Поддержка DTrace в MySQL: способы решения типичных проблем с производительностью

Alexey.Kopytov@Sun.com

April 9, 2009

Введение в DTrace

- DTrace — технология динамической трассировки
- Не нужно перезапускать приложение
- Работает на обычной, не “отладочной” сборке
- Используется для отладки, поиска источника проблем и анализа производительности.
- Поддержка в Solaris, OpenSolaris, Mac OS X, FreeBSD и Linux (в разработке).

Как это работает?

- Пробы — “датчики” в исполняемом коде
- Сообщают своё состояние, когда выполнение доходит до нужного места
- Неактивные пробы не ухудшают производительность
- Могут быть в ядре ОС, библиотеках или прикладных программах
  - Ядро Solaris: 36 000 проб
  - Ядро + библиотеки: 50 000 проб
  - Firefox: 50 проб

Имена проб

идентификатор_пробы := provider:module:function:name

provider:	Логическая группа проб
module:	Имя исполняемого файла или библиотеки
function:	Имя функции в исходном тексте
name:	Имя пробы

Как использовать?

Сценарии на языке **D**:

- указывают пробы, состояние которых нужно наблюдать
- описывают действия при срабатывании пробы

Пример

reads.d

```
syscall::read:entry
/execname == "ssh"/
{
  printf("%s: read(%i, %p, %i)\n",
         execname, arg0, arg1, arg2);
}
```

```
# dtrace -qs ./reads.d
ssh: read(3, bfffef64, 32)
ssh: read(4, 80c200, 1675)
^C
```

Динамические пробы

Динамические пробы:

- создаются «на лету»
- не требуют поддержки от приложения

Пример

mysql_parse.d

```
pid$target::*mysql_parse*:entry
{
    printf("%s\n", copyinstr(arg1))
}
```

```
# dtrace -p `pgrep mysqld` -qs ./mysql_parse.d
BEGIN
SELECT c from sbtest where id=5042
...
```

Статические пробы

Статические пробы:

- вставляются в исходный текст приложения

Пример

sql/sql_parse.cc

```
bool dispatch_command(...)
{
  switch (command) {
  case COM_QUERY:
    MYSQL_QUERY_START(query, thread_id, db, user, host);
    /* Query processing starts here */
    ...
  }
}
```

```
# dtrace -qn 'mysql*:::query-start \
{ printf("%s\n", copyinstr(arg0)) }'
BEGIN
SELECT c from sbtest where id=5042
...
```

Сравнение статических и динамических проб

	Динамические	Статические
Работают с любой версией MySQL	да	нет
Стабильный интерфейс	нет	да
Знакомство с исходным кодом	нужно	не нужно
Покрытие	высокое	низкое

Статические пробы в MySQL

~60 проб в MySQL 6.0:

- создание/завершение соединения
- Query Cache
- MyISAM key cache
- сетевой ввод/вывод
- Выполнение SQL запросов:
  - лексический разбор (parsing)
  - блокировки таблиц
  - отдельные типы запросов (`SELECT`, `INSERT`, `UPDATE`, ...)
  - операции с отдельными строками в «движках» (storage engines)
  - сортировка результатов (filesort)

Производительность

DTrace скрипт

```
mysql*:::
{
  @nprobes = count();
}
```

sysbench, CPU bound, OLTP read-only:

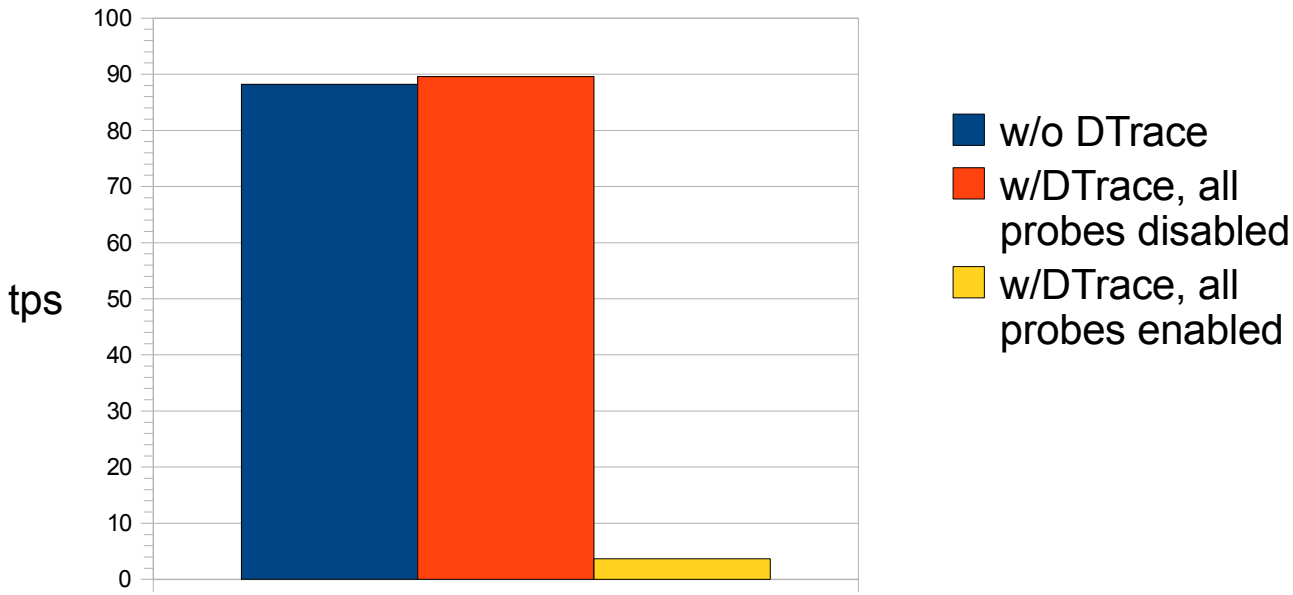
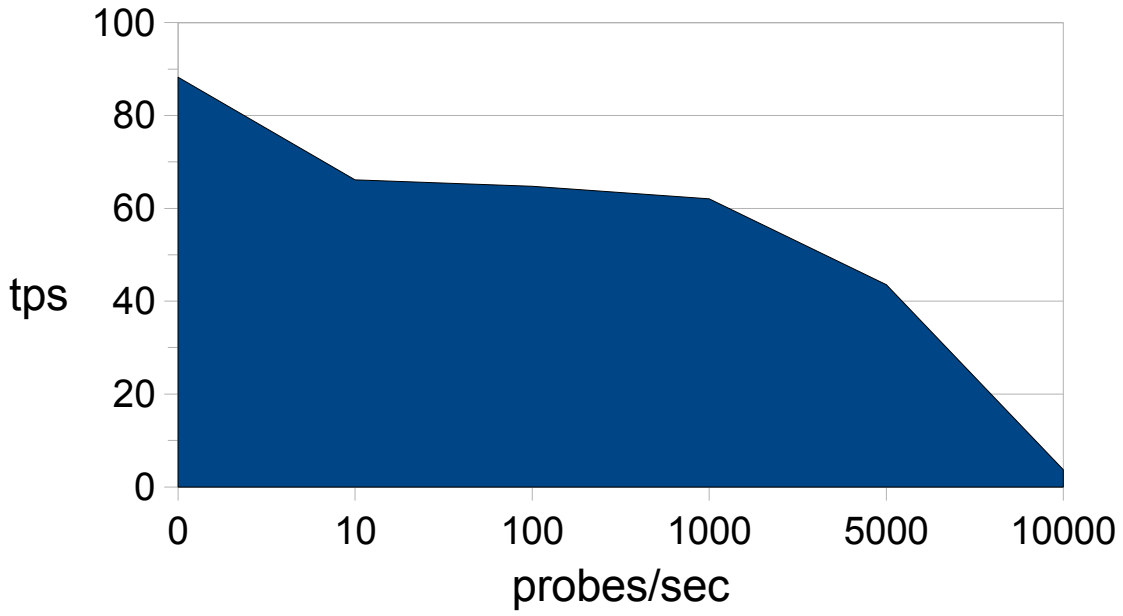


Figure: Со всеми включенными пробами производительность снижается примерно в 24 раза

$\sim$10 000 проб/сек.

Производительность



probes/sec	transactions/sec
0	88,22
10	66,17
100	64,77
1000	62,04
5000	43,56
10000	3,69

DTrace скрипт

```
profile-$1 /pid == $target /
{
  @nprobes = count();
}
```

Примеры

Пример: «Горячие» таблицы

DTrace скрипт

hot_tables.d

```
mysql$target:::*-row-start
{
  @tables[copyinstr(arg0), copyinstr(arg1)] = count();
}

END
{
  printa("%s.%s %@u\n", @tables);
}
```

Используемые пробы

```
insert-row-start(db, table)
update-row-start(db, table)
delete-row-start(db, table)
read-row-start(db, table, scan_flag)
index-read-row-start(db, table)
```

Пример: «Горячие» таблицы

Результаты работы

```
# dtrace -qp `pgrep mysqld` -s ./hot_tables.d
^C
test.t4 8
test.t8 12
test.t12 14
test.t11 18
test.t0 54
test.t3 88
test.t2 611
test.t1 12499
```

Пример: запросы на заданную таблицу

DTrace скрипт

table_scans.d

```
mysql$target:::query-exec-start
{
  self->query = copyinstr(arg0);
  self->nrows = 0;
}

mysql$target:::read-row-start
/ self->query != 0 && copyinstr(arg1) == "t1" /
{
  self->nrows++;
}

mysql$target:::query-exec-done
/ self->nrows > 0 /
{
  printf("Query: %s\n", self->query);
  printf("Rows scanned: %u\n", self->nrows);
}
```

Используемые пробы

```
query-exec-start(query, connid, db_name, user, host, exec_type)
query-exec-done(status)
read-row-start(db, table, scan_flag)
```

Пример: запросы на заданную таблицу

Результаты работы

```
# dtrace -qp ‘pgrep mysqld‘ -s ./table_rows.d
Query: SELECT a,
(SELECT REPEAT(’ ’,250) FROM t1 i1
WHERE i1.b=t1.a ORDER BY RAND() LIMIT 1) AS a
FROM t1 ORDER BY a LIMIT 5
Rows scanned: 8193
...
```

Пример: трассировка хранимых процедур

DTrace скрипт

sp_trace.d

```
mysql$target:::query-exec-start
/ arg5 == 3 /
{
  self->query_start = timestamp;
  self->query = copyinstr(arg0);
}

mysql$target:::query-exec-done
/ self->query != 0 /
{
  printf("%uus %s\n", (timestamp - self->query_start) / 1000,
         self->query);
  self->query = 0;
}
```

Используемые пробы

```
insert-row-start(db, table)
update-row-start(db, table)
delete-row-start(db, table)
read-row-start(db, table, scan_flag)
index-read-row-start(db, table)
```

Пример: трассировка хранимых процедур

Результаты работы

```
# dtrace -qp `pgrep mysqld` -s ./sp_trace.d
563us select * from t1 order by data
138us select id,data into x,y from test.t1 limit 1
...
```

Ссылки

- **Раздел в руководстве пользователя:**
  “5.7. Tracing mysqld Using DTrace”.
  http://dev.mysql.com/doc/refman/6.0/en/dba-dtrace-server.htm
- **Документация по DTrace в Wiki:**
  http://wikis.sun.com/display/DTrace/
- **Feature preview tree:**
  `bzr branch lp:~akopytov/mysql-server/mysql-6.0-dtrace`

Вопросы и ответы