First 5 Minutes Troubleshooting A Server

Back when our team was dealing with operations, optimization and scalability at [our previous company](http://wiredcraft.com/), we had our fair share of troubleshooting poorly performing applications and infrastructures of various sizes, often large (think CNN or the World Bank). Tight deadlines, "exotic" technical stacks and lack of information usually made for memorable experiences.

The cause of the issues was rarely obvious: here are a few things we usually got started with.

Get some context

Don't rush on the servers just yet, you need to figure out how much is already known about the server and the specifics of the issues. You don't want to waste your time (trouble) shooting in the dark.

A few "must have":

* What exactly are the symptoms of the issue? Unresponsiveness? Errors?
* When did the problem start being noticed?
* Is it reproducible?
* Any pattern (e.g. happens every hour)?
* What were the latest changes on the platform (code, servers, stack)?
* Does it affect a specific user segment (logged in, logged out, geographically located...)?
* Is there any documentation for the architecture (physical and logical)?
* **Is there a monitoring platform?** Munin, Zabbix, Nagios, [New Relic](http://newrelic.com/)... Anything will do.
* **Any (centralized) logs?**. Loggly, Airbrake, Graylog...

The last two ones are the most convenient sources of information, but don't expect too much: they're also the ones usually painfully absent. Tough luck, make a note to get this corrected and move on.

Who's there?

$ w

$ last

Not critical, but you'd rather not be troubleshooting a platform others are playing with. One cook in the kitchen is enough.

What was previously done?

$ history

Always a good thing to look at; combined with the knowledge of who was on the box earlier on. Be responsible by all means, being admin shouldn't allow you to break ones privacy.

A quick mental note for later, you may want to update the environment variable HISTTIMEFORMATto keep track of the time those commands were ran. Nothing is more frustrating than investigating an outdated list of commands...

What is running?

$ pstree -a

$ ps aux

While ps aux tends to be pretty verbose, pstree -a gives you a nice condensed view of what is running and who called what.

Listening services

$ netstat -ntlp

$ netstat -nulp

$ netstat -nxlp

I tend to prefer running them separately, mainly because I don't like looking at all the services at the same time. netstat -nalp will do to though. Even then, I'd ommit the numeric option (IPs are more readable IMHO).

Identify the running services and whether they're expected to be running or not. Look for the various listening ports. You can always match the PID of the process with the output of ps aux; this can be quite useful especially when you end up with 2 or 3 Java or Erlang processes running concurrently.

We usual prefer to have more or less specialized boxes, with a low number of services running on each one of them. If you see 3 dozens of listening ports you probably should make a mental note of investigating this further and see what can be cleaned up or reorganized.

CPU and RAM

$ free -m

$ uptime

$ top

$ htop

This should answer a few questions:

* Any free RAM? Is it swapping?
* Is there still some CPU left? How many CPU cores are available on the server? Is one of them overloaded?
* What is causing the most load on the box? What is the load average?

Hardware

$ lspci

$ dmidecode

$ ethtool

There are still a lot of bare-metal servers out there, this should help with;

* Identifying the RAID card (with BBU?), the CPU, the available memory slots. This may give you some hints on potential issues and/or performance improvements.
* Is your NIC properly set? Are you running in half-duplex? In 10MBps? Any TX/RX errors?

IO Performances

$ iostat -kx 2

$ vmstat 2 10

$ mpstat 2 10

$ dstat --top-io --top-bio

Very useful commands to analyze the overall performances of your backend;

* Checking the disk usage: has the box a filesystem/disk with 100% disk usage?
* Is the swap currently in use (si/so)?
* What is using the CPU: system? User? Stolen (VM)?
* dstat is my all-time favorite. What is using the IO? Is MySQL sucking up the resources? Is it your PHP processes?

Mount points and filesystems

$ mount

$ cat /etc/fstab

$ vgs

$ pvs

$ lvs

$ df -h

$ lsof +D / /\* beware not to kill your box \*/

* How many filesystems are mounted?
* Is there a dedicated filesystem for some of the services? (MySQL by any chance..?)
* What are the filesystem mount options: noatime? default? Have some filesystem been re-mounted as read-only?
* Do you have any disk space left?
* Is there any big (deleted) files that haven't been flushed yet?
* Do you have room to extend a partition if disk space is an issue?

Kernel, interrupts and network usage

$ sysctl -a | grep **...**

$ cat /proc/interrupts

$ cat /proc/net/ip\_conntrack /\* may take some time on busy servers \*/

$ netstat

$ ss -s

* Are your IRQ properly balanced across the CPU? Or is one of the core overloaded because of network interrupts, raid card, ...?
* How much is swappinness set to? 60 is good enough for workstations, but when it come to servers this is generally a bad idea: you do not want your server to swap... ever. Otherwise your swapping process will be locked while data is read/written to the disk.
* Is conntrack\_max set to a high enough number to handle your traffic?
* How long do you maintain TCP connections in the various states (TIME\_WAIT, ...)?
* netstat can be a bit slow to display all the existing connections, you may want to use ssinstead to get a summary.

Have a look at [Linux TCP tuning](http://www.lognormal.com/blog/2012/09/27/linux-tcpip-tuning/) for some more pointer as to how to tune your network stack.

System logs and kernel messages

$ dmesg

$ less /var/**log**/messages

$ less /var/**log**/secure

$ less /var/**log**/auth

* Look for any error or warning messages; is it spitting issues about the number of connections in your conntrack being too high?
* Do you see any hardware error, or filesystem error?
* Can you correlate the time from those events with the information provided beforehand?

Cronjobs

$ ls /etc/cron\* + cat

$ **for** user **in** $(cat /etc/passwd | cut -f1 -d:); **do** crontab -l -u $user; **done**

* Is there any cron job that is running too often?
* Is there some user's cron that is "hidden" to the common eyes?
* Was there a backup of some sort running at the time of the issue?

Application logs

There is a lot to analyze here, but it's unlikely you'll have time to be exhaustive at first. Focus on the obvious ones, for example in the case of a LAMP stack:

* **Apache & Nginx**; chase down access and error logs, look for 5xx errors, look for possible limit\_zone errors.
* **MySQL**; look for errors in the mysql.log, trace of corrupted tables, innodb repair process in progress. Looks for slow logs and define if there is disk/index/query issues.
* **PHP-FPM**; if you have php-slow logs on, dig in and try to find errors (php, mysql, memcache, ...). If not, set it on.
* **Varnish**; in varnishlog and varnishstat, check your hit/miss ratio. Are you missing some rules in your config that let end-users hit your backend instead?
* **HA-Proxy**; what is your backend status? Are your health-checks successful? Do you hit your max queue size on the frontend or your backends?

Conclusion

After these first 5 minutes (give or take 10 minutes) you should have a better understanding of:

* What is running.
* Whether the issue seems to be related to IO/hardware/networking or configuration (bad code, kernel tuning, ...).
* Whether there's a pattern you recognize: for example a bad use of the DB indexes, or too many apache workers.

You may even have found the actual root cause. If not, you should be in a good place to start digging further, with the knowledge that you've covered the obvious.