
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

```
