Dstat

Pluggable real-time performance monitoring

Dagit Linux Solutions
dag@wieers.com

and now dag@linux.com !

Who am I ?

- Started with Linux in 1994 using Slackware
- Worked 6 years at IBM Belgium Linux team
- Since 3 years: freelance Linux and Open Source consultant for big companies
- Founded RPMforge repository in 2003
- Also member of the new ELRepo project
- Ex-member of the CentOS team
- Authored and developed some tools:
  - wiipresent, unoconv, mrepo, dconf, proxytunnel and *dstat*

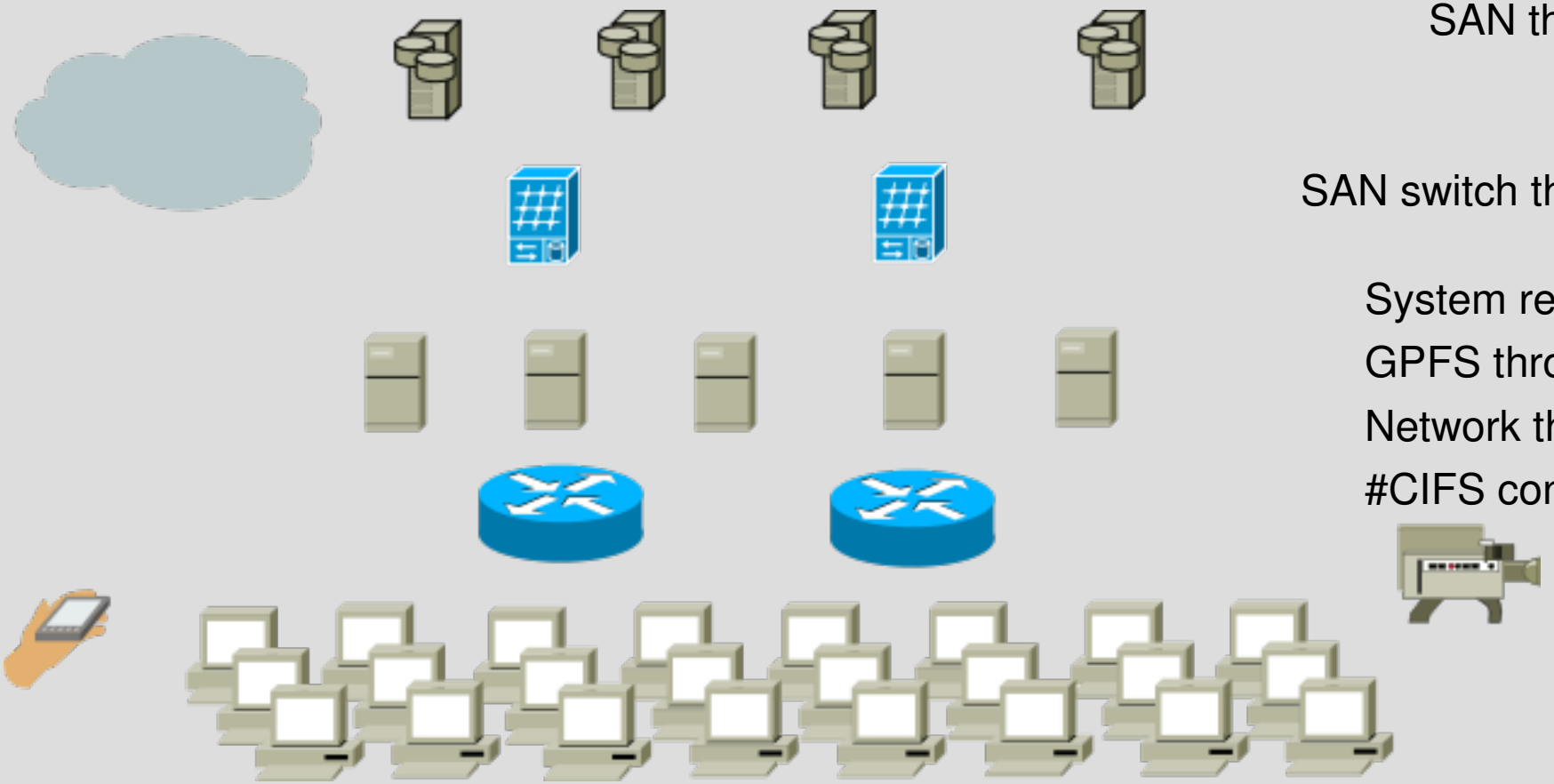
Commercial break: WiiPresent

- Giving presentations with Nintendo Wiimote
  - Browsing through slides
  - Changing workspaces or applications
  - Moving your mouse and controlling the desktop
  - Leds indicate time progress
  - Rumbling (shaking) to indicate time progress
  - Can also be used to control your Linux-based media center or software
- Get it at: http://dag.wieers.com/home-made/wiipresent/
- Less than ¥3000 across the street :-)

A case for Dstat - 1

- Customer project: install and optimize a 5 node GPFS cluster connected via 2 FC to 3 SANs (more than 128 LUNs per system)
  - 60 windows NLE clients using CIFS to connect to Samba front-ends sharing GPFS
  - GPFS allows to stripe (in parallel) to all available disks on all SANs to optimize bandwidth usage of local HBA, multipath, SAN controllers and disk expansion units
- Proof that we get the performance the customer require under different workloads

A case for Dstat - 2



SAN throughput

SAN switch throughput

System resources
GPFS throughput
Network throughput
#CIFS connections

A case for Dstat - 3

- How can I monitor multiple nodes in a cluster simultaneously ?
- How can I select only those system and application counters to validate system performance ?
- How can I make it easier to correlate counters and see usage patterns ?
- How can I follow progress and behavior and validate performance and balance during and after performance tests ?

A case for Dstat - 4

- Many tools exist to monitor resources
- Some allow to customize or write own counters
  - mrtg, nagios, cacti, munin, zabbix, ...
- Some are command line
  - vmstat, ifstat, top, *htop*, sar, ...
- None allow combining both
- Most command line tools feel old

A case for Dstat - 5

- ...and it provided an excuse to learn python at the time

A case for Dstat - 6

- Design goals (problems with eg. vmstat)
  - Needs to be easily extensible
  - Selection of counters
  - Human readable and easy to interprete
  - Show progress before showing average
  - Ability to export data for processing and reporting
- So without further ado....

Dstat output example

Dstat features

- History of counters (use terminal buffer)
- Adding unit indication (B: bytes, k: kilobytes)
- Fixed width columns
- Color highlighting
- Intermediate updates (*feel* progress)
- Adding own counters and plugins
- Exporting to CSV
- Use terminal capabilities
- Works with python 1.5.2 and later

Dstat plugins

- Comes with plenty of plugins already:
  - *time*, cpu, disk, net, mem, vm, interrupts, system, load, swap, filesystem, paging, tcp, udp, raw, unix, locks, ipc, process, ...
  - dbus, freespace, gpfs, innodb, lustre, memcache, mysql, mysql5, nfs, ntp, postfix, rpc, sendmail, utmp
  - openvz and vmware plugins
  - battery, cpufreq, power, thermal, wifi
  - topcpu, topio/topbio, topmem, topoom
- but please, let's not stop here...

Using Dstat: selecting plugins

- Internal vs. external plugins
- Internal plugins: short and long options
- External plugins: long options or -M *name*
- Example:
  - dstat -tcd
  - dstat –time –cpu –disk
  - dstat -M time,cpu,disk
  - dstat -M time -M cpu -M disk

Using Dstat: ordering plugins

- The order of the options influence the order of the counters
- Anomaly? Try this:
  - dstat -ccccccc
  - dstat -c –cpu -M cpu -c –cpu -M cpu

Total or individual counters ?

- Some of the plugins show total values
- You can override the behaviour
  - -f or –full to see all individual counters
  - -C, -D, -I, -N, -S (capital options) to select individual counters
- Use 'total' to see the total together with individual counters, eg:
  - dstat -c -C total,0,1
  - dstat -d -D total,sda,sdb

Influencing output

- Disabling colors: --nocolor
- Disabling header repetition: --noheader
- Disabling intermediate updates: --noupdate

- or simply use Unix as it was designed :-)
  - dstat -af | cat

- White background: --bw or --blackonwhite
- Appending detailed output to CSV: --output

Dstat use-cases - 1

- Simple system check
  - dstat -taf
- What is the system doing now ?
  - dstat -c -M topcpu -dng -M topmem
  - dstat -dr -M topio -M topbio
- Is my SWRAID performing as it claims ?
  - dstat -td -D md0,md1,sda,sdb,hda

Dstat use-cases - 2

- What process is using all my CPU, memory or I/O at 4:20 AM ?
  - screen dstat -tcy -M topcpu 120
  - screen dstat -tmgs -M topmem 120
  - screen dstat -tdi -M topbio 120
- What device is slowing down my system ?
  - dstat -t -y -i -f
  - dstat -t -y -i -I 12,58,iwlagn -f 5

Dstat use-cases - 3

- How much ticks per second on my kernel ?
  - dstat -t -i -I0 -M snooze –debug
- Is my system clock being updated as it should ?
  - dstat -t -M ntp
- What process is going to be killed when I run out of memory ?
  - dstat -t -M topoom

Dstat use-cases - 4

- How is my laptop/battery/wifi doing ?
  - dstat -t -M cpufreq,power,thermal,battery,wifi

Using Dstat as a module

- Dstat itself can be used as a python module
- Accessing counters (raw values and differences)
- Examples in sources:
  - read.py: get raw values from plugins
  - mstat.py (milli-stat): shows sub-second values, useless but ubergeeky

Known issues

- Counter rollovers (be aware !)
  - eg. 32bit counters using bonded 10Gbit NICs...
- Performance issues ?
  - Dstat is **NOT** optimized for performance !
  - It's ironic, for a performance monitoring tool
  - Debugging dstat performance with --debug
- Writing plugins in C
  - Possible, but needs expertise

Future development - 1

- Implement caching between plugins
  - /proc/pid files don't need to be re-opened
- Improvements to color and meaning ?
- Exporting to syslog ?
- Configuration file ?

Future development - 2

- Add more plugins
  - More I/O related counters (iostat)
  - Xen plugins
  - Samba plugin (lacks interface ?)
  - Xorg resources, maybe topx (see xrestop)
  - Slab counters (need expert to group counters)
  - Systemtap/kernel perf/ftrace template plugin
  - SNMP template plugin
- Provide integration scripts for other apps
  - Plotting, alerting, ...

Dstat pointers

- Website and download
  - http://dag.wieers.com/home-made/dstat/
- Documentation
  - Included manual page: dstat(1)
  - Included Dstat paper
- Subversion/sourcecode
  - http://svn.rpmforge.net/svn/trunk/tools/dstat/
- Mailinglist
  - tools@lists.rpmforge.net

Writing Dstat plugins - 1

- Plugin instantiates dstat() python class
- Infrastructure is provided by the class
- Extra functions exist to simplify the actual plugins, eg:
  - dopen: keeps filedescriptors open and seek(0)
  - dpopen: keeps a pipe open to an application to write to and read from
  - readpipe/greppipe/matchpipe: parsing information

Writing Dstat plugins - 2

- Introducing the helloworld plugin
  - see the dstat paper
  - or simply look at dstat_helloworld.py
- Parsing counters
  - see the dstat paper
  - or simply look at eg. dstat_postfix.py

What is next ?

- Create an abstract object model and namespace for counters ?
- Ripping the counters/plugins out of Dstat into a framework
  - Getting rid of the Dstat specific fluff
- Lots of possibilities:
  - Framework could allow to write C, perl or python plugins
  - Reusing plugins from rrdtool, nagios, mrtg, munin