot),



you can install with:

```
@powershell -NoProfile -ExecutionPolicy Bypass -Command
 ➥"iex ((New-Object System.Net.WebClient).DownloadString
➥('https://chocolatey.org/install.ps1'))" && SET
 ➥"PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"
```

Or even better, visit https://chocolatey.org/install for more options and a chance to look at the installation script before installing. The site actually recommends looking at the installation code before running it to make sure it's safe. That doesn't make me less confident of the code, but it makes me happy to see smart security choices.

So, thanks to making Windows a bit more like Linux and easing the process of keeping your software up to date, Chocolatey earns this month's Editors' Choice award. If you use Windows, head over to the website and check out this awesome system. It's especially useful for brand-new Windows installs, because managing all your third-party software with a single tool is wonderful. Thanks again, Kris!—Shawn Powers

RETURN TO CONTENTS

Novelty and Outlier Detection

Which of these data points doesn't belong? Machine learning can tell you.

REUVEN M. LERNER

Reuven M. Lerner, a longtime Web developer, offers training and consulting services in Python, Git, PostgreSQL and data science. He has written two programming ebooks (*Practice Makes Python* and *Practice Makes Regexp*) and publishes a free weekly newsletter for programmers, at http://lerner.co.il/newsletter. Reuven tweets at @reuvenmlerner and lives in Modi'in, Israel, with his wife and three children.

PREVIOUS
Editors' Choice

NEXT
Dave Taylor's
Work the Shell

IN MY THE LAST FEW ARTICLES, I've looked at a number of ways machine learning can help make predictions. The basic idea is that you create a model using existing data and then ask that model to predict an outcome based on new data.

So, it's not surprising that one of the most amazing ways machine learning is being applied is in predicting the future. Just a few days before writing this piece, it was announced that machine learning models actually might be able to predict earthquakes—a goal that has eluded scientists for many years and that has the potential to save thousands, and maybe even millions, of lives.

But as you've also seen, machine learning can be used to "cluster" data—that is, to find patterns that humans either can't or won't see, and to try to put the data into various "clusters", or machine-driven categories. By asking the computer to divide data into distinct groups, you gain the opportunity to find and make use of previously undetected patterns.

Just as clustering can be used to divide data into a number of coherent groups, it also can be used to decide which data points belong inside a group and which don't. In "novelty detection", you have a data set that contains only good data, and you're trying to determine whether new observations fit within the existing data set. In "outlier detection", the data may contain outliers, which you want to identify.

Where could such detection be useful? Consider just a few questions you could answer with such a system:

- Are there an unusual amount of login attempts from a particular IP address?
- Are any customers buying more than the typical number of products at a given hour?
- Which homes are consuming above-average amounts of water during a drought?
- Which judges convict an unusual number of defendants?
- Should a patient's blood tests be considered normal, or are there outliers that require further checks and examinations?

In all of those cases, you could set thresholds for minimum and maximum values and then tell the computer to use those thresholds in determining what's suspicious. But machine learning changes that around, letting the computer figure out what is considered "normal" and then identify the anomalies, which humans then can investigate. This allows people to concentrate their energies on understanding whether the outliers are indeed problematic, rather than on identifying them in the first place.

So in this article, I look at a number of ways you can try to identify outliers using the tools and libraries that Python provides for working with data: NumPy, Pandas and scikit-learn. Just which technique and tools will be appropriate for your data depend on what you're doing, but the basic theory and practice presented here should at least provide you with some food for thought.

Finding Anomalies

Humans are excellent at finding patterns, and they're also quite good at finding things that don't fit a pattern. But, what sort of algorithm can look at a group of data sets and figure out which is unlike the others?

One simple way to do this is to set a cutoff, often done at one or two standard deviations. For those of you without a background in statistics (or who have forgotten what a "standard deviation" is), it's a measurement of how spread out the data is. For example:

```
>>> a = np.array([10,10,10,10,10,10,10])
>>> print("std = {}, mean = {}".format(a.std(), a.mean()))

std = 0.0, mean = 10.0
```

In the above example, I have a NumPy array containing seven instances of the number ten. People often think of the mean as describing the data, and it does, but it's only when combined with the standard deviation that you can know how much the numbers differ from one another. In this case, they're all identical, so the standard deviation is 0.

In this example, the mean remains the same, but the standard deviation is quite different:

```
>>> a = np.array([5,15,0,20,-5,25,10])
>>> print("std = {}, mean = {}".format(a.std(), a.mean()))

std = 10.0, mean = 10.0
```

Here, the mean has not changed, but the standard deviation has. You can see, from just those two numbers, that although the numbers remain

centered around 10, they also are spread out quite a bit.

One simple way to detect unusual data is to look for all of the values that lie outside of two standard deviations from the mean, which accounts for about 95% of the data. (You can go further out if you want; 99.73% of data points are within three standard deviations, and 99.994% are within four.) If you're looking for outliers in an existing data set, you can do something like this:

```
>>> a = np.array([-5,15,0,20,-5,25,1000])
>>> print(a.std())

347.19282415231044

>>> min_cutoff = a.mean() - a.std()*2
>>> max_cutoff = a.mean() + a.std()*2

>>> print(a[(a<min_cutoff) | (a>max_cutoff)])

array([1000])
```

Sure enough, that found an outlier in the data.

It's even easier if you have a bunch of new data and want to determine whether those values would fit inside or outside your existing data set:

```
>>> new_data = np.array([-5000, -3000, -1000, -500, 20, 60, 500, 800,
>>> 900])
>>> print(new_data[(new_data<min_cutoff) | (new_data>max_cutoff)])

array([-5000, -3000, -1000,   900])
```

The good news is that this is simple—simple to understand, simple to implement and simple to automate.

However, it's also too simple for most data. You're unlikely to be looking at a single-dimensional vector. The baseline (mean) is likely to shift over time. And besides, there must be other, better ways to measure whether something is "inside" or "outside", right?

Getting More Sophisticated

For real-world anomaly detection, you're going to need to improve on a few fronts. You'll need to consider the data and determine what's "in" and what's "out". You'll also need to figure out ways to evaluate your model.

Let's consider novelty detection: there is initial data, and you want to know if a new piece of data would fit inside the existing data or if it would be considered an outlier. For example, consider a patient who comes in with values from a blood test. Do those tests indicate that the patient is normal, because the data's values are similar to the ones you've already seen? Or are those new values statistical outliers, indicating that the patient needs additional attention?

In order to experiment with novelty and outlier detection, I downloaded historic precipitation data for an area of Pennsylvania (Wyncote), just outside Philadelphia, for every day in 2016. Because I'm a scientific kind of guy, I downloaded the data in metric units. The data came from the US government, at https://www.climate.gov/maps-data/dataset/past-weather-zip-code-data-table.

That site contains clear instructions for downloading data from here: https://www.ncdc.noaa.gov/cdo-web/datasets.

It's quite amazing what government data is freely available, and the sorts of analysis you can do with it once you've retrieved it.

I downloaded the data as a CSV file and then used Pandas to read it into a data frame:

```
>>> df = pd.read_csv('/Users/reuven/downloads/914914.csv',
    usecols=['PRCP', 'DATE'])
```

Notice that I was interested only in PRCP (precipitation) and DATE (the date, in YYYYMMDD format). I then manipulated the data to break apart the DATE column and then to remove it:

```
>>> df['DATE'] = df['DATE'].astype(np.str)
>>> df['MONTH'] = df['DATE'].str[4:6].astype(np.int8)
>>> df['DAY'] = df['DATE'].str[6:8].astype(np.int8)
>>> df.drop('DATE', inplace=True, axis=1)
```

Why would I break the date apart? Because it'll likely be easier for models to work with three separate numeric columns, rather than a single date-time column. Besides, having these columns as part of my model will make it easier to understand whether snow in July is abnormal. I ignore the year, since it's the same for every record, which means that it can't help me as a predictor in this model.

My data frame now contains 353 rows—I'm not sure why it's not 365—of data from 2016, with columns indicating the amount of rain (in mm), the date and the month.

Based on this, how can you build a model to indicate whether rainfall on a given day is normal or an outlier?

In scikit-learn, you always use the same method: you import the estimator class, create an instance of that class and then fit the model. In the case of supervised learning, "fitting" means teaching the model which inputs go with which outputs. In the case of unsupervised learning, which I'm doing here, you use "fit" with just a set of inputs, allowing the model to distinguish between inliers and outliers.

Creating a Model

In the case of this data, there are several types of models that I can build. I experimented a bit and found that the `IsolationForest` estimator gave me the best results. Here's how I create and train the model:

```
>>> from sklearn.ensemble import IsolationForest
>>> model = IsolationForest()
>>> model.fit(df)
```

The model now has been trained, so I can find out whether a given amount of rain, on a certain month and day, is considered normal.

To try things out, I check the model against its own inputs:

```
>>> Series(model.predict(df)).value_counts()
```

In the above code, I run `model.predict(df)`. This gives the inputs to the model and asks it to predict whether these are normal, expected values (indicated by 1) or outlier values (indicated by –1). By turning the

result into a Pandas series and then calling `value_counts`, I see:

```
 1    317
-1     36
```

Although it falsely marked 36 days as outliers, maybe those days were unusual. The model certainly would be improved if it had multiple years' worth of data, rather than just one year's worth.

Now what? I can ask the system to make some predictions:

```
for i in range(1, 13):
    print(model.predict([[15, i, 16]]))
```

This will tell whether it's normal to get 15 mm rain on the 16th of each month. The conclusion of the model: yes, it's perfectly normal in February–July, but not so in August–January. What about if there's zero precipitation:

```
for i in range(1, 13):
    print(model.predict([[0, i, 16]]))
```

It turns out that no matter what month, it's never an outlier to have zero rain on the 16th of the month.

Of course, those are just crude tests. The real thing to do is use our old friend `train_test_split`:

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test = train_test_split(df)
>>> model.fit(X_train)
>>> Series(model.predict(X_test)).value_counts()
```

The model did pretty well, given that I didn't even try to tune it:

```
 1    77
-1    12
dtype: int64
```

In other words, given data that should all be classified as inliers, you can see here that the overwhelming majority is indeed classified correctly.

There are other types of estimators you can use as well. In particular, the One-Class SVM estimator has had a good track record of working with input data. That, combined with a larger data set, might well improve the results shown above—although in trying One-Class SVM for this article, I didn't see any such results. It's possible that if I were to add several more years' worth of data, other estimators would work better.

Conclusion

Novelty and outlier detection is (yet another) large, exciting and growing use for machine learning. As usual with machine learning, the problem is not one of coding, but rather of massaging the data into a format that you can use, and then tinkering with model definitions until you find one that predicts or identifies outliers with a high degree of confidence.■

RESOURCES

I used Python (http://python.org) and many parts of the SciPy stack (NumPy, SciPy, Pandas, Matplotlib and scikit-learn) in this article. All are available from PyPI (http://PyPI.python.org) or from SciPy.org (https://www.scipy.org).

The documentation for scikit-learn has some (but not a great deal of) documentation on novelty/outlier detection: http://scikit-learn.org/stable/modules/outlier_detection.html

A simple Python package for detecting anomalies, lsanomaly, is available on PyPI and GitHub: https://github.com/lsanomaly/lsanomaly. It might be worth consideration for simple data sets.

As I mentioned previously, the US government's NOAA (National Oceanic and Atmospheric Administration) site contains a treasure trove of weather and climate data, which you can download for free at https://www.ncdc.noaa.gov/cdo-web/datasets.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

Working with YouTube and Extracting Audio

A fun and simple script to make YouTube work for you.

DAVE TAYLOR

Dave Taylor has been hacking shell scripts on UNIX and Linux systems for a really long time. He's the author of *Learning Unix for Mac OS X* and *Wicked Cool Shell Scripts*. You can find him on Twitter as @DaveTaylor, or reach him through his tech Q&A site: http://www.AskDaveTaylor.com.

PREVIOUS
Reuven M. Lerner's
At the Forge

NEXT
Kyle Rankin's
Hack and /

IN MY LAST FEW ARTICLES, I've been exploring the capabilities of ImageMagick, showing that just because you're working on a command line doesn't mean you're stuck processing only text. As I explained, ImageMagick makes it easy to work with images, adding watermarks and analyzing content far more accurately than with the standard Linux `file` command, and much, much more.

Continuing in a similar vein, I want to look at audio and video in this article. Well, maybe "listen"

Let's start with the most basic functionality: downloading a video from YouTube so you can watch it on your Linux system.

to audio and "look" at video, but again, I'm still focusing on the command line, so in both instances, player/viewer apps are required.

YouTube to MP3 Audio

As someone who watches a lot of lectures online, I'm also intrigued by the online services that can extract just the audio portion of a YouTube or Vimeo video and save it as an MP3. Listening to a lecture while driving is far safer than trying not to watch a video on the move, for example.

Since there are so many live concert performances online, many people also like to use a video-to-MP3 service to add those songs to their music libraries.

Note: be leery of copyright issues with any download and conversion of content. Just because it's on Vimeo, YouTube or other online service, doesn't mean you have permission to extract the audio or even download it and save it on your computer.

Let's start with the most basic functionality: downloading a video from YouTube so you can watch it on your Linux system. There are a lot of browser plugins and even websites devoted to this task, but who wants to risk malware or be plagued by porn site ads? Yech.

Fortunately, there's a terrific public domain program called youtube-dl on GitHub that covers all your needs. At its most basic, it lets you download video content from YouTube and a variety of other online video repositories, but as you'll learn, it can do quite a bit more.

You can grab a copy for your system at https://github.com/rg3/youtube-dl/blob/master/README.md.

Let's start by downloading a copy of one of my own YouTube videos. It's a review of the splendid 1More quad-driver headphones, and its URL is https://www.youtube.com/watch?v=BFL1E77hTHQ.

As an aside: I have a YouTube channel where I review consumer electronics and gadgets. You should subscribe! Find all my videos at http://youtube.com/askdavetaylor.

YouTube has a bunch of ways it can assemble a URL, however, including using its URL-shortener youtu.be, but fortunately, youtube-dl can handle the variations.

Downloading a copy of the video to the current working directory is now as simple as:

```
youtube-dl 'https://www.youtube.com/watch?v=BFL1E77hTHQ'
```

The full output of the command is a bit, um, hairy, however:

```
$  youtube-dl 'https://www.youtube.com/watch?v=BFL1E77hTHQ'
[youtube] BFL1E77hTHQ: Downloading webpage
[youtube] BFL1E77hTHQ: Downloading video info webpage
[youtube] BFL1E77hTHQ: Extracting video information
[youtube] BFL1E77hTHQ: Downloading MPD manifest
WARNING: Requested formats are incompatible for merge and
will be merged into mkv.
[download] Destination: 1More Quad Driver In-Ear Headphones
Reviewed-BFL1E77hTHQ.f137.mp4
[download] 100% of 118.74MiB in 02:49
[download] Destination: 1More Quad Driver In-Ear Headphones
Reviewed-BFL1E77hTHQ.f251.webm
[download] 100% of 4.81MiB in 00:03
[ffmpeg] Merging formats into "1More Quad Driver In-Ear
Headphones Reviewed-BFL1E77hTHQ.mkv"
Deleting original file 1More Quad Driver In-Ear Headphones
Reviewed-BFL1E77hTHQ.f137.mp4 (pass -k to keep)
Deleting original file 1More Quad Driver In-Ear Headphones
Reviewed-BFL1E77hTHQ.f251.webm (pass -k to keep)
$
```

You can wade through the output messages, but it's the message from companion open-source program ffmpeg that's

most important: `merging formats into ... mkv`.

In other words, the download format of the video is MKV by default. MKV is part of the increasingly popular Matroska Multimedia Container format, and it works with a lot of video players (including VideoLan, aka VLC, my favorite cross-platform video player).

A quick `ls` reveals the result and that the default filename is taken from the title of the video, something that might not be particularly desirable:

```
$ ls -lh *mkv
-rw-r--r--  1 taylor  staff   124M Jan 31 16:56 1More Quad
Driver In-Ear Headphones Reviewed-BFL1E77hTHQ.mkv
```

Do you prefer to specify the output name and have the output file in MP4 (MPEG4) format instead? That's doable:

```
$ youtube-dl -o 1more-review.mp4 -f mp4 \
    'https://www.youtube.com/watch?v=BFL1E77hTHQ'
[youtube] BFL1E77hTHQ: Downloading webpage
[youtube] BFL1E77hTHQ: Downloading video info webpage
[youtube] BFL1E77hTHQ: Extracting video information
[youtube] BFL1E77hTHQ: Downloading MPD manifest
[download] Destination: 1more-review.mp4
[download] 100% of 57.63MiB in 00:27
```

As a bonus, you get less ominous informational messages from the program too, so it's cleaner. And the output, sure enough, is in MP4 format:

```
$ ls -lh *mp4
-rw-r--r--@ 1 taylor  staff  58M Jan 31 16:57 1more-review.mp4
```

As a second bonus, it's also more efficient in its video encoding, so the MP4 version of the downloaded video is only 58M as opposed to the 124M of the MKV-merged version.

So how do you watch it? Most likely, do a double-click and it'll be

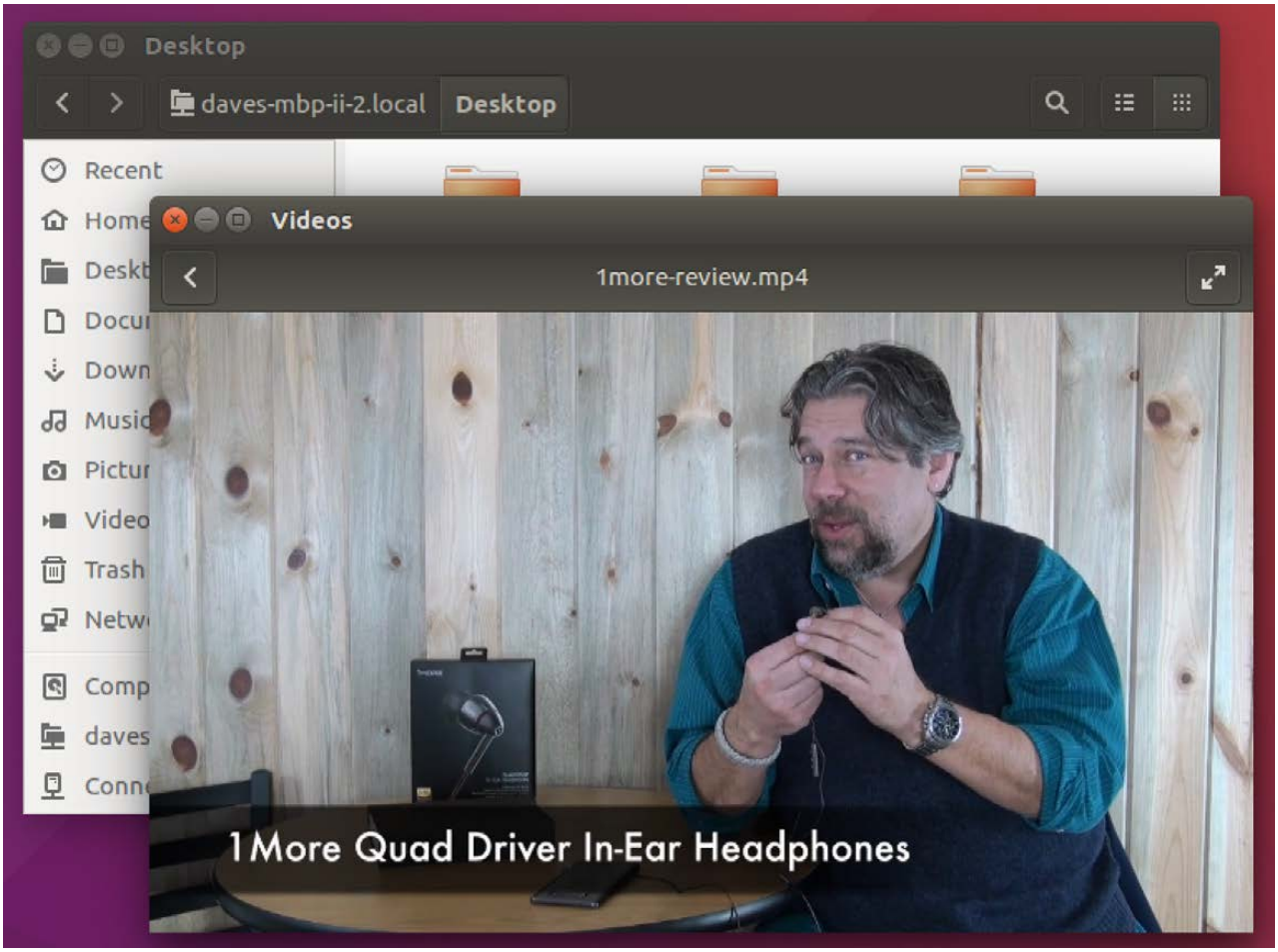


Figure 1. Downloaded YouTube Video Playing in Ubuntu Player

up and running, as shown in Figure 1.

That's easy enough, but the original goal was to be able to extract just the audio component of a YouTube video, so let's look at that task.

Downloading Just the Audio Track

Since I've already started to delve into the command-line options for the youtube-dl program, it's not a leap to find out that there's yet another command-line option that lets you save just the audio portion of a video:

```
$ youtube-dl -x --audio-format mp3 \
    'https://www.youtube.com/watch?v=BFL1E77hTHQ'
[youtube] BFL1E77hTHQ: Downloading webpage
[youtube] BFL1E77hTHQ: Downloading video info webpage
```

```
[youtube] BFL1E77hTHQ: Extracting video information
[youtube] BFL1E77hTHQ: Downloading MPD manifest
[download] Destination: 1More Quad Driver In-Ear Headphones
Reviewed-BFL1E77hTHQ.webm
[download] 100% of 4.81MiB in 00:07
[ffmpeg] Destination: 1More Quad Driver In-Ear Headphones
Reviewed-BFL1E77hTHQ.mp3
Deleting original file 1More Quad Driver In-Ear Headphones
Reviewed-BFL1E77hTHQ.webm (pass -k to keep)
$ ls -lh *mp3
-rw-r--r--  1 taylor  staff   4.0M Jan 31 18:22 1More Quad
Driver In-Ear Headphones Reviewed-BFL1E77hTHQ.mp3
```

That's easy enough, and the output is delightfully small: 4MB total. The problem is, there's the same awkward naming issue, so the addition of `-o output-filename` definitely will be a win. But, really, youtube-dl makes these tasks trivially easy, as long as you're willing to figure out all of its command-line options.

Writing a Wrapper Script

Instead of worrying about the obscure command-line flag notation, let's just write a script that does the heavy lifting for you. I'm going to call it ytdl for "youtube download", and by default, it'll accept just a URL and output an MP4 format video file that has the same name as the YouTube shortcut (for example, the above video would become BFL1E77hTHQ.mp4).

Add a second parameter, and that becomes the output filename. Specify the `-a` flag, and it saves audio output only, in MP3 format instead.

Let's start with a usage block if the user forgets to specify anything or just needs a simple reminder:

```
if [ $# -eq 0 ] ; then
  echo "Usage: $(basename $0) {-a} YouTubeURL {outputfile}"
  echo "   where -a extracts the audio portion in MP3 format"
  exit 1
fi
```

That's easy enough. The script is also going to use some predefined combinations of flags to make it easier to write:

```
youtubedl="/usr/local/bin/youtube-dl"
audioflags="-x --audio-format mp3"
videoflags="-f mp4"
flags=$videoflags       # default set of command flags
audioonly=0             # default is audio + video
```

If the user specifies the `-a` flag, `audioonly` will be set to true (that is, 1), and the default flags will switch from video to audio:

```
if [ "$1" = "-a" ] ; then
  audioonly=1
  flags=$audioflags
  shift
fi
```

You'll recall that the `shift` command moves all the parameters "down" one to the left, so `$2` becomes `$1` and so on. It's an easy way to process and discard parameters in a script, of course.

The biggest block of code creates a default output filename from the YouTube URL:

```
if [ $# -eq 1 ] ; then
  # no output filename specified
  outfile=$(echo "$1" | cut -d= -f2)
  if [ $audioonly -eq 1 ] ; then
    outfile="$outfile.mp3"
  else
    outfile="$outfile.mp4"
  fi
else
  outfile="$2"
fi
```

This isn't the most robust code, because it assumes that the URL specified is in a format like the examples used herein, `youtube-yadda-yadaa?value=shortcode`. It extracts the shortcode and simply appends an appropriate filename suffix. There are better ways to do this, but that's okay, this'll work for now. Just realize that your output format might be a bit weird if you have a very different type of YouTube URL or a URL from another site.

And, finally, the actual invocation of the `youtube-dl` command:

```
$youtubedl $flags -o "$outfile" "$1"
```

That's it! Now you can download a video as simply as:

```
$ ytdl 'https://www.youtube.com/watch?v=5yXDzg_QDGw' wiper.mp4
```

And an audio portion with:

```
$ ytdl -a 'https://www.youtube.com/watch?v=5yXDzg_QDGw'
```

Nice, eh?

I've way overrun my space for this column, but this is such a fun and simple script atop a terrific, powerful program, that it's worth it, right? And now you know how to make YouTube work for you, rather than vice versa!■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

Sysadmin 101: Leveling Up

There's more to sysadmin experience than a name. Find out the skills behind these sysadmin titles.

KYLE RANKIN

Kyle Rankin is VP of engineering operations at Final, Inc., the author of many books including *Linux Hardening in Hostile Networks*, *DevOps Troubleshooting* and *The Official Ubuntu Server Book*, and a columnist for *Linux Journal*. Follow him @kylerankin.

PREVIOUS
Dave Taylor's Work the Shell

NEXT
Shawn Powers' The Open-Source Classroom

THIS IS THE FOURTH IN A SERIES OF ARTICLES ON SYSTEMS ADMINISTRATOR FUNDAMENTALS. These days, DevOps has made even the job title "systems administrator" seems a bit archaic like the "systems analyst" title it replaced. These DevOps positions are rather different from sysadmin jobs in the past with a much larger emphasis on software development far beyond basic shell scripting and as a result often are filled with people with software development backgrounds without much prior sysadmin experience.

In the past, a sysadmin would enter the role at a junior level and be mentored by a senior sysadmin on the team, but in many cases these days, companies go quite a while with cloud outsourcing before their first DevOps hire. As a result, the DevOps engineer might be thrust into the role at a junior level with no mentor

around apart from search engines and Stack Overflow posts.

In the first article in this series, I explained how to approach alerting and on-call rotations as a sysadmin. In the second article, I discussed how to automate yourself out of a job. In the third, I covered why and how you should use tickets. In this article, I describe the overall sysadmin career path and what I consider the attributes that might make you a "senior sysadmin" instead of a "sysadmin" or "junior sysadmin", along with some tips on how to level up.

Keep in mind that titles are pretty fluid and loose things, and that they mean different things to different people. Also, it will take different people different amounts of time to "level up" depending on their innate sysadmin skills, their work ethic and the opportunities they get to gain more experience. That said, be suspicious of anyone who leveled up to a senior level in any field in only a year or two—it takes time in a career to make the kinds of mistakes and learn the kinds of lessons you need to learn before you can move up to the next level.

Junior Systems Administrator

Junior sysadmins are early on in their sysadmin training. It might be their first sysadmin job where they are learning everything from scratch, or they might have a few years of experience under their belts. Either way, a few attributes are common among junior sysadmins:

- Tasks will require help from other members of the team to complete.
- They will rely heavily on documentation and may not understand what individual tasks do.
- It may take weeks or even months to be productive at a new job.
- Most of their time will be spent with daily tickets.
- Eventually they might take on a project, but will need quite a bit of help to complete it.

One of the first attributes that defines junior sysadmins is the

amount of outside help they will need to do their jobs. Generally speaking, they will need help and direction to perform day-to-day tasks, especially at first. If you document your routine tasks (and you should!), you will find that junior sysadmins will dutifully follow your procedures step by step, but they may not understand exactly what those steps do. If a task deviates from the norm, or if for some reason a step fails, they will escalate up to a more senior member of the team for help—this is a good thing, because this mentoring is one of the main ways that junior sysadmins build their experience besides making mistakes and fixing them.

It might take sysadmins at this level a few weeks or even months at a new organization until they are productive and can start doing daily sysadmin tasks independently without help. These are great opportunities for a team to audit documentation and for junior members of the team to flag gaps in documentation or places where they are out of date. If you have junior team members add documentation themselves, just make sure that a more senior team member goes over it to make sure it's correct and complete.

A sysadmin's task list is usually divided into two main categories: day-to-day tasks and projects. Junior sysadmins often end up being assigned more of the day-to-day "grunt work", not as a punishment, but just because projects usually require more experience—experience they will get as they master daily tickets.

That said, at some point, it will be important for junior sysadmins to take on their first project. Ideally, this will be a project without a strict deadline, so they can take the time they need to research and get it right. At this level, a more senior team member will need to devote a fair amount of time to act as a mentor and help direct the planning and research for the project and answer any questions.

Both daily tasks and projects are important for junior sysadmins, as it's the mastery of daily tasks and the successful completion of a couple projects that will help prepare junior sysadmins to level up. Each task they master will add a certain level of confidence and proficiency in routine sysadmin tasks, and projects will help develop their research skills and the ability to complete tasks that fall outside a playbook.

Mid-Level Systems Administrator

It can be difficult to draw the exact line where a sysadmin levels up past the junior level. There isn't an exact number of years' experience needed; instead, it has more to do with sysadmins' competency with their craft and their overall confidence and independence. Here are a few attributes that are common to mid-level sysadmins:

- They generally perform day-to-day tasks independently.
- They understand some of the technology behind their routine tasks and don't just parrot commands they see in documentation.
- It takes a few weeks up to a month to be productive at a new job.
- Their time is pretty equally balanced between daily tickets and longer-term projects.
- They are able to come up with new approaches and improvements to existing tasks.
- They can complete simple projects independently and more complex projects with some help from more senior team members.

The main difference between junior sysadmins and mid-level sysadmins has to do with their independence. As sysadmins become more comfortable with servers in general, and the processes within an organization specifically, they start to be able to perform typical tasks by themselves. Mid-level sysadmins should be able to handle all of the normal tasks that are thrown at them without outside help. It's only when they get an odd "curve ball", such as a one-off task that hasn't been done before or some unique emergency, that mid-level sysadmins may need to reach out to the more senior members of the team for some guidance. As with junior sysadmins, this type of help is very important, and it would be a mistake for mid-level sysadmins not to ask for help with odd requests just to try to be "more senior". Asking questions and getting advice from more experienced sysadmins will help them level up. If they try to go it

completely alone, no matter what, it will take much longer.

Mid-level sysadmins also take on more projects than their junior counterparts, and they are able to complete simple projects independently. Junior sysadmins might be able to maintain an existing system, but mid-level sysadmins actually might be able to set it up from scratch. They also can start tackling larger, more complicated projects that may require them to learn new technologies and come up with some approaches independently, although in those cases, they'll still sometimes need to reach out to more experienced team members to make sure they are on the right track.

As sysadmins master all of the day-to-day tasks, they also naturally will start to come up with improvements and efficiencies for those tasks, and they may make some suggestions to the team along those lines. These improvements may become projects for them in their own right. They also should be able to provide some level of mentorship and training for junior members on the team, at least with daily tasks.

One of the most important things for mid-level sysadmins to do if they want to level up is to take on projects and help triage emergencies. Projects and emergencies often provide opportunities to think outside established playbooks. It's this kind of critical thinking, research and problem-solving that builds the experience that's so important for sysadmins. They will start to notice some common patterns the more emergencies and projects they work through, and that realization builds a certain level of confidence and deeper understanding that is vital for moving to the next level.

Senior Systems Administrator

Although some may consider people to be senior sysadmins based on a certain number of years' experience, to me, what makes someone a senior sysadmin versus a mid-level sysadmin isn't years of experience or number of places worked at, it's more a particular state of mind that one can get to via many different means. Many people get the title before they get the state of mind, and often it takes getting the title (or some of the responsibilities associated with it) to make a person level up.

The main difference between senior sysadmins and mid-level sysadmins is that one day, something clicks in senior sysadmins' minds when they

realize that basically every emergency they've responded to and every project they've worked on to that point all have a common trait: given enough time and effort, they can track down the cause of just about any problem and complete just about any sysadmin task. This is a matter of true confidence, not false bravado, and it's this kind of real independence that marks senior sysadmins.

Early on in your career, certain tasks or projects just seem over your head, and you absolutely need help to complete them. Later on, you master daily tasks, but weird emergencies or complex projects still may intimidate you. Senior sysadmins have completed so many projects and responded to so many emergencies, that they eventually build the confidence such that they aren't intimidated by the next project, the next emergency or the prospect of being responsible for important mission-critical infrastructure. Like mid-level sysadmins might approach their daily tickets, senior sysadmins approach any task that comes their way.

Here are some attributes common to senior sysadmins:

- They can perform both daily tasks and complex projects independently.
- They understand the fundamentals behind the technologies they use and can distill complex tasks down into simple playbooks everyone on the team can follow.
- They can be productive at a new job within a week or two.
- Their time is spent more on large projects and odd requests that fall outside the norm.
- They mentor other team members and have a good sense of best practices.
- They come up with new projects and improvements and can suggest appropriate designs to solve new problems.
- They understand their own fallibility and develop procedures to protect themselves from their own mistakes.

Again, it's the confidence and independence of senior sysadmins that separates them from mid-level sysadmins. That's not to say that senior sysadmins never ask for help. Along with the confidence of being able to tackle any sysadmin task is the humility that comes with a career full of mistakes. In fact, part of their experience will have taught them the wisdom of asking other people on the team for feedback to make sure they haven't missed anything. Often senior sysadmins will come up with multiple ways to tackle a problem, each with pros and cons, and use the rest of the team as a sounding board to help choose which approach would work best in a specific case.

Senior sysadmins' experiences expose them to many different technologies, systems and architectures through the years. This means they start to notice which approaches work, which don't, and which work at first but cause problems in the long run. In particular, they might track some project they completed themselves through its lifetime and note how their initial solutions worked to a particular point and then either failed as it scaled, or needed to change with the advent of some new technology. The more this happens, the more senior sysadmins start to develop a natural sense of best practices and what I call the "sysadmin sense", which, like Spiderman's "spidey sense", starts to warn them when they see something that is going to result in a problem down the road, like a backup system that's never been tested or a system that has a single point of failure. It's in developing this expertise that they are able to level up to the last major level outside management.

Systems Architect

Although every organization is a bit different, there are two main career paths senior sysadmins might choose from as they gain experience. The most common path is in management. Senior sysadmins over time end up spending more time mentoring their team and often are promoted to team leads and from there into full managers over their teams. The other path continues on with the "individual-contributor" role where they may or may not act as team leads, but they don't have any direct reports and don't spend time doing employee evaluations or things of that sort. Of course, there also

are paths that blend those two extremes. In this last section, I describe one of the last levels for an individual-contributor sysadmin to move to: systems architect.

In many organizations, the line between a systems architect and a senior sysadmin can be blurry. Equally blurry are the qualifications that may make someone a systems architect. That said, generally speaking, systems architects have spent a number of years as senior sysadmins. During the course of their careers, they have participated in a large number of projects, both with a team and independently, and they have started to see what works and what doesn't. It's this accumulation of experience with a wide variety of technologies and project designs that starts to build this inherent sense of best practices that makes someone a systems architect.

The following are some attributes common to systems architects:

- They are familiar with many different technologies that solve a particular problem along with their pros and cons.
- When solving a problem, they come up with multiple approaches and can explain and defend their preferred approach.
- They understand the limitations to a solution and where it will fail as it scales.
- They can distill a general problem down to individual tasks as part of a larger project that can be divided among a team.
- They can evaluate new technologies based on their relative merits and not be distracted by hype or popularity.

Systems architects aren't necessarily married to a particular approach, although they may have a set of approaches for tackling certain problems based on what's worked for them in the past. Because they have operated at a senior level for some time, they have developed a deeper understanding of what defines a good architecture versus a bad one and how to choose one technology

over the other. Technology moves in trends, and those trends tend to repeat themselves over a long enough timeline. Systems architects have been around long enough that they have seen at least one of those trend cycles for some hyped technology, and they probably have been burned at some point in the past by adopting an immature technology too quickly just because it was popular. Whereas junior administrators are more likely to get caught up in the hype behind a particular new technology and want to use it everywhere, systems architects are more likely to cut through the hype and, for any new technology, be able to identify where it would be useful and where it wouldn't.

Conclusion

I hope this description of levels in systems administration has been helpful as you plan your own career. When it comes to gaining experience, nothing quite beats making your own mistakes and having to recover from them yourself. At the same time, it sure is a lot easier to invite battle-hardened senior sysadmins to beers and learn from their war stories. I hope this series in Sysadmin 101 fundamentals has been helpful for those of you new to the sysadmin trenches, and I also hope it helps save you from having to learn from your own mistakes as you move forward in your career.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

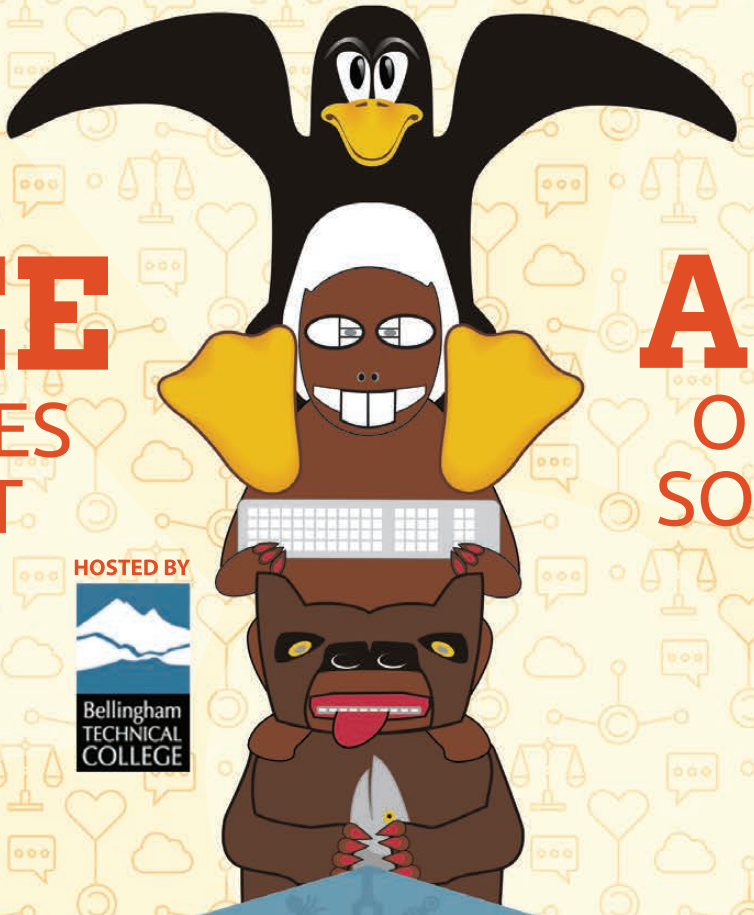
RETURN TO CONTENTS

LINUXFEST NORTHWEST

2017

MAY 6TH & 7TH

BELLINGHAM, WA



FREE
ALL AGES
EVENT

ALL
OPEN
SOURCE

HOSTED BY
Bellingham TECHNICAL COLLEGE

1500+
Attendees

40+
Exhibitors

80+
Sessions

LINUXFESTNORTHWEST.ORG

Tracking Down Blips

Wonder who (or what) is hogging your bandwidth? Install a network monitor.

SHAWN POWERS

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via email at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

PREVIOUS
Kyle Rankin's Hack and /

NEXT
New Products

IN A PREVIOUS ARTICLE, I explained the process for setting up Cacti, which is a great program for graphing just about anything (see the November 2016 issue). One of the main things I graph is my internet usage. And, it's great information to have, until there is internet activity you can't explain. In my case, there was a "blip" every 20 minutes or so that would use about 4mbps of bandwidth (Figure 1). In the grand scheme of things, it wasn't a big deal, because my connection is 60mbps down. Still, it was driving me crazy. I don't like the idea of something on my network doing things on the internet without my knowledge. So, the hunt began.

Most folks immediately told me to use Wireshark to analyze the data. That's good advice, but the problem

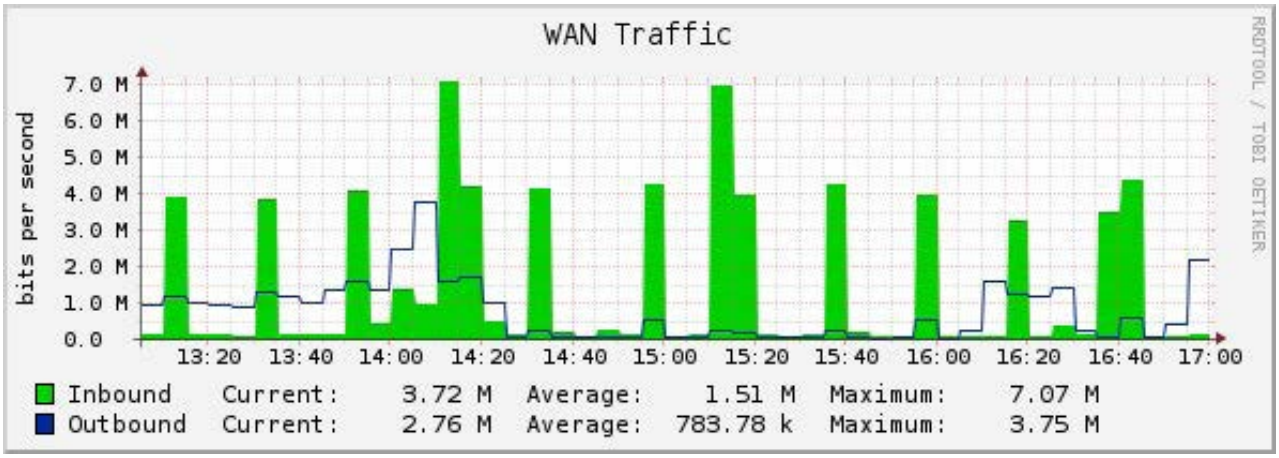


Figure 1. That blip drove me crazy for weeks.

makes me want a real-time monitoring system rather than a one-off packet search. Plus, even with Wireshark, you need to address the issue of capturing *all* the data flowing to and from the internet. Modern switching hardware purposefully directs traffic only to the ports on your switch where the traffic is intended. That means you can't just "sniff" the whole network without some effort. So regardless of how I was going to analyze the traffic, I had to be able to see the traffic. Thankfully, there are a few ways to accomplish that.

Sniffing All the Data

Network hubs were very common 20 years ago. The idea with a hub is that the network data coming in is repeated to every port on the hub, and whichever computer the packet was intended for accepts it. Every other computer just ignored the data. This worked fine when the amount of data was low and the speed of the data was slow, but as more devices were added to the network, it quickly became congested. About that time, "switching" technology entered the picture. A switch would accept data on every port, but repeat the packets only to the single port on which the intended device was listening. At first, switches were extremely expensive, so it wasn't uncommon to see a four-port rackmount switch that had hubs connected to each port. It was a way to segregate the congestion into manageable chunks. Eventually, switching technology became mainstream. Now even the $10 eight-port devices you can buy online are switches instead of hubs, and the idea of a hub is outdated.

Wi-Fi Hubs?

Although Ethernet networks rarely use hub technology anymore, Wi-Fi doesn't have that same luxury. In fact, the reason Wi-Fi access points can support only a certain number of devices before they become unusable is that Wi-Fi functions conceptually like a hub. All the devices on a Wi-Fi network receive all the packets and "accept" only packets destined for them. That's the reason it is so important to be security-conscious when using a public Wi-Fi network. Anyone else connected can see all your traffic, so make sure any sensitive data is encrypted via SSL, or even better, encrypt all your traffic via VPN.

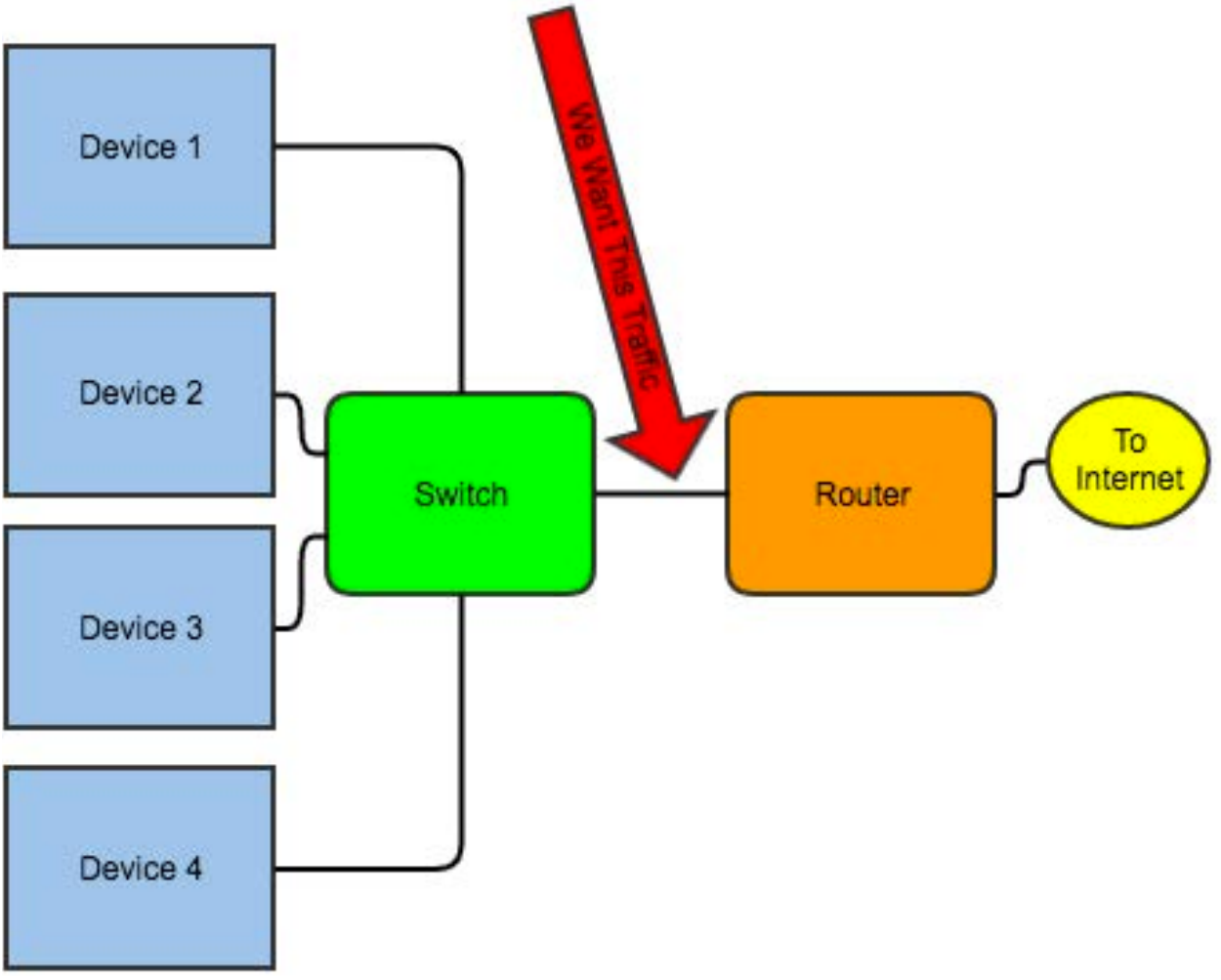


Figure 2. This is the bottleneck through which all internet traffic flows.

Why is all that important? Well, if you're trying to monitor an entire network, switches work against you. Back in the day of hubs, every computer saw all the traffic on a network, which made it easy to monitor what was happening. (It also made it easy to sniff other people's packets, but that's another story altogether.) Thankfully, there are a few options for capturing all the data so you can analyze traffic on your network.

The first thing to determine is where you want to monitor. You can monitor only traffic flowing through a certain place, so you need to determine where that place is. In my case, I want to monitor internet traffic, so from a port standpoint, I want to see all the traffic flowing into and out of where my router plugs in to my LAN (Figure 2). There are a few ways to capture that data—let me run through the options.

Option 1: Use a Hub

If you look at Figure 3, you can see how this option works. All the internet traffic flows through the hub into the router. Because hubs repeat all data to every port on the hub, you simply plug the monitoring computer in

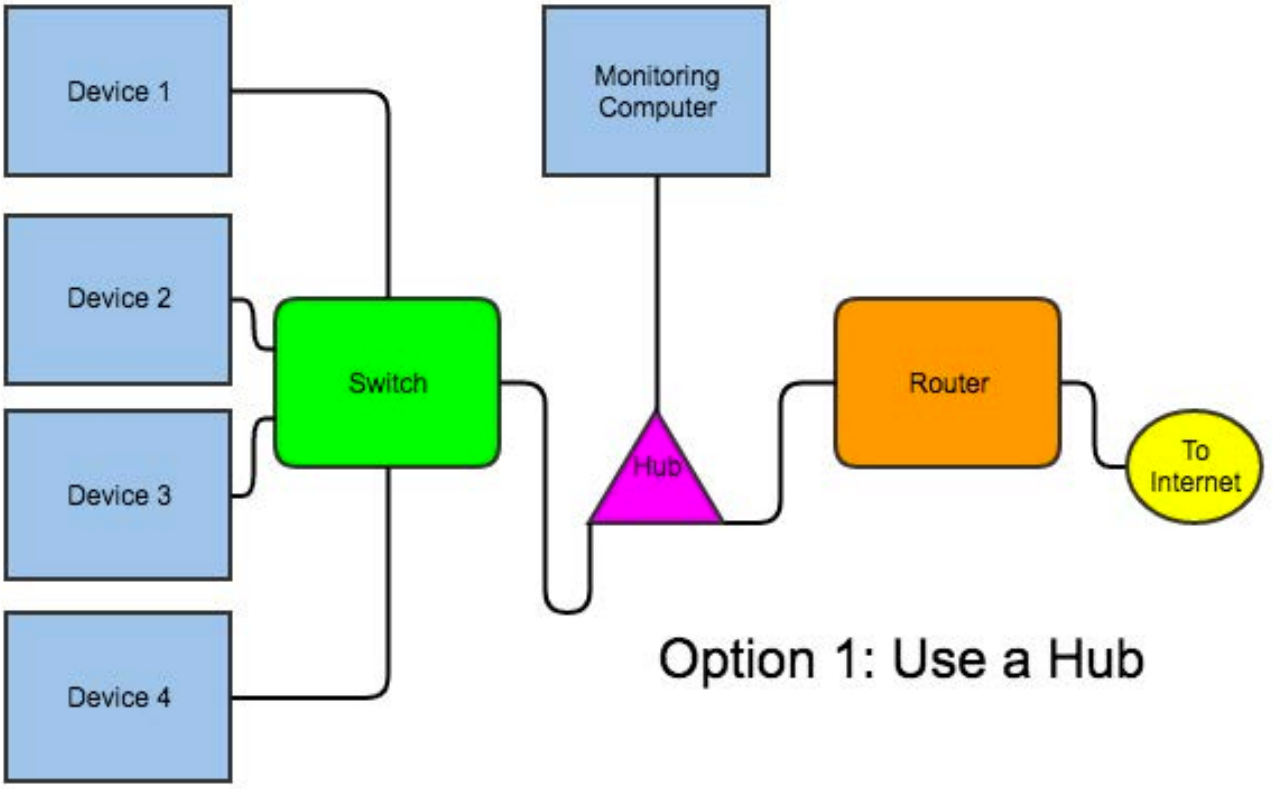


Figure 3. A hub is technically an option, but not a good one.

to that same hub, and it "hears" all the network traffic to and from the internet. There are a few problems with this method. One, it's difficult to find hubs anymore, especially those capable of 100mbps. Even if you can find a hub, it will be a cheap desktop type that is likely not reliable enough to handle all your data. Quite honestly, even though option 1 is technically still feasible, it's a really bad idea, and I don't recommend it.

Option 2: Mirror a Port on Your Switch

This is probably the best way to monitor network traffic in a modern LAN environment. Figure 4 shows what it looks like. The only problem with this method is that it requires a "smart" or "managed" switch. That usually increases the cost drastically, but it gives you other management features like VLANs. Still, if you already have an unmanaged switch, this can be a large expenditure. If you want to use this method in an environment already populated with unmanaged switches, perhaps consider getting a small managed switch and put it in place of the hub shown in Figure 3.

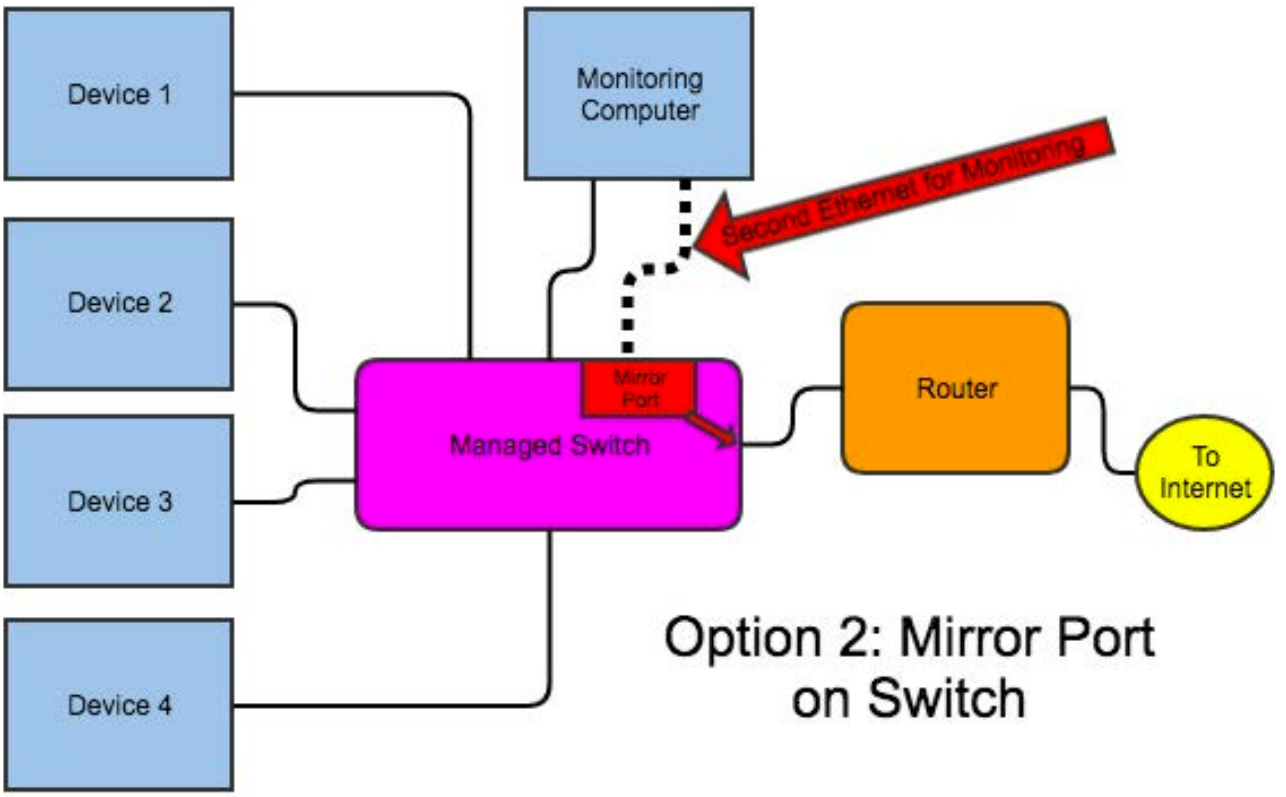


Figure 4. This is probably the best way to monitor traffic, but it requires a managed switch, which I didn't have.

The actual process for mirroring a port works differently on different brands of switches. Regardless of the specific method, however, all managed switches should allow you to mirror the traffic from one port to another. Then your monitoring computer can listen on that mirror port and analyze the traffic. It's sort of like creating an internal two-port hub that sends traffic only one way. Usually if you do this, it's wise to "sniff" with a second Ethernet card on the monitoring computer. That way, you're not confusing traffic to and from the monitoring computer with the internet traffic.

Option 3: Inline Linux Bridge

Figure 5 shows what I actually did on my network. I didn't use this method because it's better than option 2; rather, I used it because I didn't

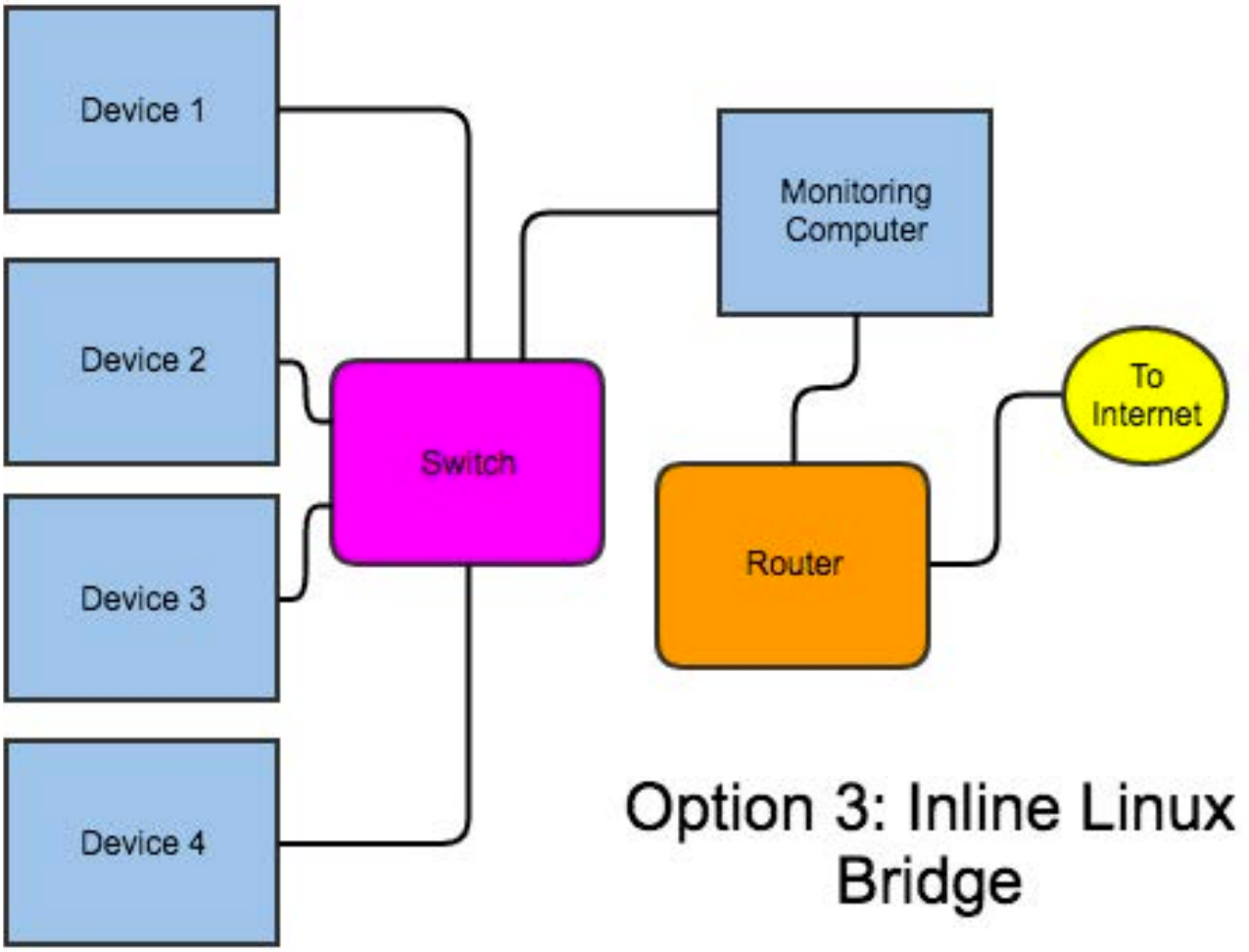


Figure 5. Although possibly not the best way to do it, this is probably the nerdiest. It's also the way I did it, because it didn't require any new hardware.

have a managed switch. Basically, in this setup, you need a monitoring server with two Ethernet cards. You create a "bridge" interface, and plug the computer in between the switch and the router. This does capture all the traffic because it literally flows through the computer. Unfortunately, it means if something goes wrong with your monitoring computer, it can take down internet access for your entire network, along with DHCP and DNS if your router hosts those for you. There are special Ethernet cards with multiple ports that "fail open", so that if the machine does, traffic still flows. If you're going to spend money on this, however, I recommend just getting a managed switch and doing a mirrored port.

Option 4: Use Your Router

This option works only if your router is robust enough to run bandwidth-monitoring tools. Some are, especially if you're using a full-blown server as a router. Most can do it, but not well. Still, if you only want simple monitoring and your router can handle the load, it's an easy way to do it.

My Bridge Install

I went with Option 3, which meant I needed to install a network bridge on my Linux machine. On Ubuntu, the process is really simple. It's not difficult on other distros like CentOS, but if you're using something other than Ubuntu, you'll need to google the specific steps. For Ubuntu, do:

```
apt-get install bridge-utils
```

Then edit your /etc/network/interfaces file:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The bridge interface
auto br0
iface br0 inet dhcp
    bridge_ports eth0 eth1
    bridge_stp off
```

```
    bridge_fd 0
    bridge_maxwait 0
```

You'll need to adjust your interface names if your Ethernet devices don't come up as eth0 and eth1, but the configuration should make sense from looking at my example. You either can restart networking or, even better, reboot your system to make sure it comes up properly on startup. Once up and running, the `ifconfig` command should look something like mine:

```
spowers@pooky:~$ ifconfig
br0    Link encap:Ethernet  HWaddr 00:25:90:34:d4:3a
       inet addr:192.168.1.6  Bcast:192.168.1.255  Mask:255.255.255.0
       inet6 addr: fe80::225:90ff:fe34:d43a/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:1820381471 errors:0 dropped:0 overruns:0 frame:0
       TX packets:124514207 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:1850742830285 (1.8 TB)  TX bytes:34896441989 (34.8 GB)

eth0  Link encap:Ethernet  HWaddr 00:25:90:34:d4:3a
       inet6 addr: fe80::225:90ff:fe34:d43a/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:1040590501 errors:0 dropped:56421 overruns:0 frame:0
       TX packets:782968757 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:1101548247906 (1.1 TB)  TX bytes:493789819966 (493.7 GB)
       Interrupt:16 Memory:fb5e0000-fb600000

eth1  Link encap:Ethernet  HWaddr 00:25:90:34:d4:3b
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:1001689311 errors:0 dropped:55961 overruns:0 frame:0
       TX packets:1165557388 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:797026007540 (797.0 GB)  TX bytes:1134936725326 (1.1 TB)
       Interrupt:17 Memory:fb6e0000-fb700000
```

Note the only interface with an IP address is br0. The other two interfaces are bridged together, so traffic can freely flow through them. It's interesting that this is the way distributions like Untangled work. They create a bridge device and then filter/block/redirect traffic as it passes through.

The Software

So far, I've gotten only to the point where the monitoring computer can listen to the traffic going to and from the internet. I haven't actually installed any software to do the listening. Quite a few different packages exist for capturing traffic and analyzing it. Depending on the type of network trends you're looking for, you might choose a different software package from me. I actually installed a few, but rely most on BandwidthD for analyzing traffic. I'll talk more about BandwidthD, but be sure to check out some others too:

- Darkstat: https://unix4lyfe.org/darkstat
- Etherape: http://etherape.sourceforge.net
- Ntop: http://www.ntop.org
- Wireshark: https://www.wireshark.org
- BandwidthD: http://bandwidthd.sourceforge.net

I like BandwidthD because it shows traffic graphs for each device on my network. If you remember my initial problem, I was trying to figure out what device on my network was downloading something every 20 minutes. I figured it was a game system or cell phone stuck in a failed download loop or something.

Installing BandwidthD (or most any of the utilities) is a simple `apt-get install` away. The software is most likely in your distribution's software repository, and even if the version is a little outdated, it should work perfectly fine. The only thing I needed to do is edit /etc/bandwidthd/bandwidthd.conf and set the network I wanted to monitor and the interface I wanted to listen on. Otherwise, I left everything the default.

BandwidthD installs an Apache configuration file, so you should be able to access its interface at http://server.ip.address/bandwidthd/.

After it's running for a while, you should see statistics like those in Figure 6, which shows the top 20 bandwidth users on my network. It's fun information to see, but if you're looking for a specific traffic pattern, you'll need to scroll down a bit to see network graphs for every device on your network. Figure 7 shows the traffic to and from my Plex Media Server. Keep in mind this is only traffic going to and from the internet, but still, you can see when friends and family were watching videos from my server over the internet. It's important to note that although the default page of BandwidthD shows only the top 20 users, you can click on

Programmed by David Hinkle, Commissioned by DerbyTech wireless networking.

- Daily -- Weekly -- Monthly -- Yearly -

Pick a Subnet:
- Top20 -- 192.168.1.0 -

Top 20 IPs by Traffic - Daily

Ip and Name	Total	Total Sent	Total Received	FTP	HTTP	MAIL	P2P	TCP	UDP	ICMP
Total	104.4G	40.2G	64.2G	0	34.0G	0	2.1M	63.8G	40.5G	20.9M
192.168.1.17	41.1G	5.8G	35.3G	0	0	0	0	1.1G	40.0G	254.7K
192.168.1.12	22.3G	21.8G	506.3M	0	125.0M	0	0	22.3G	1.2M	0
192.168.1.16	6.5G	6.2G	260.8M	0	158.1M	0	44.0K	6.5G	12.4M	13.5K
192.168.1.179	5.4G	359.6M	5.1G	0	5.3G	0	0	5.3G	75.8M	79.0K
192.168.1.39	5.2G	2.5G	2.7G	0	5.2G	0	0	5.2G	3.1M	86.4K
192.168.1.225	4.3G	75.5M	4.3G	0	4.3G	0	0	4.3G	625.1K	0
192.168.1.112	3.8G	63.2M	3.7G	0	3.8G	0	0	3.8G	670.2K	0
192.168.1.182	3.5G	89.4M	3.4G	0	3.5G	0	0	3.5G	1.0M	5.2K
192.168.1.6	2.4G	1.1G	1.3G	0	1018.8M	0	20.8K	2.3G	82.7M	13.8M
192.168.1.150	2.4G	81.9M	2.3G	0	2.2G	0	0	2.2G	225.5M	38.2K
192.168.1.186	2.1G	700.3M	1.4G	0	2.1G	0	0	2.1G	800.0K	1008
192.168.1.138	1.8G	79.8M	1.7G	0	1.8G	0	0	1.8G	1.1M	3.1K
192.168.1.222	1.6G	272.4M	1.3G	0	1.6G	0	0	1.6G	3.3M	9.9K
192.168.1.146	1.2G	654.8M	550.9M	0	1.2G	0	0	1.2G	610.4K	3.6K
192.168.1.111	1.0G	12.3M	1.0G	0	1.0G	0	0	1.0G	812.9K	0
192.168.1.145	479.8M	12.7M	467.0M	0	477.2M	0	0	477.9M	1.9M	0
192.168.1.196	368.5M	7.4M	361.1M	0	365.9M	0	0	366.0M	2.5M	233
192.168.1.153	272.6M	15.9M	256.7M	0	260.5M	0	2.1M	263.9M	8.8M	3.7K
192.168.1.224	263.4M	37.0M	226.4M	0	217.5M	0	0	231.6M	31.8M	6.6K
192.168.1.117	139.5M	9.8M	129.6M	0	135.2M	0	0	135.2M	3.3M	955.2K

Figure 6. I love BandwidthD; it's full of so much juicy data.

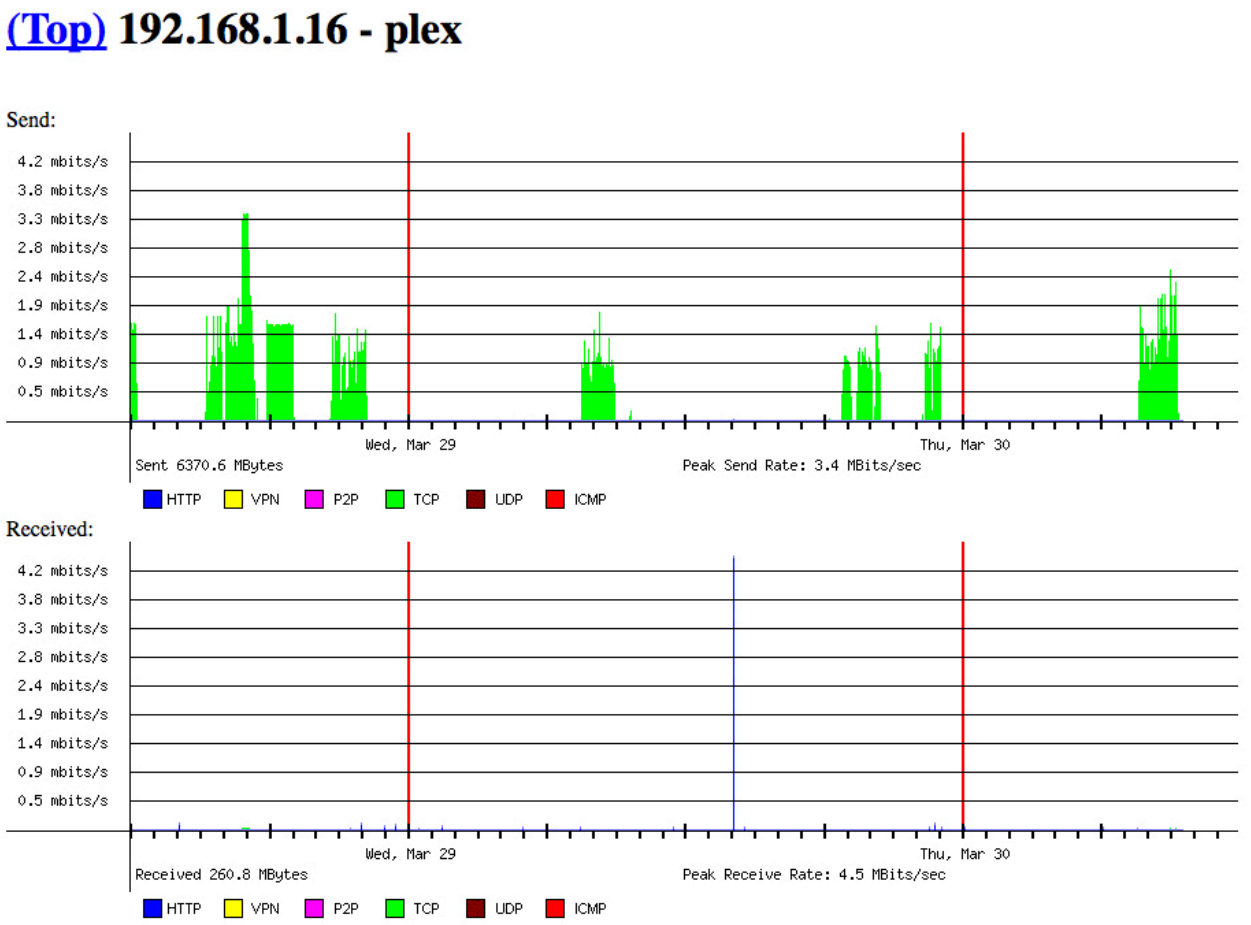


Figure 7. Having a set of graphs for each device on the network is amazingly convenient. Having it done automatically is just plain amazing.

the network address to see every user who accesses the internet. It's an amazing tool for figuring out what's happening on your network.

What about My Blip?

It turns out that I couldn't find anything on my network causing those network usage spikes every 20 minutes. I looked at the graphs for every single device on my network and compared it to the spikes on my Cacti bandwidth graphs. I just couldn't find a match. Then I realized that my total bandwidth graph from BandwidthD should come pretty close to matching my WAN bandwidth graph from Cacti. And, it didn't. My entire network monitoring server setup seemed to be for nothing, because I couldn't track down what was causing the blips.

I decided to troubleshoot my Cacti installation to see if there was

something happening every 20 minutes to cause an error. It was then that I noticed that while the WAN interface on my router had the blip every 20 minutes, the LAN side of my router (which I graph, but never really look at because it's just an inverse of the WAN graph) didn't have the blip. It turns out that my UniFi router has a feature that runs a speed test every 20 minutes to graph the health of the connection. I don't remember turning that feature on, but sure enough, it was enabled. When I disabled the periodic speedtest, my network blips stopped.

So in the end, my network monitoring setup didn't find anything, but I don't regret setting it up. Now I can monitor traffic easily and see what sort of bandwidth requirements individual devices need. In fact, the only change I plan to make is to set up my server using Option 2 instead of Option 3, because I recently upgraded my server rack to managed switching hardware. That way if my monitoring computer dies, my internet connection stays up.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

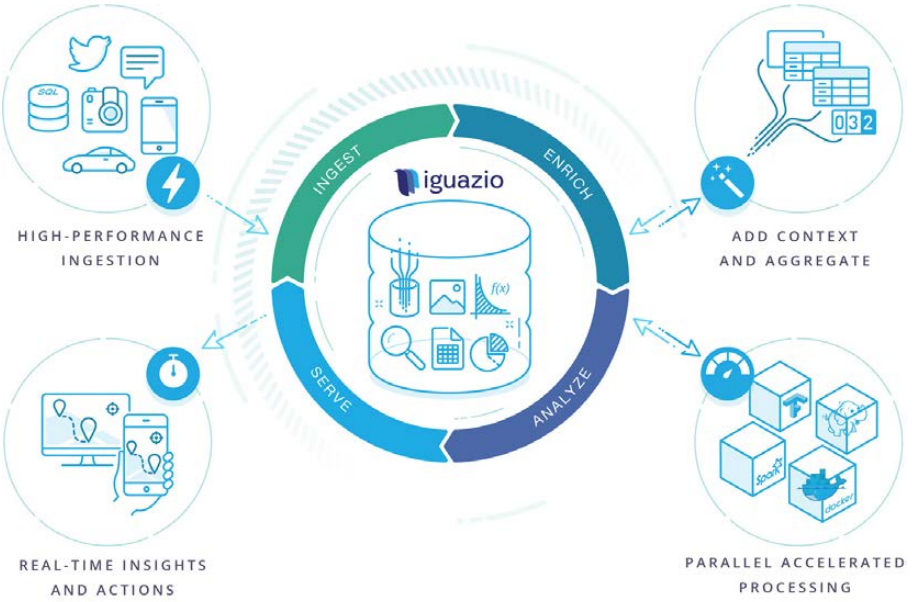
RETURN TO CONTENTS

PREVIOUS
Shawn Powers' The Open-Source Classroom

NEXT
Feature: 3D Imaging of Heart Activity with Open-Source Software

ioSafe Server 5

Until now, says ioSafe, true zero-recovery-point server solutions have been available only to the biggest of companies. However, with the arrival of ioSafe's Server 5, SMEs have access to "the industry's first fire and waterproof server" designed to eliminate data loss and minimize downtime. Server 5 protects data in real time and delivers instant disaster recovery with a true zero-recovery point and the best recovery time objectives for terabytes of data. Both fire- and waterproof, Server 5 protects data from fire with temperatures to 1550°F for 30 minutes per ASTM E-119 and from floods up to a ten-foot depth for three days. Running either Linux or Windows Server 2012, the solution is equipped with an Intel Xeon processor, 16–128GB of DDR4 RAM, dual 10 Gigabit Ethernet adapters for increased bandwidth and automatic failover and capacities between 5–40TB.
http://iosafe.com

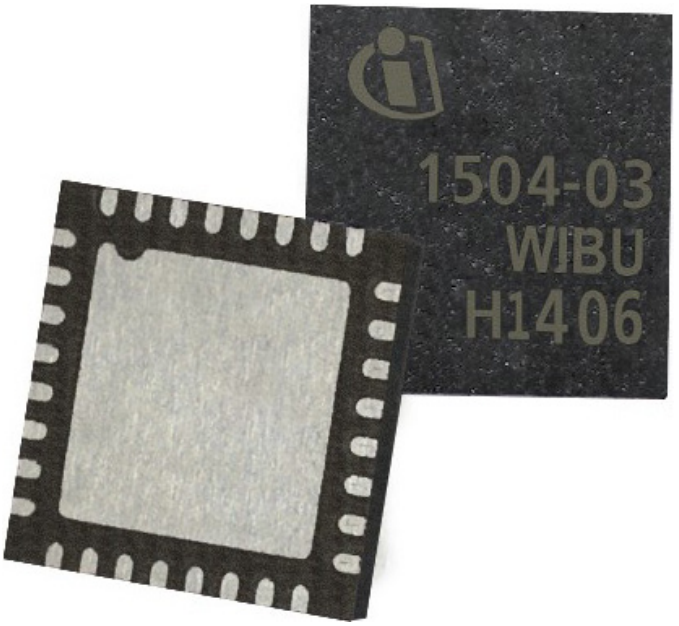


iguazio's Continuous Analytics Solution

In industries like financial services, healthcare and IoT, organizations are faced with the challenge of complexity across the entire data lifecycle. To help enterprises solve big data operational challenges and generate real-time insights, iguazio has developed a new Continuous Analytics Solution. Deploying a continuous data consumption approach, the solution reduces time to insights from hours to seconds, eliminating data pipeline complexities while seamlessly integrating with Apache Spark and Kubernetes. Data is at the center of iguazio's real-time continuous analytics platform, which simplifies the data pipeline and speeds it up—ingesting, enriching, analyzing and serving it—all in one unified platform. By integrating with the open-source Spark and Kubernetes frameworks, it accelerates insight generation and enables rapid deployment of a variety of stateless analytics services and data processing tasks. iguazio's platform secures the data and allows accessing the same records simultaneously through streaming, object and database APIs. iguazio's business value is driven by three key capabilities: real-time insights and faster analytics, developer and operator simplicity and agility and fine-grained security.
http://iguazio.com

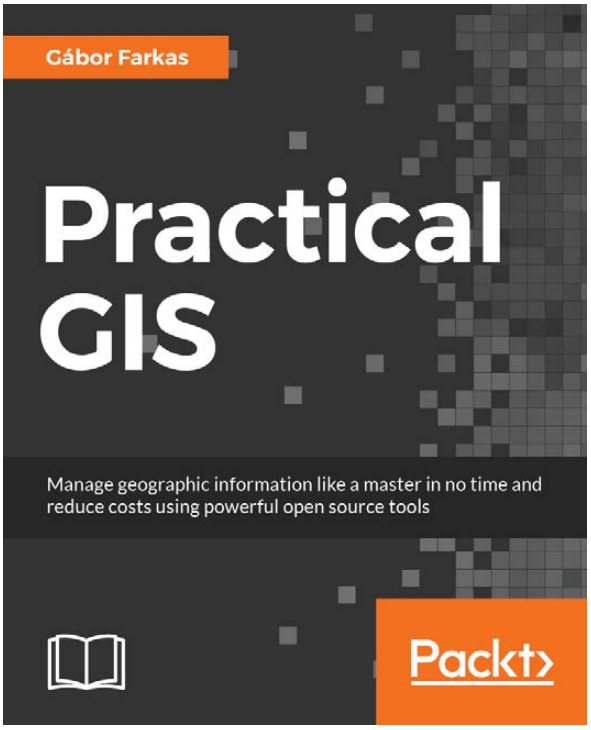
Retro United Ltd.'s Write!

Even when you're sequestered in a monastery with nothing to do but write, you still need a tool to record your thoughts. Realizing the overabundance of bloatware out there—replete with features, icons, templates, check boxes and other stuff that consumes screen space and RAM—Retro United, Ltd., released Write!, a new distraction-free text editor for Linux, Windows and MacOS. Write! is an elegant workspace for any kind of writing, from notes to to-do lists, tweets and novels—writing projects and texts of any kind. It has all the features of a good word processor, notes Retro, packed into a clutter-free interface. Write! is as functional as it is aesthetically pleasing with a minimalist design and features a specialized focus mode for concentrating on a single paragraph at a time. Write!'s publishing feature lets users quickly publish their writing online for target readers—for example, Twitter followers—and a dark theme makes nighttime writing pleasant for the eyes.
http://wri.tt



Wibu-Systems CmASIC

Wibu-Systems describes the new generation of its CmASIC module as "the answer to the security-by-design needs of modern embedded computing technology leaders". CmASIC is a module that Intelligent Device Manufacturers (IDMs) can directly embed into their boards to provide out-of-the box security and entitlement management. Security-by-design leaves nothing to chance: the system is developed free from vulnerabilities from the get-go and is impervious to attack. By soldering a security module directly on to the board of the endpoint, the endpoint is clearly identified, sensitive data (such as encryption keys, production records and configuration schemes) is safeguarded, and the module itself is accessible only by physically and logically tampering with the endpoint. Integrating CmASIC is a snap, as the module is delivered in a VQFN-32, 5x5mm chip-size package. On the software side, CmASIC supports all mainstream operating systems via CodeMeter Embedded, including Linux, MacOS and Windows for PCs, Embedded Linux and Windows Embedded for embedded systems, Linux RT, VxWorks and QNX for RTOS, and CODESYS, B&R and Rockwell for PLCs.
http://wibu.us



Gábor Farkas' *Practical GIS* (Packt Publishing)

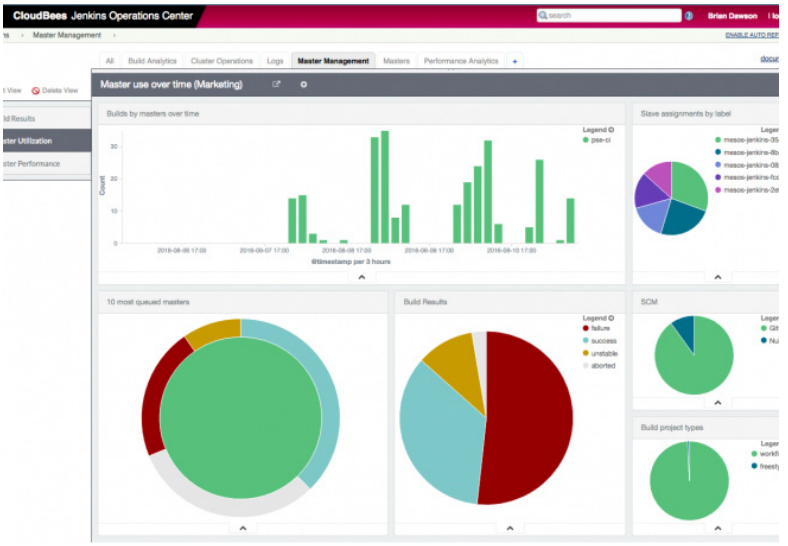
Open-source GIS tools are maturing rapidly, and Gábor Farkas' new book *Practical GIS* is a guide to applying these tools to managing geographic information like a pro at minimal cost. Farkas deploys the popular QGIS application and explains how to use it to generate useful spatial data. From there, readers are guided through the basics of queries, data management and geoprocessing. After mastering the essentials, readers practice their knowledge on real-world examples, solving various types of geospatial analyses with appropriate methods. Finally, readers learn how to publish data (and results) on the web with QGIS Server and GeoServer and create a basic web map with the API of the lightweight Leaflet web-mapping library. Storing data in a PostGIS database also is treated in the book.

http://packtpub.com

Crank Software's Storyboard Suite

Crank Software is working hard to change how embedded user interface (UI) solutions are developed. Traditional development methodologies, asserts the Crank group, leave designers on the sideline after the artwork has been handed off to the software developers. This linear process typically leads to a UI that isn't a true reflection of the design vision and is difficult to update. Enter Storyboard Suite 5.0, Crank Software's recently updated turnkey UI development software that gives designers and developers the tools to work collaboratively and iterate the UI and UX to perfection in a fraction of the time. Storyboard Suite supports graphic designers and engineers equally with innovative features and an intuitive workflow that allows them to work in parallel to deliver UIs that are modern, scalable and optimized for embedded products. The platform is powerful enough to create high-end, sophisticated UIs yet easy to learn and use without requiring hardware decisions to be made up front. Teams can be productive from the beginning of a project through to final delivery, saving valuable resources and costs and delivering powerful user experiences in less time. New features in version 5.0 include finer execution control and custom easing rates for improved animations and movement; ready-to-use, sharable components for quick prototyping and test UI interactions; a new render extension to facilitate drawing and graphing programmatically; and an exports option that simplifies sharing of Storyboard applications for quick testing to improve the feedback loop for design iteration. Storyboard Suite offers Ubuntu Linux, Mac OS and Windows versions.
http://www.cranksoftware.com

CloudBees, Inc.'s CloudBees Jenkins Enterprise



Modern IT departments are adopting continuous delivery (CD) and automating software pipelines to accelerate and scale their software development and delivery across environments. This means that CD platforms are now business-critical and need to be scalable, secure, stable and reliable. To address these requirements, CloudBees, Inc., developed CloudBees Jenkins Enterprise, a new unified CD platform offering that allows enterprises to achieve CD for their entire software portfolio, securely and in a manner that handles a large volume of projects. CloudBees Jenkins Enterprise is based on Docker container technology and architected from the ground up to allow for scale and failure backup. It can run anywhere, including public cloud or private cloud, virtual environments (VMware vSphere), Red Hat Enterprise Linux and from legacy to container-based technology. CloudBees warns that most continuous delivery solutions focus on a specific technology stack or process, leading to the deployment of multiple competing or overlapping solutions lacking holistic support of the enterprise-wide IT delivery processes. In contrast, CloudBees Jenkins Enterprise leverages Jenkins' 1,200+ extensions and is the only solution on the market that makes it possible for enterprises to have a unified view across their entire software portfolio. As a result, it is possible to evaluate how teams are progressing in their adoption of DevOps, continuous delivery and ultimately time-to-market goals.
http://cloudbees.com

AdaCore's GNAT Pro, CodePeer, QGen and SPARK Pro

AdaCore recently announced the concurrent annual release of four flagship products in its portfolio of software development and verification tools for mission-critical, safety-critical and security-critical systems. These include version 17.1 of GNAT Pro, CodePeer, QGen and SPARK Pro. All products aid organizations with the challenges they face in the development and verification of critical systems, especially when certification against software standards, such as DO-178C or EN 50128, is required. GNAT Pro is a development environment for Ada and C, on native and cross platforms. CodePeer is a deep static analysis tool for Ada that can identify bugs and vulnerabilities both during development and retrospectively on existing code bases. SPARK Pro is a verification environment that brings mathematics-based assurance to high-integrity software, and QGen is a model-based development and verification toolset for Simulink and Stateflow models, which generates code in MISRA-C or SPARK. Each tool in the suite offers its own diverse set of enhancements and improvements in the new version 17.1 release.
http://adacore.com

Please send information about releases of Linux-related products to newproducts@linuxjournal.com or New Products c/o Linux Journal, PO Box 980985, Houston, TX 77098. Submissions are edited for length and content.

RETURN TO CONTENTS

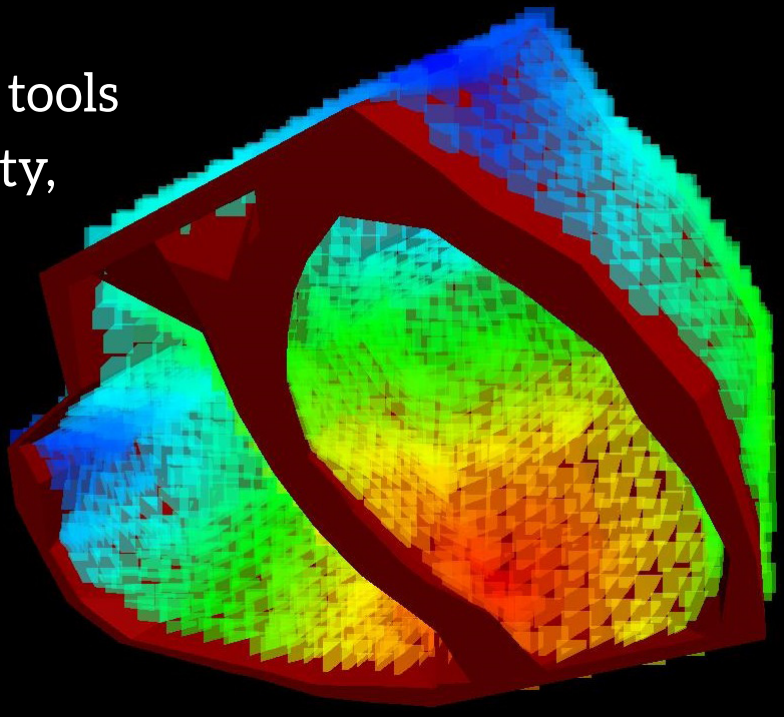
FEATURE

3D Imaging of Heart Activity

with Open-Source Software

We chose open-source tools for their superior quality, not because they're free. It turned out we needed some closed source tools as well.

JACQUES DE HOOGE



PREVIOUS
New Products

NEXT
Feature: BYOC: Build Your Own Cluster, Part I—Design

Heart diseases are common all over the world. They lower life expectancy and quality, and weigh heavily upon healthcare budgets. At the HAGA teaching hospital in The Hague, we're developing a software package named Sculptor, which allows detailed localization of heart problems while avoiding exploratory surgery. Development took us along a path that involved both Linux and Windows. I share our experiences in this article.

Apart from being a pump, the heart is a current generator. To find out what's wrong with it, a cardiologist needs a 3D image of its electrical behavior. To obtain this image from skin electrodes, we had to turn around the path traveled by the electrical signals from heart to body surface. This "inverse problem", as mathematicians call it, requires solving a set of equations that have been known since the time of Faraday.

On their way to the body surface, the electrical signals undergo the influence of the organs they meet. Until recently, it was hard to visualize this influence, but with the availability of more and more computing power on the desktop, it is possible to generate a 3D map that exactly shows how the lungs, the spine and the chest bone bend the electrical currents. To trace the signal path, we overlay a 3D grid on the heart, using the splendid GMSH open-source package that has been under development for many years.

We then apply a mathematical trick we took from construction engineering to solve the equations. Fenics-Dolphin, another great open-source package developed in an international cooperation of four universities, helps achieve this feat. Both packages are part of some Linux distributions, and they can be installed on others easily.

To fine-tune our equations, we also needed a method to guide the flow of electrical currents inside the heart itself. For this, we use a combination of two algorithms, both of which are open source. To keep the electric currents inside the conductive parts of the heart, we use a gem from the computer game industry: the Moeller-Trumbore triangle intersection algorithm. To make sure the electrical signals travel straight to their targets, we use another popular recipe, although few know its name: the Floyd-Warshall shortest path algorithm. This algorithm and its cousin, the Dijkstra shortest path algorithm, are routinely used in car navigation

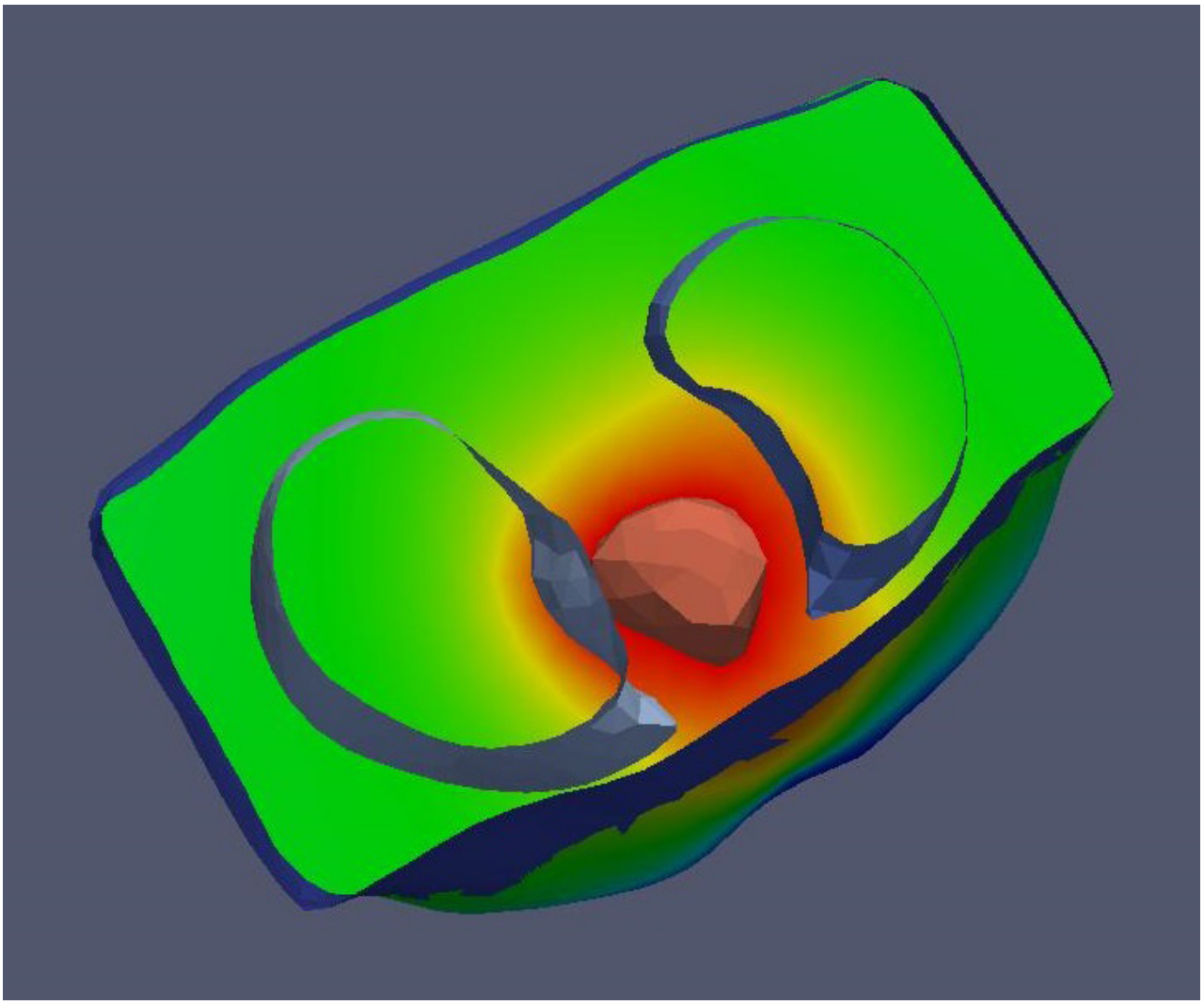


Figure 1. Electrical Field of the Heart Penetrating the Lungs (Top View)

systems. So, no rocket science.

Algorithms, algorithms, algorithms—fast, well documented, resulting from years of dedicated research work, with implementations available for free on Linux—we felt like kids in a candy shop. Why write anything ourselves? It looked like open source had all we needed. The main thing we had to do was tie it together, which, we thought, would be simple. But, it wasn't. We had underestimated this hidden booby trap of software engineering: accumulated complexity. We had pieces of knitted Matlab code, featuring fuzzy module boundaries and cryptic one-letter variable names. We had lumps of archaic Kernighan and Ritchie C code, littered with macros. We even had some well conserved Fortran IV code, shouting

at us in all capitals. Call sequences, memory models and tool versions to compile the whole bunch were all incompatible.

In short, we had a set of beautiful Christmas balls, but how to hang them on one single tree—that was the question. Early attempts had used a lot of messy glue code, and our first task was to scrape the glue off the balls and give them all the same hooks: a clean interface. Matlab, Fortran and C balls were all converted to C++, sticking to the solid concepts of object orientation: encapsulation, inheritance and polymorphism, but avoiding the jungle of template- and overload resolution.

Oh yes, and there was this one ball we didn't yet have: we needed a simple, static 3D model of the heart tissue to allow for computing our currents. None of the many research articles we read had paid any attention to this step, so we assumed it was trivial and that something like Blender could be used. Not so.

Will It Still Be Here Next Year?

As it turned out, drawing up a 3D static model of the heart wasn't simple at all. We tried Blender and at least a dozen things, but they all failed at the task. The reason is that they were not made for computation but for visualization, typically involving steps like smoothing to make things look nice.

As I am writing this, I am sitting behind a steel desk, my bare wrists resting upon the surface. On my desk, there's a lamp with a power cord, running over the edge to a wall outlet. Now suppose I want to draw up a 3D model of that desk, including the lamp and the power cord. The model is built out of tiny triangles and is smoothed to look good, with tiny gaps puttied automatically. For visualization, this model is great. But unfortunately, the smoothing closed one gap too many: the tiny gap between the copper core of the power cord and the edge of my desk, originally separated by a tenth of an inch of plastic insulator material. From an aesthetic point of view, my model is great. But from an electrical point of view, it's completely worthless. The distinction between typing quietly and being subjected to the full power-grid voltage on my wrists is what scientists would call "a significant difference".

What this meant for us is that, although tools like Blender give nice-looking results, they only rarely produce something accurate enough

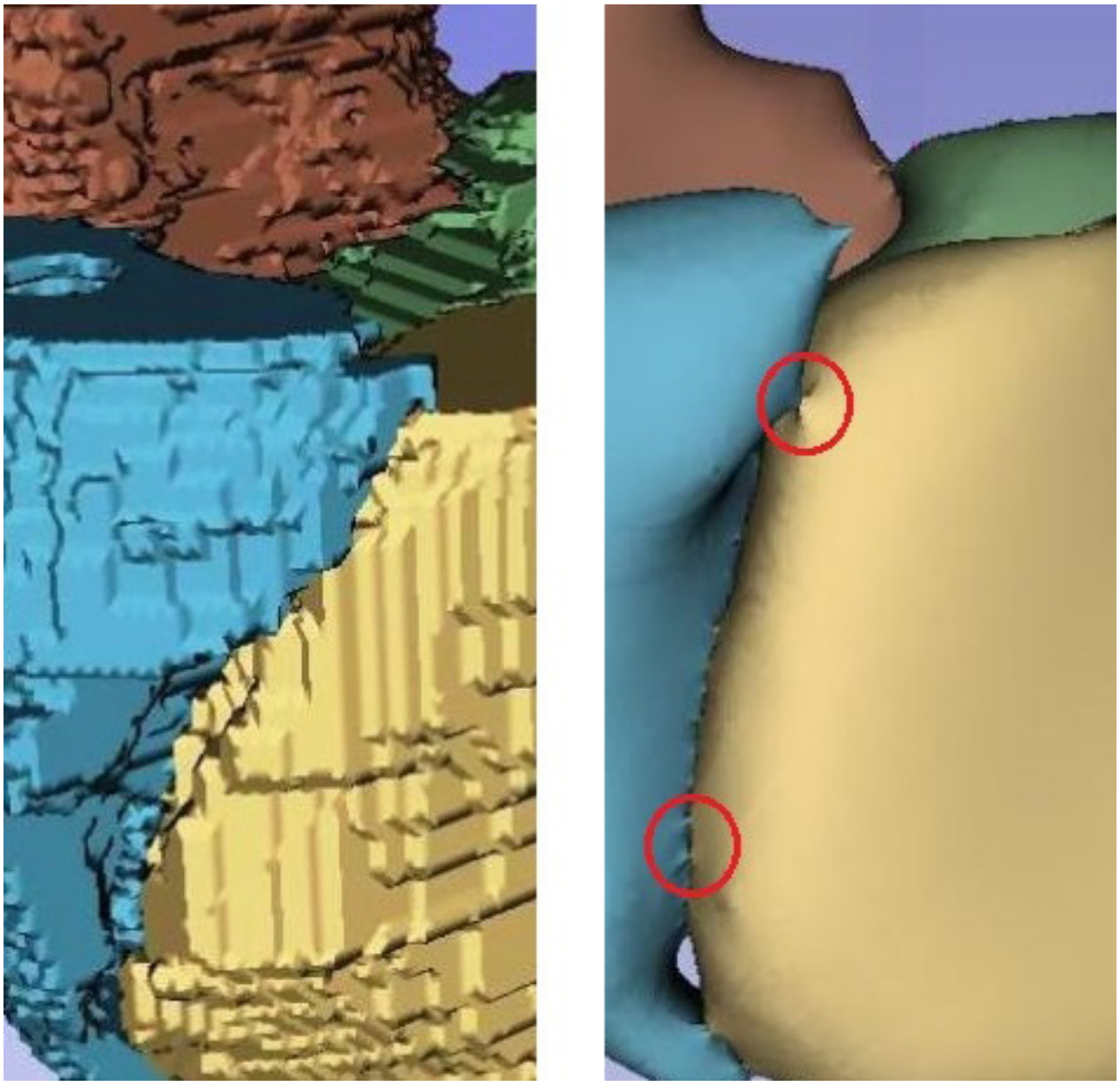


Figure 2. Smoothing causes short-circuits between tissue patches.

for simulation of electrical behavior. There are many (open-source, you guessed it) tools to repair small defects in 3D models, and we spent at least two months trying them in every possible combination. In the end, we had to admit total failure, and we concluded that we had to make this Christmas ball ourselves. We decided upon Python, NumPy and OpenGL to do it. To be able to read images generated by an MRI scanner, we needed another open-source package called PyDicom.

By now our software depended upon a large number of open-source

components, some of them mainstream, some of them less so. Having installed several versions of Ubuntu on a diversity of hardware in the course of the project, something else became clear: although kernel versions followed a single upward-bound track, Linux distributions as a whole were more like a grab bag filled by a drunken Santa Claus. Sometimes basic tools were completely lacking; sometimes several incompatible versions were on the same distribution.

We spent quite some time composing our own "standard distribution" on one of our machines, and we were nearly there when we decided to replace Python 2.7 with the Conda Python 3.5 distribution, which includes OpenGL, Numpy and even PyDicom. As Python 2.7 kept popping up at unexpected moments, we uninstalled it...which left us with a Linux that could just barely bring up a command-line console. Since installing everything properly had taken so much time, we attempted to repair our installation, spending hours reading about the adventures of other developers as laid down on Stack Overflow. In the end, we had to give up. Simply uninstalling a development tool turned out to have wrecked our OS beyond repair by mere mortals.

That wasn't all bad. We decided to grab the opportunity to make the move from Ubuntu 12 to Ubuntu 14, which had newer versions of nearly everything we needed—yet another mistake. Fenics-Dolphin turned out to depend on Python 2.7, which it couldn't find, since it wasn't the default anymore. The GCC C++ compiler had become more picky and refused several constructs for reasons that we have yet to understand. CLang was said to be the solution, and indeed, it was better at some points, but worse at others.

To cut a long story short, we had poured months into development of software that we planned to use for five to ten years at least, but installing a new Linux version after only a year left us with a dysfunctional system that was near to impossible to repair. What to expect in three years? Or six? With regard to every component we used, we had to pose one serious question: will it still be here next year? After reading through many blogs about life-cycle policies, we finally came up with a clear conclusion: 1) We don't know. 2) Nobody knows. It's hard to base a multi-year investment decision on that. Choosing to stay with a fixed set of legacy tools eventually would cut us off from new developments, which

The only viable path out of this dilemma was to start with a very bare-bones Linux and follow a rigid script to install exactly everything we wanted in exactly the right order, manually accounting for uncharted dependencies.

in the case of research, isn't an option. Using state-of-the-art tools would cripple our system over and over again.

The only viable path out of this dilemma was to start with a very bare-bones Linux and follow a rigid script to install exactly everything we wanted in exactly the right order, manually accounting for uncharted dependencies. In the end, it worked out. We now can install the newest version of everything we need on any system capable of running it. But the illusion that Linux is a kind of prepackaged development paradise fell to pieces. We all love Linux, and we know we aren't full-time system administrators. But surely the Linux world could benefit from some unification and clear policies in this area.

The Right Tool for the Job

Computations are best done on one or more Linux boxes, but physicians are used to Windows. That's one problem we still had to solve. The other problem was about the one missing Christmas ball: obtaining an accurate static 3D model of the heart without short circuits. We embarked on writing this missing ball, and since Windows was available on every laptop in the hospital, we started development in the closed-source world. No problem—Python, C++ and OpenGL were all platform-independent, so once everything was ready, we would just recompile for Linux. We thought.

It took us three months to develop a fast, reliable application to draw up the required accurate 3D models from fuzzy, misaligned MRI scans. Obtaining a good model for each patient requires human intervention, and

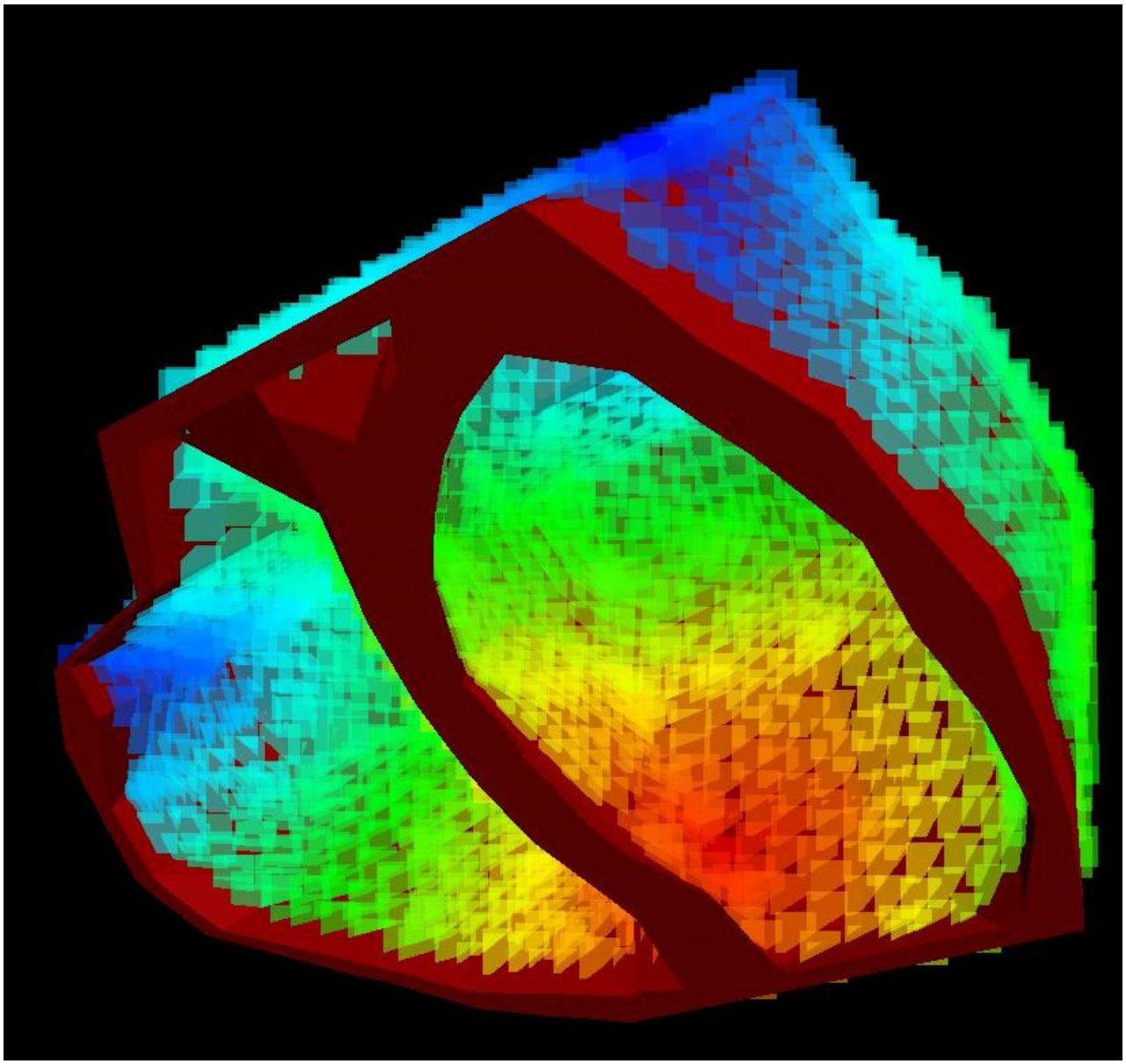


Figure 3. Electrical Activity in a Male Adult Heart, Viewed from the Inside

since radiologists were to spend many hours using this application, we paid attention to things like minimizing the number of mouse clicks and allowing intuitive manipulation of the anatomical parts that make up the heart. We could remove valves, leave out the atria or the ventricles, look from the inside out and so on—all of this fast and in high resolution. Doctoral researchers loved the result and spent many hours drawing up topologically correct models of hearts of men, women and children—a whole stock of them.

The disappointment indeed came when we wanted to move this ball

to our Linux Christmas tree. Suddenly our graphics were slow, memory corruption occurred frequently, and weird striping patterns haunted our displays. We didn't panic. This must be just a matter of getting the right drivers. We thought. This is one area where long-standing cooperation between hardware and software manufacturers has paid off in the Windows world. Being able to entertain such a stable cooperation between separate industries is one of the benefits of the contract-driven closed-source world, and it is not to be underestimated. If you buy a Windows box, you can be quite certain all capabilities of the graphics card are used to their utmost. For Linux, it's different.

Since we already had learned enough to doubt our capacities as system administrators, we sought specialist help on this one. The result didn't please us. Making the most of a state-of-the-art graphics card in Linux is often just impossible. The drivers aren't there, or they're buggy, or they refuse to cooperate with the Linux distro at hand—many reasons that in the end all boil down to the same response: forget it. Oh yes, results sometimes were quite nice. But currently I am staring at the screen of a stereovision laptop, wirelessly controlling a pair of NVIDIA shutter glasses. The results on Windows for our heart models are stunning; seeing depth really makes interpretation of disease patterns much easier. Only, after three years, we couldn't make it run on Linux. Shame on us.

Since physicians prefer Windows anyhow, and in the end, our computations will run remotely on a cluster, we decided to have the GUI part running on Windows and the computational back end on Linux, not an unusual combination. For synchronization, we considered dinosaurs like CORBA and mouses like JSON, but finally settled for simple file versioning and locking, which worked flawlessly, even over a network. For demonstration purposes, we still wanted it all to run on one (very fast) laptop. A dual-boot system was no help; we needed Linux and Windows to cooperate, running simultaneously. VMware Player made it possible, and finally, we had the system of our dreams: razor-sharp visualization with stereovision capabilities under Windows, open-source computational tools running efficiently on a quad core under Linux on the same machine and standardization on two mainstream, highly complementary languages: C++ for fast execution and Python for fast development. I dare say we were pleased with the result. The next step was scaling up our research.

Deployment for Non-Travelers

As the first publications were accepted by science journals, we were open to future international cooperation. There were several candidates inside and outside Europe. Even though we'd documented it well, our complicated setup could become a problem. The Netherlands are small, and if someone has an installation problem that can't be solved using a remote desktop, it's at most a short drive to provide personal assistance. Reliably deploying and supporting a setup over the borders is an entirely different matter. This made us think about thin clients and a browser GUI. For a start, we reserved a few URLs and began to look into security regulations for patient data.

Since we already had separated computation and visualization, having the latter run in a browser seemed doable. Not without reason, we'd chosen something ubiquitously supported like OpenGL. While getting OpenGL to run in a browser proved simple, our interactive 3D modelling Python code certainly wasn't. We did some experiments in JavaScript, but things like class-based object orientation and multiple inheritance were embedded so deeply in our thinking, that using JavaScript for this sent us back to square number one. So we looked into some alternatives, most notably Pythons that could run in the browser. We found several, some requiring special browser plugins, others producing slow, bulky code, yet others lean and mean but lacking some of the basic Python features we needed.

So we had to roll our own. Having such positive experiences with open source, it seemed only logical to choose this path ourselves. We embarked on developing an open-source Python-to-JavaScript compiler called Transcrypt, which was able to run our graphics routines. We wanted it to be fast, compact and able to use any JavaScript graphics library. The first versions uploaded to GitHub were accompanied by the warning that the code was unfit for any serious use and might go away any moment. But, that changed earlier than we had expected. Writing a compiler is a well documented undertaking, and Python itself comes with an excellent parser, so in a few months' time, we had all the functionality we ourselves needed. Execution speed and code size matched JavaScript, and we were quite satisfied. Since we didn't plan to go any further, we referred to our tool as a "Small Sane Subset"

classes.py

```
class A:
    def __init__ (self, x):
        self.x = x

    def show (self, label):
        print ('A.show', label, self.x)

class B:
    def __init__ (self, y):
        alert ('In B constructor')
        self.y = y

    def show (self, label):
        print ('B.show', label, self.y)

class C (A, B):
    def __init__ (self, x, y):
        alert ('In C constructor')
        A.__init__ (self, x)
        B.__init__ (self, y)
        self.show ('constructor')

    def show (self, label):
        B.show (self, label)
        print ('C.show', label, self.x, self.y)

a = A (1001)
a.show ('america')

b = B (2002)
b.show ('russia')

c = C (3003, 4004)
c.show ('netherlands')

show2 = c.show
show2 ('copy')
```

classes.js

```
var A = __class__ ('A', [object], {
    get __init__ () {return __get__ (this, function (self
        self.x = x;
    });},
    get show () {return __get__ (this, function (self, la
        print ('A.show', label, self.x);
    });}
});
var B = __class__ ('B', [object], {
    get __init__ () {return __get__ (this, function (self
        alert ('In B constructor');
        self.y = y;
    });},
    get show () {return __get__ (this, function (self, la
        print ('B.show', label, self.y);
    });}
});
var C = __class__ ('C', [A, B], {
    get __init__ () {return __get__ (this, function (self
        alert ('In C constructor');
        A.__init__ (self, x);
        B.__init__ (self, y);
        self.show ('constructor');
    });},
    get show () {return __get__ (this, function (self, la
        B.show (self, label);
        print ('C.show', label, self.x, self.y);
    });}
});
var a = A (1001);
a.show ('america');
var b = B (2002);
b.show ('russia');
var c = C (3003, 4004);
c.show ('netherlands');
var show2 = c.show;
show2 ('copy');
```

Figure 4. Class-Based Object Orientation in the Browser

compiler. That was when the dynamics of open source took over.

People started to notice our little compiler; they even used it professionally, and our first GitHub stars were acquired. Feature requests came in. Could you build in this? Could you build in that? It all wasn't too hard, and some of the new stuff we could use ourselves. Although Transcrypt was open source from the start, and its development decoupled from Sculptor, it now gained momentum of its own. Some very experienced and competent developers from all over the world made large contributions, and people started to use it for a diversity of projects, every bit as serious as our research. It was pure fun, and on top of that, it took away our last reservations about Linux in particular and open source in general. We clearly had been drawn into something bigger than ourselves.

Up to this point, we'd always wondered about reliability. Just how reliable can anything as complex and vulnerable as an operating system be, if it's developed in a loose cooperation of hundreds of developers that may withdraw their support at any time? How about code quality? How about continuity? From experience, we now can tell the answer. Let me make this personal. For some 35 years I've been a lead developer in many teams with many companies large and small, but I've never encountered such craftsmanship, such dedication, such enthusiasm, such energy and such a sense of responsibility as in the open-source world. Here's a message to any company still shy of trying: there's a world of competent, top-notch developers out there. Do you think you can survive without them, creating your own proprietary world with ten, 20, 100 developers being managed hierarchically? You don't stand a chance. It's as simple as that. ■

Jacques de Hooge is a developer of technical and scientific applications, and he lives in Rotterdam, the Netherlands, with his wife, two kids and a cat. After graduating from the Delft University of Technology, he started his own company, GEATEC engineering, which specializes in software for computation, visualization, real-time control and simulation. He is co-author of several scientific publications in the area of medical technology and initiator of the Transcrypt open-source project. He's also a part-time lecturer at the Rotterdam University of Applied Sciences, and he has provided in-company training for many years, on subjects ranging from programming in C++ and Python to software development quality control.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

BYOC

Build Your Own Cluster, Part I—Design

Cluster computing mainly has been relegated to the professional realm. It doesn't have to be that way. Anyone with modest Linux experience can build their one.

NATHAN R. VANCE, MICHAEL L. POUBLON and WILLIAM F. POLIK

PREVIOUS
Feature: 3D Imaging of Heart Activity with Open-Source Software

NEXT
Doc Searls' EOF

Computer clusters are standard tools in scientific computing. Clusters speed up calculations by computing a single task in parallel or by computing multiple tasks simultaneously, thus rapidly solving extremely large or computationally intensive problems. They utilize commodity hardware, resulting in an excellent performance-to-price ratio.

In a basic computer cluster, one computer (the head node) relays instructions to the rest of the computers (compute nodes) across an isolated local network. The compute nodes then carry out their assigned tasks, optionally communicate among themselves, and return the results to the head node. This structure is analogous to a work force: the head node is the manager, who receives jobs from a customer and subcontracts with the compute node workers. When a worker is done, it signals the manager that it is available for another task. When the job is completed, the manager returns the final product to the customer—in this case, a calculated result.

Building a cluster typically is accomplished in one of two different ways. One way is to use configuration software, such as ROCKS, to set up a cluster automagically. Although that method has the obvious strength of convenience, weaknesses include being constrained to the assumptions and supported architectures of the tool. Additionally, it can be difficult to diagnose problems that arise, because such tools conceal what's happening under the hood.

The other method of cluster building is to build your own. It may take more time to complete, but the level of control and understanding of the cluster gained by this method makes it worth it in the long run. Cluster design goals include:

- **Robustness:** the cluster is flexible enough to support many different applications, whether or not known at the time of its initial setup.
- **Reliability:** the cluster, both hardware and software, should be stable for the long-term after the initial setup.
- **Portability:** the same process and tools used in this guide can be used on different distributions with little change.

- **Scalability:** the procedures should be practical if there are 2, 10 or 100 nodes.
- **No "Magic":** common problems are resolved by straightforward, well documented, easy-to-follow solutions.
- **Heterogeneity:** upgrading in the future is still possible when the hardware may be different.
- **Simplicity:** the approach outlined here should allow people with modest Linux/UNIX experience to build a cluster of their own.
- **Low-Cost:** the cluster uses readily available hardware and free software (Linux).

This three-part series walks through the process of building a cluster that has all of those attributes. This first article gives an overview of the cluster's design, including the setup of computing hardware, networking and storage. The second article will deal with installing the operating system and software on the cluster in a scalable manner. The third article will cover the configuration of tools and services that transform the loose collection of computers to a tightly integrated cluster.

Cluster Hardware

Many hardware components go into building your cluster. These components can be broken down into six general categories: the computers themselves, networking supplies, physical storage of the computers, power distribution equipment, a console to access the computers and spare parts.

Computers Clusters can be constructed from generic tower computers, but for large clusters, computers specifically designed for high-performance computing can be purchased from suppliers like Supermicro. These systems are preferable to tower PCs because they come in high-density packages, such as 1 or 2 U rack-mountable cases (a U is 1.75" of vertical rack space). In either case, the computers must work well with Linux. For scalability, the compute nodes need to

support PXE booting and IPMI control.

- The head node typically is more capable than the compute nodes since it is the entry point for the entire cluster. It should support RAIDed hard drives (more on that later) and must have at least two Ethernet ports, one to connect to the outside world and the other for the isolated internal compute node network.

- Compute nodes ideally should be small, cheap and plentiful. Depending on the application of the cluster, they could have any combination of powerful CPUs, large amounts of RAM, large hard disks or GPUs.

Networking Your network switch needs to have at least as many ports as you have compute nodes. Extra ports always are handy in case you decide to add to your cluster in the future. Network cables also are essential.

Physical Configuration A cluster can be constructed from tower PCs on a shelf; however, a professionally built cluster typically will use special rack-mounted computers. Computers often will be on rails allowing you to slide them out far enough to remove the lid without physically detaching them from the rack. It is advisable to leave enough slack in the cables on the backs of the computers so they can be running while pulled out for diagnostic purposes.

An important consideration is the location for the cluster. A cluster can be rather noisy due to the fans, so put it in a place where you won't mind some extra white noise. A cluster also can generate a lot of heat. If it's large, you'll need ventilation or a dedicated air conditioning unit.

Power Distribution Computers draw a lot of power, and lots of computers draw lots of power. The circuits they're on must be able to handle the draw, and you'll need power strips to distribute the power. If your cluster is small, a few consumer-grade power strips should be adequate. Otherwise, large rack-mountable power strips exist that report current.

Additionally, the head node and storage unit should be plugged in to an uninterruptible power supply (UPS), so that they don't immediately halt

on a power outage, potentially corrupting data.

Access It isn't practical to have a keyboard, video monitor and mouse (KVM) for every node in the cluster. It is a good idea, however, to have a local KVM hooked up to the head node. This guarantees that you always will be able to access your cluster to perform administrative tasks. There are specialty products, such as a rack-mountable LCD monitor and keyboard, that can serve this purpose well.

Once the cluster is set up, you will be able to access compute nodes using SSH from the head node. Under normal operation, the nodes then can be headless (operate without a KVM). Under abnormal operation, such as when initially setting up the cluster or when diagnosing a problem, you can access the compute nodes using a crash-cart, which is a mobile KVM with long cables that you can plug in to whatever node is having troubles. Another option is a KVM switch. These switches can use standard IO cables (such as USB and VGA), or they can work entirely over IP if the computers' BIOSes support it. The more nodes involved, the pricier the KVM and cables will be.

Spares Stuff breaks. When you have a lot of stuff (as in a cluster), it breaks often. For example, let's say that you have 100 hard disks among all of the computers in your cluster, and each hard disk is rated to operate for 20 years. This is an annual failure rate of 5%, so you can expect five of them to fail in a year, or roughly one every ten weeks. This analysis applies to all computer components, meaning that in addition to spare hard disks, it's also a good idea to purchase spare RAM, power supplies and possibly even motherboards and CPUs. In a large cluster, it's wise to have spare networking equipment, and in a production environment, an entire spare head node. Spare parts for compute node repairs are not as necessary since dysfunctional nodes simply can be taken offline or cannibalized for parts.

In summary, the parts needed to build your own cluster are as follows:

- Head node (optionally a storage node and a spare as well).
- RAID storage (integrated directly into the head node or storage node, or as a separate device).

- Compute nodes.
- Networking switch(es).
- Networking cables.
- Rack and mounting hardware.
- Power strips.
- Uninterruptible power supply.
- KVM switch and cables.
- Spare parts kit (hard drives, RAM, power supplies).

Network Setup

Once you settle on hardware, you need to plan how to connect it up. Let's start with communication. First, give your cluster a name. Names are used as aliases for IP addresses, making it much easier for a human to identify individual computers on a network. A cluster computer uses two different networks: the external network (aka the internet) that only the head node connects to, and the internal network that the cluster uses for internal communication. Therefore, two names must be configured, one for the external network and one for the internal network:

- **External network:** this is used only by the head node. The name on the external network typically is formatted as `hostname.domain.suffix`, where the `hostname` is whatever you want, and the `domain.suffix` pertains to the organization using the cluster. The example used in this guide is `name.university.edu`.
- **Internal network:** this is used by all nodes in the cluster. The internal name is typically the hostname component from the external name used in conjunction with a numbering scheme. For example, we append two digits to the end of the hostname for each

node: `name00` (head node), `name01` (first compute node) and so on. This scheme limits us to 100 nodes, but it easily can be expanded to accommodate future upgrades.

Naming computers is vital for humans to be able to maintain the cluster, but the computers themselves deal with numeric IP addresses. The method for obtaining an IP address on the external network is up to your network administrator, but you get full reign over the internal network. Two methods exist for assigning IP addresses in the internal network: static and dynamic assignment.

- **Static assignment:** each compute node is configured individually with its own IP address. This contradicts the scalability goal of this guide, because manually configuring IP addresses for a large number of nodes is not practical.

- **Dynamic assignment:** each compute node has an identical configuration and receives its IP address from the head node through the network based on its unique MAC address. This guide uses dynamic IP assignment.

Storage Node

So far in our description of a cluster, we have mentioned a single head node that acts as an access point to the cluster, along with many compute nodes to perform the tasks the cluster receives. Many large clusters will separate out tasks further, especially if the head node becomes a bottleneck for cluster operation. For example, it is common to have a separate storage node to manage the files to which the compute nodes need access, such as application software and each user's home directory.

Disk Partitioning

As opposed to Windows, where partitions are referred to as lettered drives, in Linux, they are mounted under directories called "mount points" in the filesystem. Partitions are useful for keeping data, applications and system software separate for easy backups and re-installations. This section highlights useful Linux partitions assumed in this article:

- **root (/)** — This partition is where the actual operating system resides, and other partitions will be mounted in its filesystem.

- **/boot** — The files Linux uses to boot, including the kernel itself, are located here. For mostly historical reasons, some administrators prefer to keep this on a separate partition, but we will keep them on the same partition as root.

- **/admin** — Disk images, software distributions, kickstart files and backups are stored here. This is vital for the installation of all compute nodes in a scalable way.

- **/home** — User files are located here. We will make this a separate partition from root for ease of backups, upgrading and re-installations.

- **/export** — System-wide application software to be run on compute nodes is stored here. Although sometimes a partition of its own, it can be subsumed under /home/export instead.

- **/scratch** — Hefty computations like those done on clusters often involve writing large temporary files to the hard disk over the process of the computation, then reading those files to complete a result. It is recommended to have a large partition on all compute nodes set aside for this purpose.

- **swap** — Swapping is the process by which, should Linux run out of memory, it writes pages of memory to the swap partition on the hard disk. This can result in allowing memory-intensive software to run on systems with too little RAM. However, swapping is orders of magnitude slower than using RAM. Perhaps a decade ago when RAM was expensive, it would have been advisable to have a large swap partition. But now that RAM is cheap, it is best not to swap at all but instead buy more RAM if memory constraints become a problem, or use software that is designed to use the /scratch partition.

If your nodes use multiple disks, you will have the choice of which ones

Table 1. Head Node

/ (includes boot)	200GB
/admin	200GB
/home (includes export)	rest of space

Table 2. Compute Node

/	200GB
/scratch	rest of space

to use for which partitions. By convention, the root partition should go on the first hard disk, but the rest is up to you. Tables 1 and 2 are examples of single disk partitioning schemes using 1TB hard drives.

RAID Devices

When storing large amounts of data, it is highly recommended to utilize a RAID (Redundant Array of Independent Disks) device. This may be integrated directly into the head node, a separate component connected to the head node or part of a separate storage node.

A RAID device works by combining several small physical drives to form one larger, faster virtual drive. A RAID device also can introduce data redundancy, which allows a drive to fail while still preserving the data. This is absolutely vital in a production environment. As mentioned earlier, with many components comes frequent component failures. A drive on a compute node failing isn't the end of the world, because there's nothing on it that can't be re-installed, so clusters often will use a single hard drive for each compute node. However, losing all of the cluster's application or user data would be a disaster, making RAIDing of head node partitions a must.

Several commonly used RAID levels achieve increased size and speed, redundancy or both of these goals.

- **RAID 0** provides a storage size and performance increase by "striping" data across two or more drives. This means that consecutive data segments are stored on different disks. This may improve read time significantly in some applications; however, one failed drive causes all of the data to be lost. A RAID 0 drive is as large as the size of its smallest drive times the number of drives.

- **RAID 1** provides redundancy with no storage size or performance increase by "mirroring" data writes to two or more disks, allowing one to go down while still preserving the data. The size of a RAID 1 drive is the same as its smallest drive.

- **RAID 5** is similar to RAID 0 except that it includes redundant parity information spread across the three or more disks. This allows any one disk to fail without the loss of data. The RAID as a whole will store as much as the smallest drive times one less than the number of drives.

- **RAID 6** is similar to RAID 5 except that it has two disks worth of redundant parity information spread across four or more drives. This allows for two disks to fail without the loss of data. Storage will be limited to the size of the smallest drive times two less than the number of drives.

Hot Spares are blank drives included with a RAID 5 or 6 device. When one drive fails, the RAID device rebuilds the information formerly on that drive onto the hot spare using the redundant information spread across the other drives. If hot spares are not included, this process would begin only when you manually swap out the failed disk. If you happen to be on vacation and can't get a friend to perform this task, you run the risk of additional drives failing and resulting in data loss before you return.

No matter how many hot spares you provision yourself with, your data isn't 100% safe from disk failures wiping it out. Using a RAID device may make the likelihood of losing your data smaller, but it won't eliminate it. Therefore, if your data is at all important (you're going through the effort of building a cluster in order to obtain it, so it is), make sure you have access to another machine in a different physical location to which you can back up files.

Assembly

After you have gathered all your hardware and planned the configuration, you can begin the fun part of cluster work: actually assembling your cluster. Arrange your nodes for good air flow. When running cables, make sure to use differently colored cables for the internal and external

networks, and label them on both ends. It can be both time-consuming and frustrating to track down a problem only to find that it was caused by a swapped network cable. If things are kept consistent among nodes, your life will be much easier when it comes to managing your cluster. Ideally when starting out, your compute nodes all should be identical, both in terms of internal hardware and external cable configuration. In our experience, however, this ideal is seldom maintained over the long-term as the cluster expands or specialized capabilities are added.

Maintenance

Scalability for a cluster requires that it is easy to set up all of the compute nodes with identical configurations. This ability is useful in several scenarios: to install the cluster initially, to re-install a node for maintenance, or to add new nodes to the cluster.

There are two methods for achieving this goal. The first is to perform a complete installation on a single node, save a disk image and write that image to all other nodes. Unfortunately, this strategy results in a loss of support for heterogeneity. If you desire to add nodes of a different architecture than what's already in the cluster, you'd be forced to start from scratch in installing them.

The other method is to script all the changes to the operating system so that they can be applied during an automated install. Such installation scripts typically record basic settings similar to those that would be configured in the system installer, a list of software packages to install beyond the initial system and a script to handle all other modifications.

This installation method solves the issue of heterogeneity in that the installer handles the choice of software, allowing the same script to be used on multiple architectures assuming that all requested software is packaged for the different architectures. Furthermore, a script containing an exhaustive list of modifications from a clean install is an excellent resource when diagnosing future issues or for performing an operating system upgrade. Most major distributions support this method. On CentOS, it is called kickstart.

In practice, a combination of these two methods is used. For example, scripts are used to build the cluster, and an image can be used to replace a bad hard disk on a compute node.

System Software

In Part II of this series we will dive into installing CentOS using kickstart. This procedure involves performing a single manual installation to generate the base kickstart file, then iteratively making modifications until the operating system installs on the head node and compute nodes without manual intervention. The end result will be a functional operating system on every node with networking in place.

In Part III, we will describe the installation and configuration of software that makes these networked computers function as an integrated cluster. This software includes DHCP for IP addresses assignment, NFS to share filesystems over the internal network, passwordless SSH between all nodes, a suite of administrative tools, a local software repository for supplying RPMs to compute nodes, Ganglia for monitoring the cluster and SLURM as a resource manager.

Application Software

When building a cluster, it's important to know what you're going to do with it. You already should have done some research to be sure that software exists to achieve your goal. For example, we built a high-throughput computational chemistry cluster that runs quantum chemical programs and molecular dynamics simulation software. This requires compute nodes with multicore CPUs, GPUs, large amounts of RAM and significant scratch space.

An important consideration is licensing of the application software that will run on the cluster. For example, there are many free or open-source computational chemistry programs for which licensing isn't a problem, such as MOPAC, GAMESS and ORCA. One can purchase a site license for commercial software, such as Gaussian, and use it across the cluster. Other commercial programs like QChem require being keyed to the specific nodes on which they will be running.

Conclusion

In this article, we discussed the considerations that go into designing a cluster computer. To start, we outlined the design goals, a vital one being that the setup is scalable to accommodate any number of nodes without their installation and administration becoming impractical. We

then discussed the hardware that goes into the cluster, including the computers, networking equipment, physical storage, power distribution, access and spare parts. We also designed a disk partitioning scheme for the head node and compute nodes that allows for easy backups, upgrades and re-installations. We described the networking of a cluster, including an external network connection and an isolated internal network. Finally, we discussed the physical assembly of the cluster, introduced the importance of maintenance and touched on the cluster's application.

In the next article, we will install the base operating system and set up network connectivity. In the process, we will create two kickstart files, one for the head node and the other for the compute nodes. In the third article, we will turn the group of computers into a single cluster by configuring vital system services to communicate and run cluster software. ■

Nathan Vance is a computer science major at Hope College in Holland, Michigan. He discovered Linux as a high-school junior and currently uses Arch Linux. In his free time, he enjoys running, skiing and writing software.

Mike Poublon is a senior data-center network engineer and technical lead at Secant Technologies in Kalamazoo, Michigan. He has extensive professional experience in networking and high-performance computing systems. As a student, he built Hope College's first production computer cluster.

William Polik is a computational chemistry professor at Hope College in Holland, Michigan. His research involves high-accuracy quantum chemistry using computer clusters. He co-founded WebMO LLC, a software company that provides web and portable device interfaces to computational chemistry programs.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

SUPERMICRO®
MARKETPLACE

Powered by Silicon Mechanics

Broad Selection

Zero Defects

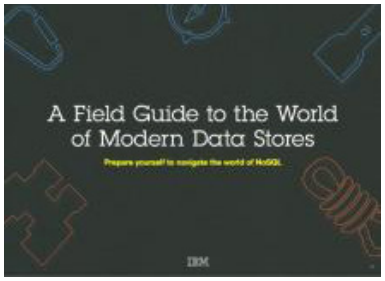
3-Year Warranty

Your Source for Supermicro Platform Technology

Configure Now

Talk to a Supermicro Expert!

866.352.1173



A Field Guide to the World of Modern Data Stores

There are many types of databases and data analysis tools to choose from when building your application. Should you use a relational database? How about a key-value store? Maybe a document database? Is a graph database the right fit? What about polyglot persistence and the need for advanced analytics?

If you feel a bit overwhelmed, don't worry. This guide lays out the various database options and analytic solutions available to meet your app's unique needs.

You'll see how data can move across databases and development languages, so you can work in your favorite environment without the friction and productivity loss of the past.

Sponsor: IBM

> **https://geekguide.linuxjournal.com/content/field-guide-world-modern-data-stores**

Why NoSQL? Your database options in the new non-relational world

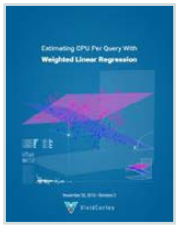
The continual increase in web, mobile and IoT applications, alongside emerging trends shifting online consumer behavior and new classes of data, is causing developers to reevaluate how their data is stored and managed. Today's applications require a database that is capable of providing a scalable, flexible solution to efficiently and safely manage the massive flow of data to and from a global user base.

Developers and IT alike are finding it difficult, and sometimes even impossible, to quickly incorporate all of this data into the relational model while dynamically scaling to maintain the performance levels users demand. This is causing many to look at NoSQL databases for the flexibility they offer, and is a big reason why the global NoSQL market is forecasted to nearly double and reach USD3.4 billion in 2020.

Sponsor: IBM

> **https://geekguide.linuxjournal.com/content/why-nosql-your-database-options-new-non-relational-world**

Estimating CPU Per Query With Weighted Linear Regression



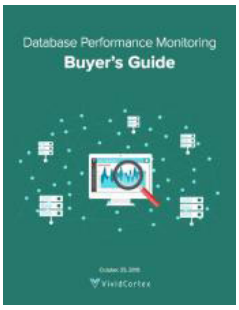
Your database server is suddenly using a lot of CPU resources. Quick, what caused it? This is a familiar question for engineers of all persuasions. And it's often impossible to answer.

There are good reasons why it's hard to figure out what consumes resources like CPU, IO, and memory in a complex piece of software such as a database. The first problem is that most database server software doesn't offer any way to measure or inspect that type of performance data. The database server isn't observable. This problem arises in turn from the complexity of the database server software and the way it does its work, which actually precludes measuring resource consumption accurately!

Author: Baron Schwartz

Sponsor: VividCortex

> **https://geekguide.linuxjournal.com/content/estimating-cpu-query-weighted-linear-regression**



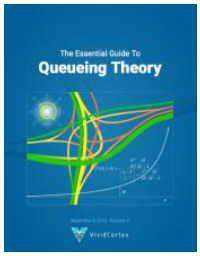
Database Performance Monitoring Buyer's Guide

More and more companies have begun to recognize database performance management as a vital need. Despite its widespread importance, good database performance management requires specialized expertise with custom approaches--yet all too often, organizations rely on one-size-fits-all solutions that theoretically "check the box" but in practice do little or nothing to help them find or prevent database-related outages and performance problems.

This buyer's guide is designed to help you understand what database management really requires, so your investments in a solution provide the greatest possible ultimate value.

Sponsor: VividCortex

> **https://geekguide.linuxjournal.com/content/database-performance-monitoring-buyer%E2%80%99s-guide**



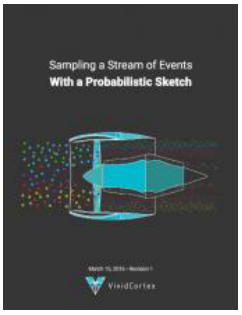
The Essential Guide To Queueing Theory

Whether you're an entrepreneur, engineer, or manager, learning about queueing theory is a great way to be more effective. Queueing theory is fundamental to getting good return on your efforts. That's because the results your systems and teams produce are heavily influenced by how much waiting takes place, and waiting is waste. Minimizing this waste is extremely important. It's one of the biggest levers you will find for improving the cost and performance of your teams and systems.

Author: Baron Schwartz

Sponsor: VividCortex

> **https://geekguide.linuxjournal.com/content/essential-guide-queueing-theory**



Sampling a Stream of Events With a Probabilistic Sketch

Stream processing is a hot topic today. As modern Big Data processing systems have evolved, stream processing has become recognized as a first-class citizen in the toolbox. That's because when you take away the how of Big Data and look at the underlying goals and end results, deriving real-time insights from huge, high-velocity, high-variety streams of data is a fundamental, core use case. This explains the explosive popularity of systems such as Apache Kafka, Apache Spark, Apache Samza, Apache Storm, and Apache Apex—to name just a few!

Author: Baron Schwartz

Sponsor: VividCortex

> **https://geekguide.linuxjournal.com/content/sampling-stream-events-probabilistic-sketch**

Will Anything Make Linux Obsolete?

Or is Linux immortal?

DOC SEARLS

Doc Searls is Senior Editor of *Linux Journal*. He is also a fellow with the Berkman Center for Internet and Society at Harvard University and the Center for Information Technology and Society at UC Santa Barbara.

PREVIOUS
Feature: BYOC: Build Your Own Cluster, Part I—Design

Remember blogging? Hell, remember magazine publishing? Shouldn't be hard. You're reading some now.

Both are still around, but they're obsolete—at least relatively. Two cases in point: my blog (http://blogs.harvard.edu/doc) and *Linux Journal*.

Back when blogging was a thing, in the early 2000s, about 20,000 people subscribed to RSS feeds of my original blog (1999–2007, still mothballed here: http://doc.weblogs.com). At its peak, I posted many times per day and had a strong sense of connection with my readership.

Same went, by the way, for my postings in *Linux Journal*, on our website and on one of our own blogs, called IT Garage—lots of readers, lots of engagement.

Most early bloggers were journalists by profession or avocation—good writers, basically. Some blogs turned into online pubs. BoingBoing, TechCrunch and TPM all started as blogs.

But blogging began to wane after Twitter and Facebook showed up in 2006. After that journalism also waned, as "content generation" became the way to fill online publications. Participating in "social media" also became a requisite function for journalists still hoping to stay active online (if not also employed).

These days, I blog only a few times per month, for readers that range in number from dozens to hundreds. Usually I duplicate those posts in *Medium*, where they get about the same numbers. Meanwhile, I have 23.7k followers on Twitter (as @dsearls). Although that's a goodly number, you could say the same for the average parking space. (Which, if it could speak, might say "Hey, I had 25k impressions on passing drivers today and engaged 15.") From what I can tell from counting clicks of shortlinks I produce with Bit.ly, most of my tweets are clicked on by a few dozen people, tops. I'd gladly trade all my followers (and my Klout score of 81) for the actual readers I had in my old blog. But alas, this is now.

Thanks to loyal subscribers, *Linux Journal* is still going strong, proving it's still possible to operate a journal that isn't just another sluice for "content".

But we have to face the facts here: content production has clearly obsolesced journalism—just like TV obsolesced radio, cars obsolesced horses and printing obsolesced scribes. All of the obsolesced things do persist, but in a diminished state relative to what obsolesced them.

To understand why and how, it helps to raid the works of Marshall McLuhan, the media scholar best known for saying "the media is the message" (or "the massage"—he said both) and coining the term "global village" decades before the internet materialized one.

The analytic system McLuhan and his colleagues created for understanding how one medium obsolesces another is the *tetrad* (Greek for *group of four*). Every medium, he said, does four things. These are discovered in answers to four questions:

- What does the medium enhance?
- What does the medium make obsolete?
- What does the medium retrieve that had been obsolesced earlier?
- What does the medium reverse, or flip into, when pushed to extremes?

Graphically, it is represented as shown in Figure 1.

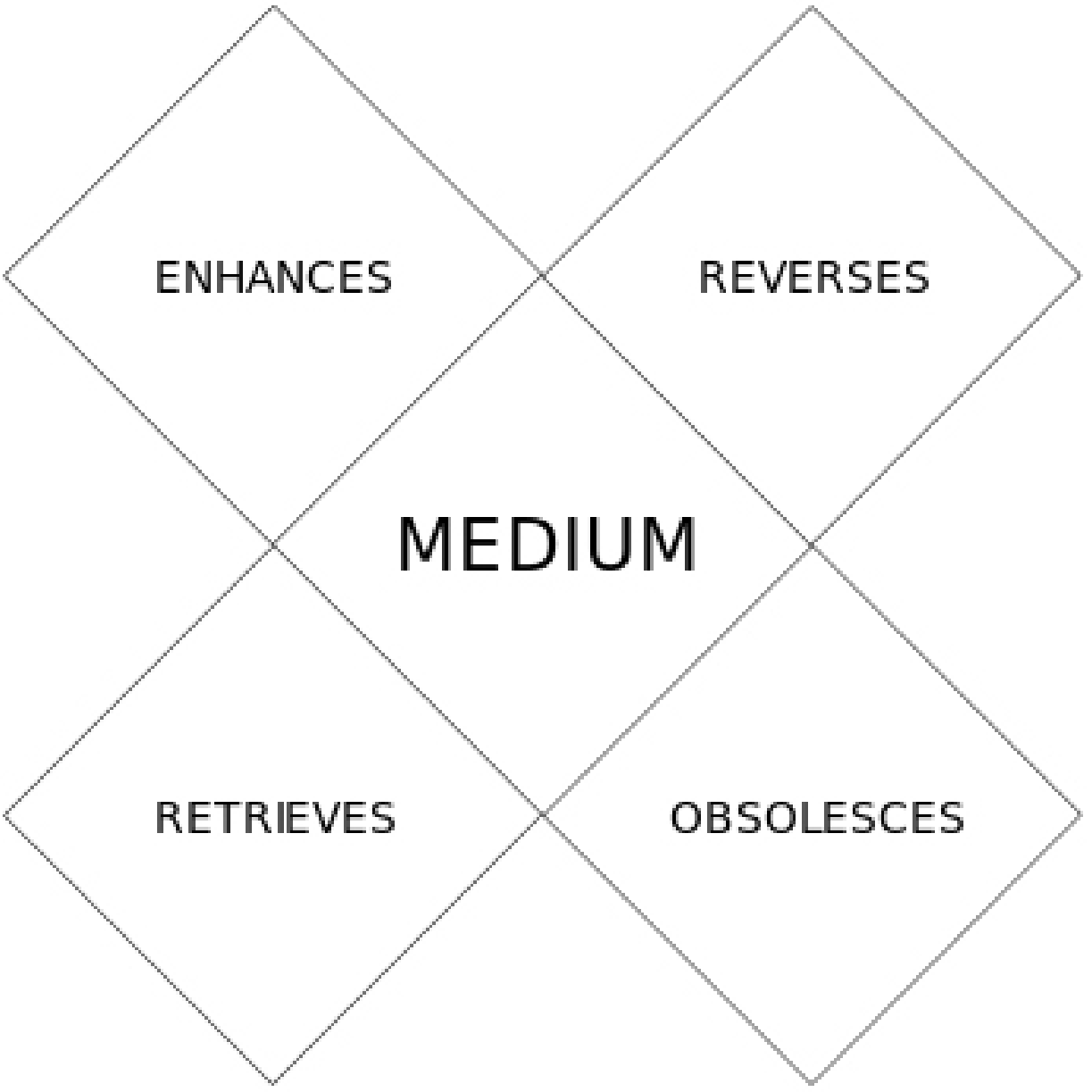


Figure 1. Media Tetrad

So, for example, radio—

- enhanced music, news and talk.
- obsolesced live performance—you didn't have to be there. (Note: I use *obsolesce* and *obsolesced* to be consistent both with the McLuhans' usage and norms more established than those of the computing world, where using *obsolete* and *obsoleted* as present and past verb forms of *obsolesce* is more common.)
- retrieved recordings.
- reversed or flipped into isolation (of individuals and families from social interaction).

There can be many other answers, all arguable. Note: the above are my answers, not McLuhan's. That's why the tetrad poses questions. McLuhan wanted us to understand that each of the four "laws of media" (McLuhan's term) apply to every medium, from stone tools to machine learning. Every new thing does all four, one way or another—or in many ways.

So let's apply the tetrad to social media. It—

- enhanced conversation (many more people participating, over all distances).
- obsolesced journalism (now anybody could spread the word about anything).
- retrieved gossip.
- reversed or flipped into tribalism (people talking inside algorithmically isolated echo chambers, both feeding and feeding on each others' beliefs and prejudices).

Again, there can be other answers, but those are ones I'm prepared to argue for. (And to consider yielding on, if anyone wants to engage in some fun co-thinking about the topic.)

So now, let's put Linux through the tetrad. I'll say it—

- enhanced compatibility and usefulness.
- obsolesced proprietary UNIX.
- retrieved tools, code and methods developed for UNIX.
- reversed or flipped into empire: world domination was achieved.

Again, it's arguable. But let's not stop there.

Eric McLuhan (Marshall's son and co-author of several McLuhan books) writes, "If you know in advance that any innovation will have at least four (and possibly many more) kinds of side effects, then you can reasonably predict that fact and you know where to begin looking for specifics." So let's look through the tetrad toward what might obsolesce Linux. Whatever it is, it will—

- enhance _____.
- obsolesce Linux.
- retrieve _____.
- reverse or flip into _____.

To approach these questions, it helps to understand formal causality. In *Media and Formal Cause*, Eric McLuhan writes:

> Formal causality kicks in whenever "coming events cast their shadows before them." Formal cause is still, in our time, hugely mysterious. The literate mind finds it is too paradoxical and irrational. It deals with environmental processes and it works outside of time. The effects—those long shadows—arrive first; the causes take a while longer.

Formal cause was one of four outlined by Aristotle in his treatises on

physics and metaphysics. Those were (and I simplify here):

- Material—what something is made of.
- Efficient—how one thing acts on another, causing change.
- Formal—what makes the thing form a coherent whole.
- Final—the purpose to which a thing is put.

In *Understanding Media*, Marshall McLuhan writes, "Any technology gradually creates a totally new human environment", adding:

> Environments are not passive wrappings but active processes....The railway did not introduce movement or transportation or wheel or road into society, but it accelerated and enlarged the scale of previous human functions, creating totally new kinds of cities and new kinds of work and leisure.

In other words, railways were the formal cause that scaled up new kinds of cities, work and leisure. Railways, as a formal cause, were also what McLuhan called an "anti-environment". What made it anti- was that people tended not to look at it.

"People don't want to know the cause of anything", Marshall said (and Eric quotes, in *Media and Formal Cause*). "They do not want to know why radio caused Hitler and Gandhi alike. They do not want to know that print caused anything whatever. As users of these media, they wish merely to get inside, hoping perhaps to add another layer to their environment...."

In *Media and Formal Cause*, Eric also sources Jane Jacobs, the reigning authority on cities:

> Current theory in many fields—economics, history, anthropology—assumes that cities are built upon a rural economic base. If my observations and reasoning are correct, the reverse is true: that rural economies, including agricultural work, are directly built upon city economies and city work....Rural production is literally the creation of city

consumption. That is to say, city economics invent the things that are to become city imports from the rural world.

In this same way we can see Linux as a formal cause of cloud services, content delivery networks, Android devices and much more. All those are so interesting as environments that they misdirect observers from Linux as a formal cause of them.

I believe this is one reason why Linux kernel development is dedicated toward all possible uses, rather than the special purposes of any one application. The Linux anti-environment is a medium that formally causes the vast theater of attractions in all the environments that run atop it, just as railways cause coal mines and power plants, and cities cause farms.

So, now that world domination has been achieved, can we look forward and see what may or may not portend obsolescence for Linux?

Maybe we can see one possibility in Figure 2.

Figure 2 is from Netcraft's February 2017 Web Server Survey (https://news.netcraft.com/archives/2017/02/27/february-2017-web-server-survey.html), which shows Apache's precipitous drop since last

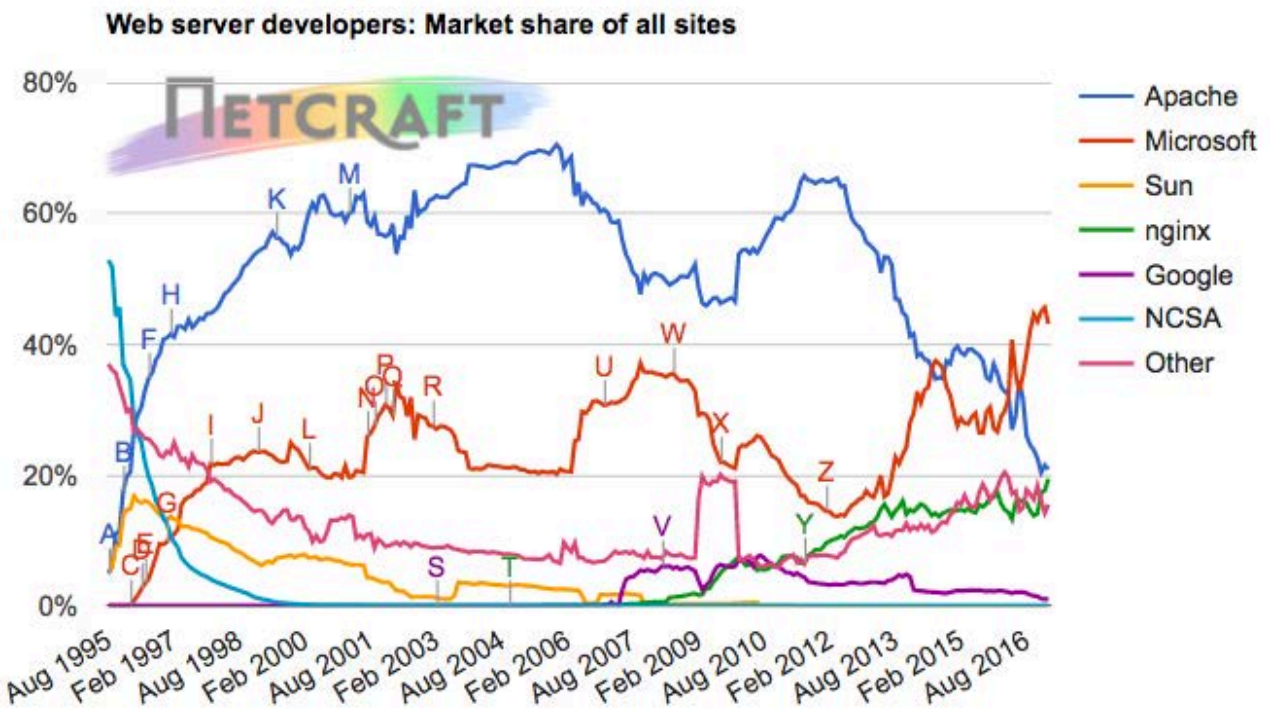


Figure 2. Netcraft's February 2017 Web Server Survey

AWS is a triumph of centralization. So are Twitter, Facebook and other social-media giants that obsolesced journalism and blogging.

peaking in 2012, yielding first to Microsoft's proprietary offerings and then to nginx and Other.

Credit where due: like Apache, nginx is open source (with a BSD license), and it was purposefully created as an open-source alternative to Apache. So maybe one should add Apache and nginx together. We do that with Linux and BSD sometimes.

But my point here isn't about open source. It's about centralization of environments in both the technical and McLuhanesque senses of the word. nginx and Microsoft are most successful in the big server and cloud worlds. It is now standard for developers to back-end all kinds of stuff in the clouds of Amazon, Rackspace and other central giants of what we might call utility computing. What are *they* causing in a formal way? Well, let's run AWS through the tetrad. It has—

- enhanced ease and scale for development and hosting.
- obsolesced distributed on-site development and hosting.
- retrieved centralization.
- reversed or flipped into massive vulnerability (such as when a typo in March brought down some huge services that run on AWS: https://aws.amazon.com/message/41926).

AWS is a triumph of centralization. So are Twitter, Facebook and other social-media giants that obsolesced journalism and blogging. And, even though I wrote in my last article that we're due for a pendulum swing in a distributed direction, I do have to give silos their due (http://www.linuxjournal.com/content/giving-silos-their-due)—especially

when so many of them run on Linux.

Given that nine out of the ten most reliable hosting sites in Netcraft's most recent list (February 2017: https://news.netcraft.com/archives/2017/03/03/most-reliable-hosting-company-sites-in-february-2017-3.html) run on Linux (the tenth runs on BSD), I've got to wonder if Linux is capable of obsolescence. If it is, and you can tell us what will obsolete it—or retrieve it the same way Linux retrieved what was already developed for UNIX—maybe we can start a magazine for it.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

ADVERTISER INDEX

Thank you as always for supporting our advertisers by buying their products!