Principled workflow-centric tracing of distributed systems

Raja Sambasivan
Ilari Shafer, Jonathan Mace, Ben Sigelman, Rodrigo Fonseca, Greg Ganger
Today’s distributed systems

E.g., Twitter

Twitter “death star”: https://twitter.com/adrianco/status/441883572618948608
Today’s distributed systems

E.g., Netflix

Machine-centric tools insufficient

E.g., Twitter

GDB, gprof, strace, linux perf. counters

Amazingly complex

Netflix “death star”: http://www.slideshare.net/adriancockcroft/fast-delivery-devops-israel
Workflow-centric tracing

Provides the needed coherent view

Get 17 µs
27 ms

App Server
Trace point (e.g., at functions)

Table store

Server

Distributed FS

Metadata (e.g., IDs)
### It is useful / being adopted

<table>
<thead>
<tr>
<th>Category</th>
<th>Management task</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Diagnosis</strong></td>
<td>ID anomalous workflows</td>
</tr>
<tr>
<td></td>
<td>ID workflows w/ steady-state problems</td>
</tr>
<tr>
<td></td>
<td>Profiling</td>
</tr>
<tr>
<td><strong>Resource mgmt.</strong></td>
<td>Attribution</td>
</tr>
<tr>
<td></td>
<td>Performance tuning</td>
</tr>
<tr>
<td><strong>Multiple</strong></td>
<td>Dynamic monitoring</td>
</tr>
</tbody>
</table>

- **Stardust** [SIGM’06]
- **Stardust+** [NSDI’11]
- **X-Trace** [NSDI’07]
- **X-Trace+** [WREN’10]
- **Retro** [NSDI’15]
- **PivotTrace** [SOSP’15]
- **Pip** [NSDI’06]
- **Pinpoint** [NSDI’04]
- **Mace** [PLDI’07]
- **Dapper** [TR10-14]
- **HTrace** \ **Zipkin**
- **UberTrace**

**But, no clarity for tracing developers**
But, **no clarity** for tracing developers
We provide **clarity** for tracing developers

**Methodology:**

Use experiences to distill design axes

ID design choices best for different tasks

Compare to existing infrastructures
Key results

1. Different design decisions needed for diagnosis and resource management

2. Batching causes some design decisions across some axes to interact poorly

3. Existing tracing infrastructures suited to a task make similar choices to our suggestions
Anatomy & design axes

Management tasks

Trace construction

Trace storage

In-band

Out-of-band

App Server

Table store

File system

Sample?

Causal relationships?

How to define a request?

Conc./Sync. needed?

Inter-request needed?

How will trace points be added?

In-band / out-of-band?

What to use to reduce ovhd?
How original Stardust defined requests

10 µs
WRITE START

CACHE WRITE

20 ms
INSERT BLOCK

WRITE REPLY

Response time: ~20 ms

Unaccounted latency

Trace not useful for diagnosis tasks
Two valid ways to define a request’s workflow

```
WRITE START
CACHE WRITE
INSERT BLOCK
WRITE REPLY
EVICT BLOCK
DISK START
DISK END

WRITE START
CACHE WRITE

WRITE REPLY
INSERT BLOCK
WRITE START
CACHE WRITE

Latent work

Resource management: Assign latent work to original submitter
```
Two valid ways to define a request’s workflow

**Diagnosis:** Assign latent work to request on whose critical path it is executed.
Future research directions

- Reducing difficulty of adding trace points
- Lowering overhead when identifying anomalous workflows
- Exploring new analyses
Key design choices dictate workflow-centric utility for different tasks

We identify choices best suited for different tasks