fixing bugs in illumos

Ryan Zezeski // illumos day // Sep 2015
"zfs holds" is $O(n^2)$

Added by Matthew Ahrens over 3 years ago.

Status: New  Start date: 2011-12-08
Priority: Normal  Due date: 
Assignee: -  % Done: 0%
Category: -
Target version: -
Difficulty: Bite-size  Tags: needs-triage

Description

"zfs holds snap ..." is $O($number snapshots listed $)^2)$. For every snapshot, it gets all the holds on all of the named snapshots. This can be observed with truss.

The problem is that when the python code was removed, it was replaced with incorrect C code, which calls zfs_for_each() for every argument. But zfs_for_each() iterates over the arguments itself
my favorite part of illumos is the tooling
Debugging Tools

- truss — syscall and userspace tracer
- ptools: proc(1) — various process-focused introspection
- mdb(1) — modular debugger, core dump analysis, live kernel introspection with -k
- dtrace(1) — dynamic tracing of the entire system, the best debugging tool ever made
How My Printer Caused Excessive Syscalls & UDP Traffic

DTracing in Anger

http://dtrace.org/blogs/brendan/2012/11/14/dtracing-in-anger/
Shouting in the Datacenter

https://www.youtube.com/watch?v=tDacjrSCeq4
The Setup
Pre-Setup

• A lot of ex-Sun people.

• Their traditions and nomenclature carried over.

• Confusing, and perhaps frustrating, at first. Be patient.

• Decades of history. Give their process a chance before criticizing it. You might end up liking it!

• http://zinascii.com/2014/my-first-illumos-build.html
Install OmniOS


- Recently gained ability to build gate. I use 151014.

- I run OmniOS on SmartOS KVM, but recommend a spare machine or VMWare Fusion to start.

- Tell Fusion to use Solaris 10 64-bit and don’t forget to add a floppy (to appease the installer).

“Zero Out” Your Environment

• Before trying to fix anything make sure you can build gate without any modifications.

• Many versions of instructions on how to build gate, lots of opinions. Often results in pain for newcomers.

• Use my nightly-setup script to save pain.
First Build

# wget http://zinascii.com/pub/rpz-misc/latest/rpz-misc.tar.gz

# tar -zxvf rpz-misc.tar.gz

# ./rpz-misc/bin/nightly-setup

# cd /code/illumos-gate

# sed -i.bk 's/export NIGHTLY_OPTIONS.*/export NIGHTLY_OPTIONS="-FnCDAprt"/' illumos.sh

# /opt/onbld/bin/nightly illumos.sh || echo "BUILD FAILED -- CHECK LOGS"
Build Failed?

- timestamped dir created under log/
- mail_msg contains summary of build
- nightly.log contains all output, where you need to look for errors
- search for three asterisks in nightly.log: ***
- focus on first error, often the cause of later errors
- ask for help in #illumos
RTI
Request to Integrate
RTI Steps


2. When satisfied with your patch then perform a full build (with lint) and perform an ONU.

3. Run git pbchk to verify style.

4. Submit webrev to developer@lists.illumos.org. Incorporate any changes and return to step 1.

5. Upon consensus submit for RTI at advocates@lists.illumos.org.
1. Code & Test
Incremental Builds

- A full nightly build, especially with lint, takes a long time.

- Use incremental to compile and lint.

- Must be in a build environment, see bldenv(1ONBLD).

- # cd usr/src/cmd/zfs && dmake lint && dmake install
Proto Area

• Proto area is where build artifacts are placed (i.e. when running dmake install).

• Certain changes can be tested from the proto area alone. E.g. command fixes and manual pages.

• Changes to libraries or kernel typically require an ONU to test.

• 

  # .proto/root_i386/sbin/zfs holds -r rpool/foo@1
bldenv(1ONBLD)

- Puts you in interactive shell with proper build environment (based off your illumos.sh).

- Needed to perform incremental builds and build cscope index.

- Requires a bootstrap nightly run.

- # /opt/onbld/bin/bldenv -d illumos.sh
cscope(1)

- Allows powerful search of entire usr/src.
- Must be in build environment to build index.
- # cd /code/illumos-gate/usr/src
- # dmake cscope.out
- # cscope-fast -dq
2. Full Build
nightly(1ONBLD)

- "The advantage to using nightly is that you build things correctly, consistently and automatically, with the best practices" — nightly(1ONBLD)

- Builds "ON" (OS-Network) aka the entire illumos kernel and system libraries.

- Need to do this so a) you have a mail_msg and b) you can test a full ONU (OS-Net Update).
Env File — illumos.sh

- Used to configure both nightly(1ONBLD) and bldenv(1ONBLD).

- User customizations to the build environment.

- NIGHTLY_OPTIONS — controls steps taken during nightly, e.g. lint.
beadm(1M)

• Boot Environment (BE) is basically the OS you boot into.

• BEs allow you to have multiple instances of your operating system that you can jump between via system reboot.

• Mostly used to perform safe atomic upgrades of the kernel, system, and user applications: i.e. the entire OS.

• It's also the tool used, with help from onu(1ONBLD), to create BE for your build.
ONU(1ONBLD)

- "ON Update"—create a new BE with the latest build so you can boot into it and test it.
- Uses pkg(5) and beadm(1ONBLD) underneath.
- `# /opt/onbld/bin/onu -t nightly -d packages/i386/nightly`
- `# reboot`
3. pbchk
git-pbchk(1ONBLD)

- A "putback" check. Final checks before RTI that help keep consistency across the entire codebase.

- Verify commit format.

- Verify illumos C Style (google "C Style and Coding Standards for SunOS")

- Verify header format.

- Verify copyright (optional).
4. Submit Webrev
webrev(1ONBLD)

• Generates standalone webpage showing all changes made by your patch.

• Generates diffs in many different formats for reviewer preference.

• Manual pages have both raw diffs and rendered diffs. Thank you Yuri Pankov!

• http://zinascii.com/pub/illumos/gate/6205/
5. Submit RTI
RTI Email

- List of reviewers: name + email.
- Summary of testing you performed.
- Copy of pbchk output.
- Patch file.
- Compressed mail_msg.
- Additional information you wish to add.
Bug #1869: "zfs holds" is \(O(n^2)\)

Added by Matthew Ahrens over 3 years ago.

<table>
<thead>
<tr>
<th>Status:</th>
<th>New</th>
<th>Start date:</th>
<th>2011-12-08</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assignee:</td>
<td>-</td>
<td>% Done:</td>
<td>0%</td>
</tr>
<tr>
<td>Category:</td>
<td>-</td>
<td>Tags:</td>
<td>needs-triage</td>
</tr>
<tr>
<td>Target version:</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Difficulty:</td>
<td>Bite-size</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description**

"zfs holds snap ..." is \(O(\text{number snapshots listed }^2)\). For every snapshot, it gets all the holds on all of the named snapshots. This can be observed with truss.

The problem is that when the python code was removed, it was replaced with incorrect C code, which calls zfs_for_each() for every argument. But zfs_for_each() iterates over the arguments itself.
Must Reads

- http://illumos.org/books/dev/workflow.html
- http://wiki.illumos.org/display/illumos/How+To+Build+illumos
- https://www.illumos.org/issues?query_id=15
Yes, still relevant.
Anyone can do this. It takes time and patience, not a background in OS dev or even a CS degree. Don't let anyone tell you otherwise.
Questions?
ryan@zinascii.com