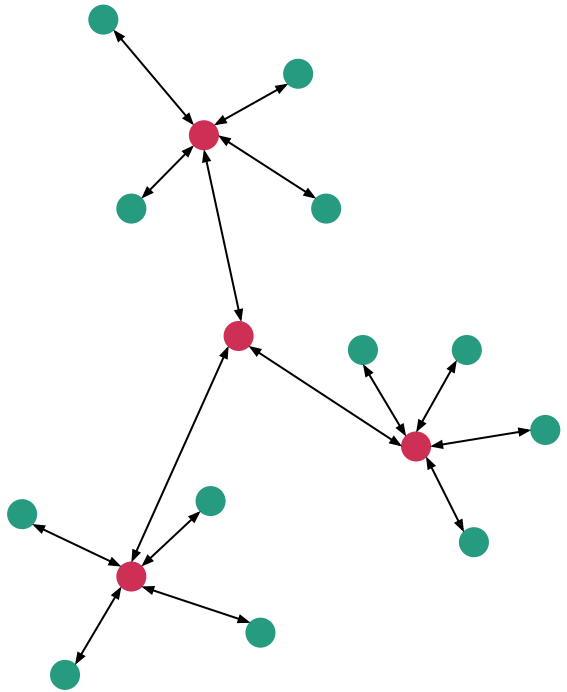
Justin Findlay
Systems Engineer

Use one or more weird tricks to speed up your salt master

Distributed model

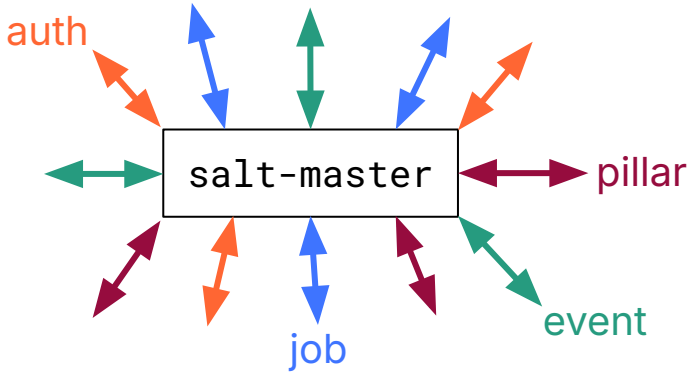
- Salt wants you to think of configuration management as a type of distributed computing
- Privileged nodes are a liability that will set the scale factor by the inefficiency imposed



`salt-minion`

`salt-master`

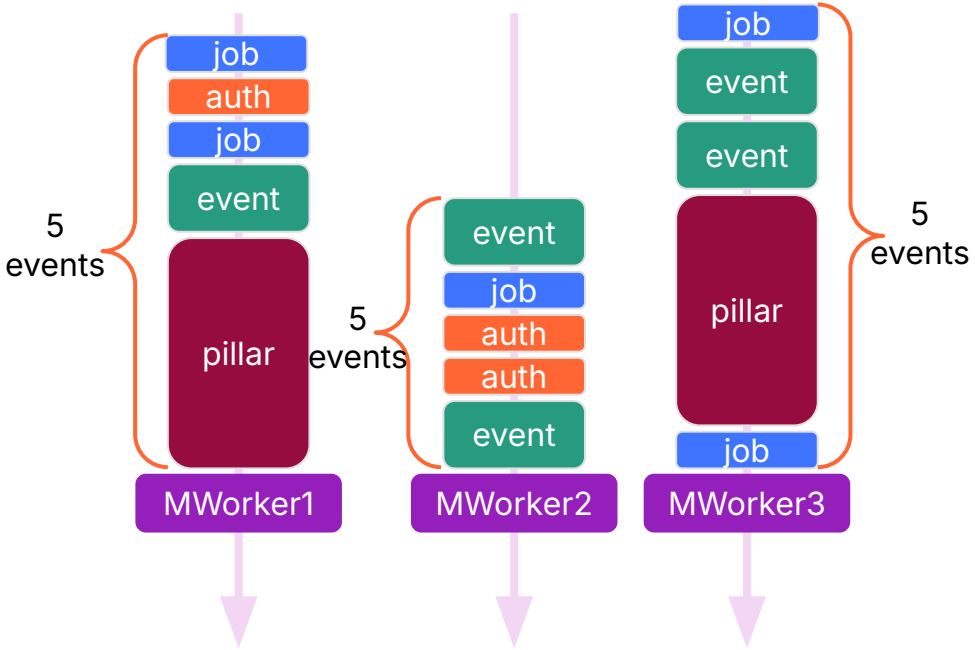
Distributed model



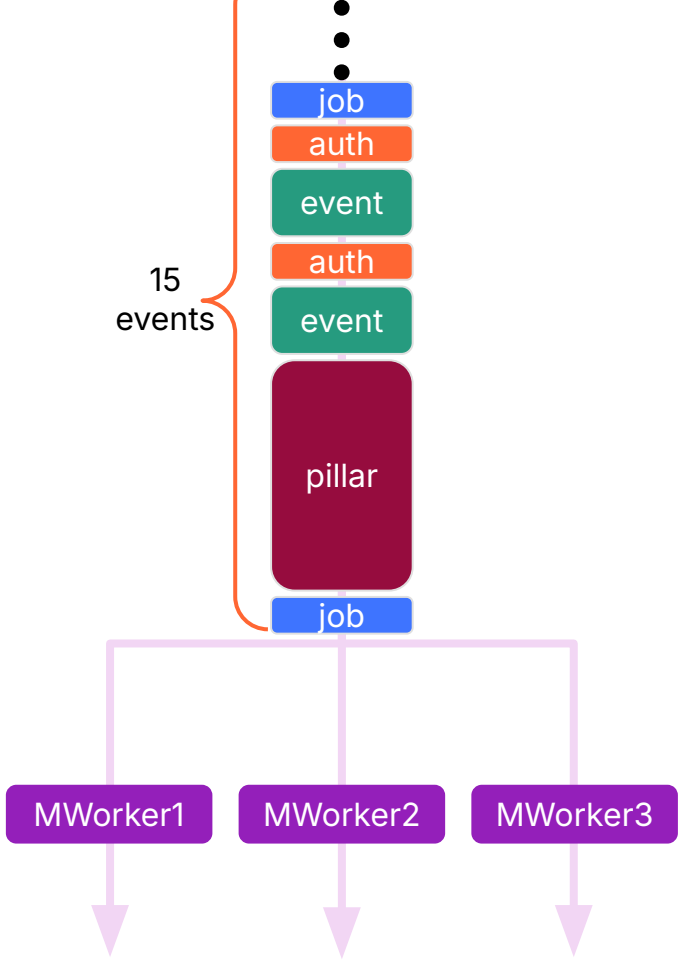
Salt masters:

- Auth with minions
- Broker `jobs` between clients (`salt` cli, `salt-api`) and minions
- Render pillar data
- React to events

`MWorker` queueing



Before: round robin



After: single queue

`MWorker` queueing

Performance improvement:
~10% `MWorker` throughput

Author: **Sze Chuen Tan**
The code: https://github.com/saltstack/salt/pull/67215
Reference:
https://zguide.zeromq.org/docs/chapter3/#A-Load-Balancing-Message-Broker

No initial pillar

Salt renders pillar at esoteric events hardcoded in the pillar code

- One of these events is master↔minion reconnection
- Pushing a minion config update triggers N_{minion} pillar renders

No initial pillar

Restarting N_{minion}s triggers N_{minion} pillar renders

- Because a master **faithfully executes every event** in its MWorker queue and a **minion abandons and rerequests events** that exceed their TTL
- At ~10k minions, depending on render time per pillar, a master **can runaway endlessly rerendering** the same pillars that the minions have already discarded

No initial pillar

Restarting N_{minion}s triggers N_{minion} pillar renders

- Restarting the master won't fix this, each minion has to **stop endlessly asking** for the same data

No initial pillar

Restarting N_{minion}s triggers N_{minion} pillar renders

- So either **shutdown most of the minions** long enough and **bring them online slowly enough** for the backlog to clear (forget about it)
- Or fix minion **event TTL** and **priority** (good luck fixing salt's **lossy connectivity** without completely rewriting the codebase)
- Or ...

No initial pillar

Restarting N_{minion}s triggers N_{minion} pillar renders

- **Completely disable initial pillar**
- A minion will request a new pillar at the start of highstate anyway
- It is mostly* unused

* The few cases for which is was used at Cloudflare turned out to be trivial to mitigate

No initial pillar

Disabling initial pillar meant going from a multi SRE shift ordeal to an event nobody notices

- "We can probably scale indefinitely. The **reauth** thundering herd is nothing compared to **initial pillar**"
- Until, well, you can probably guess where we are getting now

No initial pillar

How to do it

- `$ salt-call --skip-initial-pillar`
- `echo skip_initial_pillar: true >> /etc/salt/minion`

Performance improvement: negligible
Toil elimination: priceless

Author: **Yew Leong**
The code: https://github.com/saltstack/salt/pull/67216

Caching jinja `import_*`

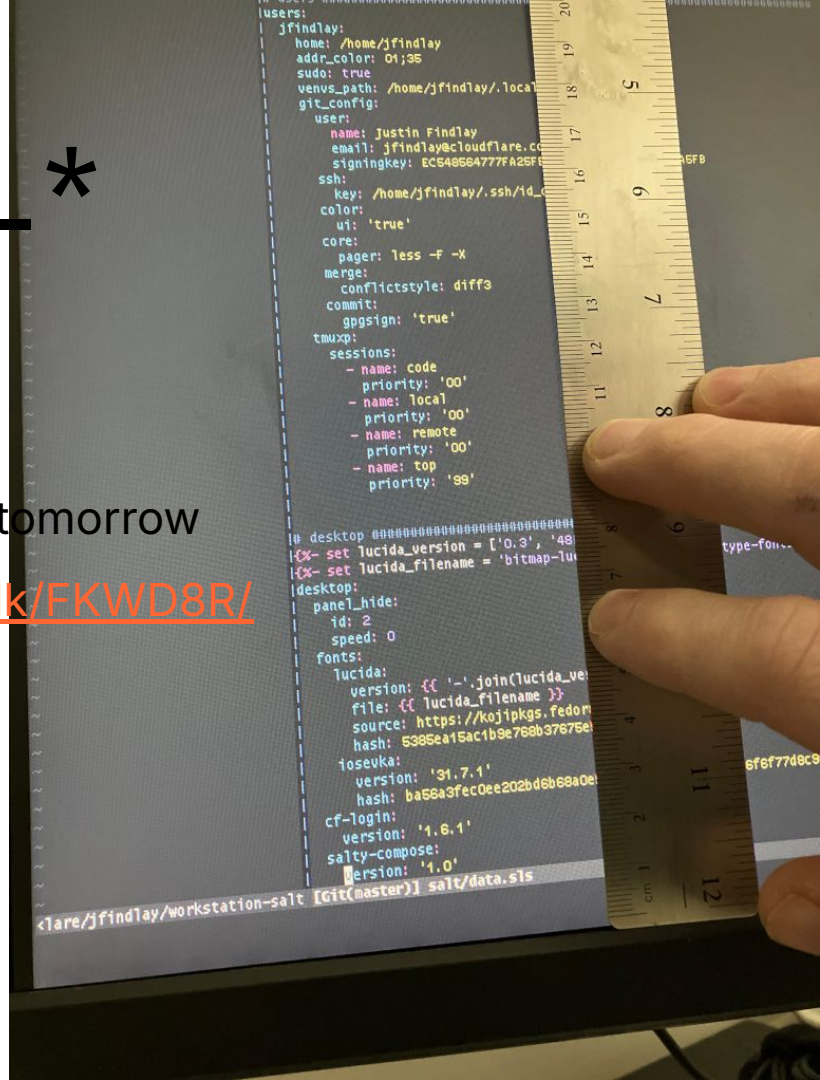
Jinja template rendering in python is not fast

- In general, caching in salt could be much better in coverage and robustness
- For example, `{%- import_* %}` calls are a convenient point of memoization for repeated identical data

Caching jinja `import_*`

Jinja template rendering in python is not fast

- Or, you know, you could **stop using jinja**
- For more on this topic, see my next session tomorrow
- https://cfp.cfgmgmtcamp.org/ghent2025/talk/FKWD8R/



Caching jinja `import_*`

Depending on the number of import_* calls used in the codebase

- At Cloudflare, we got a

Performance improvement:
~8% pillar render

Author: **Brian Harring**
The code: https://github.com/saltstack/salt/pull/67217

Bonus: One last weird trick

Masterless?

- 0(1) **scalability** of salt (scalability within salt itself)
- Convert **nebulous role** of salt into something bounded by a **well-defined contract**
- Layers of **unnecessary abstraction** within salt removed in favor of newer external **purpose-built** services

Bonus: One last weird trick

Removing the master nodes from the service graph makes each minion node fully **autonomous** and the service graph **completely horizontal**

- Layers of unnecessary abstraction within salt replaced by newer purpose-built systems for
  - Secrets (pillar→vault)
  - Orchestration (reactors→temporal/airflow/kafka)
  - Realtime lookup (grains/mine/beacons→clickhouse/prometheus/consul/osquery)

Bonus: One last weird trick

Masterless also means

- No need for builtin transport and crypto
- No minions forking their process tree per master
- Just highstate

Bonus: One last weird trick

If all we are doing is highstate

- It is much easier to **PoC greenfield** config management solutions
- And **iterate** on them even in a complex staging or production environment

All code in presentation and more at:
https://github.com/saltstack/salt/pulls/jfindlay

CLOUDFLARE
Justin Findlay
Systems Engineer

Thank you

Cloudflare Platform Configuration team

- Cian Leow
- Walter Clark
- Menno Bezema
- Marek Schwann

- Joe Groocock
- Vasilii Alferov

Code fixes provided by

- Sze Chuen Tan
- Yew Leong
- Brian Harring