Automated diagnosis without predictability is a recipe for failure

Raja Sambasivan

Greg Ganger

PARALLEL DATA LABORATORY

Carnegie Mellon University

What is important for design? (I)

The designer usually finds himself floundering in a sea of possibilities, unclear about how one choice will limit his freedom...or affect the size and performance of the entire system.

--Butler Lampson

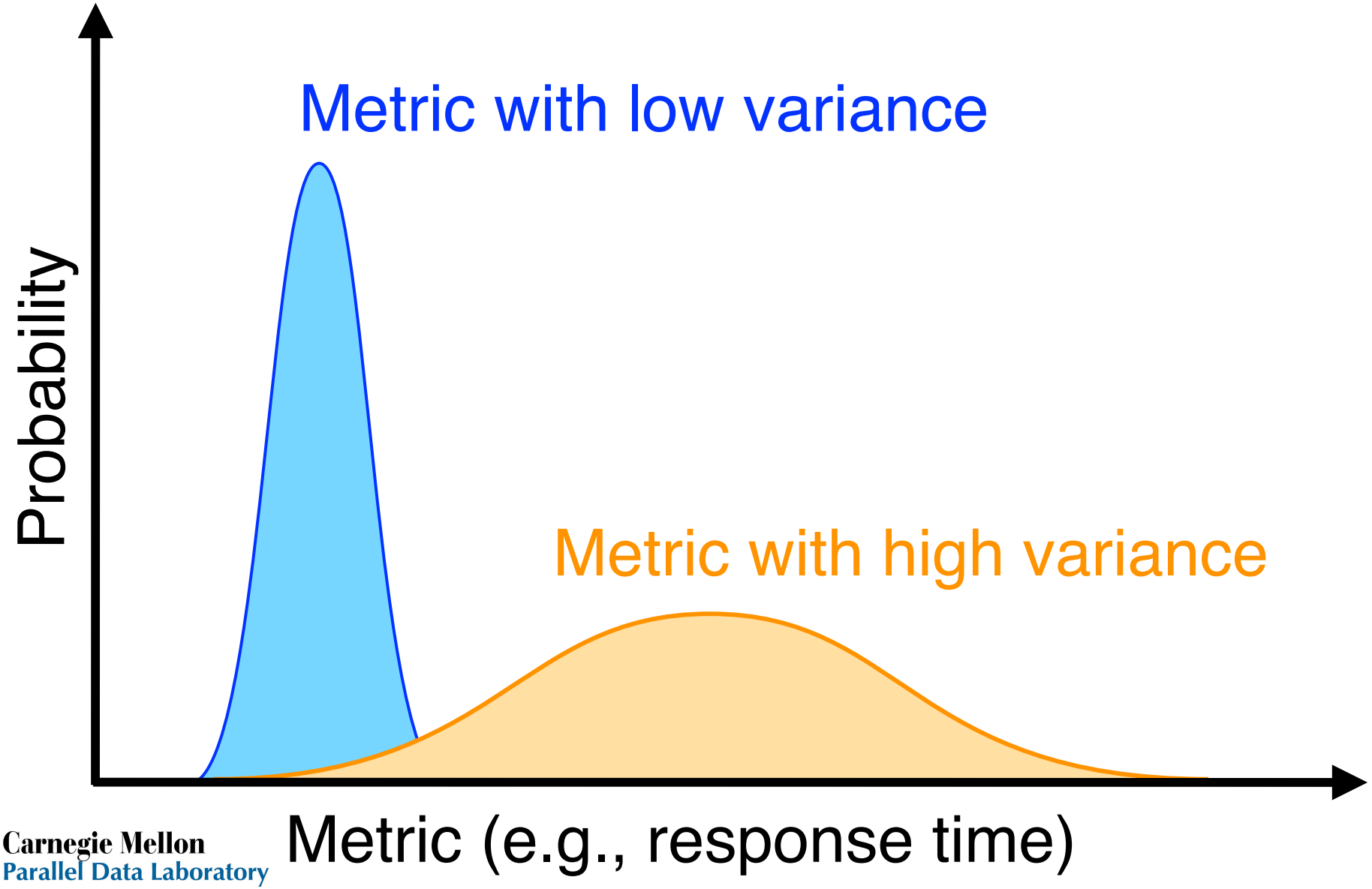
What is important for design? (II)

- Commonly agreed upon axes:
  - Correctness, perf., reliability, power...

- This talk: Predictability also important
  - Especially for large distributed systems
  - Affects ability to optimize other axes
  - Affects manageability

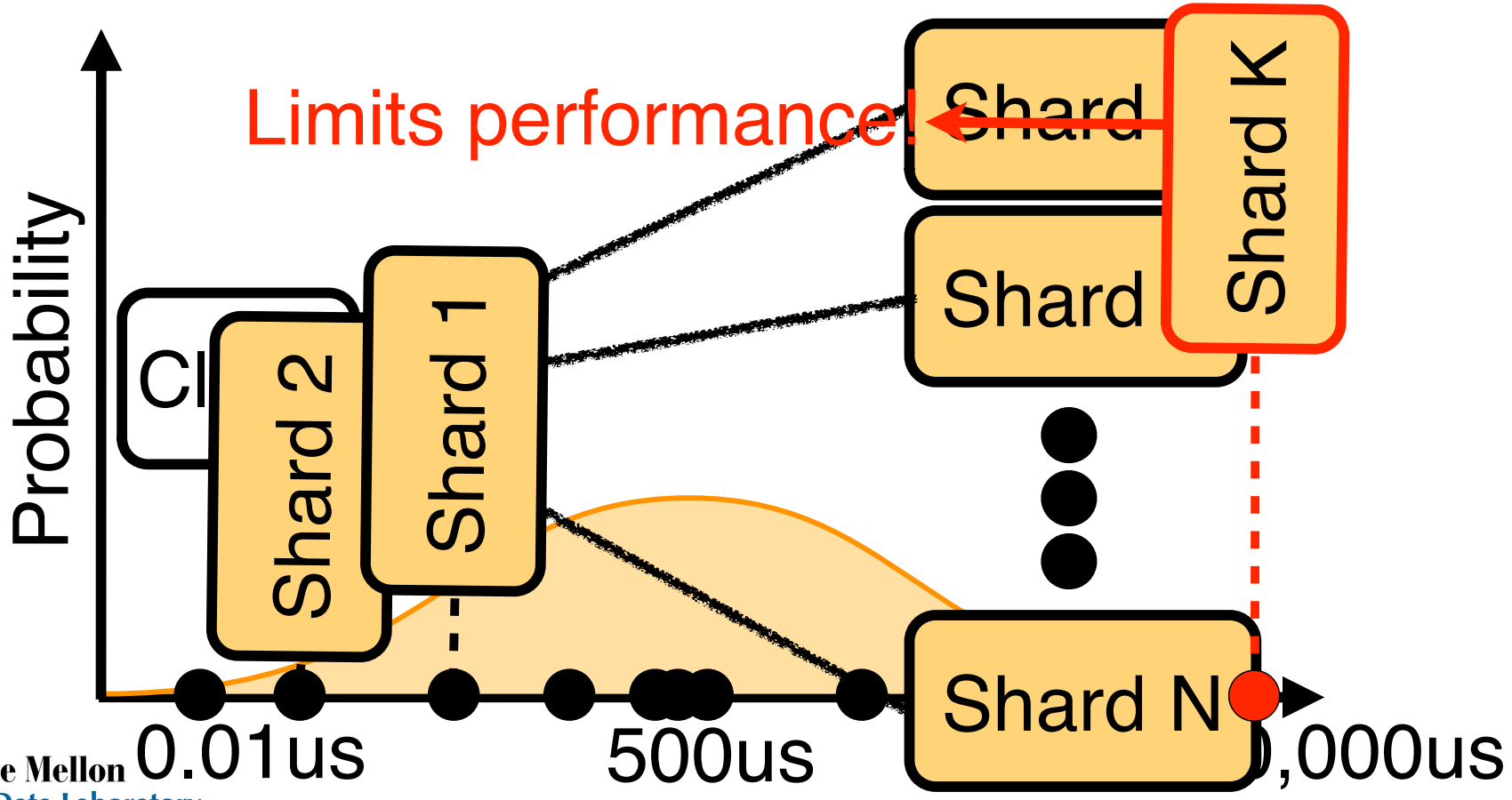
Achieving it will require hard work

Predictability means low variance



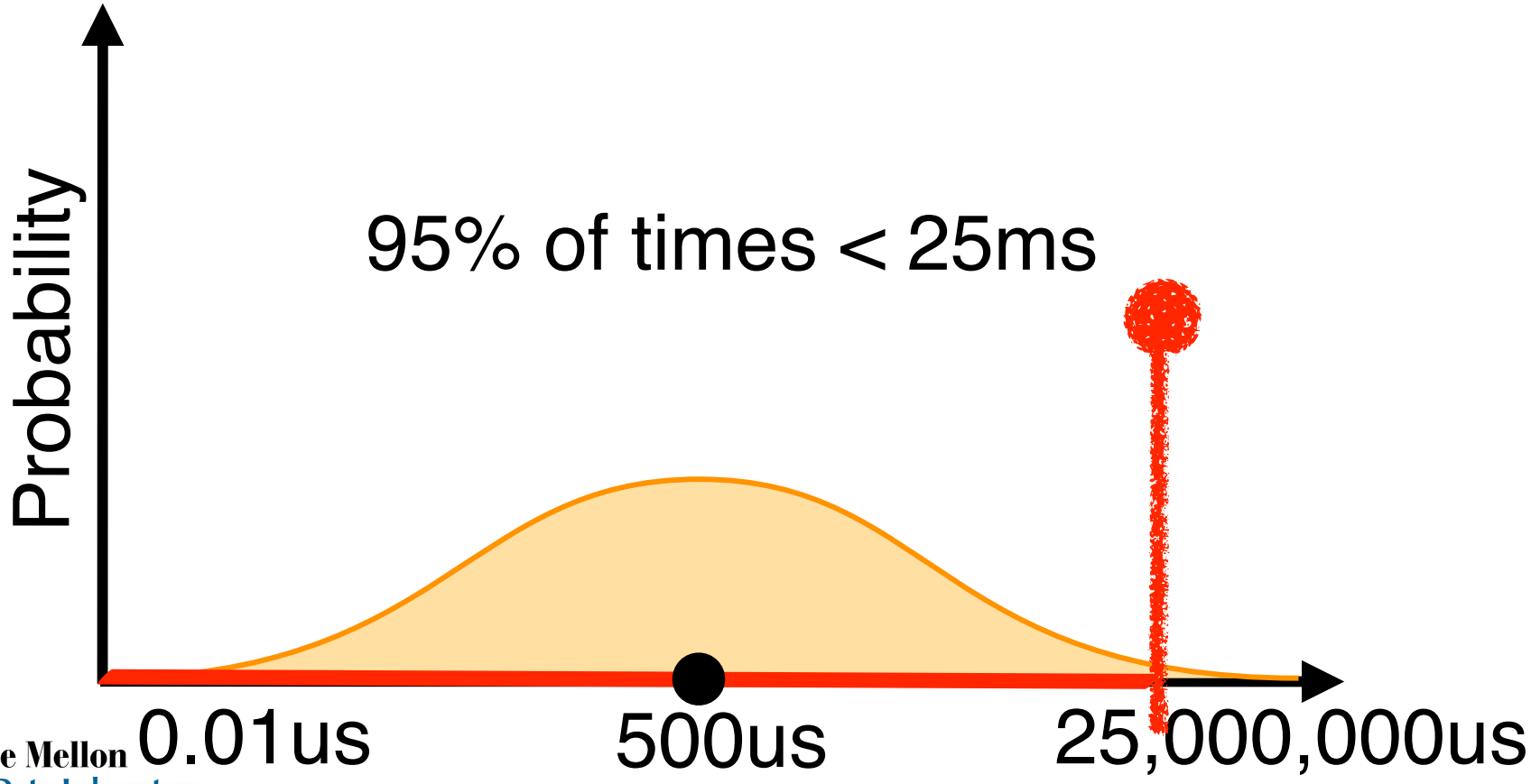
Predictability is important (I)

- For performance and efficiency of very large-scale computing workloads



Predictability is important (II)

- For setting & enforcing SLAs and QoS

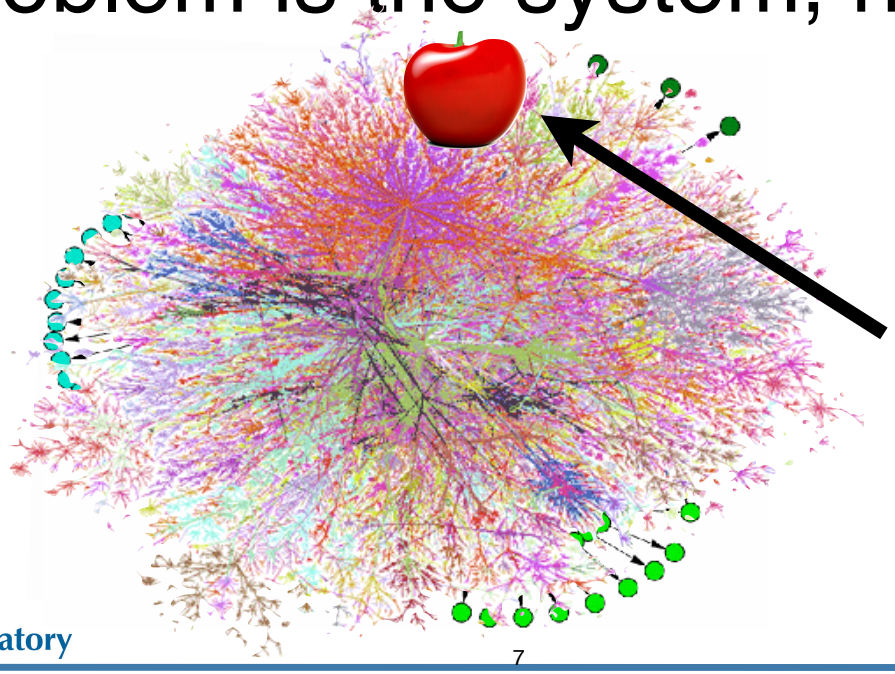


Predictability is important (III)

- For success of automated diagnosis

Magpie	NetMedic	Pinpoint	Spectroscope	...

- Automation tools not used in production
  - Problem is the system, not the tools...

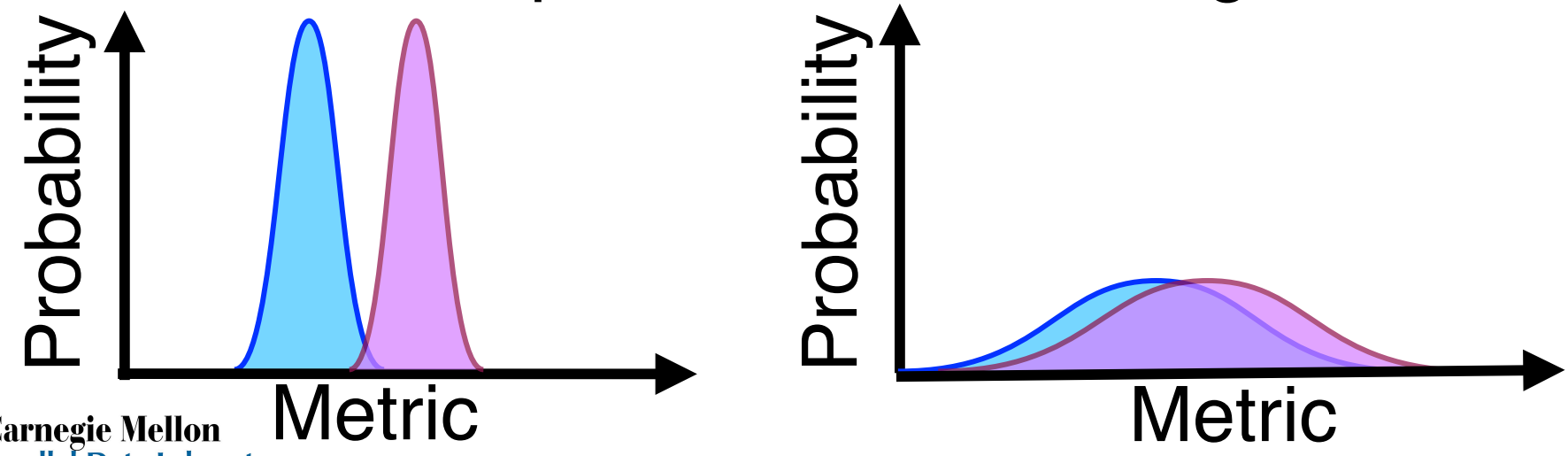


Cherry-on-top approach to automation

The issue with diagnosis tools

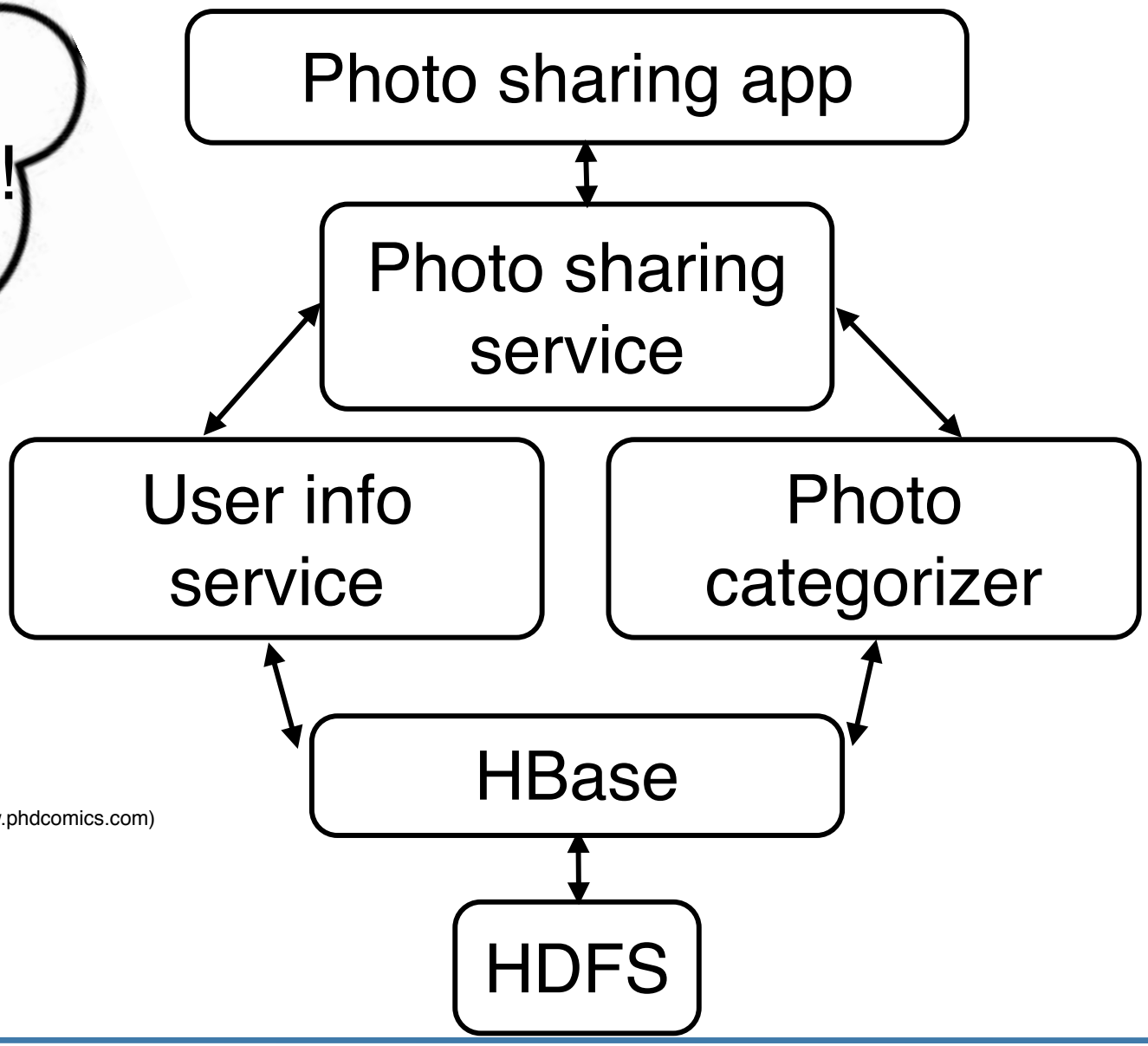
- Use deviations in metrics to localize perf. problems to a few likely culprits

- Increased variance in metrics yields...
  - more false positives or false negatives



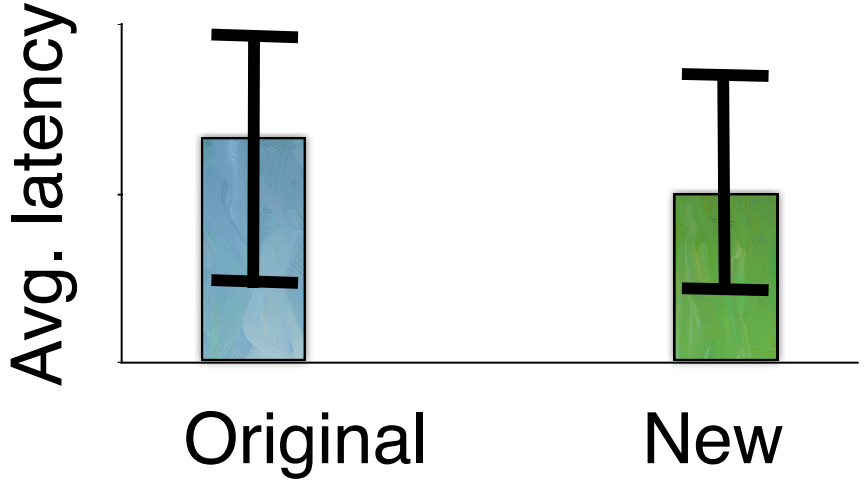
Grad student vs. diagnosis tool

Image from Piled Higher & Deeper (www.phdcomics.com)



The easy life of a grad student

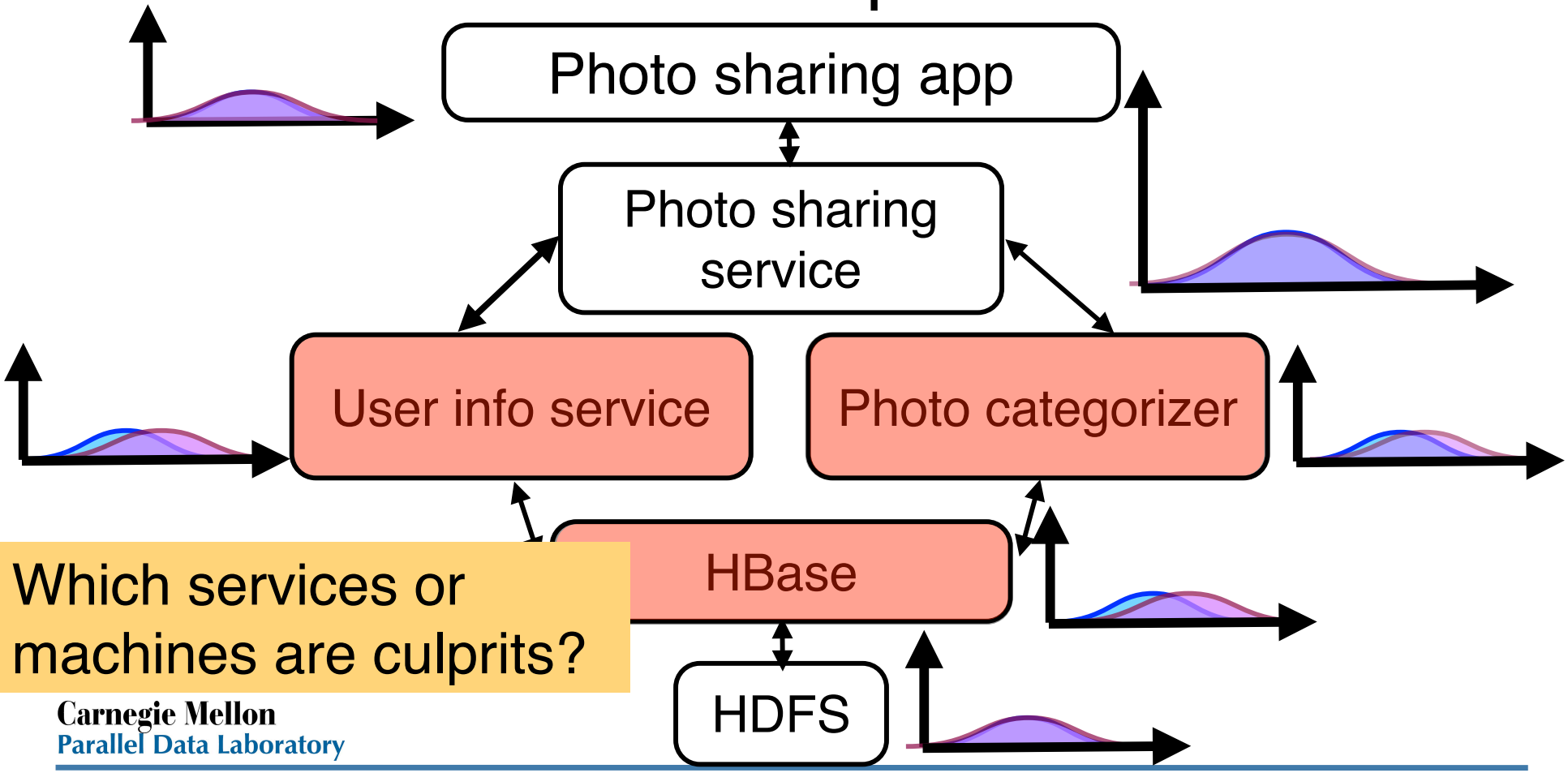
- Student's research idea improves avg. response time by 25%



- But, variance too high for confidence:
  - Can run more experiments
  - Can ignore variance ;)

The hard life of a diagnosis tool

- Tool must find code responsible for an increase in mean response time



Which services or machines are culprits?

The hard life of a diagnosis tool

- Can't ignore underlying variance
  - Will yield more false positives/negatives
  - FPs: Tool is misleading developer
  - FNs: Tool is not useful

- Can't 're-run' problem in production

Utility of diagnosis tools limited unless system has predictablity properties

How to increase predictability?

- Variance is treated as a key metric
  - Must be measured during dev/testing
- Devs must ID high-variance sources
  - Must *explicitly* decide on tradeoffs
- Must rigorously isolate irreducible variance sources from rest of system

Grokking variance

(Developer)

VarianceFinder

Identifies similar behaviour w/high variance and localizes it

Variance nomenclature

Helps devs reason about possible actions

The three I's of variance

- **Inadvertent** variance is unintentional
  - E.g., spaghetti code or a bug — Reduce
- **Intrinsic** variance is fundamental
  - E.g., Hard disks/non-networks — Isolate
- **Intentional** variance result of tradeoff
  - E.g., low-latency schedulers — Isolate

Open questions about variance

- How much needs to be reduced?
- If intrinsic variance is significant are hardware changes needed?
- If intentional variance is significant do we need to re-evaluate tradeoffs?

Conclusion

- Predictability important for systems
- No answer but hard work during system development and testing
- Need tools/techniques to help developers increase predictability