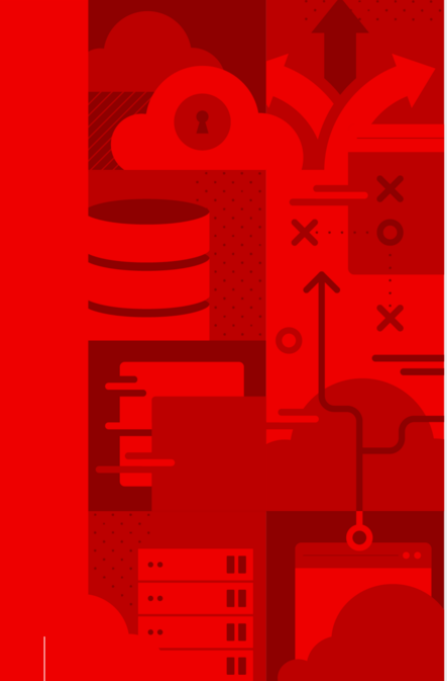
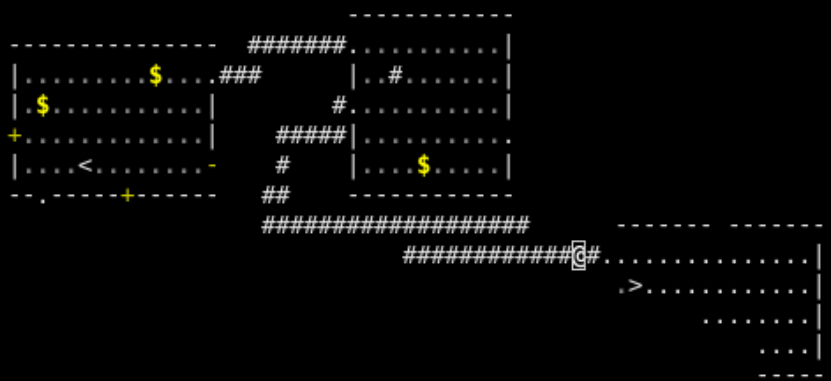


Party tricks for PostgreSQL perf, ftrace and bpftrace

DMITRII DOLGOV

26-10-2022

Red Hat



Erthalion the Digger St:11 Dx:8 Co:17 In:15 Wi:15 Ch:8 Neutral
Dlvl:1 $:0 HP:13(13) Pw:2(2) AC:9 Xp:1

```
/* XXX guessed-at value */
#define PGPROC_MAX_CACHED_SUBXIDS 64


/* has PGPROC->subxids overflowed */
bool        overflowed;
```

```
$ perf record -e probe_postgres:GetNewTransactionId
$ perf script
    postgres 29959 probe_postgres
    GetNewTransactionId: nxids=0
```

Instrumentation basics

Static tracing: Tracepoints, USDT

Dynamic tracing: k[ret]probe, u[ret]probe

Dynamic tracing: k[ret]probe vs k[ret]func

```
<handle_event>:   push %rbp
<handle_event+1>: mov  %rsp,%rbp
<handle_event+4>: sub  $0×60,%rsp

<handle_event>:   int3
<handle_event+1>: mov  %rsp,%rbp
<handle_event+4>: sub  $0×60,%rsp
```

Static tracing

```
--enable-dtrace

$ bpftrace -l 'usdt:bin/postgres:*'
usdt:bin/postgres:postgresql:buffer__checkpoint__done
usdt:bin/postgres:postgresql:buffer__checkpoint__start
[ ... ]
```

```
$ bpftrace -p <pid> -f json -e
    'usdt:bin/postgres:postgresql:checkpoint__start
    {printf("Checkpoint, Flags %d\n", arg0);}'

{"type": "attached_probes", "data": {"probes": 1}}
{"type": "printf", "data": "Checkpoint, Flags 108\n"}
{"type": "printf", "data": "Checkpoint, Flags 108\n"}
```

Dynamic tracing

```
$ perf probe -x bin/postgres \
    -L GetNewTransactionId

[ ... ]
187 int nxids = MyProc→subxidStatus.count;
[ ... ]
```

```
$ perf probe -x bin/postgres \
    -V GetNewTransactionId:187

Available variables at GetNewTransactionId:187
        @<GetNewTransactionId+1040>
                FullTransactionId       full_xid
                TransactionId   xid
                XidCacheStatus* substat
                _Bool   isSubXact
                char*   __func__
                int     nxids
```

```
$ perf probe -x bin/postgres \
    GetNewTransactionId:187 nxids

$ perf record -e \
        probe_postgres:GetNewTransactionId_L187_2

$ perf script
        postgres probe_postgres
        GetNewTransactionId_L187_2: nxids=0
```

Limitations

Context switch overhead

Context switch overhead

Debugging symbols

Context switch overhead

Debugging symbols

Post-processing

Context switch overhead

Debugging symbols

Post-processing

Incorporate into infrastructure?

Context switch overhead

Debugging symbols

Post-processing

Incorporate into infrastructure?

Privileges?

```
BPF_OBJ_PIN
    Pin an eBPF program or map referred
    by the specified *bpf_fd* to the
    provided *pathname* on the filesystem.
```

```
$ cat events/synthetic/query_latency/hist
# event histogram

hitcount: 1
{ lat_us: 223,
  query_string: select a from test1;}

hitcount: 1
{ lat_us: 314,
  query_string: select b from test2;}
```

```
$ perf probe -x bin/postgres \
    exec_simple_query query_string:string

$ perf probe -x bin/postgres \
    exec_simple_query%return
```

Belzebubs: *belzebubsofficial.tumblr.com*

Red Hat

```
synthetic_events
query_latency u64 lat_us; char[] query_string

events/probe_postgres/exec_simple_query/trigger
hist:keys=common_pid:t0=common_timestamp.usecs

events/probe_postgres/exec_simple_query__return/trigger
hist:keys=common_pid:lat_us=common_timestamp.usecs-$t0:
   onmatch(probe_postgres.exec_simple_query).
   trace(query_latency,$lat_us,query_string_string)

events/synthetic/query_latency/trigger
hist:keys=lat_us,query_string:sort=lat_us
```

```
synthetic_events
query_latency u64 lat_us; char[] query_string
```

```
events/probe_postgres/exec_simple_query/trigger
hist:keys=common_pid:t0=common_timestamp.usecs
```

```
events/probe_postgres/exec_simple_query__return/trigger
hist:keys=common_pid:lat_us=common_timestamp.usecs-$t0:
   onmatch(probe_postgres.exec_simple_query).
   trace(query_latency,$lat_us,query_string_string)
```

```
events/synthetic/query_latency/trigger
hist:keys=lat_us,query_string:sort=lat_us
```

```
synthetic_events
query_latency u64 lat_us; char[] query_string

events/probe_postgres/exec_simple_query/trigger
hist:keys=common_pid:t0=common_timestamp.usecs

events/probe_postgres/exec_simple_query__return/trigger
hist:keys=common_pid:lat_us=common_timestamp.usecs-$t0:
   onmatch(probe_postgres.exec_simple_query).
   trace(query_latency,$lat_us,query_string_string)

events/synthetic/query_latency/trigger
hist:keys=lat_us,query_string:sort=lat_us
```

```
synthetic_events
query_latency u64 lat_us; char[] query_string

events/probe_postgres/exec_simple_query/trigger
hist:keys=common_pid:t0=common_timestamp.usecs

events/probe_postgres/exec_simple_query__return/trigger
hist:keys=common_pid:lat_us=common_timestamp.usecs-$t0:
   onmatch(probe_postgres.exec_simple_query).
   trace(query_latency,$lat_us,query_string_string)

events/synthetic/query_latency/trigger
hist:keys=lat_us,query_string:sort=lat_us
```

```
synthetic_events
query_latency u64 lat_us; char[] query_string

events/probe_postgres/exec_simple_query/trigger
hist:keys=common_pid:t0=common_timestamp.usecs

events/probe_postgres/exec_simple_query__return/trigger
hist:keys=common_pid:lat_us=common_timestamp.usecs-$t0:
   onmatch(probe_postgres.exec_simple_query).
   trace(query_latency,$lat_us,query_string_string)

events/synthetic/query_latency/trigger
hist:keys=lat_us,query_string:sort=lat_us
```

Development

```
$ perf record -a -g --event=mem:0x7ff3bc1f9000/8:rw
```

```
$ bpftrace -e 'kprobe:__x64_sys_pwrite64
    /comm == "postgres"/ { override(-12) }' --unsafe
Attaching 1 probe...

=# create table test();
PANIC:  XX000: could not write to
log file 000000010000000000000002B at offset 11444224,
length 106496: Cannot allocate memory
LOCATION:  XLogWrite, xlog.c:2233
```

Tips & tricks

```
$ perf report --header

# ========
# captured on    : Wed Oct 12 17:07:32 2022
# header version : 1
# data offset    : 504
# data size      : 1392968
# feat offset    : 1393472
# arch : x86_64
# ...
```

```
$ perf archive

  Now please run:
  tar xvf perf.data.tar.bz2 -C ~/.debug
```

```
$ perf record --overwrite --switch-output=signal
$ kill -SIGUSR2 <perf-pid>
```

```
$ perf record -p <pid> -e intel_pt//u --filter\
 'filter tts_buffer_heap_materialize @ bin/postgres'
```

```
$ perf record -e event:[u:k:pp:b]
```

```
$ perf record -F max | -c 1000
$ perf script
$ stackcollapse-perf.pl
$ flamegraph.pl --hash
```


perf record -e probe_postgres:GetNewTransactionId perf script bpftrace -l 'usdt:bin/postg
stgres:postgresql:checkpoint__start printf("Checkpoint, Flags d", arg0);' perf probe -x bin/p
obe -x bin/postgres -V GetNewTransactionId:187 perf probe -x bin/postgres GetNewTrans
e probe_postgres:GetNewTransactionId_L187_2 perf script cat events/synthetic/query_la
x bin/postgres exec_simple_query query_string:string perf probe -x bin/postgres exec_simple_query return echo 'query_latency u64 lat_us; char[] query_string' > synthetic_events echo 'hist:keys=common_pid:ts0=common_timestamp.usecs' > events/probe_postgres/exec_simple_query/trigger echo 'hist:keys=common_pid:lat_us=common_timestamp.usecs-ts0:onmatch(probe_postgres.exec_simple_query).trace(query_latency,lat_us,query_string)' > events/p

Photo by Dmitrii Dolgov: Teufelsberg, Berlin

Red Hat

postgresql.org/docs/current/dynamic-trace.html

perf.wiki.kernel.org/index.php/Tutorial

kernel.org/doc/Documentation/trace/ftrace.txt

github.com/iovisor/bpftrace

Gregg, Brendan. ”Systems Performance: Enterprise and the Cloud”

Questions?

@erthalion

ddolgov at redhat dot com

Red Hat