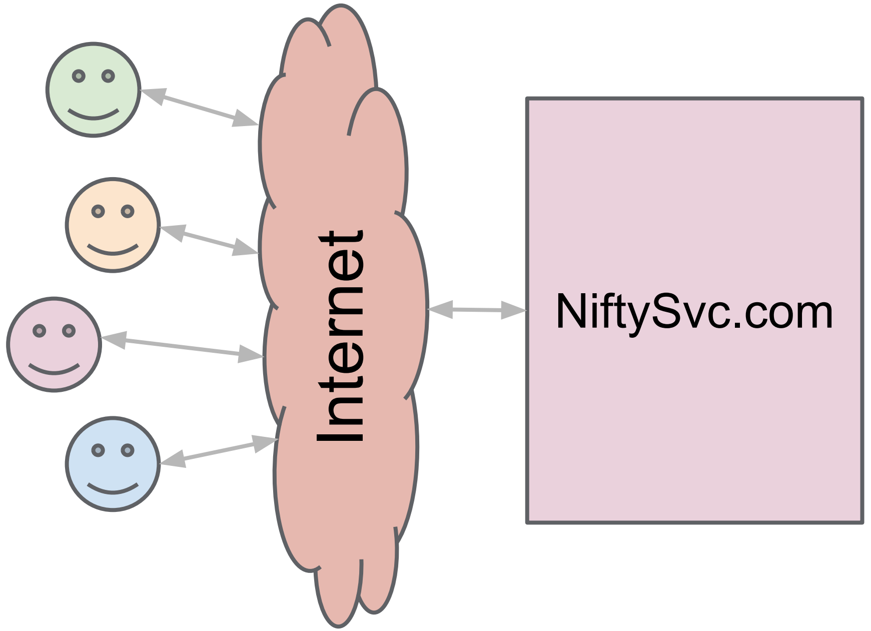
Inferring the Network Latency Requirements of Cloud Tenants

Jeff Mogul (Google)
Ramana Rao Kompella (Google + Purdue)

HotOS XV
May 2015

Users care about latency



"Systems that respond to user actions [within 100ms] feel more fluid and natural to users"

-- J Dean & L Barroso, *The Tail at Scale*, CACM 56(2)

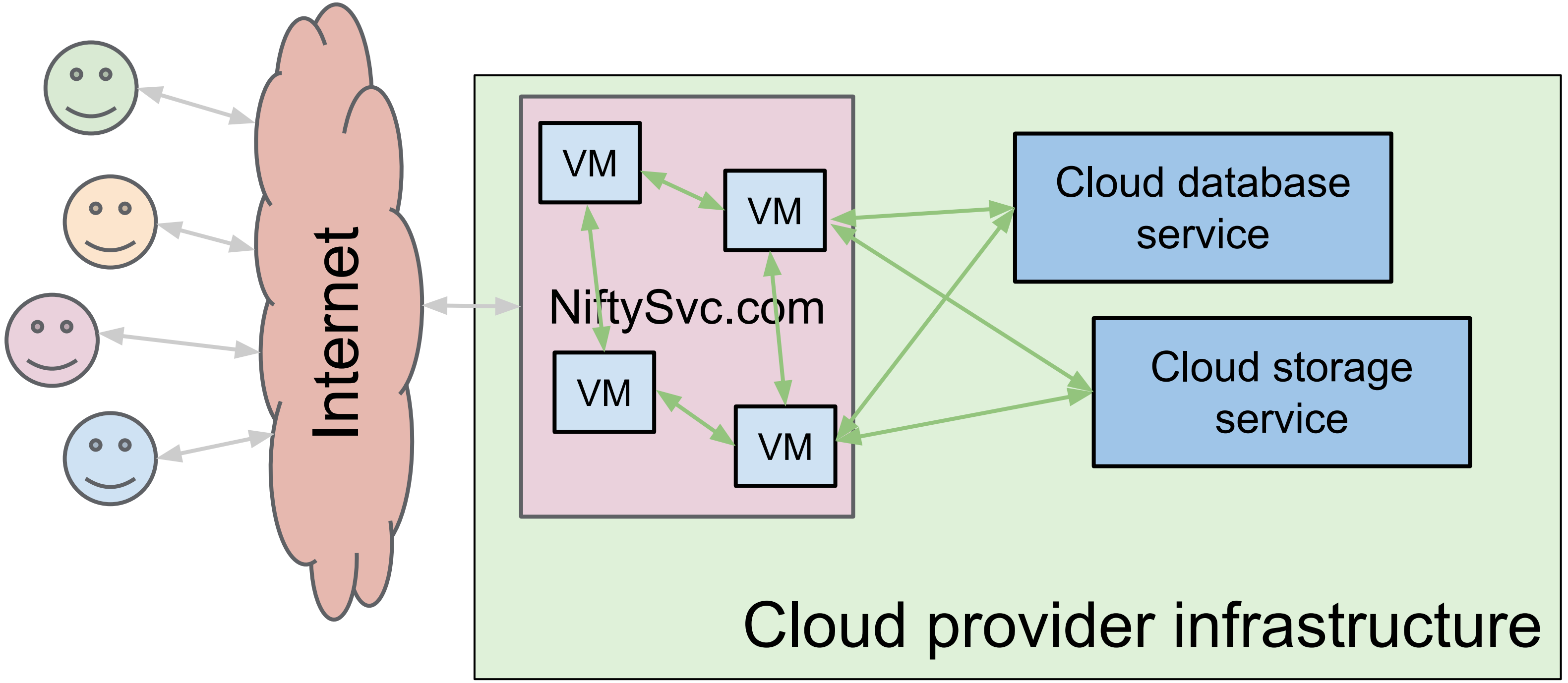
"[Amazon's] services have stringent latency requirements … measured at the 99.9th percentile"

-- G DeCandia *et al.*, *Dynamo …*, SOSP 2007

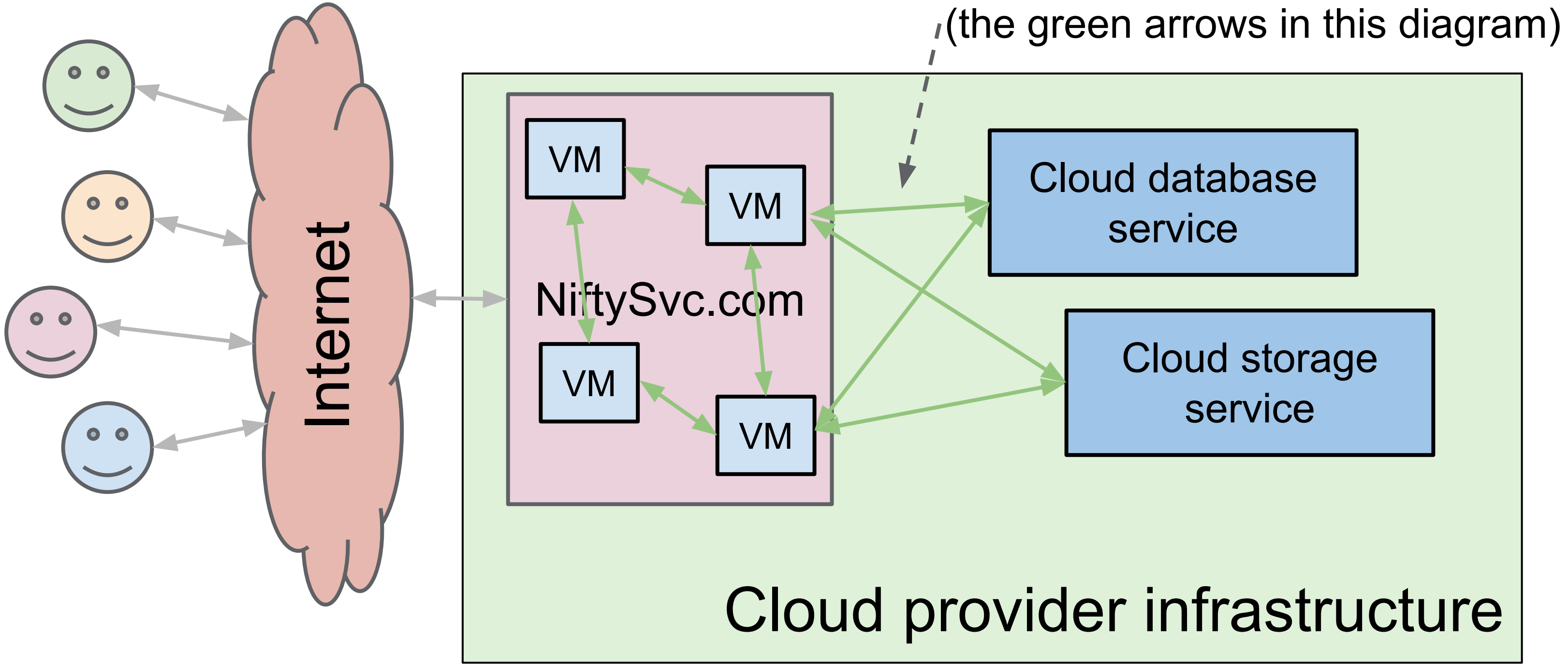
"Amazon found every 100ms of latency cost them 1% in sales"

-- (various non-original sources)

So, cloud tenants might care about intra-cloud latency



So, cloud tenants might care about intra-cloud latency



How much do tenants care about intra-cloud latency?

How much do tenants care about intra-cloud latency?

- Providers don’t generally know

How much do tenants care about intra-cloud latency?

- Providers don’t generally know
- Tenants don’t generally know

So why should we care?

So why should we care?

- Our intuition: intra-cloud latency actually does matter to tenants
- So a cloud provider with better latency will have happier tenants
    - that is, more tenants, and/or tenants who are willing to pay more
- But building infrastructure to support low latency isn’t cheap
    - especially for low tail latency, which tends to require low utilization

Goal of our work: how much does latency matter?

How sensitive is a given cloud application (or VM) to the underlying intra-cloud network latency?

Specific focus of our work:

- Within-region vs. WAN latency
  - Intuition: local latency is easier to vary per-tenant
- Techniques requiring little or no help from the tenants
  - Intuition: tenant developers don't want to be bothered
- Not on how much bandwidth an application uses
  - previous work has looked at inferring cloud bandwidth needs
    - Proteus (Xie *et al.*, SIGCOMM '12); Cicada (LaCurts *et al.*, HotCloud '14)

One-slide summary

Our system will:

- Inject network latency using known patterns (“PN codes”)
- Measure application-level metrics
- Use correlation to detect how much latency affects these metrics

What could a provider do with this information?

Balance resource allocations between tenants:

- Use admission control, to avoid over-utilizing the network
- Place VMs to improve locality or reduce interference
  - as in Oktopus (Ballani *et al.* SIGCOMM '11) and Silo (Jang *et al.* '13)
- Rate-limit latency-insensitive tenant VMs
- Use DSCP settings to shift load between switch queues
- Adjust relative prices of VMs and guarantees for BW & latency

What could a provider do with this information?

And:

- Plan infrastructure upgrades/expansions
- Help tenants understand which provider better suits their needs

Why is inferring latency harder than inferring bandwidth?

Basic technique for inferring bandwidth needs:

- Temporarily turn off rate limiting
- Measure how much bandwidth the application (VM) uses
- Infer future needs from (measured) past behavior

It's harder to apply this method to latency:

- How do you measure "how much latency the VM uses"?

How bad is it?

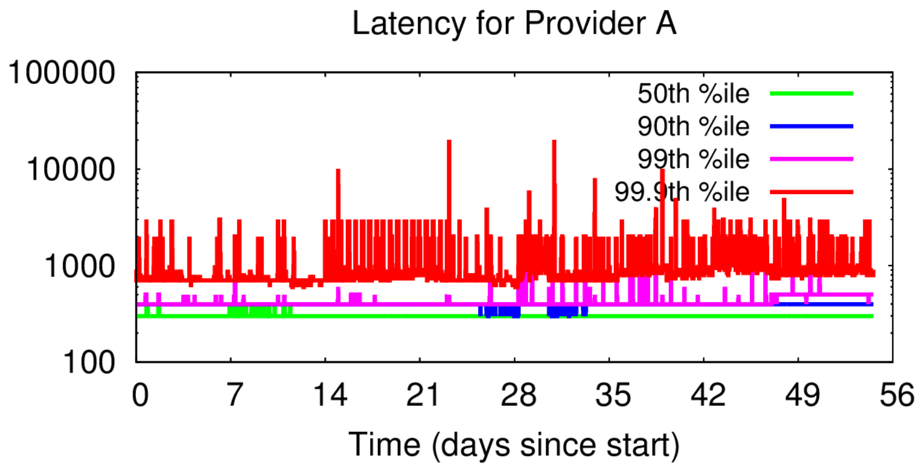
We did a simple study to quantify latency variability

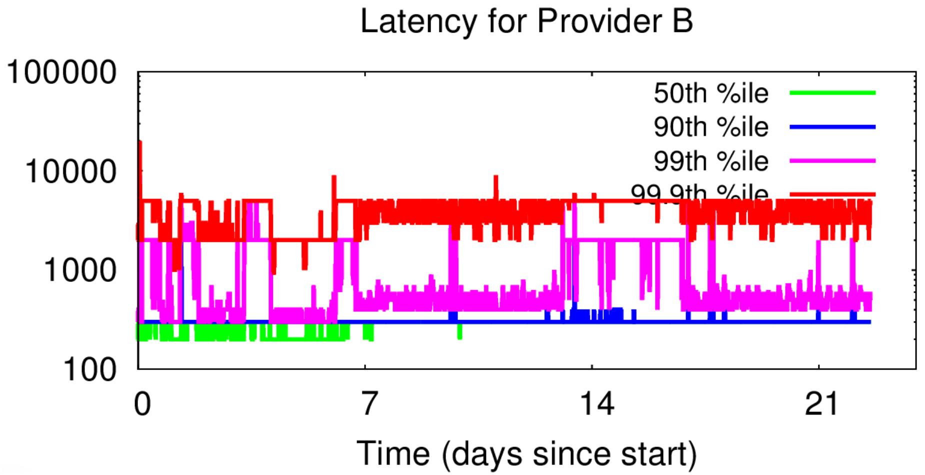
- “simple” means “WARNING: this is bad science”
- Do not attempt to compare providers using this data. Please.

Methodology:

- Buy two cheap VMs in each of several providers
- Run netperf TCP_RR for 60 seconds every 15 minutes
    - netperf reports latency histograms (in a weird way)

Latency results (see warnings on previous slide)





- Latencies can be quite large (at 99th %ile or 99.9th %ile)
- Latencies vary over both short and long time scales
- Latencies *seem* to vary between providers (**WARNING: NOT ACTUAL SCIENCE!**)

Latencies in microseconds

Our approach: Use correlation for latency inference

Goals:

- Infer a causal relationship between network latency and application-level “Service-Level Objective” (SLO) for latency
- Find threshold below which better network latency doesn’t help
- Understand how well the application tolerates latency increases

Approach:

- Measure network latency variations
- Measure SLO effects
- Correlate! Statistics!
    - As in Cohen *et al.*, “Correlating Instrumentation Data to System States”, OSDI ‘04

The hard parts:

- Measuring network latency variation
- Measuring SLO variation
- Doing this for a complex multi-VM application
- Doing this without (much) help from the tenants
- Be robust to various and unknown sources of noise
    - e.g., network traffic, application behavior, storage service latency variation, workload variations

Measuring network latency variation

Possible approaches:

- Exploit natural variation?
  - Only works if there is enough natural variation [maybe]
  - How can a VMM actually measure the latency seen by a VM?
    - add timestamps to packets? But what if there is no rapid response?
    - snoop on TCP headers? But what if no TCP? Or if VM uses IPSEC?
- Inject our own variation?
  - We can do it whenever we want (e.g., only for selected VMs)
  - No need to match up requests and responses
  - We control the frequency and amplitude
  - Relatively easy to do at the VMM layer

Measuring network latency variation

Possible approaches:

- Exploit natural variation?
  - Only works if there is enough natural variation [maybe]
  - How can a VMM actually measure the latency seen by a VM?
    - add timestamps to packets? But what if there is no rapid response?
    - snoop on TCP headers? But what if no TCP? Or if VM uses IPSEC?
- Inject our own variation? ✓
  - We can do it whenever we want (e.g., only for selected VMs)
  - No need to match up requests and responses
  - We control the frequency and amplitude
  - Relatively easy to do at the VMM layer

Measuring SLO variation

What is an application-level SLO?

Examples:

- 99.9% of HTTP requests complete within 500 msec
- Handle at least 1000 requests/sec at least 99% of the time

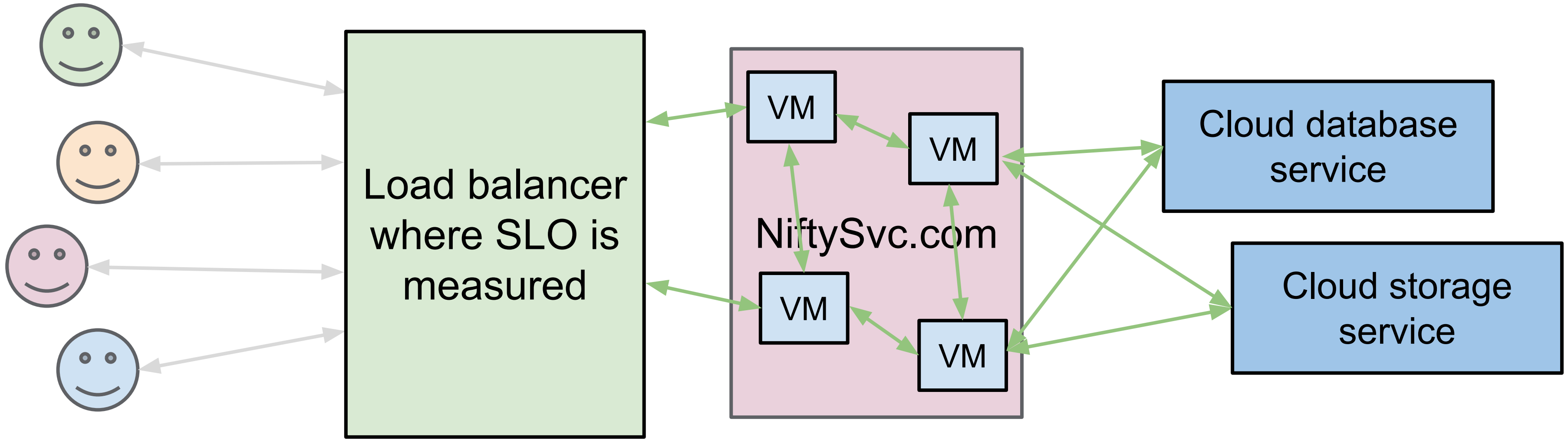
Measuring SLO variation (without help from tenants)

Options:

- Assume tenant uses provider-supplied load balancer
  - but: not all tenants use one, or they use Direct Server Return
- Measure Internet use; assume better results lead to more use
  - not always a good indicator
- VMM assumes HLT/MWAIT means VM is waiting for network
  - doesn't work if there's enough parallelism to keep cores busy
- Hook into cloud-monitoring tools (e.g., Tracelytics or AppDynamics)
  - not everyone uses these

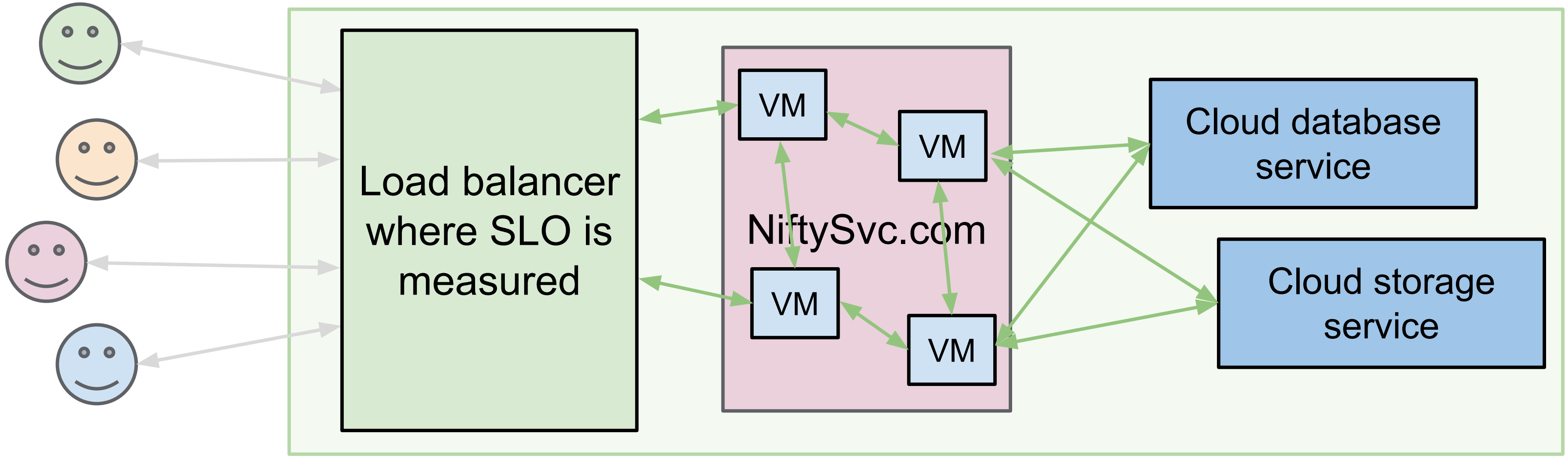
We're still trying to figure out which of these we can use (maybe several?)

How do we know which paths are latency-sensitive?



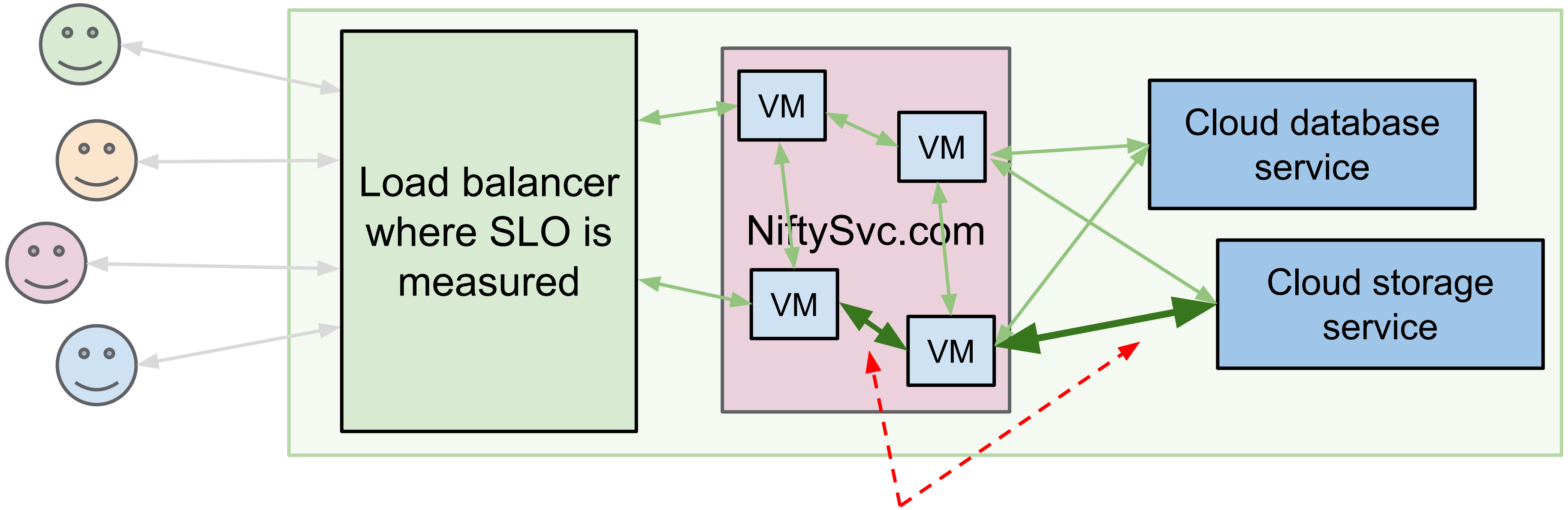
Which green arrows are the ones whose network latency affects the overall SLO?

How do we know which paths are latency-sensitive?



Which green arrows are the ones whose network latency affects the overall SLO? (within the cloud provider's domain)

How do we know which paths are latency-sensitive?

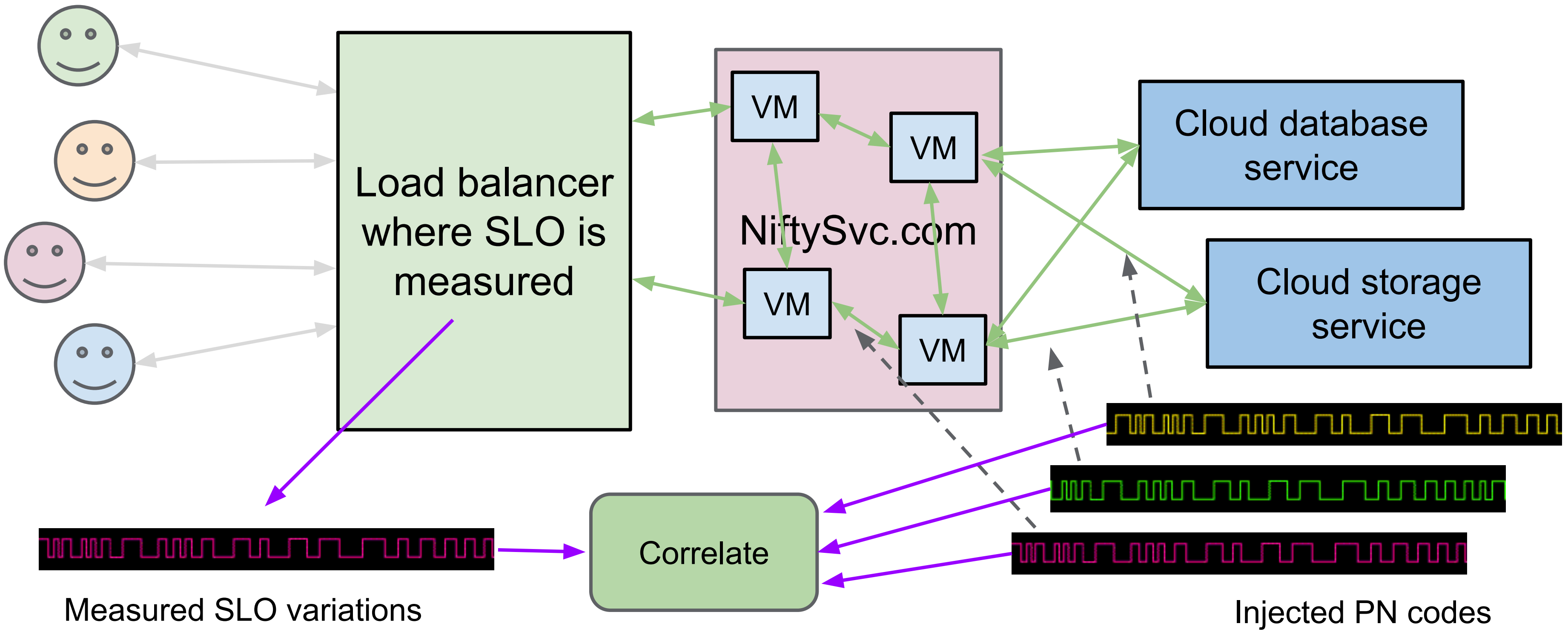


For example: these paths might matter the most

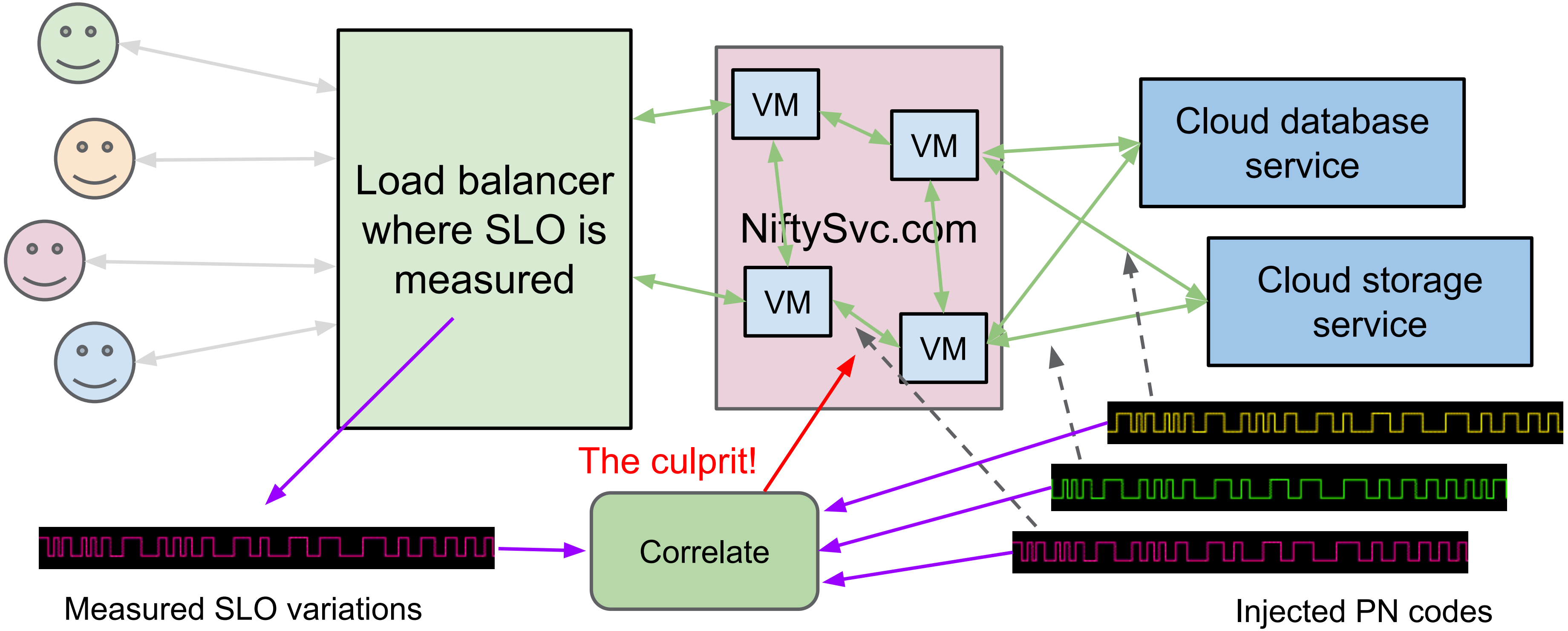
Insight: inject latency variation using pseudo-noise code

- Inject latency using time-varying pattern representing bit-sequence
  - added latency = 1
  - no added latency = 0
- Choose pattern using pseudo-noise (PN) codes
  - A set of PN codes can be chosen to be “highly orthogonal”
    - i.e., minimal correlation between pairs of PN codes
  - Assign one PN code to each latency path (i.e., each green arrow)
- Correlate time-varying SLO measurements with *known* PN codes
  - This is what GPS receivers do (more or less)
- This should allow us to:
  - Understand which network paths actually matter
  - Separate effects of network latencies from various noise sources

Cartoon version of PN codes in action



Cartoon version of PN codes in action



Characteristics of PN codes

We will have to experiment to find out the necessary

- Amplitude: how much time packets are delayed for
- Frequency: how long a “1” or “0” bit lasts
- Duration: how many bits in a PN code

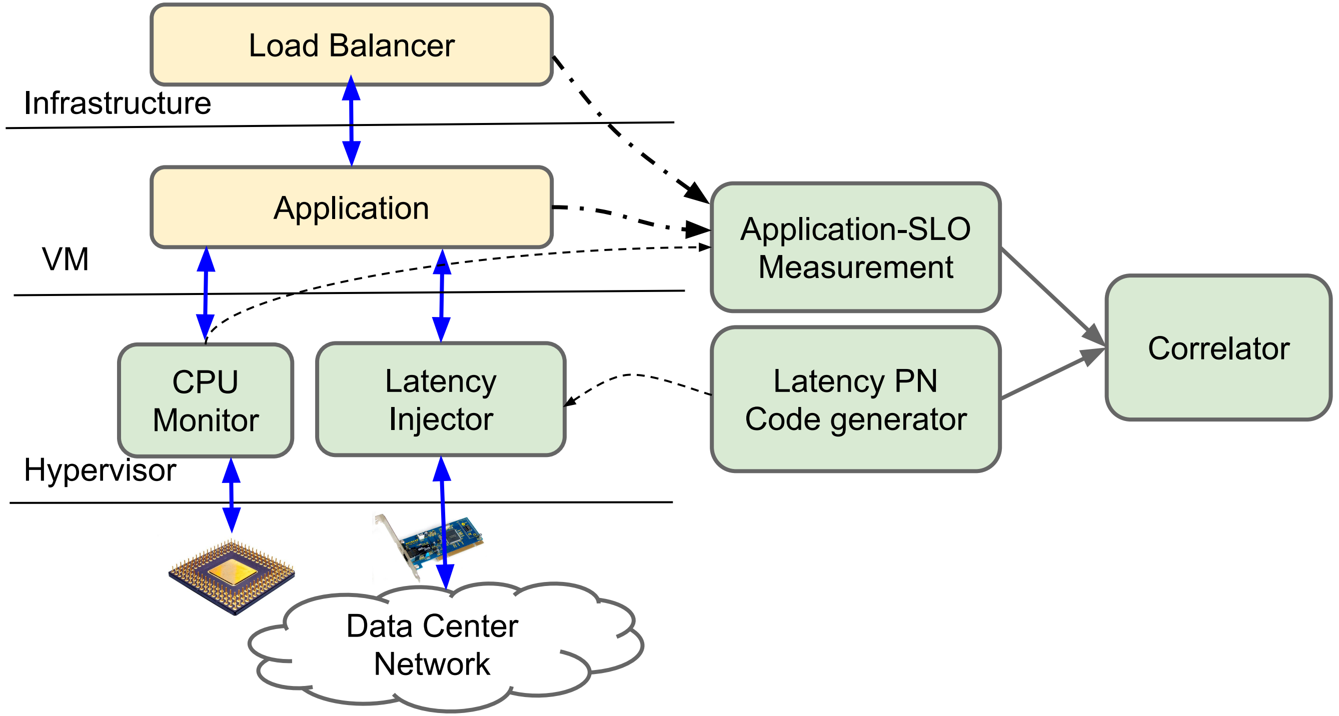
to support reasonably-fast correlation … without annoying users

Implementation

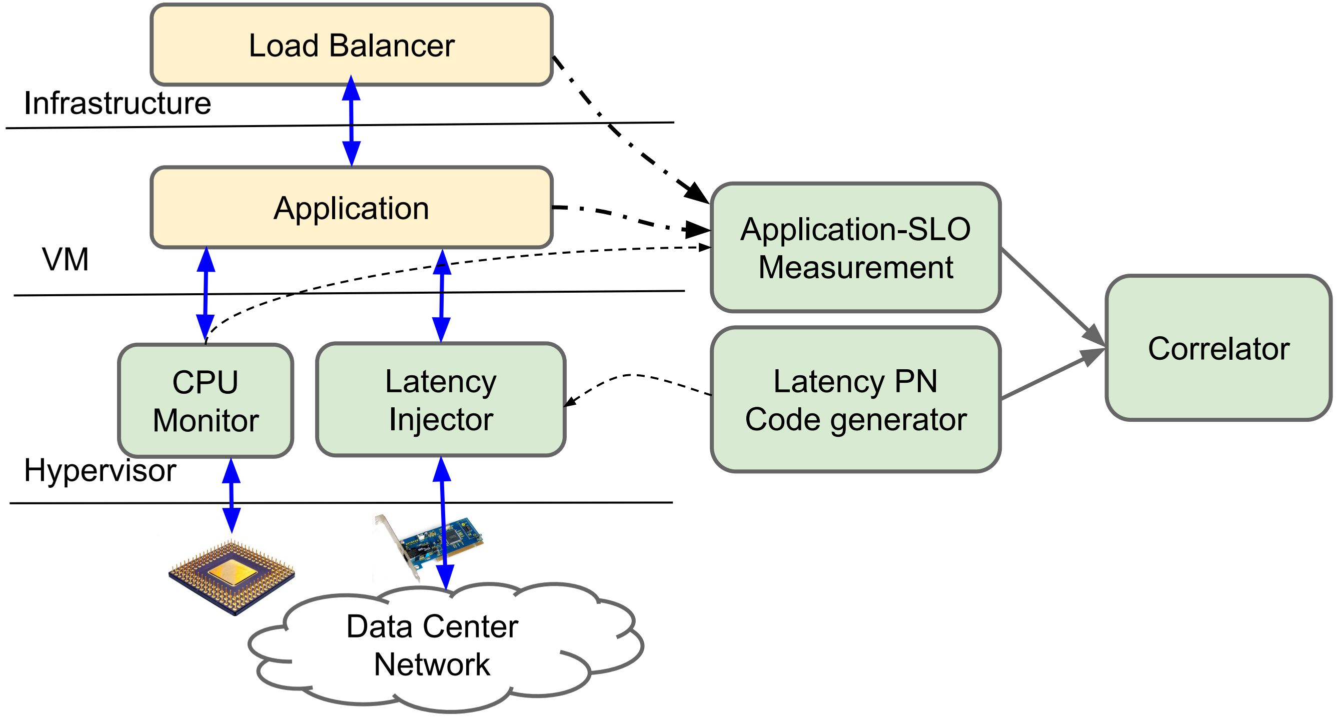
Progress so far:

- Hired a really good intern

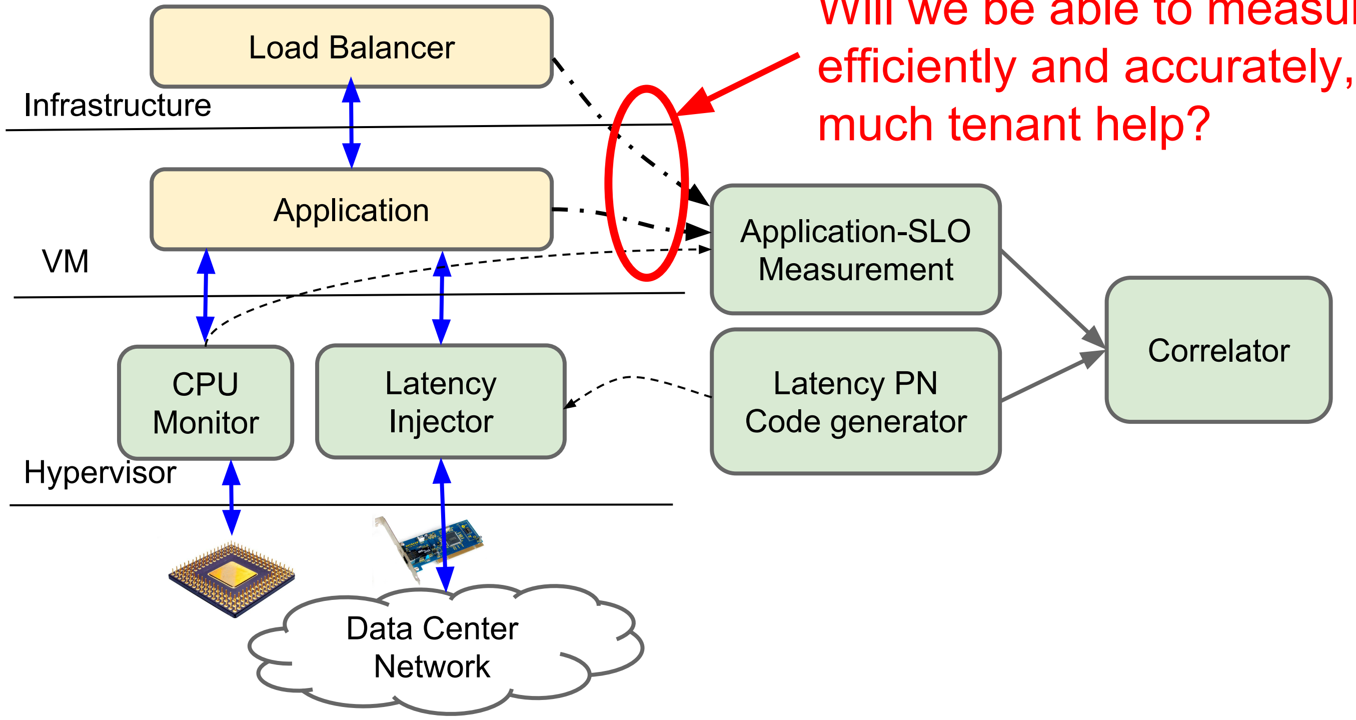
System diagram



Implementation/research challenges

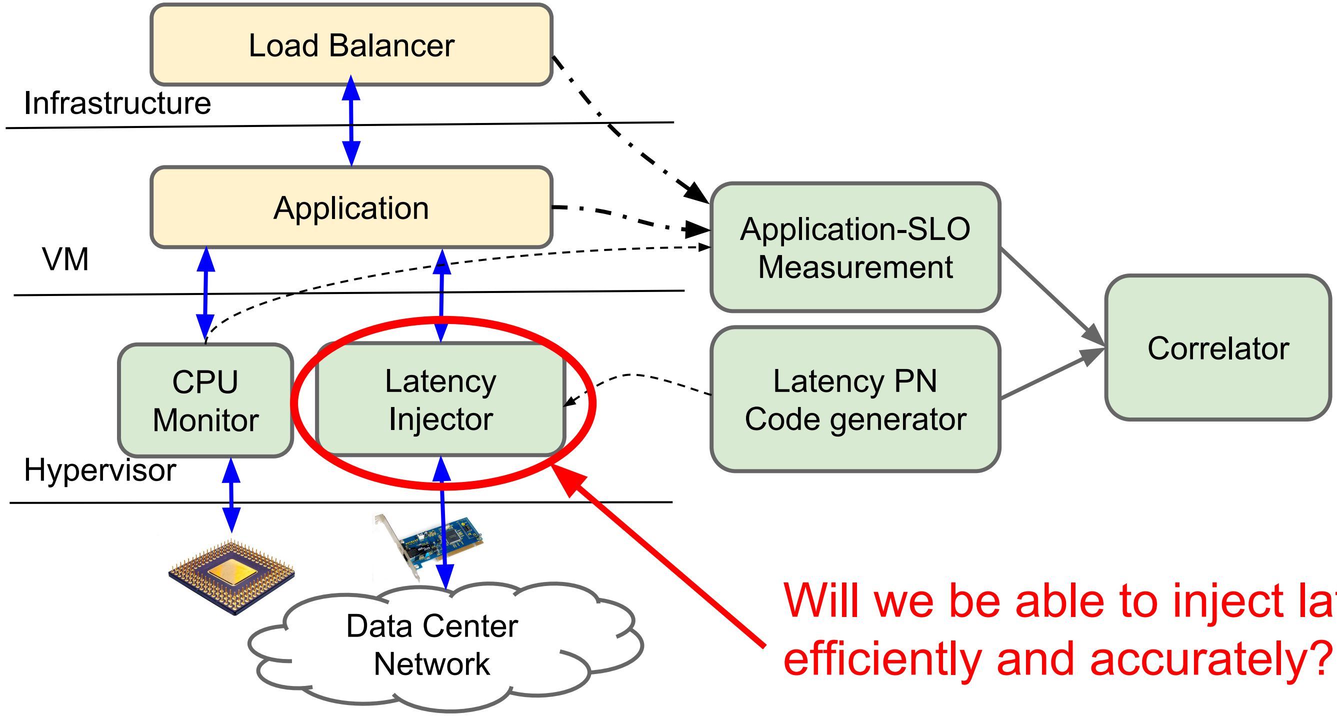


Implementation/research challenges



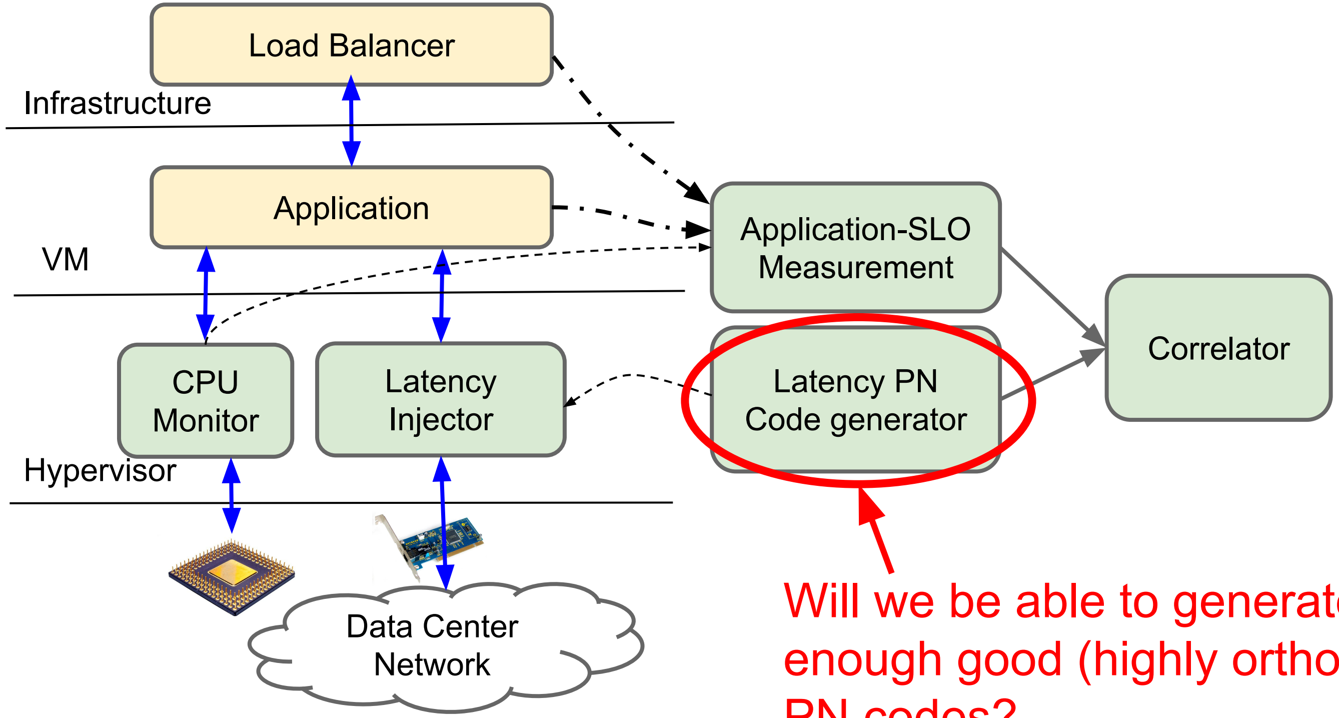
Will we be able to measure SLOs efficiently and accurately, without much tenant help?

Implementation/research challenges



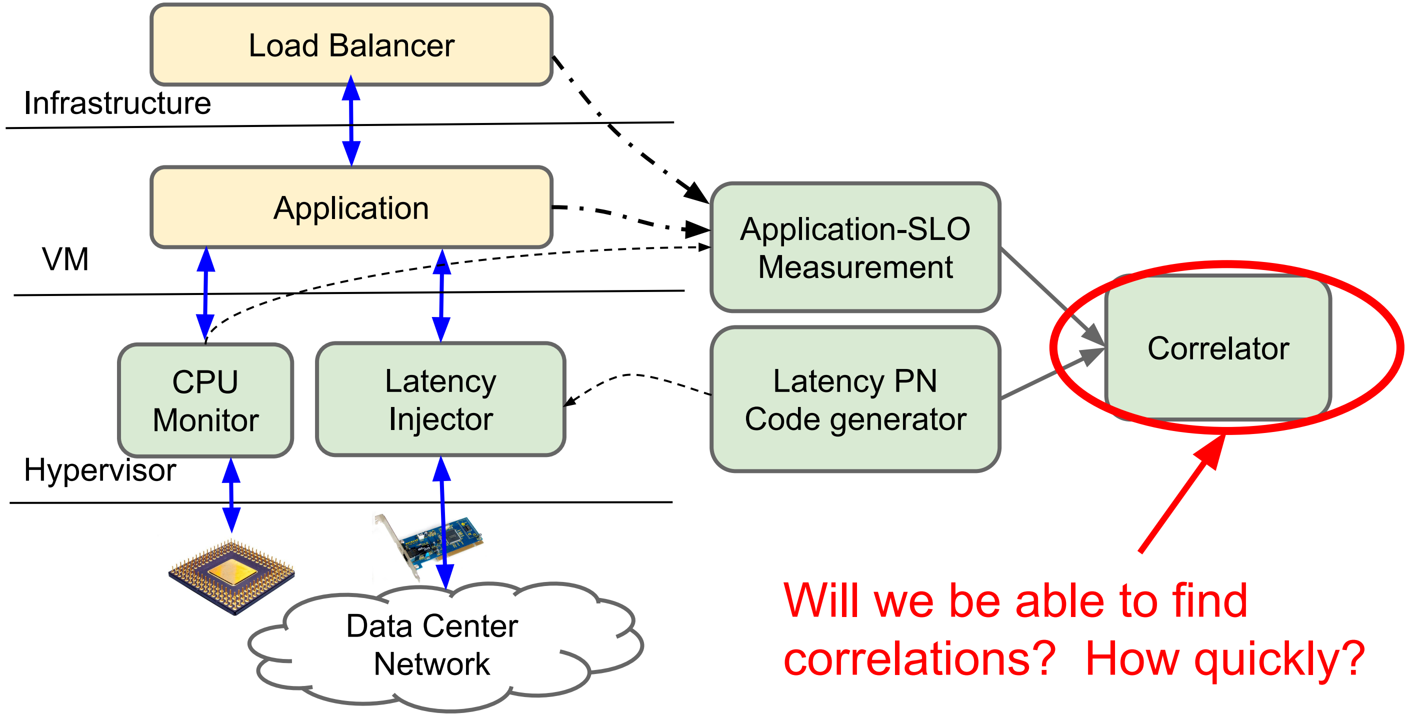
Will we be able to inject latency efficiently and accurately?

Implementation/research challenges



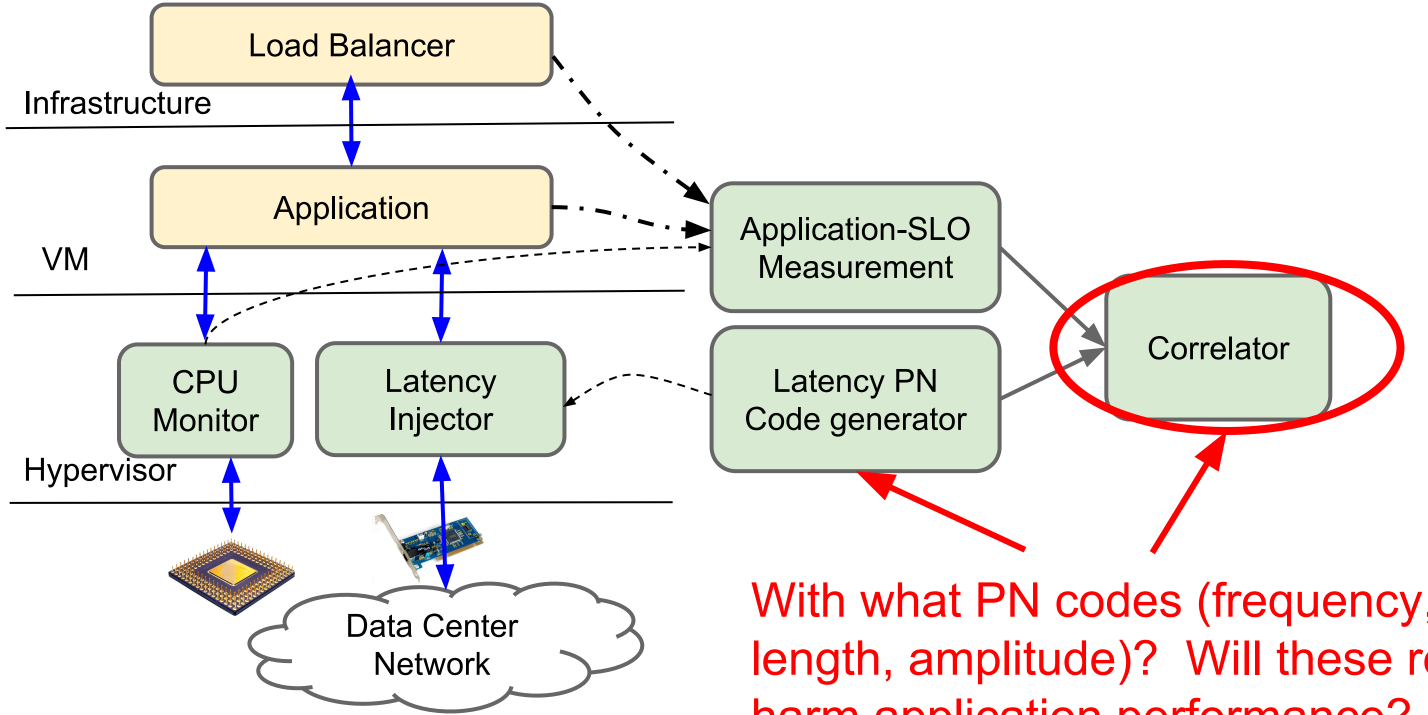
Will we be able to generate enough good (highly orthogonal) PN codes?

Implementation/research challenges



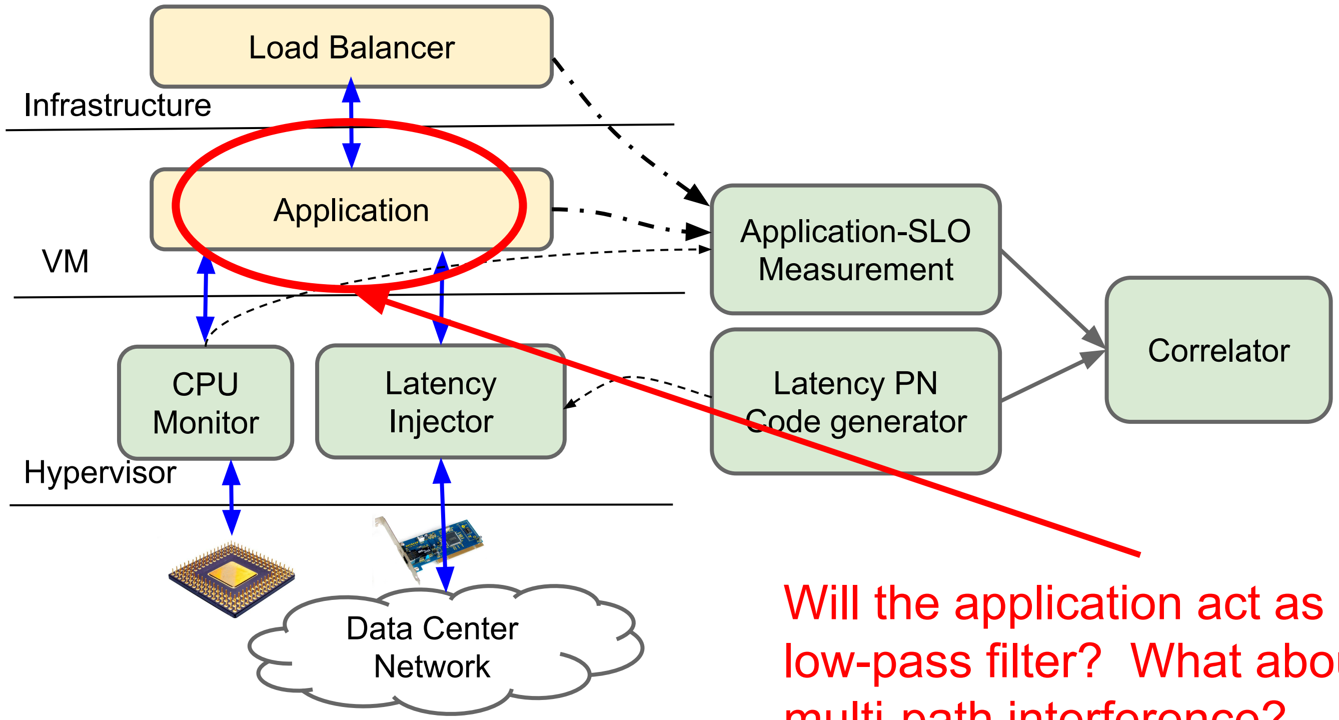
Will we be able to find correlations? How quickly?

Implementation/research challenges



With what PN codes (frequency, length, amplitude)? Will these really harm application performance?

Implementation/research challenges



Will the application act as a big low-pass filter? What about multi-path interference?

Q&A