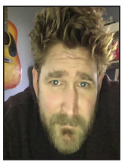
iVoyeur
Using NCPA: Nagios Cross-Platform Agent

DAVE JOSEPHSEN



Dave Josephsen is the sometime book-authoring developer evangelist at Librato.com. His continuing mission: to help engineers worldwide close the feedback loop. dave-usenix@skeptech.org

I've been working at home for over a year now, and I can't help but feel that I'm somehow doing it incorrectly. I'm wearing pants for one thing, and my hygiene habits have not changed whatsoever (although admittedly I never was a hygiene Olympian). In fact I seem to be experiencing very few of the great benefits one hears about, like drinking at inappropriate times, playing video games, not interacting with people, and, well, not working.

In their place I'm experiencing a whole slew of not very awesome side effects of having these large, luxurious blocks of uninterrupted time to dig in and work on stuff. These include failing to stop working ever and starting too many side-projects because of all the "extra time" I feel like I have (that I don't actually have). I even have meetings. Oh crap, in fact I have meetings right now; I'll be right back.

Okay sorry, that's another problem: meetings sneak up on me now, and nearly always coincide with one meal or another that I'm supposed to be eating. An unlucky consequence, I suppose, of the dissonance between the people in my life who make meetings and live two hours ago, in California, and the people in my life who make lunch and dinner and who live now, in Texas. It also has begun to seem weird that we have times for these things at all, eating, meeting, and working, that is.

When I applied for this job, my first several interviews were undertaken by way of Google Hangouts. This was a very real logistical concern for me at the time because I was running a snowflake everything-compiled-from-scratch Linux laptop, and, well, you know how that goes with cameras, soundcards, printers, and etc. I got it all sorted out in time, and experienced my first few video-chats as job interviews, which, by the way, is not a very good idea. It was extremely awkward and I kept spacing out. It felt like I was watching a job interview on TV, so I kept forgetting to answer.

Anyway, I spend an inordinate amount of time on Hangouts, appear.in, and various other hosted impromptu meeting services these days, and I've noticed that whenever Hangouts is going, my CPU fans kick on. This is pretty noticeable on my MacBook, but downright distracting on my ThinkPads. My poor little ThinkPad x120 gasps and wheezes like it's sprinting the last 30 feet of an ultra-marathon when I try to run Hangouts on it.

Being a monitoring sort of person, I got curious about this behavior, and brought some tools to bear to help me visualize the overhead, but I pretty quickly got myself entangled in the question of whether I was comparing apples to apples. I mean literally. Is it the same thing to measure CPU utilization on an Apple vs. a Linux box?

At this point I should point out that not only am I lazy by nature, but I also really don't have the time to put any actual effort into this, so I figured the shortest path was probably to get my hands on a cross-platform monitoring agent. That would at least make me feel like I was measuring both systems with the same ruler, and that'd probably help me to brute-force ignore the screams of protest from my inner engineer.

Because I spend my time these days thinking about and working with telemetry processing systems, I haven't really looked at the state of client-side data collection tools lately (especially tools that might work on a Mac). There aren't many cross-platform monitoring agents that include support for OS X. The most robust solution is probably DataDog, but that was overkill for my purposes. I wanted something I could use for a few days and then get rid of, and setting up DataDog would entail ... artifacts like emails, and passwords on Web sites, and well-intentioned pre-sales, and support representatives.

Really, I just wanted something like good-old GKrellM, so I spent a few minutes trying to get GKrellM to build on my Mac, which was fun but fruitless. I was also a little surprised to find there was no homebrew recipe for GKrellM; "brew install X" so rarely fails me nowadays. Then I remembered NCPA.

NCPA, or the "Nagios Cross-Platform Agent," is a monitoring agent built and maintained by the folks at Nagios Enterprises. It's a cross-platform Python script that is distributed in binary form (via cx_Freeze). In many ways, it's exactly what you'd expect if you asked Nagios Enterprises for an agent. It's small, easy to work with, and, out of the box, it doesn't really know how to monitor very much of anything. It can enumerate the running processes and measure CPU, Memory, Disk, and Network utilization. And it does a great job of detecting all of these things (it sees all of my vmnet network interfaces, for example), but like its big brother it depends on plugins to do the heavy lifting, and that's a good thing IMO.

I'd never tried NCPA, so I thought this would be a great opportunity. It, along with Nagios Core and the rest of the open source software made by Nagios Enterprises, is on GitHub. I must be getting old, though, because I just went and grabbed the official binary distributions of NCPA for OS X and Debian from [1]. The Linux install was pretty much what you'd imagine: one dpkg-i and it was up and running.

The Mac put up a little more resistance. NCPA came packaged in a disk-image (.dmg file), which contained an installer shell script called install.sh. I could not chmod the script to make it executable because .dmg's are a read-only file system. All of my

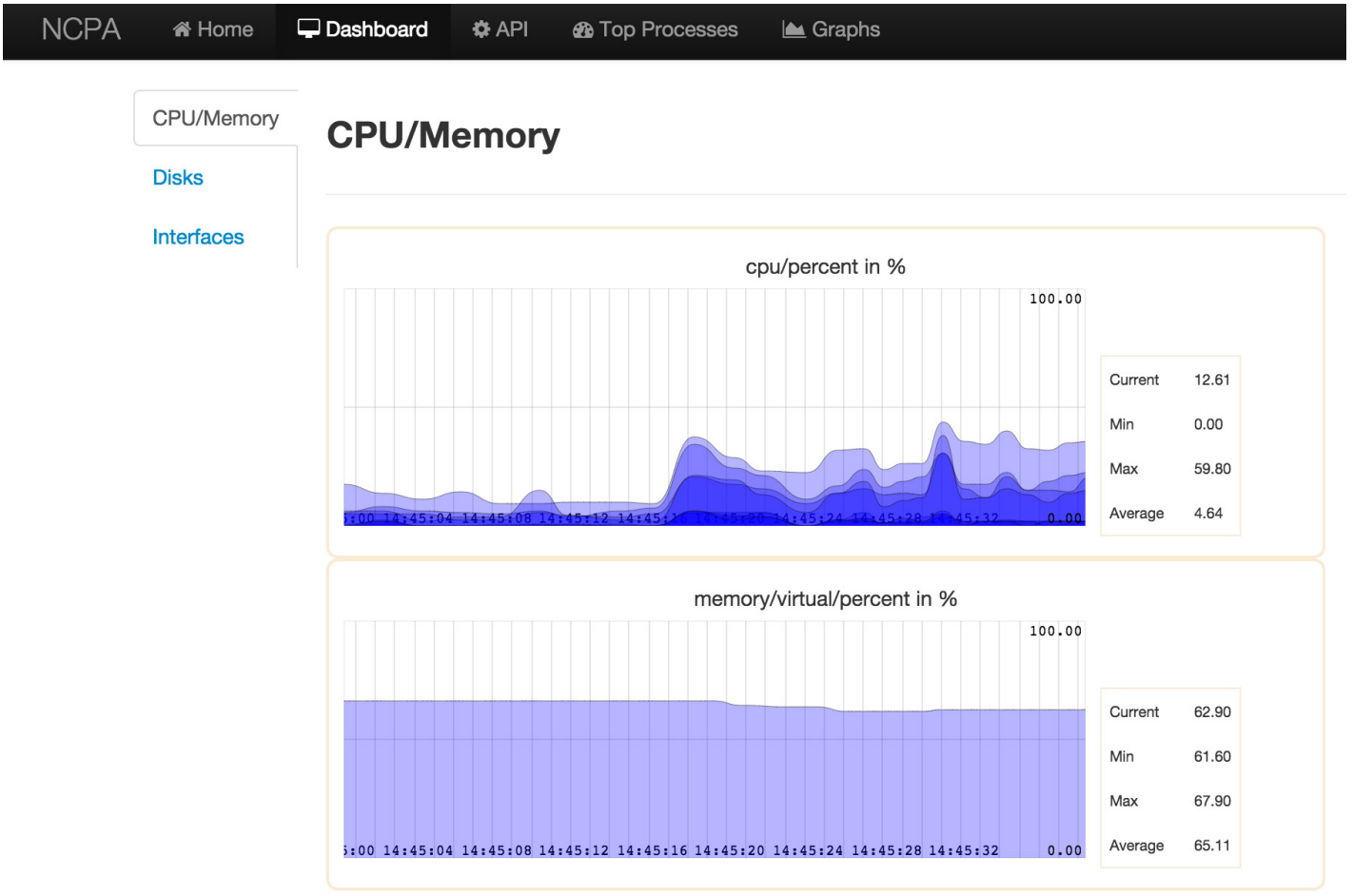


Figure 1. NCPA's spartan but functional built-in Web interface

attempts to remount it as R/W were mockingly rejected by OS X. Giving up on that, my first few attempts to indirectly execute the script with, for example, "sh -c" were also rebuffed, but...

```
/bin/sh < /Volumes/NCPA-1.8.1/install.sh
```

...worked for me.

Like Nagios, NCPA installs itself to /usr/local/ncpa by default. Inside this directory is an "etc" where you will find an ncpa.cfg file that controls NCPA's behavior. I left most of this alone, changing only the "community_string" attribute, which specifies the auth token you use to interact with NCPA.

Compared to the other agents, like Check_MK, commonly used in the Nagios solar system, NCPA is a lot easier to install and reason about. It eschews custom protocols and provides a Web-API that responds in JSON over HTTPS on tcp/5693 by default (change this along with everything else in the config file). This is pretty great. You can interact with NCPA using cURL or anything else that can speak HTTPS, and you can parse its output with jq, or anything else that groks JSON.

It even comes with a Web UI that draws graphs!

Granted, it's missing some fundamental features that I look for in a metrics analysis tool. Its y-axis handling leaves a lot to be desired, for example, but the UI is fine for ad hoc checking out individual boxes, and obviously it was more than sufficient for my current purposes. Anyway, NCPA really isn't here to be an analysis tool; it's a lightweight, easy-to-run data collection agent. One that, if I were in the market for an agent, is actually a quite compelling choice.

I mean look at this API! There are eight top-level URIs: memory, interface, agent, CPU, disk, agent, process, and services. I can, for example, get a JSON dump of the running processes on my MacBook at

```
https://localhost:5693/api/processes/
```

I can get the free memory with

```
https://localhost:5693/api/memory/virtual/available
```

I'm oversimplifying just a tad there. If you're doing this outside of a browser, you'll need to pass in the token by setting it as an attribute in the URL like so:

```
https://localhost:5693/api/memory/virtual
/available?token=zomgsecret
```

There are a slew of other attributes we can set: for example, get Nagios-style output by setting threshold attributes like so:

```
https://localhost:5693/api/memory/virtual/available?token
=zomgsecret&warning=1&critical=2&check=true
```

If you copy or symlink some standard Nagios plugins into /usr/local/ncpa, you can even run them from the API from the agent tree like so:

```
https://localhost:5693/api/agent/plugin/check_thing
/"First Arg"/"Second Arg"/?token=zomgsecret
```

You'll get back a JSON blob of the plugin's output that looks like this:

```
{ "value": { "returncode": 0, "stdout": "Thingy looks ok! First
Arg, Second Arg\n" } }
```

If you aren't already using check_mk and especially if you're running NRPE/NRDP, then you might want to consider running NCPA as a replacement for your current remote plugin-execution framework. In my admittedly teensy experience, it's been simple and painless, and has a slew of features built in for emitting to preexisting NRDP daemons and otherwise cohabitating with your existing Nagios toolchain.

It certainly scratched my itch for comparing the utilization characteristics of the various video conferencing tools I use every day (for the moment, it looks like appear.in on my MacBook is the best option). The next time I'm helping someone design and/or build out their Nagios infrastructure, NCPA will definitely play a role.

Take it easy.

Resources

[1] NCPA agent: https://assets.nagios.com/downloads/ncpa/download.php.