Falcon

Joe Malle
Janar thanan Rajendran
Detecting Failures in Distributed Systems

The all powerful Distributed System
FALCON

Fast And Lethal Component Observation Network
Motivation

Common way to tolerate crashes:

- Detect failure
- Recover from failure

Detection is usually the bottleneck
Why is failure detection hard?

- It is fundamentally difficult to be fast and accurate
Ideal properties of a Failure Detector

- **Reliability**
  - If a FD reports a process $p$ as down, then $p$ has crashed
  - If $p$ crashes, then the FD eventually reports $p$ as DOWN

- **Fast detection**
  - Time taken to report DOWN should be short

- **Little disruption**
  - Do not disrupt everything else in the server to achieve the first two goals
Existing methods: end-to-end timeouts

How long should the timeout be?

○ Too long: system unavailable for long time
○ Too short:
  ■ Falsely declare a working node DOWN
  ■ Probing incessantly -> costly
Existing methods: end-to-end timeouts

It is difficult to tell what went wrong:

- Network may have failed
- OS could have panicked
- etc
FALCON Goals

- Reliable
- Fast
- Little Disruption
FALCON : Observation

- Crash failures can be observed readily by looking at the right layer and using inside information
FALCON

- One spy for each layer
- Platform specific logic for each spy
- Spy sends UP if process is up
- Spy sends DOWN if a process is crashed
- When it is uncertain
  - Kills it and report DOWN
  - Kills the smallest possible layer
FALCON

- Large end-to-end timeout to cover rare cases that the spy misses
- A spy also monitors the spy in the higher layer
Falcon goals

**Reliable**: If process is up, report UP. Otherwise eventually report DOWN.

- If a process is behaving well, let it be and report UP (at regular intervals).
- If a spy detects a failure, it reports DOWN (as soon as it knows).
- If a spy is uncertain, it kills the process and report DOWN.
Falcon goals

**Fast:** Time taken to report DOWN should be short

- Each layer implements failure detection differently. Most depend on polling.
- If a spy detects a failure, it reports DOWN (as soon as it knows).
- Big step up from timeouts
Falcon goals

**Little Disruption:** Do not disrupt everything else in the server to achieve the first two goals

- Spies kill the smallest layer possible.
- CPU overhead and per-platform requirements for spies are small
ADMINISTRIVIA

• Send me a picture of yourself by next class
  • **Subject:** [EECS591] Picture of `<first name>` (preferred)
    `<last name>` `<UMID>`

• Make sure to subscribe to our Piazza forum
  • Announcements, discussion, etc.

• I’ve given a few more overrides
ADMINISTRIVIA

- Still missing a few pictures
- Still missing a few people on Piazza
- Piazza poll about moving class to 3:30-5
- I’ll give 1-2 overrides soon
ADMINISTRIVIA

- Over to Joe
  - Details of Spies
  - Evaluation of Flacon
This is what Falcon does

Failed process?
- Ask the OS/VM

Failed VM?
- Ask the host OS

Failed host OS?
- Ask the network

Start a spy process in the VM that monitors other processes

Start a spy process in the host OS that monitors the VM(s)

Start a spy process in the network switch that monitors the host OS(es)
This is what Falcon does

<table>
<thead>
<tr>
<th>where injected?</th>
<th>what is the failure?</th>
<th>what does the failure model?</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>forced crash</td>
<td>app. memory error, assert failure, or condition that causes exit</td>
</tr>
<tr>
<td>application</td>
<td>app inspector reports LAYER_DOWN</td>
<td>inside information that indicates an application crash</td>
</tr>
<tr>
<td>application/</td>
<td>non-responsive app inspector</td>
<td>since the app inspector is a thread inside the application, this models a buggy application (or app inspector) that cannot run but has not exited</td>
</tr>
<tr>
<td>Falcon itself</td>
<td>infinite loop</td>
<td>kernel hang or liveness problem</td>
</tr>
<tr>
<td>kernel</td>
<td>stack overflow</td>
<td>runaway kernel code</td>
</tr>
<tr>
<td>kernel</td>
<td>kernel panic</td>
<td>unexpected condition that causes assert failure in kernel</td>
</tr>
<tr>
<td>VMM/host</td>
<td>VMM error; causes guest termination</td>
<td>VMM memory error, assert failure, or condition that causes guest exit</td>
</tr>
<tr>
<td>VMM/host</td>
<td>ifdown eth0 on host</td>
<td>hardware crash (machine is separated from network)</td>
</tr>
<tr>
<td>Falcon itself</td>
<td>crash of app enforcer</td>
<td>bug in Falcon app spy</td>
</tr>
<tr>
<td>Falcon itself</td>
<td>crash of incremener</td>
<td>bug in Falcon OS spy</td>
</tr>
<tr>
<td>Falcon itself</td>
<td>crash of OS enforcer</td>
<td>bug in Falcon OS spy</td>
</tr>
<tr>
<td>Falcon itself</td>
<td>crash of VMM inspector</td>
<td>bug in Falcon VMM spy</td>
</tr>
</tbody>
</table>
Falcon performance

Static Timer (D), Chen/FDI sequential detector (C), Bertier Heartbeat failure detection (B), \( \phi \)-accrual with 0.01 error (P1), and \( \phi \)-accrual with 0.001 error (P2);
Falcon + Zookeeper performance

Not shown: Zookeeper’s painfully slow recovery time
### Falcon performance

<table>
<thead>
<tr>
<th>Component</th>
<th>CPU overhead (percent of a core’s cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>App uses no CPU</td>
</tr>
<tr>
<td>app inspector</td>
<td>0.06</td>
</tr>
<tr>
<td>app enforcer</td>
<td>0.11</td>
</tr>
<tr>
<td>increminter</td>
<td>0.58</td>
</tr>
<tr>
<td>VM total</td>
<td><strong>0.75%</strong></td>
</tr>
<tr>
<td>OS enforcer (main)</td>
<td>0.01</td>
</tr>
<tr>
<td>OS enforcer (worker)</td>
<td>0.04</td>
</tr>
<tr>
<td>libvirtd</td>
<td>0.91</td>
</tr>
<tr>
<td>QEMU</td>
<td>6.92</td>
</tr>
<tr>
<td>VMM inspector</td>
<td>0.39</td>
</tr>
<tr>
<td>VMM total</td>
<td><strong>8.27%</strong></td>
</tr>
<tr>
<td>VMM enforcer</td>
<td>0.00</td>
</tr>
<tr>
<td>switch total</td>
<td><strong>0.00%</strong></td>
</tr>
</tbody>
</table>

**Figure 14**—Background CPU overhead of our Falcon implementation, under an idle dummy application and under one that consumes 90% of its CPU. Each enforcer performs a local check 10 times per second. The switch’s CPU overhead is less than one part in 10,000 so displays as 0. QEMU’s contribution to the overhead is explained in the text.
Falcon gotchas

What if the spy fails? There is a timeout as a backup
Falcon gotchas

What happens if the network is partitioned and the client asks for the status of something on the other end of the partition?

Falcon will say “I don’t know” until the network is back up.

It’s hard to communicate without a network!
Falcon weaknesses

You have to write a custom spy program for each layer

Polling is not ideal

Only handles crash failures (no BFT)
Free Falcon! Official version
Free Falcon! Bootleg version
Architecture of Spies

Inspector and Enforcer
Unreliable Failure Detectors (UFD)

- Reports DOWN when a process is up (and just slow)
- Required complex distributed algorithms to handle it
  - Eg. Paxos