Subject: [PATCH 0/7] Profiling sleep times (v4)
Posted by Andrey Vagin on Wed, 07 Dec 2011 13:55:55 GMT
View Forum Message <> Reply to Message

Do you want to know where your code waits locks for a long time?
Yes! It's for you. This feature helps you to find bottlenecks.

It's not artificial task. Once one of my colleague was investigating a
scalability problem. He pressed sysrq-t some times and tried to merge
call-chains by hand. But perf can do that.

Problem:
The problem is that events sched_stat_* contain call-chains of
non-target tasks.
About month ago I sent series of patches:

[PATCH 0/3] trace: add ability to collect call chains of non current task.

Peter and Frederic explained me, that this solve isn't good and will be
better to make it in userspace.

Now it's in userspace. This series expands "perf inject" to be able to
merge sched_switch events and sched_stat* events. sched_switch events
contain correct call-chains and sched_stat contains a correct time
slices.

v2:
* Removed all known issues. Now it works completely.
* Improved usability of sched-stat scripts according with Arun's comments.
v3: fixed accoding to comments from David Ahem
v4: rebase to linux-tip

Andrew Vagin (6):
  perf: use event_name() to get an event name
  perf: add ability to change event according to sample (v3)
  perf: add ability to record event period
  perf: teach "perf inject" to work with files
  perf: teach perf inject to merge sched_stat_* and sched_switch events
  perf: add scripts for profiling sleep times (v2)

 tools/perf/builtin-inject.c | 132 ++++++++++++++++++++--
 tools/perf/builtin-record.c | 1 +
 tools/perf/perf.h | 1 +
 tools/perf/scripts/python/bin/sched-stat-record | 65 ++++++++++
 tools/perf/scripts/python/bin/sched-stat-report | 5 +
 tools/perf/util/event.h | 2 +
 tools/perf/util/evsel.c | 77 ++++++++++++
 tools/perf/util/header.c | 2 +-

```
 tools/perf/util/session.h                  |    9 ++
 9 files changed, 286 insertions(+), 8 deletions(-)
 create mode 100644 tools/perf/scripts/python/bin/sched-stat-record
 create mode 100644 tools/perf/scripts/python/bin/sched-stat-report
```

Subject: [PATCH 4/6] perf: teach "perf inject" to work with files
Posted by Andrey Vagin on Wed, 07 Dec 2011 13:55:59 GMT
View Forum Message <> Reply to Message

Before this patch "perf inject" can only handle data from pipe.

I want to use "perf inject" for reworking events. Look at my following patch.

Signed-off-by: Andrew Vagin <avagin@openvz.org>
---
 tools/perf/builtin-inject.c | 33 +++++++++++++++++++++++++++++++--
 1 files changed, 31 insertions(+), 2 deletions(-)

```
diff --git a/tools/perf/builtin-inject.c b/tools/perf/builtin-inject.c
index 09c1061..f87779f 100644
--- a/tools/perf/builtin-inject.c
+++ b/tools/perf/builtin-inject.c
@@ -14,7 +14,12 @@

 #include "util/parse-options.h"

-static char  const *input_name = "-";
+static char  const *input_name = "-";
+static const char *output_name  = "-";
+static int  pipe_output  = 0;
+static int  output;
+static u64  bytes_written  = 0;
+
 static bool  inject_build_ids;

 static int perf_event__repipe_synth(struct perf_tool *tool __used,
@@ -27,12 +32,14 @@ static int perf_event__repipe_synth(struct perf_tool *tool __used,
  size = event->header.size;

  while (size) {
-  int ret = write(STDOUT_FILENO, buf, size);
+  int ret = write(output, buf, size);
   if (ret < 0)
    return -errno;

   size -= ret;
   buf += ret;
```

```
+
+  bytes_written += ret;
  }

  return 0;
@@ -239,8 +246,14 @@ static int __cmd_inject(void)
  if (session == NULL)
   return -ENOMEM;

+ if (!pipe_output)
+  lseek(output, session->header.data_offset, SEEK_SET);
  ret = perf_session__process_events(session, &perf_inject);

+ if (!pipe_output) {
+  session->header.data_size += bytes_written;
+  perf_session__write_header(session, session->evlist, output, true);
+ }
  perf_session__delete(session);

  return ret;
@@ -254,6 +267,10 @@ static const char * const report_usage[] = {
 static const struct option options[] = {
  OPT_BOOLEAN('b', "build-ids", &inject_build_ids,
       "Inject build-ids into the output stream"),
+ OPT_STRING('i', "input", &input_name, "file",
+       "input file name"),
+ OPT_STRING('o', "output", &output_name, "file",
+       "output file name"),
  OPT_INCR('v', "verbose", &verbose,
    "be more verbose (show build ids, etc)"),
  OPT_END()
@@ -269,6 +286,18 @@ int cmd_inject(int argc, const char **argv, const char *prefix __used)
  if (argc)
   usage_with_options(report_usage, options);

+ if (!strcmp(output_name, "-")) {
+  pipe_output = 1;
+  output = STDOUT_FILENO;
+ } else {
+  output = open(output_name, O_CREAT| O_WRONLY | O_TRUNC,
+        S_IRUSR | S_IWUSR);
+  if (output < 0) {
+   perror("failed to create output file");
+   exit(-1);
+  }
+ }
+
  if (symbol__init() < 0)
```

```
  return -1;

--
1.7.1
```

Subject: [PATCH 5/6] perf: teach perf inject to merge sched_stat_* and sched_switch events
Posted by Andrey Vagin on Wed, 07 Dec 2011 13:56:00 GMT
View Forum Message <> Reply to Message

You may want to know where and how long a task is sleeping. A callchain
may be found in sched_switch and a time slice in stat_iowait, so I add
handler in perf inject for merging this events.

My code saves sched_switch event for each process and when it meets
stat_iowait, it reports the sched_switch event, because this event
contains a correct callchain. By another words it replaces all
stat_iowait events on proper sched_switch events.

Signed-off-by: Andrew Vagin <avagin@openvz.org>
---
 tools/perf/builtin-inject.c | 99 ++--
 1 files changed, 94 insertions(+), 5 deletions(-)

```
diff --git a/tools/perf/builtin-inject.c b/tools/perf/builtin-inject.c
index f87779f..fa6b6f0 100644
--- a/tools/perf/builtin-inject.c
+++ b/tools/perf/builtin-inject.c
@@ -13,6 +13,8 @@
 #include "util/debug.h"

 #include "util/parse-options.h"
+#include "util/trace-event.h"
+

 static char  const *input_name = "-";
 static const char *output_name  = "-";
@@ -21,6 +23,9 @@ static int  output;
 static u64  bytes_written  = 0;

 static bool  inject_build_ids;
+static bool  inject_sched_stat;
+
+struct perf_session *session;

 static int perf_event__repipe_synth(struct perf_tool *tool __used,
         union perf_event *event,
```

```
@@ -47,7 +52,7 @@ static int perf_event__repipe_synth(struct perf_tool *tool __used,

 static int perf_event__repipe_op2_synth(struct perf_tool *tool,
      union perf_event *event,
-     struct perf_session *session __used)
+     struct perf_session *s __used)
 {
  return perf_event__repipe_synth(tool, event, NULL);
 }
@@ -59,7 +64,7 @@ static int perf_event__repipe_event_type_synth(struct perf_tool *tool,
 }

 static int perf_event__repipe_tracing_data_synth(union perf_event *event,
-       struct perf_session *session __used)
+       struct perf_session *s __used)
 {
  return perf_event__repipe_synth(NULL, event, NULL);
 }
@@ -114,12 +119,12 @@ static int perf_event__repipe_task(struct perf_tool *tool,
 }

 static int perf_event__repipe_tracing_data(union perf_event *event,
-        struct perf_session *session)
+        struct perf_session *s)
 {
  int err;

  perf_event__repipe_synth(NULL, event, NULL);
- err = perf_event__process_tracing_data(event, session);
+ err = perf_event__process_tracing_data(event, s);

  return err;
 }
@@ -205,6 +210,86 @@ repipe:
  return 0;
 }

+struct event_entry
+{
+ struct list_head list;
+ u32 pid;
+ union perf_event event[0];
+};
+
+static LIST_HEAD(samples);
+
+static int perf_event__sched_stat(struct perf_tool *tool,
+          union perf_event *event,
```

```
+          struct perf_sample *sample,
+          struct perf_evsel *evsel __used,
+          struct machine *machine)
+{
+ int type;
+ struct event *e;
+ const char *evname = NULL;
+ uint32_t size;
+ struct event_entry *ent;
+ union perf_event *event_sw = NULL;
+ struct perf_sample sample_sw;
+ int sched_process_exit;
+
+ size = event->header.size;
+
+ type = trace_parse_common_type(sample->raw_data);
+ e = trace_find_event(type);
+ if (e)
+  evname = e->name;
+
+ sched_process_exit = !strcmp(evname, "sched_process_exit");
+
+ if (!strcmp(evname, "sched_switch") ||  sched_process_exit) {
+  list_for_each_entry(ent, &samples, list)
+   if (sample->pid == ent->pid)
+    break;
+
+  if (&ent->list != &samples) {
+   list_del(&ent->list);
+   free(ent);
+  }
+
+  if (sched_process_exit)
+   return 0;
+
+  ent = malloc(size + sizeof(struct event_entry));
+  ent->pid = sample->pid;
+  memcpy(&ent->event, event, size);
+  list_add(&ent->list, &samples);
+  return 0;
+
+ } else if (!strncmp(evname, "sched_stat_", 11)) {
+  u32 pid;
+
+  pid = raw_field_value(e, "pid", sample->raw_data);
+
+  list_for_each_entry(ent, &samples, list) {
+   if (pid == ent->pid)
```

```
+    break;
+  }
+
+  if (&ent->list == &samples) {
+   pr_debug("Could not find sched_switch for pid %u\n", pid);
+   return 0;
+  }
+
+  event_sw = &ent->event[0];
+  perf_session__parse_sample(session, event_sw, &sample_sw);
+  sample_sw.period = sample->period;
+  sample_sw.time = sample->time;
+  perf_session__synthesize_sample(session, event_sw, &sample_sw);
+  perf_event__repipe(tool, event_sw, &sample_sw, machine);
+  return 0;
+ }
+
+ perf_event__repipe(tool, event, sample, machine);
+
+ return 0;
+}
 struct perf_tool perf_inject = {
   .sample  = perf_event__repipe_sample,
   .mmap  = perf_event__repipe,
@@ -230,7 +315,6 @@ static void sig_handler(int sig __attribute__((__unused__)))

 static int __cmd_inject(void)
 {
- struct perf_session *session;
   int ret = -EINVAL;

   signal(SIGINT, sig_handler);
@@ -240,6 +324,9 @@ static int __cmd_inject(void)
    perf_inject.mmap  = perf_event__repipe_mmap;
    perf_inject.fork  = perf_event__repipe_task;
    perf_inject.tracing_data = perf_event__repipe_tracing_data;
+ } else if (inject_sched_stat) {
+   perf_inject.sample = perf_event__sched_stat;
+   perf_inject.ordered_samples = true;
   }

   session = perf_session__new(input_name, O_RDONLY, false, true, &perf_inject);
@@ -267,6 +354,8 @@ static const char * const report_usage[] = {
 static const struct option options[] = {
   OPT_BOOLEAN('b', "build-ids", &inject_build_ids,
        "Inject build-ids into the output stream"),
+ OPT_BOOLEAN('s', "sched-stat", &inject_sched_stat,
+      "correct call-chains for shed-stat-*"),
```

```
  OPT_STRING('i', "input", &input_name, "file",
      "input file name"),
  OPT_STRING('o', "output", &output_name, "file",
--
1.7.1
```

Subject: [PATCH 6/6] perf: add scripts for profiling sleep times (v2)
Posted by Andrey Vagin on Wed, 07 Dec 2011 13:56:01 GMT
View Forum Message <> Reply to Message

E.g.:
 # perf script record -- sched:sched_stat_sleep -- ./foo
 # perf script report sched-stat
or
 # perf script record -- -e sched:sched_stat_sleep

v2: Add ability to record events for a defined process. It executes a small
bash script, then executes perf with filtering by pid and then a bash script
executes a target process.

Signed-off-by: Andrew Vagin <avagin@openvz.org>
---
 tools/perf/scripts/python/bin/sched-stat-record | 65 ++++++++++++++++++++++
 tools/perf/scripts/python/bin/sched-stat-report | 5 ++
 2 files changed, 70 insertions(+), 0 deletions(-)
 create mode 100644 tools/perf/scripts/python/bin/sched-stat-record
 create mode 100644 tools/perf/scripts/python/bin/sched-stat-report

diff --git a/tools/perf/scripts/python/bin/sched-stat-record
b/tools/perf/scripts/python/bin/sched-stat-record
new file mode 100644
index 0000000..02da7c1
--- /dev/null
+++ b/tools/perf/scripts/python/bin/sched-stat-record
@@ -0,0 +1,65 @@
+#!/bin/bash
+# perf script record -- sched:sched_stat_[smth] -- CMD
+# perf script record -- -e sched:sched_stat_[smth]
+#
+set -o monitor
+
+usage()
+{
+ echo "Usage:"
+ echo " perf script record sched-stat -- sched:sched_stat_[smth] -- CMD"
+ echo " perf script record sched-stat -- [PERF OPTS] -e sched:sched_stat_[smth]"
+ exit 1;

```
+}
+
+declare -a opt
+declare -a cmd
+f=0;
+for i in "${@:2}"; do
+ if [ "$i" == "--" ]; then
+  f=1
+  continue
+ fi
+ if [ $f -eq 1 ]; then
+  cmd[${#cmd[*]}]="$i"
+ else
+  opt[${#opt[*]}]="$i"
+ fi
+done
+
+if [[ "${opt[@]}" != *sched_stat_* ]]; then
+ usage;
+fi
+
+if [ ${#cmd[@]} -eq 0 ]; then
+ if [ ${#opt[@]} -eq 0 ]; then
+  usage;
+ fi
+ exec perf record -agP \
+  -e sched:sched_switch \
+  --filter "prev_state == 1 || prev_state == 2" \
+  "${opt[@]}"
+fi
+
+if [ ${#opt[@]} -ne 1 ]; then
+ usage;
+fi
+
+# Wait until a target process is stopped.
+bash -c 'kill -STOP $$; exec "$@"' -- "${cmd[@]}" &
+pid=$!
+wait %1
+[ $? -eq 147 ] || exit 1;
+
+perf record -agP \
+  -e sched:sched_switch \
+  --filter "prev_pid == $pid && prev_state == 1 || prev_state == 2" \
+  -e sched:sched_process_exit -e "${opt[@]}" --filter "pid == $pid" &
+pid_perf=$!
+kill -CONT %1
+while :; do
```

```
+ wait %1
+ [ $? -eq 127 ] && break;
+done
+kill -INT %2
+wait %2
diff --git a/tools/perf/scripts/python/bin/sched-stat-report
b/tools/perf/scripts/python/bin/sched-stat-report
new file mode 100644
index 0000000..e5114ce
--- /dev/null
+++ b/tools/perf/scripts/python/bin/sched-stat-report
@@ -0,0 +1,5 @@
+#!/bin/bash
+# description: profiling sleep times
+perf inject -s -i perf.data -o perf.data.d || exit
+perf report -i perf.data.d || exit
+unlink perf.data.d
--
1.7.1
```