Arvados - Story #3826

[Crunch] Display network activity in crunchstat

09/05/2014 02:48 PM - Peter Amstutz

Status:	Resolved	**Start date:**	10/10/2014
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:	Crunch	**Estimated time:**	0.00 hour
Target version:	2014-10-29 sprint		

Description

(TC) sysfs provides traffic statistics. I tried this in a docker container, after doing a few pings:

```
# head /sys/devices/virtual/net/*/statistics/*_bytes
==> /sys/devices/virtual/net/eth0/statistics/rx_bytes <==
1116

==> /sys/devices/virtual/net/eth0/statistics/tx_bytes <==
1170

==> /sys/devices/virtual/net/lo/statistics/rx_bytes <==
168

==> /sys/devices/virtual/net/lo/statistics/tx_bytes <==
168
```

In order to read these stats from outside the container, use the "setns" syscall to set your network namespace to match the netns of any process inside the container. Close the filehandle after each use, though: otherwise it will keep the container alive after all processes exit.

- Update: "setns" turns out to be complicated: e.g., threaded programs can't use it. Instead, we settle for parsing /proc/PID/net/dev after choosing an appropriate PID.

Subtasks:

Task # 4198: Review 3826-crunchstat-netstats	**Resolved**
Task # 4181: Collect container-level network stats	**Resolved**
Task # 4225: Review 3826-crunchstat-netstats -- esp for language issues	**Resolved**
Task # 4152: Talk to Ward	**Resolved**

Related issues:

Related to Arvados - Bug #3843: [Crunchstat] Report CPU accounting numbers ac...	**Resolved**	
Related to Arvados - Bug #4185: [Crunch] crunchstat memory reports seem suspe...	**Resolved**	**10/14/2014**
Blocked by Arvados - Bug #3824: [Crunch] crunch-job should create task execut...	**Resolved**	**10/10/2014**

Associated revisions

Revision 89f38df7 - 10/22/2014 03:01 PM - Tom Clegg

Merge branch '3826-crunchstat-netstats' closes #3826

Revision a522b584 - 10/22/2014 03:03 PM - Tom Clegg

Add services/crunchstat to test suite. refs #3826

Revision a522b584 - 10/22/2014 03:03 PM - Tom Clegg

Add services/crunchstat to test suite. refs #3826

History

#1 - 09/05/2014 02:50 PM - Peter Amstutz

- Subject changed from Crunchstat log network usage for jobs. to Crunchstat monitor network usage for jobs.

- Description updated

#2 - 09/08/2014 09:20 AM - Ward Vandewege

- Subject changed from Crunchstat monitor network usage for jobs. to [Crunch] Crunchstat monitor network usage for jobs.

#3 - 09/15/2014 09:12 PM - Tom Clegg

- Description updated

- Category set to Crunch

#4 - 09/16/2014 05:13 PM - Tom Clegg

- Story points set to 1.0

#5 - 09/16/2014 05:13 PM - Tom Clegg

- Subject changed from [Crunch] Crunchstat monitor network usage for jobs. to [Crunch] crunchstat should output network activity statistics

#6 - 09/17/2014 03:00 PM - Tom Clegg

- Story points changed from 1.0 to 0.5

#7 - 09/17/2014 03:22 PM - Tom Clegg

- Target version changed from Arvados Future Sprints to 2014-10-08 sprint

#8 - 09/17/2014 03:35 PM - Peter Amstutz

- Assigned To set to Peter Amstutz

#9 - 09/17/2014 03:59 PM - Peter Amstutz

- Story points changed from 0.5 to 1.0

#10 - 09/18/2014 10:12 AM - Tom Clegg

- Description updated

#11 - 09/18/2014 02:35 PM - Tom Clegg

- Description updated

#12 - 09/18/2014 02:52 PM - Peter Amstutz

it looks like between /proc/stat, /proc/meminfo, /proc/diskstats and /proc/net/netstat that probably gives us everything we want

#13 - 09/26/2014 07:36 PM - Tom Clegg

- Target version changed from 2014-10-08 sprint to Arvados Future Sprints

#14 - 09/26/2014 07:38 PM - Tom Clegg

- Description updated

#15 - 09/26/2014 07:38 PM - Tom Clegg

- Assigned To deleted (Peter Amstutz)

#16 - 10/08/2014 02:02 PM - Tom Clegg

- Subject changed from [Crunch] crunchstat should output network activity statistics to [Crunch] Create new tool "logsysstats" similar to crunchstat but using /proc/stat et al, not cgroup. Show network, CPU, memory, and IO stats on stderr.

- Story points changed from 1.0 to 2.0

#17 - 10/08/2014 05:50 PM - Ward Vandewege

- Target version changed from Arvados Future Sprints to 2014-10-29 sprint

#18 - 10/08/2014 07:44 PM - Tom Clegg

- Assigned To set to Tom Clegg

#19 - 10/10/2014 01:51 PM - Tom Clegg

- Subject changed from [Crunch] Create new tool "logsysstats" similar to crunchstat but using /proc/stat et al, not cgroup. Show network, CPU, memory, and IO stats on stderr. to [Crunch] Display network activity in crunchstat

- Description updated

#20 - 10/10/2014 05:24 PM - Tom Clegg

I'm finding that FindStat comes with a race condition. Sometimes it finds the container stats file during startup, and prints container statistics. Other times, it doesn't find container stats so it falls back on system-wide stats, and prints those instead.

Try 1:

```
crunchstat: Running [docker run --rm=true --cidfile=/tmp/cid -t --sig-proxy ubuntu sleep 1]
crunchstat: reading stats from /sys/fs/cgroup/cpuacct/docker/aa1e7ca054cce6804c522b8cf931c9dd4337d4d469d7f5429
298fbbafdc0c076/cpuacct.stat
crunchstat: reading stats from /sys/fs/cgroup/blkio/docker/aa1e7ca054cce6804c522b8cf931c9dd4337d4d469d7f542929
8fbbafdc0c076/blkio.io_service_bytes
crunchstat: reading stats from /sys/fs/cgroup/cpuset/cpuset.cpus
crunchstat: did not find stats file (root /sys/fs/cgroup, parent docker, cid aa1e7ca054cce6804c522b8cf931c9dd4
337d4d469d7f5429298fbbafdc0c076, statgroup memory, stat memory.stat)
crunchstat: reading stats from /sys/fs/cgroup/cpuacct/docker/aa1e7ca054cce6804c522b8cf931c9dd4337d4d469d7f5429
298fbbafdc0c076/cgroup.procs
```

Try 2:

```
crunchstat: Running [docker run --rm=true --cidfile=/tmp/cid -t --sig-proxy ubuntu sleep 5]
crunchstat: reading stats from /sys/fs/cgroup/cpuacct/cpuacct.stat
crunchstat: reading stats from /sys/fs/cgroup/blkio/blkio.io_service_bytes
crunchstat: reading stats from /sys/fs/cgroup/cpuset/cpuset.cpus
crunchstat: did not find stats file (root /sys/fs/cgroup, parent docker, cid 772a04981029c7ff4ddabb9edbf32de7d
bd8689f924499339f78c12832b84387, statgroup memory, stat memory.stat)
crunchstat: did not find stats file (root /sys/fs/cgroup, parent docker, cid 772a04981029c7ff4ddabb9edbf32de7d
bd8689f924499339f78c12832b84387, statgroup cpuacct, stat cgroup.procs)
```

#21 - 10/10/2014 09:18 PM - Tom Clegg

3826-crunchstat-netstats at 4be40a4

running ping

```
crunchstat: cpuacct.stat user 0.0000 sys 0.0000 cpus 4 interval 0.1989
crunchstat: net eth0 tx 1086 +0 rx 1070 +0 interval 0.1995
64 bytes from red.tomclegg.net (64.94.249.196): icmp_seq=4 ttl=48 time=52.7 ms
crunchstat: cpuacct.stat user 0.0000 sys 0.0000 cpus 4 interval 0.1995
crunchstat: net eth0 tx 1184 +98 rx 1168 +98 interval 0.2002
crunchstat: cpuacct.stat user 0.0000 sys 0.0000 cpus 4 interval 0.1995
crunchstat: net eth0 tx 1184 +0 rx 1168 +0 interval 0.2000
```

Running sha1sum </dev/zero

```
crunchstat: net eth0 tx 90 +0 rx 90 +0 interval 0.0002
crunchstat: cpuacct.stat user 0.8006 sys 0.1001 cpus 4 interval 0.1998
crunchstat: net eth0 tx 180 +90 rx 258 +168 interval 0.2005
crunchstat: cpuacct.stat user 0.9518 sys 0.0501 cpus 4 interval 0.1996
crunchstat: net eth0 tx 258 +78 rx 258 +0 interval 0.2032
crunchstat: cpuacct.stat user 0.9657 sys 0.0000 cpus 4 interval 0.1968
```

#22 - 10/14/2014 07:04 PM - Tom Clegg

Another example, showing another race condition (note interface name changing from veth4bfb to eth0)

```
crunchstat: Running [docker run --rm=true --volume=/usr/bin/pigz:/usr/bin/pigz --cidfile=/tmp/cid -t --sig-pro
xy ubuntu bash -c head -c1000000000 /dev/zero | pigz - >/dev/null]
crunchstat: reading stats from /sys/fs/cgroup/cpuacct/docker/b95747a416d261e349ebbe9cb909bc9ad5c48f25d15c696e8
ca402e661dfe349/cpuacct.stat
crunchstat: reading stats from /sys/fs/cgroup/blkio/docker/b95747a416d261e349ebbe9cb909bc9ad5c48f25d15c696e8ca
402e661dfe349/blkio.io_service_bytes
crunchstat: reading stats from /sys/fs/cgroup/cpuset/cpuset.cpus
crunchstat: did not find stats file (root /sys/fs/cgroup, parent docker, cid b95747a416d261e349ebbe9cb909bc9ad
5c48f25d15c696e8ca402e661dfe349, statgroup memory, stat memory.stat)
crunchstat: net:veth4bfb 90 tx 90 rx
crunchstat: cpu 0.0000 user 0.0100 sys 4 cpus
crunchstat: net:veth4bfb 90 tx 90 rx -- interval 0.0010 seconds 0 tx 0 rx
crunchstat: cpu 1.9400 user 0.5300 sys 4 cpus -- interval 1.0003 seconds 1.9400 user 0.5200 sys
crunchstat: net:eth0 348 tx 258 rx
crunchstat: cpu 3.8600 user 1.2900 sys 4 cpus -- interval 0.9995 seconds 1.9200 user 0.7600 sys
crunchstat: net:eth0 598 tx 418 rx -- interval 1.0003 seconds 250 tx 160 rx
```

```
crunchstat: cpu 6.1200 user 1.8700 sys 4 cpus -- interval 0.9995 seconds 2.2600 user 0.5800 sys
crunchstat: net:eth0 598 tx 508 rx -- interval 1.0000 seconds 0 tx 90 rx
crunchstat: cpu 8.3200 user 2.4000 sys 4 cpus -- interval 0.9995 seconds 2.2000 user 0.5300 sys
crunchstat: net:eth0 598 tx 508 rx -- interval 1.0002 seconds 0 tx 0 rx
crunchstat: cpu 10.5200 user 3.0200 sys 4 cpus -- interval 0.9995 seconds 2.2000 user 0.6200 sys
crunchstat: net:eth0 598 tx 508 rx -- interval 1.0001 seconds 0 tx 0 rx
```

#23 - 10/14/2014 07:09 PM - Tom Clegg

Example output @ 8aeba61

```
tom@belle:~$ rm /tmp/cid 2>/dev/null; /tmp/tmp.*/bin/crunchstat -cgroup-root=/sys/fs/cgroup -cgroup-parent=doc
ker -cgroup-cid=/tmp/cid -poll=1000 docker run --rm=true --volume=/usr/bin/pigz:/usr/bin/pigz --cidfile=/tmp/c
id -t --sig-proxy ubuntu bash -c 'head -c1000000000 </dev/zero | pigz | pigz -d >/tmp/zero'
crunchstat: Running [docker run --rm=true --volume=/usr/bin/pigz:/usr/bin/pigz --cidfile=/tmp/cid -t --sig-pro
xy ubuntu bash -c head -c1000000000 </dev/zero | pigz | pigz -d >/tmp/zero]
crunchstat: reading stats from /sys/fs/cgroup/cpuset/cpuset.cpus
crunchstat: reading stats from /sys/fs/cgroup/cpuacct/docker/d8049825e6048f024bb8418c7bcd826478649167f3fa8cd52
b3e72a631d9aee0/cpuacct.stat
crunchstat: cpu 0.0000 user 0.0000 sys 4 cpus
crunchstat: reading stats from /sys/fs/cgroup/memory/docker/d8049825e6048f024bb8418c7bcd826478649167f3fa8cd52b
3e72a631d9aee0/memory.stat
crunchstat: mem 0 cache 0 pgmajfault 0 rss
crunchstat: reading stats from /sys/fs/cgroup/blkio/docker/d8049825e6048f024bb8418c7bcd826478649167f3fa8cd52b3
e72a631d9aee0/blkio.io_service_bytes
crunchstat: reading stats from /sys/fs/cgroup/cpuacct/docker/d8049825e6048f024bb8418c7bcd826478649167f3fa8cd52
b3e72a631d9aee0/cgroup.procs
crunchstat: cpu 1.4800 user 0.7700 sys 4 cpus -- interval 1.0010 seconds 1.4800 user 0.7700 sys
crunchstat: mem 23216128 cache 0 pgmajfault 3215360 rss
crunchstat: net:eth0 348 tx 258 rx
crunchstat: cpu 2.5600 user 1.4500 sys 4 cpus -- interval 1.0019 seconds 1.0800 user 0.6800 sys
crunchstat: mem 80625664 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 508 tx 418 rx -- interval 1.0035 seconds 160 tx 160 rx
crunchstat: cpu 3.5600 user 2.0100 sys 4 cpus -- interval 0.9989 seconds 1.0000 user 0.5600 sys
crunchstat: mem 149045248 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 598 tx 508 rx -- interval 1.0002 seconds 90 tx 90 rx
crunchstat: cpu 4.6400 user 2.7000 sys 4 cpus -- interval 0.9985 seconds 1.0800 user 0.6900 sys
crunchstat: mem 211337216 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 598 tx 508 rx -- interval 1.0033 seconds 0 tx 0 rx
crunchstat: cpu 5.8300 user 3.2500 sys 4 cpus -- interval 0.9946 seconds 1.1900 user 0.5500 sys
crunchstat: mem 275202048 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 598 tx 508 rx -- interval 1.0029 seconds 0 tx 0 rx
crunchstat: cpu 6.8300 user 3.9200 sys 4 cpus -- interval 0.9998 seconds 1.0000 user 0.6700 sys
crunchstat: mem 342704128 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 668 tx 578 rx -- interval 1.0013 seconds 70 tx 70 rx
crunchstat: cpu 7.5000 user 4.7100 sys 4 cpus -- interval 0.9996 seconds 0.6700 user 0.7900 sys
crunchstat: mem 390873088 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 668 tx 578 rx -- interval 1.0011 seconds 0 tx 0 rx
crunchstat: cpu 8.4900 user 5.3300 sys 4 cpus -- interval 0.9987 seconds 0.9900 user 0.6200 sys
crunchstat: mem 450969600 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 668 tx 578 rx -- interval 0.9995 seconds 0 tx 0 rx
crunchstat: cpu 9.1100 user 5.9700 sys 4 cpus -- interval 0.9990 seconds 0.6200 user 0.6400 sys
crunchstat: mem 484261888 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 668 tx 578 rx -- interval 1.0003 seconds 0 tx 0 rx
crunchstat: cpu 9.7500 user 6.6800 sys 4 cpus -- interval 0.9956 seconds 0.6400 user 0.7100 sys
crunchstat: mem 517062656 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 738 tx 648 rx -- interval 1.0003 seconds 70 tx 70 rx
crunchstat: cpu 10.7200 user 7.4300 sys 4 cpus -- interval 1.0022 seconds 0.9700 user 0.7500 sys
crunchstat: mem 577290240 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 738 tx 648 rx -- interval 1.0038 seconds 0 tx 0 rx
crunchstat: cpu 11.6800 user 8.1400 sys 4 cpus -- interval 0.9987 seconds 0.9600 user 0.7100 sys
crunchstat: mem 652132352 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 738 tx 648 rx -- interval 1.0002 seconds 0 tx 0 rx
crunchstat: cpu 12.7100 user 8.8200 sys 4 cpus -- interval 1.0006 seconds 1.0300 user 0.6800 sys
crunchstat: mem 721829888 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 738 tx 648 rx -- interval 1.0013 seconds 0 tx 0 rx
crunchstat: cpu 13.7600 user 9.4700 sys 4 cpus -- interval 0.9995 seconds 1.0500 user 0.6500 sys
crunchstat: mem 784154624 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 738 tx 648 rx -- interval 1.0003 seconds 0 tx 0 rx
crunchstat: cpu 14.7900 user 10.2000 sys 4 cpus -- interval 0.9993 seconds 1.0300 user 0.7300 sys
crunchstat: mem 846716928 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 738 tx 648 rx -- interval 1.0003 seconds 0 tx 0 rx
crunchstat: cpu 15.5100 user 10.7800 sys 4 cpus -- interval 0.9994 seconds 0.7200 user 0.5800 sys
crunchstat: mem 901103616 cache 0 pgmajfault 3219456 rss
crunchstat: net:eth0 738 tx 648 rx -- interval 1.0045 seconds 0 tx 0 rx
```

```
crunchstat: cpu 15.8900 user 11.2700 sys 4 cpus -- interval 0.9952 seconds 0.3800 user 0.4900 sys
crunchstat: mem 973750272 cache 0 pgmajfault 536576 rss
crunchstat: net:eth0 738 tx 648 rx -- interval 0.9964 seconds 0 tx 0 rx
```

#24 - 10/14/2014 07:13 PM - Tom Clegg

- Description updated

#25 - 10/14/2014 08:32 PM - Tom Clegg

Summary of changes in this branch (3826-crunchstat-netstats @ 5de6c06)

- Add network stats
- Reformat all stats output for easier parsing
- Report cumulative, delta, and time-delta numbers for cpu and io (leaving the divide-by-whatever logic for postprocessing/graphing)
- Avoid getting stuck with host-level reporting just because the first poll happens before the container is fully set up
- Improve error checking and reporting
- Clean up stats collection logic (get rid of the One Giant Loop)
- Report cache and major page fault stats along with rss (could shed light on #4185?)
- Add a tiny crunchstat_test.go so at least jenkins can do *something* with crunchstat.

Noticed but not solved:

- Location of cgroup files should come from /proc/PID/cgroup for a process in the container, rather than a hard-coded list of locations to probe. Currently, race conditions can cause the host-level stats to be reported during setup/teardown. The bandaid solution for this is to probe at each collection event, rather than probing during startup and sometimes getting stuck with host-level stats for the duration of the container.

#26 - 10/16/2014 01:18 PM - Brett Smith

Thanks for a lot of nice refactoring. My review of 5de6c06 produced one ERROR, one WARNING, and one idea for the future.

- The error: crunchstat has always had a classic time-of-check vs. time-of-open race condition, where something about the file might change between the time FindStat finds it, and the calling code opens the returned path. The new FindStat exacerbates this problem because it can potentially Stat the same path multiple times, making it hard to predict or reason about what might happen if the stats file changes while FindStat is executing.
  Ideally, we'd fix the race condition by having FindStat return an open file handle. I realize this probably requires reworking OpenAndReadAll and friends, though, so if that's too big of a change for this branch, I'd at least like to see FindStat stop its search as soon as it finds a good path.
- The warning: I was able to test crunchstat on my system pretty successfully (including the new network statistics, nice), but I couldn't get the CPU statistics to move at all. I realize I might just be missing a necessary kernel switch or something like that, so if you've had success, I'm happy. I just wanted to flag that as a possible issue.
- The idea: One way we might improve testing is to have all the file-parsing code in separate functions, and require the stats file to be read elsewhere. That way, we could test the parsers with arbitrary strings, make sure they handle deltas correctly, etc. That seems like it would be a significant win that doesn't require much restructuring. But it's definitely not immediately relevant to this branch; just throwing it out there.

#27 - 10/16/2014 02:00 PM - Tom Clegg

Brett Smith wrote:

> - The error: crunchstat has always had a classic time-of-check vs. time-of-open race condition, where something about the file might change between the time FindStat finds it, and the calling code opens the returned path. The new FindStat exacerbates this problem because it can potentially Stat the same path multiple times, making it hard to predict or reason about what might happen if the stats file changes while FindStat is executing.

Returning a file instead of a path had occurred to me too: I didn't like doing all those Stat() calls when all we really want to know is whether Open() works. I rearranged FindStat() in order to log which file ended up being used, and I noticed the redundant Stat() but didn't notice that I had created an even more confusing race (first two succeed, third fails, try the third path but not the second). I agree this is sufficiently unpredictable/confusing to count as an error.

I also noticed an interesting logging failure: When logging from file A ("no container proc yet"), then file B ("running"), then file A ("container is empty but docker hasn't exited yet"), there should be a log that we're switching back to file A. (As is, the logs will say "reading from A", then "reading from B", but nothing to suggest that we've changed back to file A.)

I'll go off and address these...

> Ideally, we'd fix the race condition by having FindStat return an open file handle. I realize this probably requires reworking OpenAndReadAll and friends, though, so if that's too big of a change for this branch, I'd at least like to see FindStat stop its search as soon as it finds a good path.

One reason I left FindStat in such a state is that (as mentioned in the comment above it) the whole approach is wrong. The documented way to do this is to look in /proc/mounts for a cgroup mount (rather than asking crunch-job to predict where it is on the worker node) and look in /proc/PID/cgroup (for a process inside the container) to get the correct cgroup path for each type of accounting. There's just one thing I don't quite see yet: How do you get a list of tasks in the container without guessing one of those cgroup paths or forking docker inspect --format '{{.State.Pid}}' c508b2fd6dd5 ...?

- The warning: I was able to test crunchstat on my system pretty successfully (including the new network statistics, nice), but I couldn't get the CPU statistics to move at all. I realize I might just be missing a necessary kernel switch or something like that, so if you've had success, I'm happy. I just wanted to flag that as a possible issue.

Hm, the only thing I had to work to enable was memory (GRUB_CMDLINE_LINUX="cgroup_enable=memory").

It looks like I neglected to mention this anywhere but I did find a ~~hack~~ way to test this without invoking docker (it relies on the FindStat fallback setup to treat an invalid container name as "measure host system instead of a container"):

```
rm /tmp/cid
/tmp/tmp.*/bin/crunchstat -cgroup-root=/sys/fs/cgroup -cgroup-parent=invalid -cgroup-cid=/tmp/cid -poll=1000 \
  sh -c 'echo -n invalid >/tmp/cid; head -c1000000000 /dev/zero | sha1sum'
```

And, um, yes -- CPU stats were totally broken. Fixed in 2b08ab2

- The idea: One way we might improve testing is to have all the file-parsing code in separate functions, and require the stats file to be read elsewhere. That way, we could test the parsers with arbitrary strings, make sure they handle deltas correctly, etc. That seems like it would be a significant win that doesn't require much restructuring. But it's definitely not immediately relevant to this branch; just throwing it out there.

Yes, that sounds like a good idea. A proper testing plan would surely have to include stats formats found on *supported* systems, not just the test system. It would also help explain what was expected, when unexpected data starts to show up.

(I've been wondering about the least ugly way to test real stats collection (like the above example) from crunchstat_test.go, but that sounds like a really good way to get more coverage before we go Full Integration Fest.)

#28 - 10/16/2014 03:08 PM - Tom Clegg

Now at a5ec416 with...

> Returning a file instead of a path had occurred to me too: I didn't like doing all those Stat() calls when all we really want to know is whether Open() works. I rearranged FindStat() in order to log which file ended up being used, and I noticed the redundant Stat() but didn't notice that I had created an even more confusing race (first two succeed, third fails, try the third path but not the second). I agree this is sufficiently unpredictable/confusing to count as an error.

Fixed.

> I also noticed an interesting logging failure: When logging from file A ("no container proc yet"), then file B ("running"), then file A ("container is empty but docker hasn't exited yet"), there should be a log that we're switching back to file A. (As is, the logs will say "reading from A", then "reading from B", but nothing to suggest that we've changed back to file A.)

Fixed.

> Ideally, we'd fix the race condition by having FindStat return an open file handle. I realize this probably requires reworking OpenAndReadAll and friends, though, so if that's too big of a change for this branch, I'd at least like to see FindStat stop its search as soon as it finds a good path.

Fixed. Search stops, returns *os.File instead of a path.

#29 - 10/16/2014 08:01 PM - Brett Smith

Tom Clegg wrote:

> Now at a5ec416

This version looks good to me. Thanks for revamping the stat file finder logic a second time. I just have one more small suggestion, but it's purely aesthetic, so feel free to ignore: would the "stat file not found" warning be easier to read if the fields were reversed? I think the cgroup-related fields are usually known, and the reader most needs to know which stat is missing. But that's highly subjective, so whatever.

#30 - 10/16/2014 08:41 PM - Tom Clegg

Brett Smith wrote:

> This version looks good to me. Thanks for revamping the stat file finder logic a second time. I just have one more small suggestion, but it's purely aesthetic, so feel free to ignore: would the "stat file not found" warning be easier to read if the fields were reversed? I think the cgroup-related fields are usually known, and the reader most needs to know which stat is missing. But that's highly subjective, so whatever.

Yes, I think you're right. Done in db04722. Example:

```
crunchstat: did not find stats file: stat cpuacct.stat, statgroup cpuacctblerp, cid invalid, parent invalid, r
oot /sys/fs/cgroup
```

#31 - 10/16/2014 10:15 PM - Tim Pierce

Reviewing @ a5ec4164:

Something we should consider is whether we can reuse any of the code in github.com/docker/libcontainer/cgroups to do some of this work. In particular, these look promising:

- fs.BlkioGroup.GetStats (https://github.com/docker/libcontainer/blob/master/cgroups/fs/blkio.go)
- fs.MemoryGroup.GetStats (https://github.com/docker/libcontainer/blob/master/cgroups/fs/memory.go)
- fs.CpuacctGroup.GetStats (https://github.com/docker/libcontainer/blob/master/cgroups/fs/cpuacct.go)

Using a channel to guarantee thread-safe writes to stderr is clever, but it would be nice to do it without passing a stderr channel everywhere in the program. Maybe something like this:

```
type LogWriter struct {
    logchan chan string
    logfile *os.File
}

func NewLogWriter(outfile *os.File) *LogWriter {
    logchan := make(chan string)
    lw := LogWriter{logchan, outfile}
    go func() {
        for msg := range logchan {
            fmt.Fprintln(outfile, msg)
        }
    }()
    return &lw
}

func (lw *LogWriter) Printf(format string, a ...interface{}) (n int, err error) {
    msg := fmt.Sprintf(format, a...)
    lw.logchan <- msg
}

var stderr *LogWriter

...
stderr := NewLogWriter(os.Stderr)
stderr.Printf("error: %s", foo)
```

Something like that would be valuable to use as the base logging framework for all of our concurrent Go code.

OpenAndReadAll

- Sorry if this is a stupid question, but could OpenAndReadAll plausibly be replaced with ioutil.ReadFile? I see some of the comments here about race conditions involved in opening and reading the file, but it's not immediately clear to me whether our current approach solves problems that ioutil.ReadFile doesn't.

PollCgroupStats

- It seems like it might make sense for PollCgroupStats to be responsible for waiting for the cid file to appear. It runs forever anyway, so it could just keep sleeping until either the child terminates or the cid file shows up. I haven't been through the nitty gritty here, though, so if there are situations where the container file basically never appears and crunchstat would hang forever, or if --wait=5 catches basically all of the cases we care about, then never mind.

OpenStatFile

- This should work for setup:

```
    var paths = []string{
        fmt.Sprintf("%s/%s/%s/%s/%s", cgroup.root, statgroup, cgroup.parent, cgroup.cid, stat),
        fmt.Sprintf("%s/%s/%s/%s", cgroup.root, cgroup.parent, cgroup.cid, stat),
        fmt.Sprintf("%s/%s/%s", cgroup.root, statgroup, stat),
        fmt.Sprintf("%s/%s", cgroup.root, stat),
    }
```

DoNetworkStats

- This is a cleaner way of writing the inner loop to parse the netstats lines:

```
    f := strings.Fields(scanner.Text())
    if len(f) != 17 {
        continue
    }
    ifName = strings.TrimRight(f[0], ":")
    if rx, err = strconv.Atoi(f[1]); err != nil {
        continue
    }
    if tx, err = strconv.Atoi(f[9]); err != nil {
        continue
    }
```

- Or even, if we're using fmt.Sscanf anyway:

```
    fmt.Sscanf(" %s %d %*d %*d %*d %*d %*d %*d %*d %d", &ifName, &rx, &tx)
```

- But I've never really trusted sscanf for more than trivial scans.
- This line was startling until I realized that it's just creating a new temporary scope for lastSample: if lastSample, ok := lastSample[ifName]; ok { ... How about "ls" or "prevsample" or some other name to be less confusing about scope?

#32 - 10/20/2014 05:06 PM - Tom Clegg

Tim Pierce wrote:

> Reviewing @ a5ec4164:
>
> Something we should consider is whether we can reuse any of the code in github.com/docker/libcontainer/cgroups to do some of this work. In particular, these look promising:
>
> - fs.BlkioGroup.GetStats (https://github.com/docker/libcontainer/blob/master/cgroups/fs/blkio.go)
> - fs.MemoryGroup.GetStats (https://github.com/docker/libcontainer/blob/master/cgroups/fs/memory.go)
> - fs.CpuacctGroup.GetStats (https://github.com/docker/libcontainer/blob/master/cgroups/fs/cpuacct.go)

My impression was that it might be worth waiting for the API to stabilize (or for the stable parts to be documented a bit more). But yes, in the long run libcontainer seems promising.

> Using a channel to guarantee thread-safe writes to stderr is clever, but it would be nice to do it without passing a stderr channel everywhere in the program. Maybe something like this:
> [...]

Changed to a global log_chan and a LogPrintf func to avoid the repetition.

> Something like that would be valuable to use as the base logging framework for all of our concurrent Go code.

Well, the "log" package already handles concurrency. The only reason we do something different here is to merge the child process's stderr data with our own. (In fact, using "log" for everything, including passing the child process's stderr, might have worked fine -- but this is how we did it.)

> OpenAndReadAll
>
> - Sorry if this is a stupid question, but could OpenAndReadAll plausibly be replaced with ioutil.ReadFile? I see some of the comments here about race conditions involved in opening and reading the file, but it's not immediately clear to me whether our current approach solves problems that ioutil.ReadFile doesn't.

Indeed, the only difference is that it emits a log message. I noticed it was only used in two places (one of which turned off the warning feature) so I rearranged the relevant bits to use ioutil.ReadFile(). Thanks for the tip.

> PollCgroupStats
>
> - It seems like it might make sense for PollCgroupStats to be responsible for waiting for the cid file to appear. It runs forever anyway, so it could just keep sleeping until either the child terminates or the cid file shows up. I haven't been through the nitty gritty here, though, so if there are situations where the container file basically never appears and crunchstat would hang forever, or if --wait=5 catches basically all of the cases we care about, then never mind.

Yeah, this whole setup is a bit weird but I don't want to tackle *everything* on this branch... I agree, it's weird that "container didn't appear in 5 seconds"

seems to put us in "print host stats" mode permanently, instead of staying in "switch to container stats if possible" mode.

> OpenStatFile
>
> - This should work for setup:
> [...]

That's much better, thanks.

> DoNetworkStats
>
> - This is a cleaner way of writing the inner loop to parse the netstats lines:
> [...]

Yes. Fixed.

- This line was startling until I realized that it's just creating a new temporary scope for lastSample: if lastSample, ok := lastSample[ifName]; ok { ... How about "ls" or "prevsample" or some other name to be less confusing about scope?

Fixed. (Used "prev" just like the other place where this pattern appears.)

Now at 41887dd

#33 - 10/22/2014 02:37 PM - Tim Pierce

Reviewing @ 41887dd:

Thanks for making these changes! LGTM.

Tom Clegg wrote:

> Tim Pierce wrote:
>
> > Reviewing @ a5ec4164:
> >
> > Something we should consider is whether we can reuse any of the code in github.com/docker/libcontainer/cgroups to do some of this work. In particular, these look promising:
> >
> > - fs.BlkioGroup.GetStats (https://github.com/docker/libcontainer/blob/master/cgroups/fs/blkio.go)
> > - fs.MemoryGroup.GetStats (https://github.com/docker/libcontainer/blob/master/cgroups/fs/memory.go)
> > - fs.CpuacctGroup.GetStats (https://github.com/docker/libcontainer/blob/master/cgroups/fs/cpuacct.go)
>
> My impression was that it might be worth waiting for the API to stabilize (or for the stable parts to be documented a bit more). But yes, in the long run libcontainer seems promising.

Sure. I just wanted to make sure we were aware of the parallel work going on over there.

> > PollCgroupStats
> >
> > - It seems like it might make sense for PollCgroupStats to be responsible for waiting for the cid file to appear. It runs forever anyway, so it could just keep sleeping until either the child terminates or the cid file shows up. I haven't been through the nitty gritty here, though, so if there are situations where the container file basically never appears and crunchstat would hang forever, or if --wait=5 catches basically all of the cases we care about, then never mind.
>
> Yeah, this whole setup is a bit weird but I don't want to tackle *everything* on this branch... I agree, it's weird that "container didn't appear in 5 seconds" seems to put us in "print host stats" mode permanently, instead of staying in "switch to container stats if possible" mode.

That's ok. Mostly I'm calling this out to find out if we are still seeing edge cases where crunchstat occasionally hangs up and does nothing and we can't tell why.

#34 - 10/22/2014 02:56 PM - Tom Clegg

Tim Pierce wrote:

> That's ok. Mostly I'm calling this out to find out if we are still seeing edge cases where crunchstat occasionally hangs up and does nothing and we can't tell why.

Ah, this sounds like a reference to #4044.

Thanks

#35 - 10/22/2014 03:05 PM - Anonymous

- Status changed from New to Resolved

- % Done changed from 83 to 100

Applied in changeset arvados|commit:89f38df7ce6e3af8e6119a111cdf985de6e0a0e9.