Dstat

plugin-based real-time monitoring

Dagit Linux Solutions

dag@wieers.com

Objective of this presentation

- Give a little bit of background information
- Explain and demonstrate this really simple tool
- Future development
- Receive feedback from developers and system administrators to advance Dstat (that means you !)

Who am I ?

- Started with Linux in 1994
- Worked 6 years at IBM Belgium Linux team
- Now freelance Linux and Open Source consultant
- Member of the CentOS development team
- Founded RPMforge repository in 2003
- Developer of a few sysadmin tools like mrepo, dconf, unoconv, proxytunnel and, of course, dstat

A case for Dstat

- Customer project in 2004: 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 frontends that shared GPFS
- GPFS allows to stripe (in parallel) to all available disks to optimize bandwidth usage of local HBA, multipath, SAN controllers and disk expansion units

A case for Dstat

- How can I monitor multiple nodes simultaneously ?
- How can I select only those system counters and application counters to validate performance numbers ?
- How can I make it easier to correlate counters and see usage patterns ?
- How can I follow progress during performance test and validate a test during and after it has finished ?

A case for Dstat

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

A case for Dstat

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

A case for Dstat

- Design goals (problems with eg. vmstat)
  - Needs to be extendable
  - 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 features

- History of counters (use terminal buffer)
- Adding unit indication (B = bytes, k = kilobytes)
- Fixed width columns
- Colour highlighting
- Intermediate updates (feel how things progress)
- Adding your own counters and selecting plugins
- Exporting to CSV
- Works with python 1.5.2 and later (CentOS 2)

Dstat features

- Use terminal capabilities
- Comes with plenty of plugins already:
  - time, cpu, disk, net, mem, interrupts, system, load, swap, paging, tcp, udp, raw, unix, locks, ipc, process, ...
  - dbus, gpfs, freespace, innodb, mysql, mysql5, nfs, postfix, rpc, sendmail, utmp
  - vmware, openvz
  - battery, cpufreq, thermal, wifi
  - topcpu, topio/topbio, topmem, topoom

Using Dstat: selecting plugins

- Internal vs. external plugins
- Internal plugins: short options and long options
- External plugins: use -M option
- 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 -cccc
- or:
  - dstat -c -M cpu -c -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 (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 colours: --nocolor
- Disabling header repetition: --noheader
- Disabling intermediate updates: --noupdate

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

- Appending detailed output to CSV: --output

Dstat use-cases

- Simple system check
  - dstat -taf
- What is the system doing now ?
  - dstat -c -M topcpu -dng -M topmem
- 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

Dstat use-cases (2)

- What device is slowing down my system ?
  - dstat -tyif
  - dstat -tyi -I 12,58,185 -f 5
- Is my SWRAID performing as it claims ?
  - dstat -td -D md0,md1,sda,sdb,hda
- How much ticks per second on my kernel ?
  - Dstat -t

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 !)
- 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
- Python 1.5 has limitations

Future development

- Improvements to colour and meaning
- Exporting to syslog
- Add more plugins
  - Xen plugins
  - Systemtap template plugin
  - SNMP template plugin
  - Samba plugin (lacks interface ?)
  - Xorg resources, maybe topx (see xrestop)
  - Slab counters (need expert to group counters)

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

Dstat pointers

- Website and download
  - http://dag.wieers.com/home-made/dstat/
- Subversion/sourcecode
  - http://svn.rpmforge.net/svn/trunk/tools/dstat/
- Mailinglist
  - tools@lists.rpmforge.net

Writing Dstat plugins

- 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