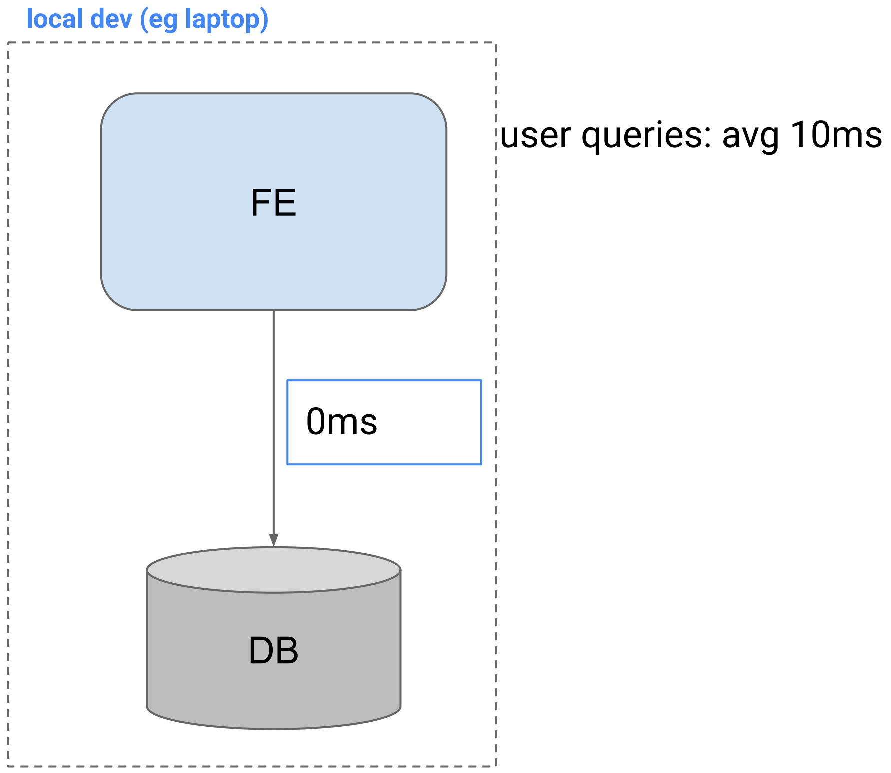
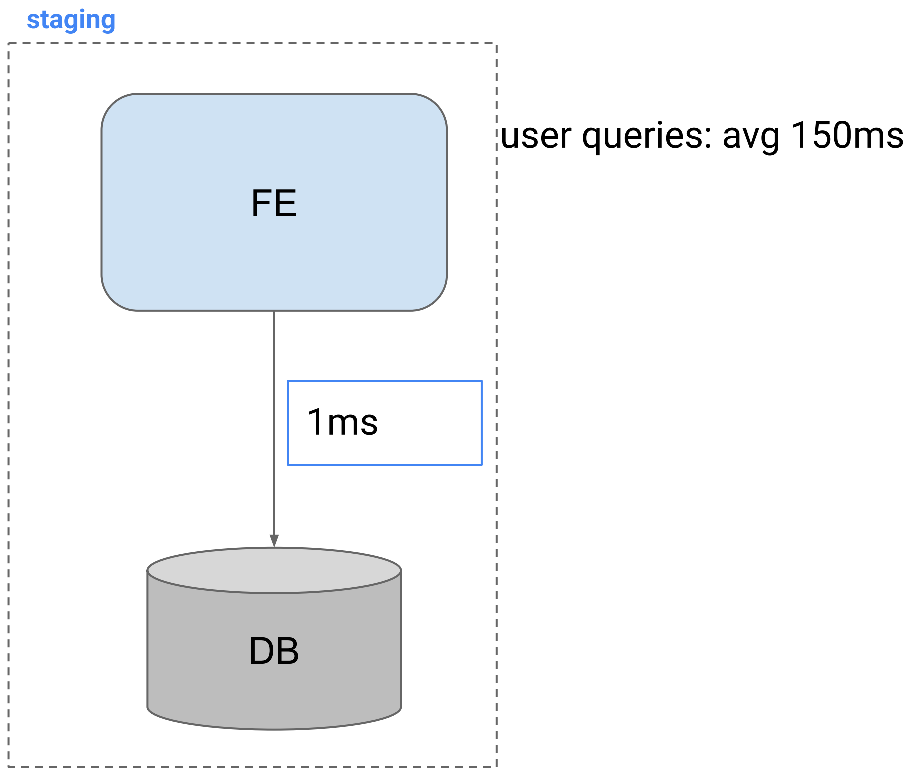
[longwire]

a toy problem to explain a system's emergent behavior

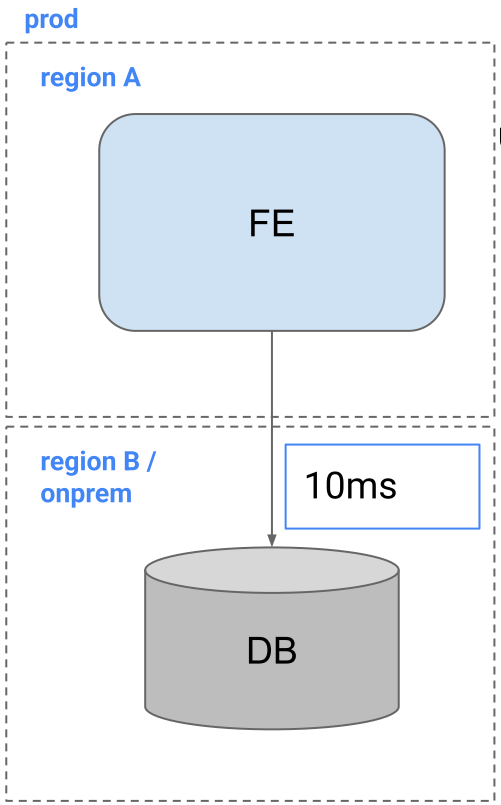
toy app - 10ms is blazing!



toy app - 150ms is nice and fast!



toy app - 15,000ms is ~~nice and fast~~ really bad



user queries: avg **15000ms** !

"what changed?"

"it's complicated" – could be many things!

- topology, distance (sadly, *c* is constant)
    - ⇒ [infra team, platform]
- prod db size (num records) vs dev/local test DBs
    - ⇒ [DBA team]
- maybe the code changed? maybe not
    - ⇒ [product devs, shared libraries, performance teams]

does [everyone] know about all changes, every time?

of course not.

underlying problem

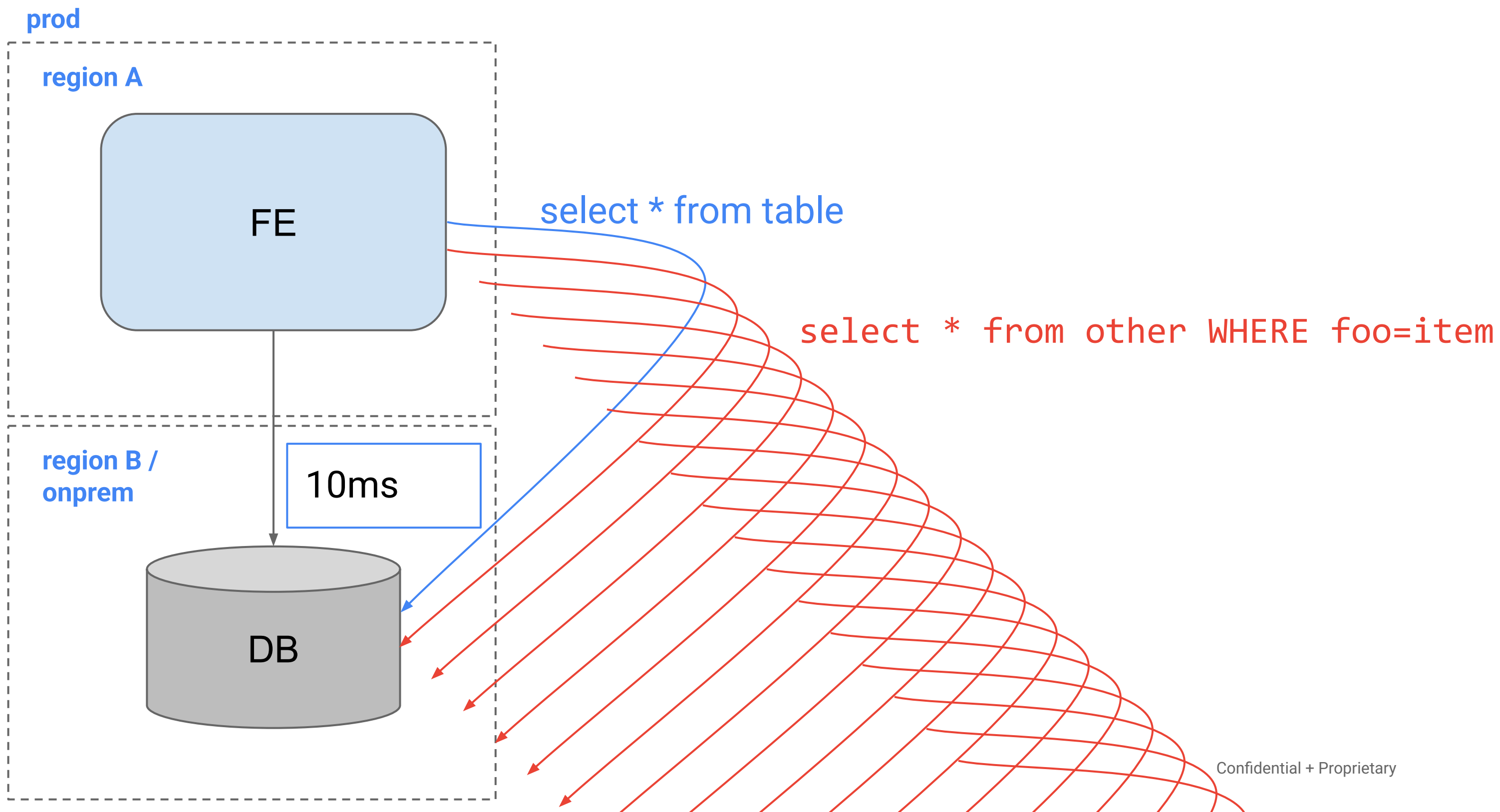
bad code?

```
items = select * from table

foreach item in items:

    select * from other WHERE foo=item
```

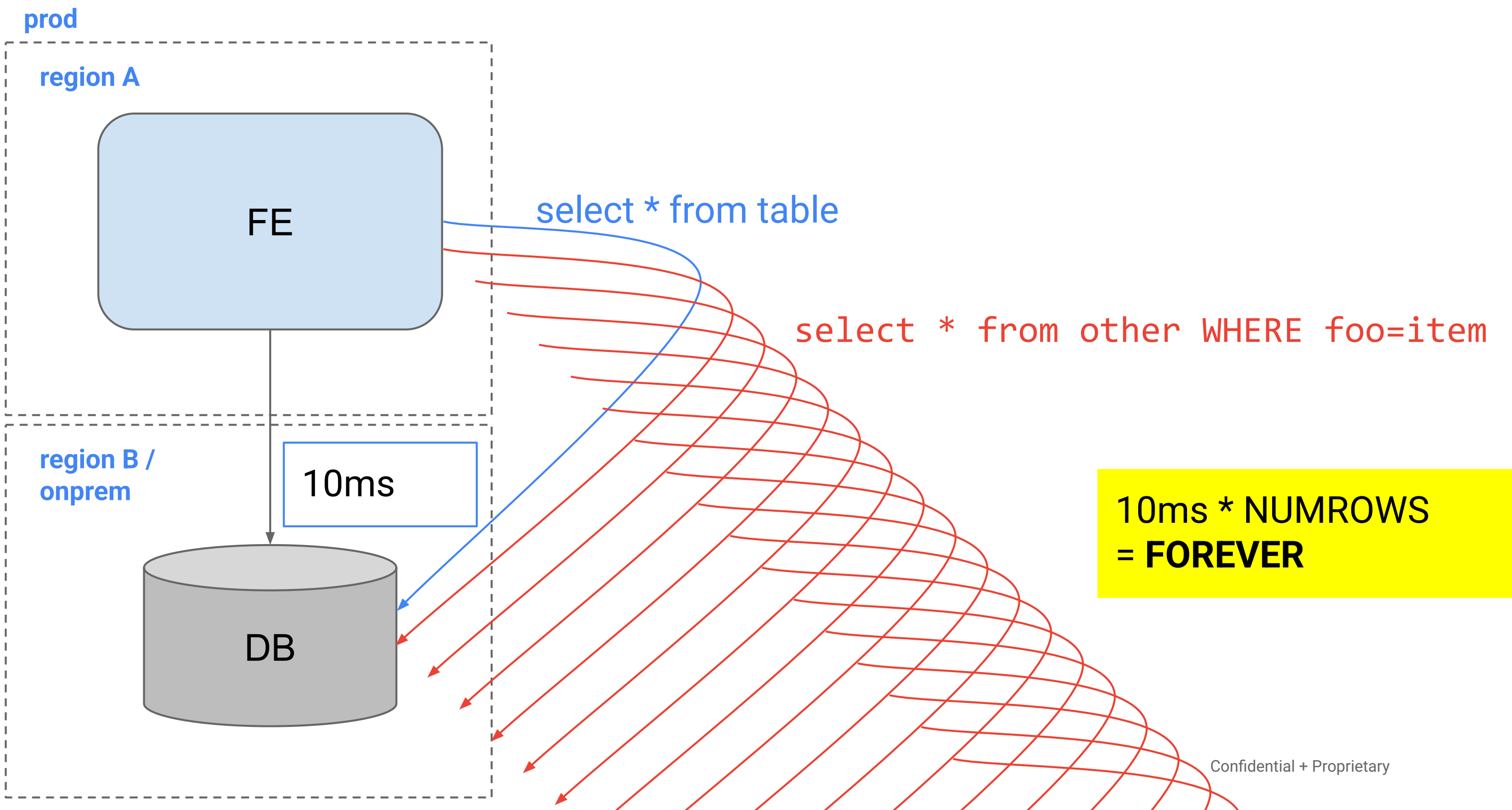
bad topology?

bad assumptions on recordset size?

toy app - 15,000ms is ~~nice and fast~~ really bad



toy app - 15,000ms is ~~nice and fast~~ really bad



10ms * NUMROWS
= **FOREVER**

infra **can't fix** bad code

but:

can we detect this?

can we mitigate this?

can we prevent this?

"shift reliability left"

what changes are relevant? whom to notify? **service catalog**

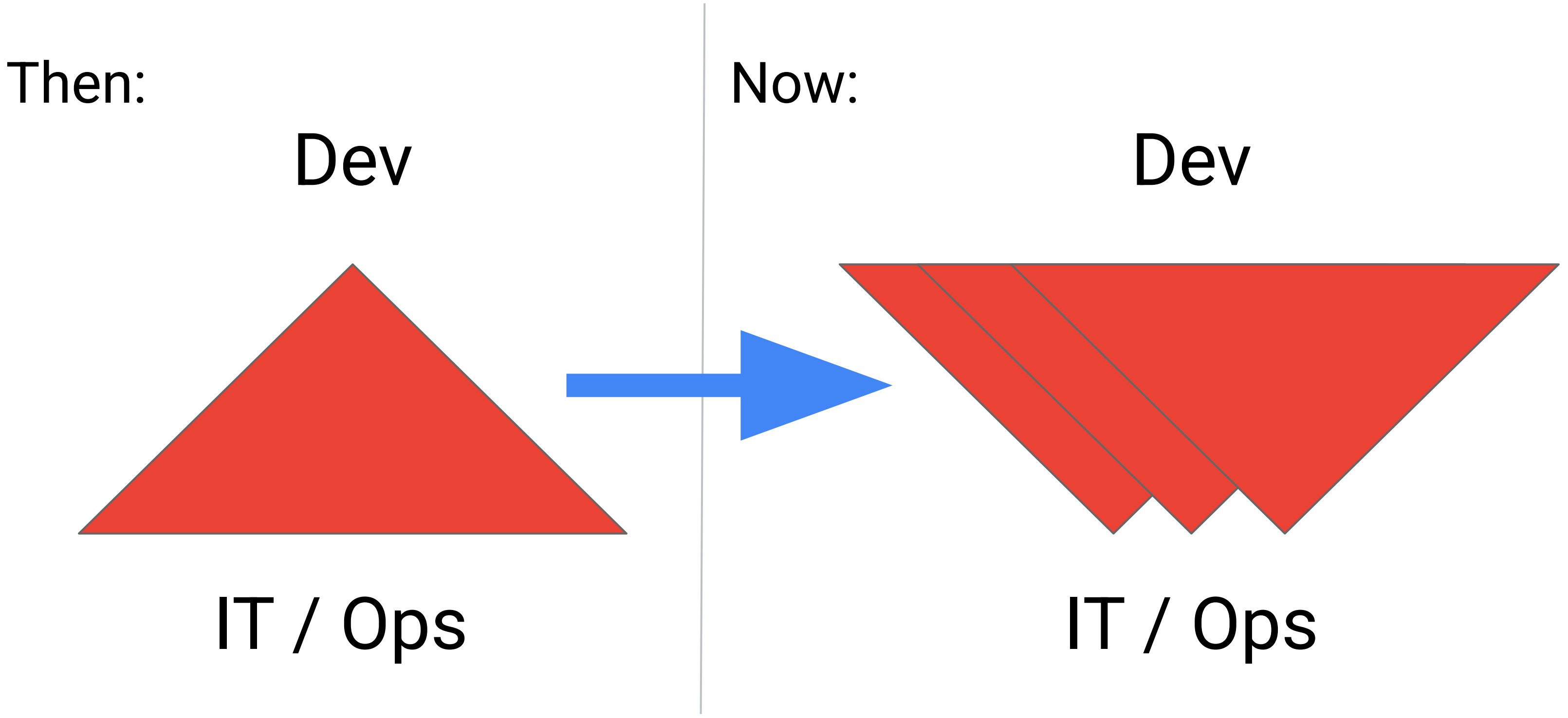
- detecting db size diffs can be done, but might **cross team boundaries** (dev, it, dba) without a way to bring them together under one "app"

"distance" can be mocked via **fault injection** (add latency)

is this a surprise dependency? **dependency description, detection, validation**

how do we know when a change is bad? **SLOs (+ loadtests)**

find these issues **sooner**, with less customer pain



Don't try to do this alone.

(eventually) most of the SRE / reliability work will be in **Dev, not in IT**. – build partnerships today!