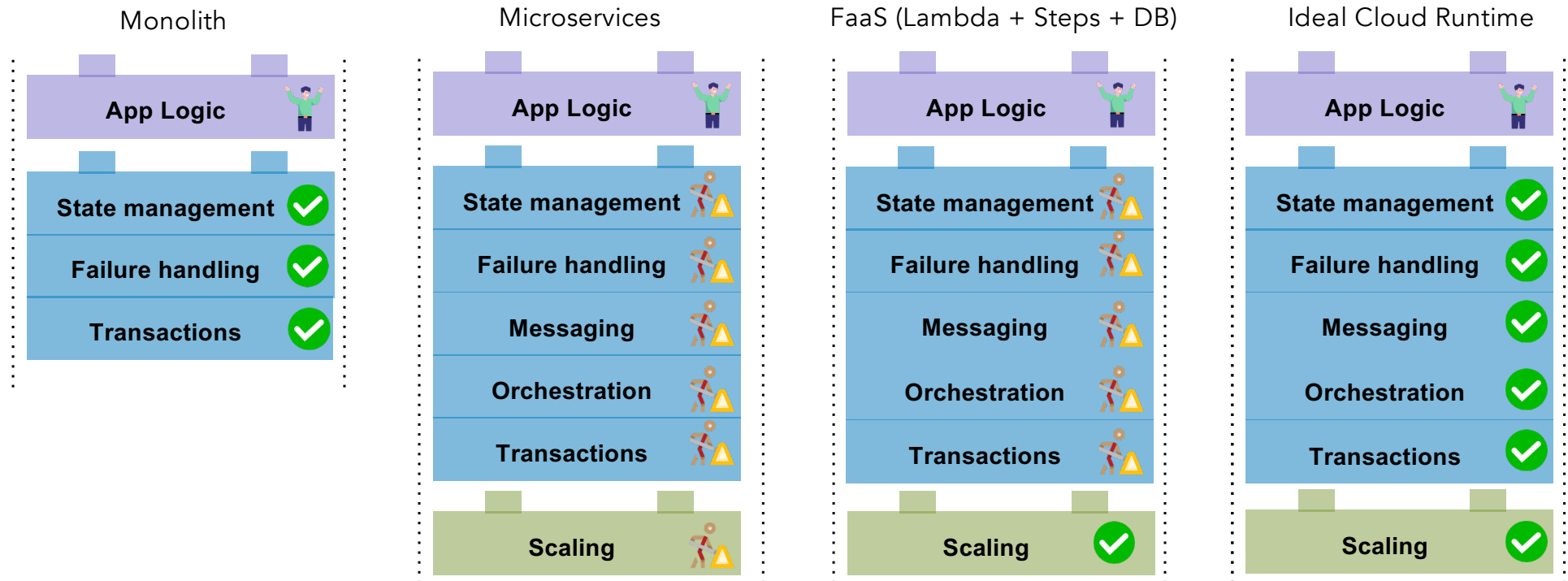


Open Problems & Future Work



Towards “ideal” runtimes for trans. Cloud apps



Transaction support

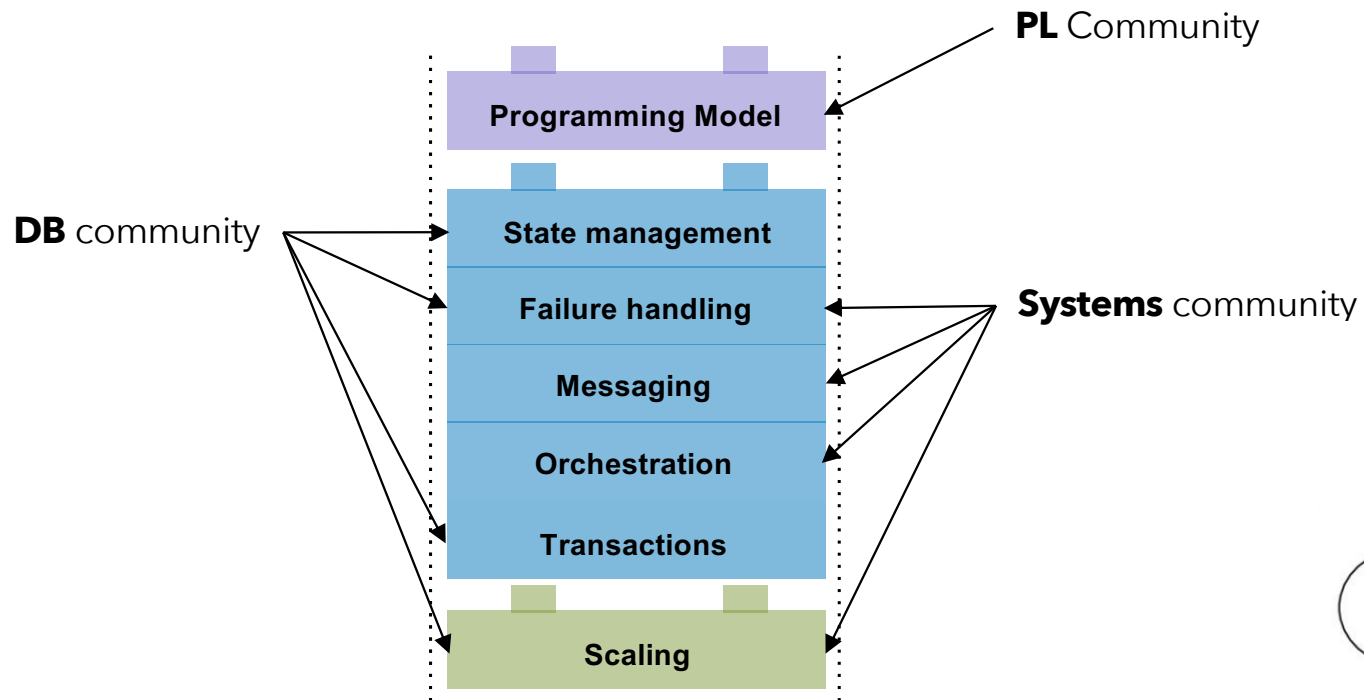


Sad Developer

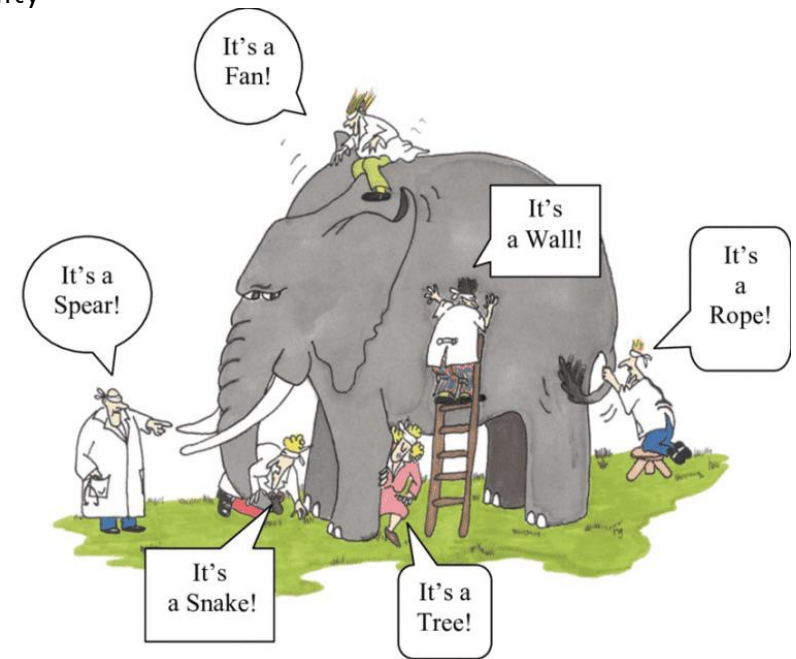


Happy Developer

Call to action



Interesting research lies in the combination of layers.

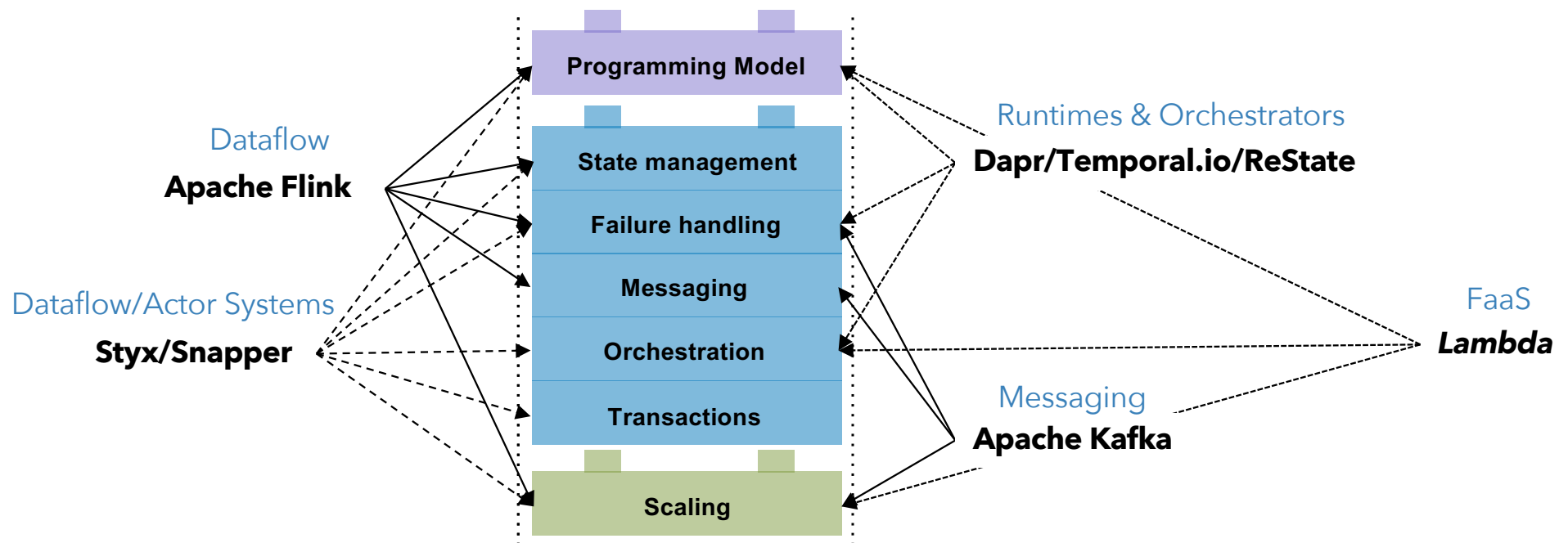


State, messaging & app logic cannot be treated separately.

"Theorem" 1: Unless a runtime fully controls state & messaging, exactly-once is impossible.
Main reason: state mutations are causally dependent on messages.

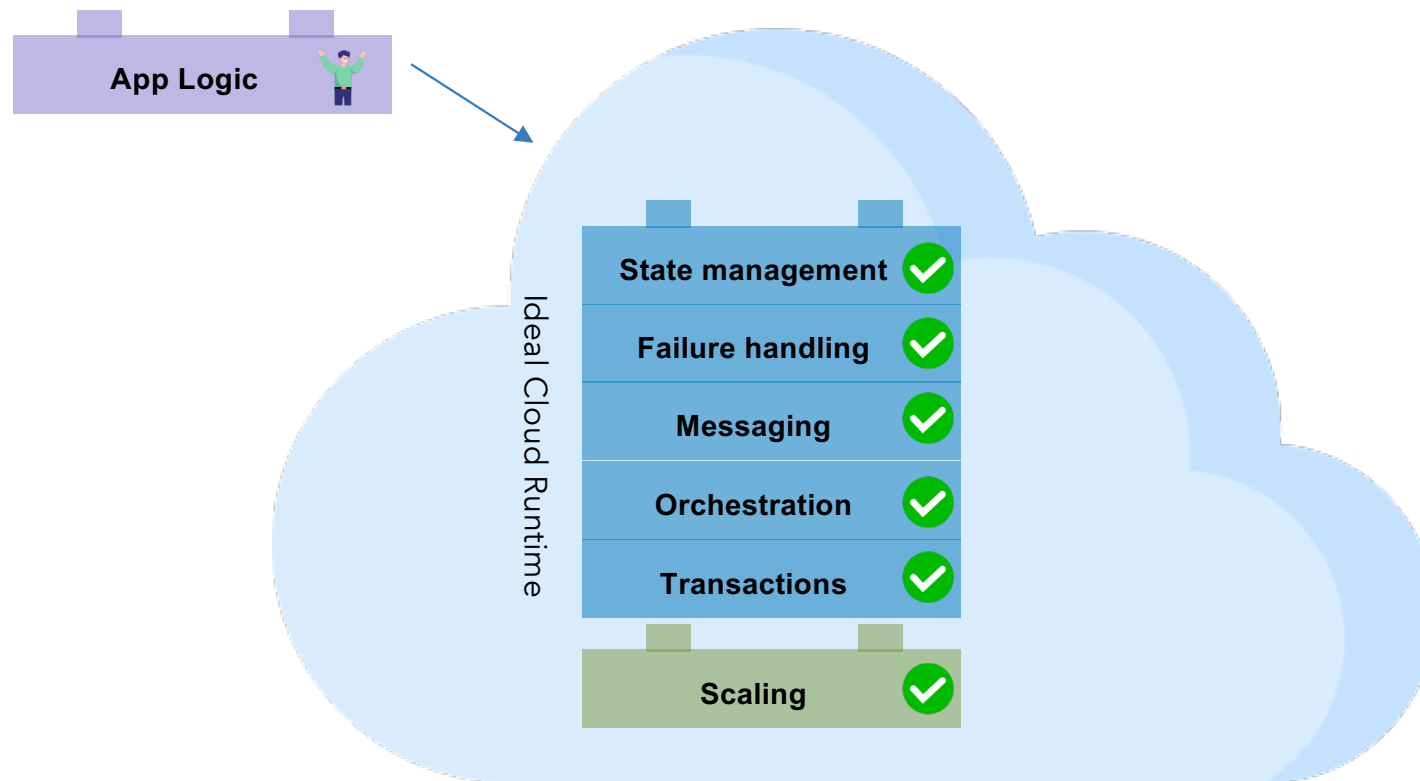
"Corollary" 1: Unless runtime guarantees exactly-once messaging, failures will leak to app logic.

"Corollary" 2: Unless runtime offers transaction primitives, transactions will "leak" to app logic.



Debate: **A)** systems implement the whole stack or... **B)** combine well-designed systems?
e.g., analytics community builds “composable” systems while Cloud providers build verticals.

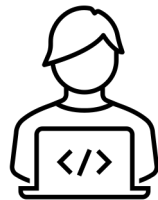
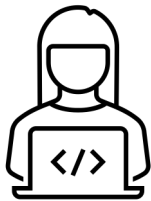
How would an ideal runtime look like?



BUT: do we really need a single runtime? Or multiple systems that agree on abstractions. What abstractions?

Before we finish...

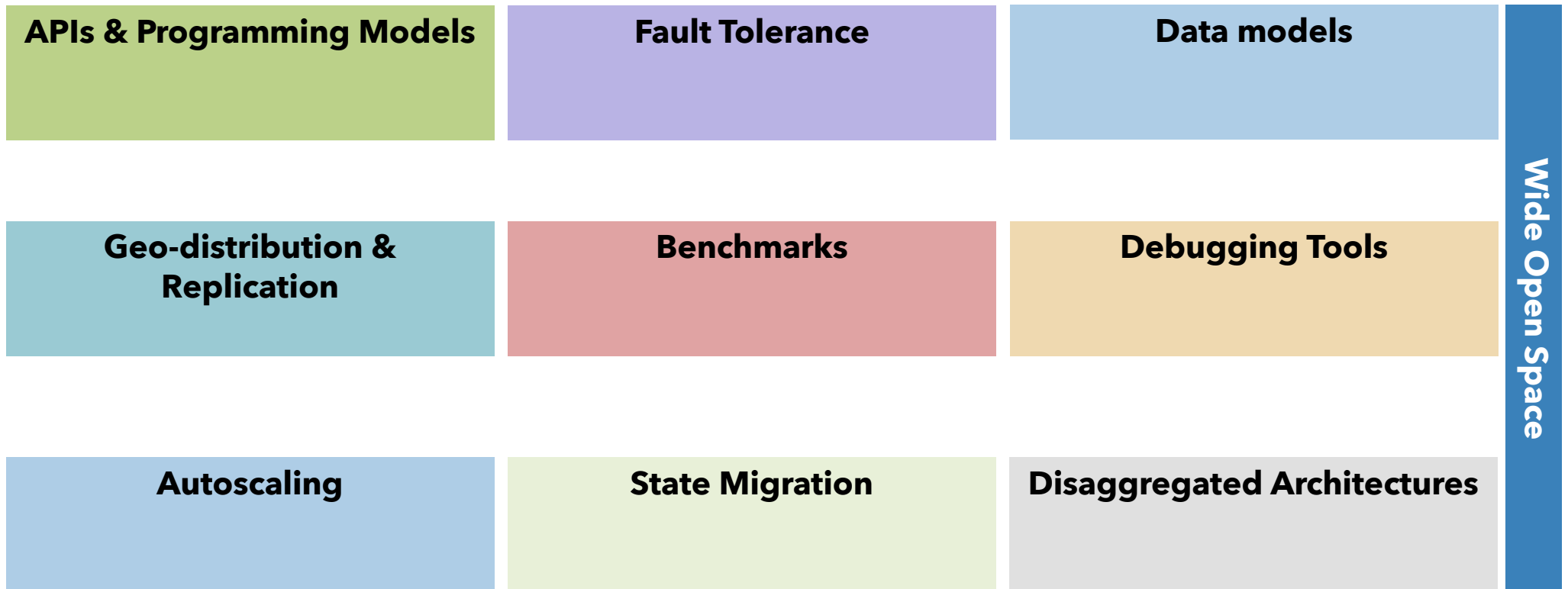
Any system advancement must be accompanied by appropriate software development support



Developers' perspective
cannot be jeopardized

Deployment, upgrading, and
deprecation of components
Data and message schemas changes
Debugging
Replayability (aka time travel)
Observability
Application logs

Open Problems: for any single color, there are multiple PhDs



Debate: Don't databases do all these?

Who are we?



**Rodrigo
Laigner**
(University of
Copenhagen)



**George
Christodoulou**
(TU Delft)



**Kyriakos
Psarakis**
(TU Delft)



**Asterios
Katsifodimos**
(TU Delft)



**Yongluan
Zhou**
(University of
Copenhagen)



Slides, and pointers:
<https://delftdata.github.io/tutorial-sigmod25/>

