Trace Everything:
When APM Meets Sysadmins

Mark Stemm,  
Sysdig Engineer
Transaction Tracing
Transaction Tracing

\[ \Delta T = ??? + ??? \]
Transaction Tracing
Transaction Tracing

\[ \Delta T = ??? + ??? + ??? \]
APM Tools
function perform_query() {
    ...
}
function serve_request(conn) {
    if (...) {
        perform_query();
    }
}
function main() {
    sock = listen();
    while (conn=sock.accept()) {
        serve_request(conn);
    }
}
function perform_query() {
    ...
}
function serve_request(conn) {
    if (...) {
        perform_query();
    }
}
function main() {
    sock = listen();
    while (conn=sock.accept()) {
        serve_request(conn);
    }
}

$\Delta T = \Delta (\text{main}) + \Delta (\text{serve_request}) + \Delta (\text{perform_query})$
Bringing it all together

\[ \Delta T = \Delta (main) + \Delta (serve\_request) + \Delta (perform\_query) \]
How developers are doing (distributed) tracing and profiling?

- OpenTracing and alternatives:
  - Zipkin, Dapper, HTrace, X-Trace
- Commercial:
  - New Relic, AppDynamics, Dyn, SPM
- Self-brewed and hacks:
  - statsd, JMX
  - print
  - logs (can be difficult!)
What if we had something...

- Open source
- Simple and easy to use (trade-off vs eBPF/bcc)
- Lightweight
- Could work everywhere (including containers)
- Combines system and application tracing
Open Source system troubleshooting with native container support
(htop, vmstat, netstat, lsof, tcpdump…)
Sysdig
• Capture system events, filter and run scripts
• Trace dumps for analysis
• Container support
• CLI or curses interface
Sysdig Tracers: System call tracing
Inject markers inside Sysdig event stream
Mark being and end of **spans**
  - Function calls
  - Network request
  - Arbitrary piece of code
From any language (write to /dev/null)
Low overhead (<1us)
Event stream

- Open
- Read
- Close
- Connect
- Read
- Write
- Read
- Read
- Read
- Write
- Close
## Event stream

<table>
<thead>
<tr>
<th>Open</th>
<th>Read</th>
<th>Close</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt; Request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connect</td>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td></td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td>&lt; Request</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Event stream

- Open
- Read
- Close
- Connect
- Read
- Write
- Read
- Read
- Write
- Close

> Request

Span

< Request
Event stream

- Open
- Read
- Close
- > Request
- Connect
- Read
- Write
- > Request.Download
- Read
- Read
- < Request.Download
- Write
- Close
- < Request

Parent Span

Child Span
Format

Simple format:
<dir>:<ID>:<tag1>..<tag2>...:<arg1>=<val1>,<arg2>=<val2>...

JSON Format:
["<direction>", <ID>, ["tag1", "tag2"...],
[{"arg1":"val1"}, {"arg2":"val2"}...]]
Example

while :
  do
    # Start a trace named 'website-latency'
    echo ">::::website-latency:::" > /dev/null
    # Download the sysdig home page
    curl -s http://sysdig.org > /dev/null
    # End the the trace
    echo "<::::website-latency:::" > /dev/null
  done
Language Support

Python
```python
from sysdig_tracers import Tracer
import time

while True:
    with Tracer():
        print "Hello World"
    time.sleep(0.5)
```

NodeJS
```javascript
const t = new Trace('some-unique-id')
t.start('request')
await request(...) 
t.stop('request')
```

Go
```go
var buf bytes.Buffer
probe := trace.Trace{ID: "unique-id", Writer: &buf}
probe.Start("db.query", trace.Args{"user": "tobi"})
...
probe.Stop("db.query", nil)
```
Fun things you can do

- Measure latencies in your code
- Save and filter traces with sysdig
- Analyze traces with csysdig
- Inspect system activity inside execution spans
- Trace-aware Log Monitoring
- Export trace latencies using statsd
What we’re going to show you

• Combined application/system transaction tracing with Sysdig
• Visualizing tracers with Csysdig
• Tracing nested distributed transactions
• Using tracers to scope activity
Further reading

● Documentation:
  ○ https://github.com/draios/sysdig/wiki/Tracers
  ○ https://sysdig.com/blog/tracking-down-application-bottlenecks-with-tracers/

● Integrations
  ○ Python https://github.com/draios/tracer-py
  ○ Node: https://github.com/tj/node-trace
  ○ Go: https://github.com/tj/go-trace
• Github
  • https://github.com/draios/sysdig
  • Pull Requests welcome!
• Wiki
  • https://github.com/draios/sysdig/wiki
• Docker Hub
  • https://hub.docker.com/r/sysdig
Thank You!