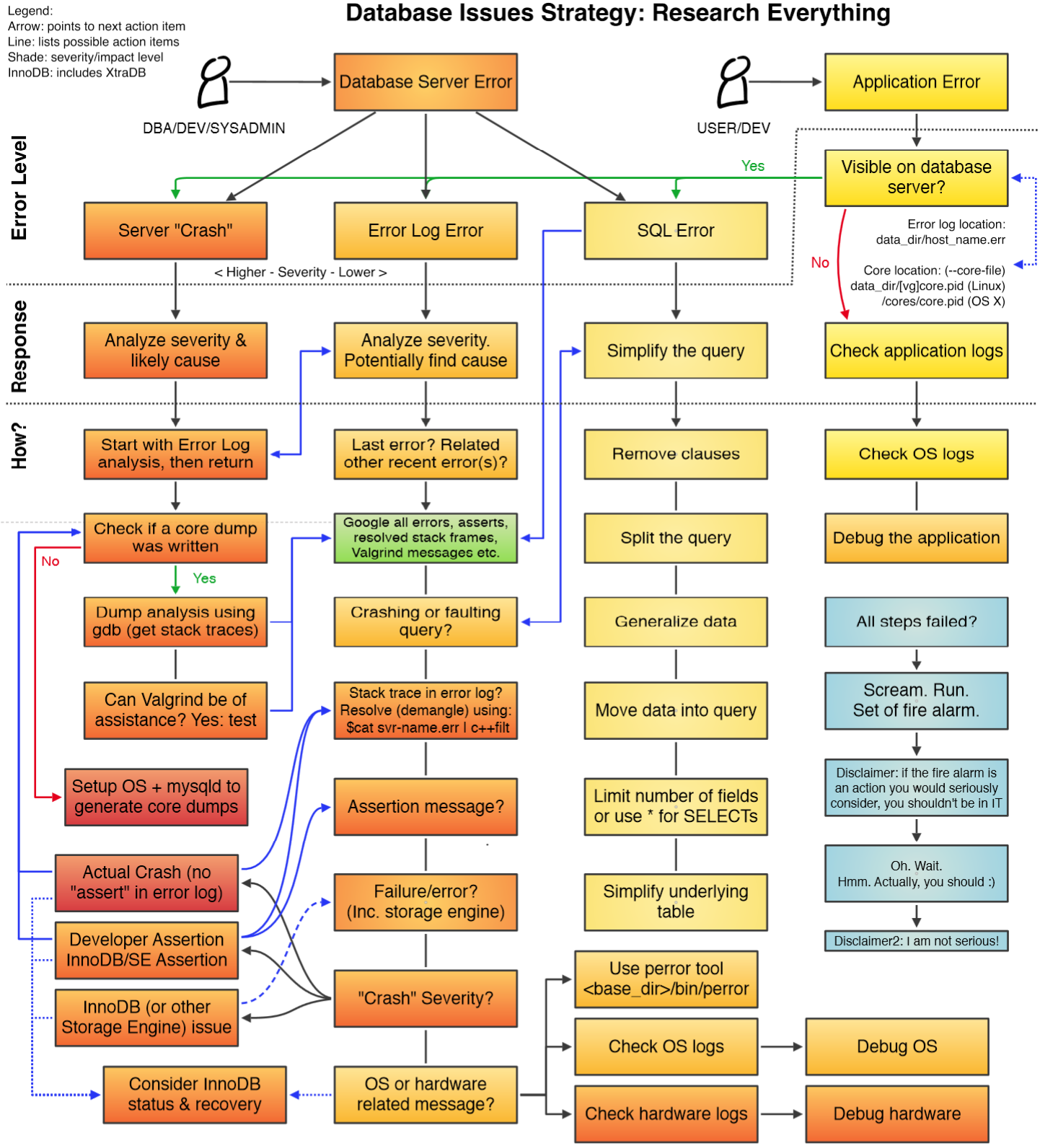
Database Issues Strategy: Research Everything



Google Strategy: Search Everything

keyword site:bugs.mysql.com keyword percona site:https://bugs.launchpad.net/bugs/ – First 3 function names of first 3 frames from coredump or Valgrind back trace – Assertion message or Valgrind message with 1-3 function names

It's great to work @ PERCONA Performance Consulting Experts

Bug Strategy: Log Everything*

In-bug-report summary: [testcase], Stack trace of failing thread, critical error log part (assert, error, ...), SEO title (function, assert, ...) etc. – (*) It is a general misconception to think that testcases are always needed, but this is often not the case, if full stack traces are logged. – Attachments to bug report: Full error log, 'thread apply all bt [full]' or full Valgrind stack output. Optional: 'How to repeat', 'Failing query' – Do your research: link to relevant bug reports, forums, hardware info's etc. Also make sure to keep track of & update your bug report

Social Strategy: Connect With Everyone

Knowing developers, engineers, managers, and other overly zealous people definitely helps in getting your bug fixed fast! – **Let's Connect!**

http://linkedin.com/in/roelvandepaar

http://twitter.com/RoelVandePaar

QA Cheat Sheet

Syntax: **Bold**=command | *Italics* = (minimum) options | {...} = Variable | [...] = Optional or choice

Server Setup: **sudo [yum/apt-get] [update/upgrade]**; **sudo [yum/apt-get] install** *wget patch make cmake automake autoconf libtool bzr gcc gcc-c++ ncurses-devel libaio libaio-devel bison valgrind perl-DBD-mysql cpan bzip2 valgrind-devel svn strace screen hdparm pam-devel sysbench;*

Gdb Cheat Sheet

See lp:percona-qa/analyze_crash.sh for an example (bzr branch lp:percona-qa)

Get the latest gdb (Currently v7.5.1., check http://ftp.gnu.org/gnu/gdb/ for latest): **sudo [yum/apt-get] remove** *gdb*; **sudo [yum/apt-get] install** *glibc** ; **wget** *ftp://sourceware.org/pub/gdb/releases/gdb-7.5.1.tar.gz*; **tar -xf** *gdb-7.5.1.tar.gz*; **cd** *gdb-7.5.1*; **./configure**; **make**; **sudo make install; gdb** *--version* #7.5.1

Usage: **gdb** *{MYSQLD_BINARY} {CORE}* (example: **gdb** */Percona-Server/bin/mysqld ./core.1113*).

Command	Comment
set auto-load safe-path /	# Avoids library loading issues. See $ info "(gdb)Auto-loading safe path"
set libthread-db-search-path /usr/lib/	# Avoids "thread debugging will not be available" warning
set trace-commands on	# Log commands to output file
set pagination off	# Don't use pagination
set print pretty on	# Print things prettier
set print array on	# Print variable arrays
set print array-indexes on	# Print array indexes
set print elements 4096	# Print variables etc. fully
set logging file gdb_standard.txt	# Set log filename
set logging on	# Turn logging on
thread apply all bt	# Backtrace all threads
set logging off	# Turn logging off
set logging file gdb_full.txt	# Want this for bug reports
set logging on	
thread apply all bt full	# With full local variables

Consider using a startup script for gdb: any commands in **~/.gdbinit** are executed when gdb starts

What to log in bug reports?
gdb_standard.txt
gdb_full.txt
full error log (servername.err)

Live Debugging
gdb *--pid {PID}* or
gdb *--args {MYSQLD_BIN} {ARGS_TO_MYSQLD}*
then **run**

Other handy commands

Command	Description
bt	backtrace
bt full	backtrace with locals
thread *nr*	select thread #*nr*
frame *nr*	select frame #*nr*
list	list code for frame
p *variable*	print named *variable*
info locals	print local vars
break	set breakpoint
c (live use only)	continue execution
quit	exit gdb

To print the LES (last executed query string): select 'do_command' frame in crashing thread using **thread** & **frame**, then use: **p** *thd->query_string.string.str*
See: http://www.mysqlperformanceblog.com/2012/09/09/obtain-last-executed-statement-from-optimized-core-dump/

WinDbg Cheat Sheet

See http://www.windbg.info/doc/1-common-cmds.html for an extensive cheat sheet

1. Download and install Microsoft .NET Framework 4.5 from http://www.microsoft.com/en-au/download/details.aspx?id=30653
2. See http://msdn.microsoft.com/en-us/windows/hardware/gg463009.aspx ("Install Debugging Tools for Windows as a Standalone Component" section)
3. This currently leads to http://msdn.microsoft.com/en-US/windows/hardware/hh852363 - click download. Start sdksetup.exe. Next > Next > Next > Accept
4. Untick: 'Windows Software Dev. Kit', 'Windows Perf. Toolkit', 'Windows App Cert. Toolkit', 'Net Framework 4.5 Software Dev. Kit'. Then click Install.
5. Start > type WinDbg > select correct version (x86=32 bit or x64=64 bit). File > Open crash dump. Tip: "Stop debugging" in "Debug" window to open new files.

Paths setup (create c:\symbolscache first)
.sympath+ *C:\Windows\System32; C:\{PATH_TO_MYSQLD_SYMBOLS_AND_BIN}*
.sympath+ *SRV*C:\symbolscache*http://msdl.microsoft.com/download/symbols*
.reload /f (Reloads all symbols immediately, except downloads which are live)

Then use the following commands:

Command	Description	Command	Description
kM	Simple backtrace	**.frame** *nr/ref*	Select a frame
!analyze *-v*	Exception analysis	**dv**	Display local vars
.help	Display help	**dt** *var [-r]*	Display a variable

Core Files Cheat Sheet

Use **sudo sysctl** *-p* to check various settings

Setup: mysqld side: ---core-file option to mysqld (in my.cnf as 'core-file' or on command line as '--core-file')
Setup OS side: (revert these changes on production systems after debugging work is done) (execute all these as root - i.e. execute **sudo su** first):
- **echo** *"core.%p.%e.%s" > /proc/sys/kernel/core_pattern* (%p=pid | %e=executable filename | %s=signal number)
- **vi** */etc/security/limits.conf* Add text: * hard core unlimited (same as making '**ulimit** *-c unlimited*' permanent)
- **vi** */etc/sysctl.conf* Add text: fs.suid_dumpable=2 (same as making '**sudo sysctl** *-w fs.suid_dumpable=2*' permanent)
  - Note: some systems use 'kernel' instead of 'fs': use kernel.suid_dumpable=2 instead if you get key or file warnings/errors. Test with ^ this command first.

Valgrind Cheat Sheet

Valgrind Documentation: http://valgrind.org/docs/manual/manual.html & http://valgrind.org/docs/manual/manual-core-adv.html

Installing the latest Valgrind (v3.8.1. at the time of writing, check http://valgrind.org/downloads/ for latest release): **sudo [yum/apt-get] remove** *valgrind*; **sudo [yum/apt-get] install** *bzip2 glibc** ; **wget** *http://valgrind.org/downloads/valgrind-3.8.1.tar.bz2*; **tar -xf** *valgrind-3.8.1.tar.bz2*; **cd** *valgrind-3.8.1*; **./configure**; **make**; **sudo make install**; **valgrind** *--version* # This should now read 3.8.1

QA'ers tip: also consider using ASan & TSan

Usage: **valgrind** *--leak-check=yes {MYSQLD_BINARY} {ARGUMENTS_TO_MYSQLD}*
Interesting options for [in-depth] debugging (ref man page for more info): *--num-callers=40 --track-origins=yes --suppressions=/server/mysql-test/valgrind.supp*

MTR+vgdb example [can also be easily adapted to work without MTR] (See http://valgrind.org/docs/manual/manual-core-adv.html#manual-core-adv.gdbserver):
$ cd /Percona-server/mysql-test/ # Percona-server or any other MySQL based server you're debugging
$ ./lib/v1/mysql-test-run.pl --start-and-exit --valgrind --valgrind-option="--leak-check=yes" --valgrind-option="--vgdb=yes" --valgrind-option="--vgdb-error=0" 1st
Now wait for a while (you may start to see "mysql-test-run: WARNING: Waited 60 seconds for /Percona-server/mysql-test/var/run/master.pid to be created, ...")
$ gdb /Percona-server/bin/mysqld # In another shell window
(gdb) target remote | vgdb # Type "c" + enter several times to continue, until the gdb prompt does not come back (some initial issues during mysqld startup).
Now start the mysql client and start running whatever you want to have Valgrind check. When it runs into a Valgrind error, it will break into the debugger with a message similar to: 'Program received signal SIGTRAP, Trace/breakpoint trap.'. Then you can start debugging as normal:
(gdb) thread apply all bt # Note also that the Valgrind error was logged to the error log at break time

RQG Cheat Sheet

Get RQG: bzr branch lp:randgen | RQG Docs: https://github.com/RQG/RQG-Documentation/wiki/Category:RandomQueryGenerator

Basic RQG Setup: Use "Server Setup" instructions above, then: **sudo perl -MCPAN** *-e 'install {MODULE_NAME}'* where {MODULE_NAME} is (one by one): *Bundle::CPAN, Test::More, Digest::MD5, XML::Writer, DBIx::MyParsePP, Statistics::Descriptive, JSON, [Test::Unit']*
See https://github.com/RQG/RQG-Documentation/wiki/RandomQueryGeneratorQuickStart for more information

Grammar Files: .yy = DML (can contain DDL also) | .zz = DDL | .cc = Combinations.pl setup (See lp:randgen/conf/percona_qa/percona_qa.* for examples)

Bash Cheat Sheet

Linux example. For Windows, use gnuwin32 (http://sourceforge.net/projects/getgnuwin32/files/), but further tweaks are required

```
$ echo -e "a\nb\nb\nc" | grep -iB1 -m1 "B" | tr '\n' '=' | awk -F"=" '{print "\n"$2}' | tr -d '\n' | sed 's/^/</;s/$/>/' | xargs -I_ echo _ | sed 's|.*\([a-z]\).*|\1|' | wc -l
        a⏎b⏎b⏎c          a⏎b⏎        a=b=        ⏎b          b        <b>        <b>        b        1
```

```
$ X='a'; Y='$XX'; Z=$(echo `echo ${Y}`); echo "$Z" > t; sed 's/x/a1/gi' t | sed "s/^.//" | sed "s/a/\t/g;s/[0-9]/3/" | awk '{print $2}' | xargs -I_ ls -_ ; rm t 2>t 1>&2
  X=a   Y=$XX   Z=$XX        file t=$XX    $a1a1      a1a1      [tab] 3 [tab] 1      1      output of ls -1      ☺
```