What can we learn from data center failures and what can we do about them?

Wei Xu

Institute for Interdisciplinary Information Sciences
Tsinghua University

About Me - Wei Xu 徐葳

- Tsinghua & U Penn (B.S) 1999-2003
- Berkeley (M.S. and Ph.D.) 2003 - 2010
  - Advisors: David Patterson and Armando Fox
  - Long term visiting researcher at Google's SRE team
- Google 2010 – 2013
  - Monitoring and Debugging tools: Logging libraries, Dapper (Tracing)
- Tsinghua from 2013
  - Institute for Interdisciplinary Information Sciences (IIIS)
- Research Area
  - Distributed systems + Machine learning
  - Interdisciplinary "Big data" Applications

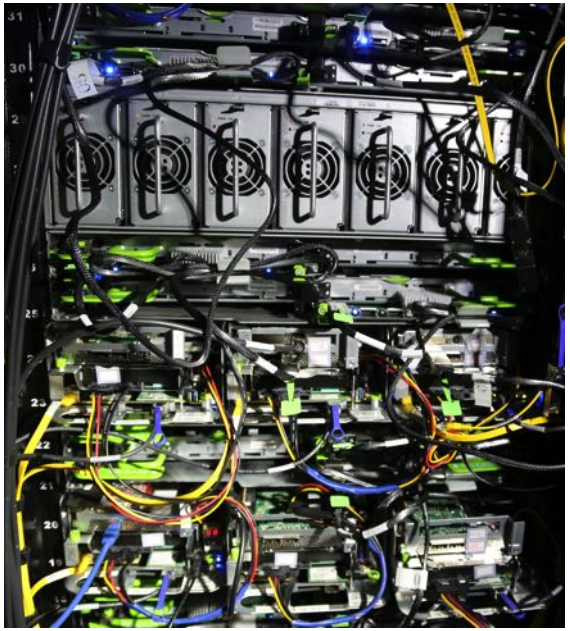
My weird job description at Tsinghua

- Assistant Professor

- Director, Open Compute Certification Lab at Tsinghua
- Advisory Committee member for Data Science institute
- Associate director of the Fintech institute
- … …

- (unofficially and unfortunately) System Administrator
  - ssh into machines to fixing misconfigured Openstack + network at 2am…

Our production cluster and Open Compute Lab



~ 300 servers

（200* 2U servers

100* Facebook OCP）

OpenStack (125 nodes)

Ceph (60 nodes)

Hadoop

Spark

…

Everything managed by …

We run smart applications

A kitten is playing with a toy.

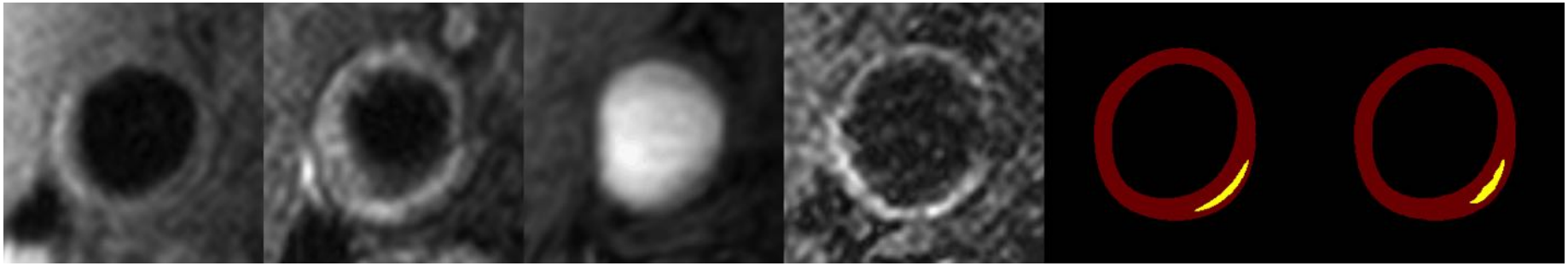
A man is singing.

A group of people are dancing.

A man is playing a piano.

A man is talking on a phone.

A man is riding a motorcycle.



But …

everything smart runs on a dumb infrastructure

needs constant baby-sitting!

Outline

- The new trends of data center failures
  - G. Wang *et. al.* What Can We Learn from Four Years of Data Center Hardware Failures? DSN'2017 [Best Paper]
- The new trends of infrastructure technologies
  - SD* : Ops -> DevOps
    - Jin *et. al.*, Optimizing Bulk Transfers with Software-Defined Optical WAN. Sigcomm'2016
    - Wei et. al., A 12-Rack, 180-Server Datacenter Network (DCN) Using Multi-wavelength Optical Switching and Full Stack Optimization. In OFC (PDP) 2016
  - Scripts / commands -> AI and natural language
    - Xiang *et. al.,* Debugging OpenStack Problems Using a State Graph Approach. APSys'16 [BEST PAPER]

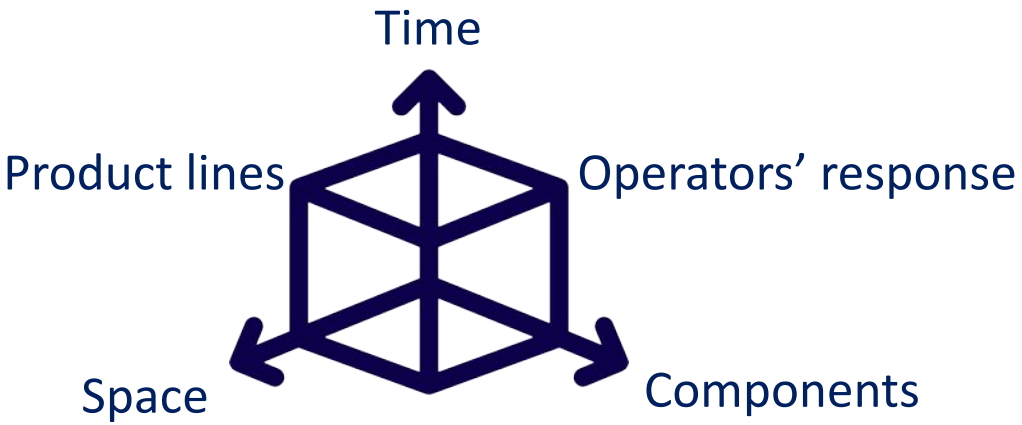
Are failures stay the same in our data centers?

- Today's data centers are different
  - 😁 Better failure detection systems, experienced operators
  - 😰 Adoption of less-reliable, commodity or custom ordered hardware, more heterogeneous hardware and workload
  - **Result:** more complex failure model
- **Goal:** comprehensive analysis of hardware failures in modern large-scale IDCs

We Re-study Hardware Failures in IDCs

Our work:

- **Large scale**: hundreds of thousands of servers with 290,000 failure operation tickets
- **Long-term**: 2012-2016
- **Multi-dimensional**: components, time, space, product lines, operators' response, etc.
- Reconfirm or extend previous findings + Observe new patterns



Interesting Findings Overview

Common beliefs

- Failures are uniformly randomly distributed over time/space
- Failures happen independently
- HW unreliability shapes the software fault tolerance design

Our findings

- HW failures are not uniformly random
  - at different time scales
  - sometimes at different locations
- Correlated HW failures are common in IDCs
- It is also the other way around: software fault tolerance indulges operators to care less about HW dependability

Failure Percentage Breakdown by Component

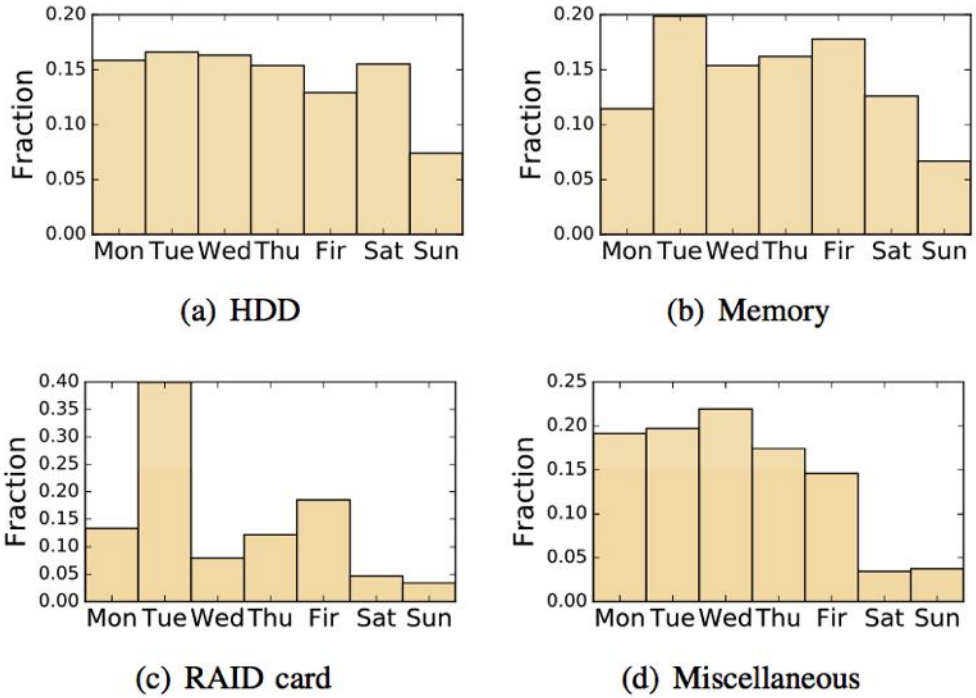
Device	Proportion
Hard Disk Drive	81.84%
Miscellaneous*	10.20%
Memory	3.06%
Power	1.74%
RAID card	1.23%
Flash card	0.67%
Motherboard	0.57%
SSD	0.31%
Fan	0.19%
HDD backboard	0.14%
CPU	0.04%

*"*Miscellaneous*" are manually submitted or uncategorized failures

TEMPORAL DISTRIBUTION OF THE FAILURES

FR is NOT Uniformly Random over Days of the Week

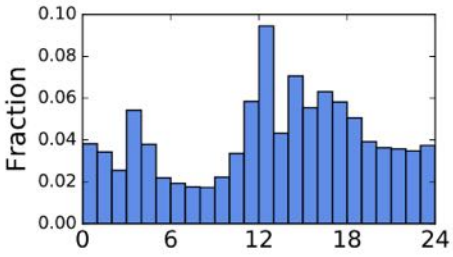
- **Hypothesis 1.** *The average number of component failures is uniformly random over different days of the week.*



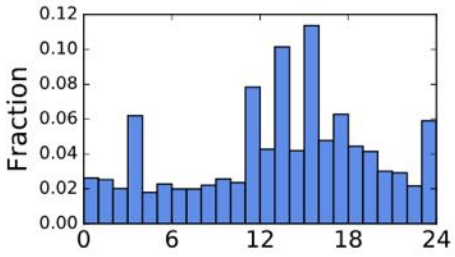
- A chi-square test can reject the hypothesis at 0.01 significance level for **all** component classes.

FR is NOT Uniformly Random over Hours of the Day

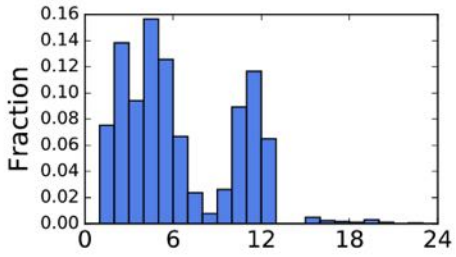
- **Hypothesis 2.** *The average number of component failures is uniformly random during each hour of the day.*



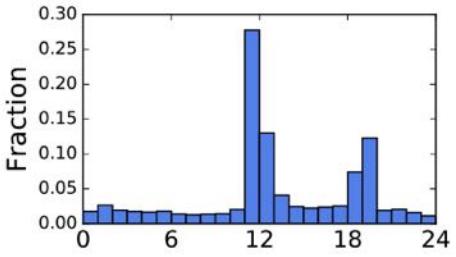
(a) HDD



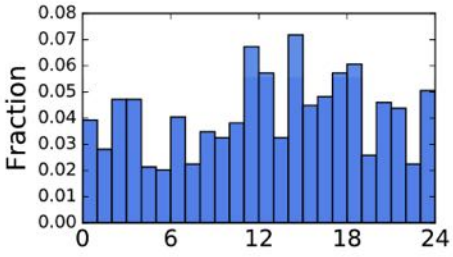
(b) Memory



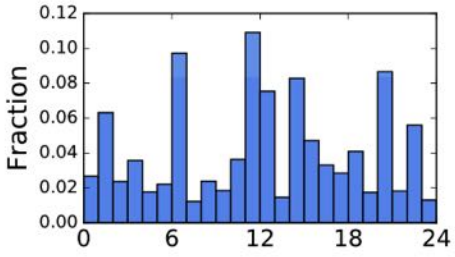
(c) Motherboard



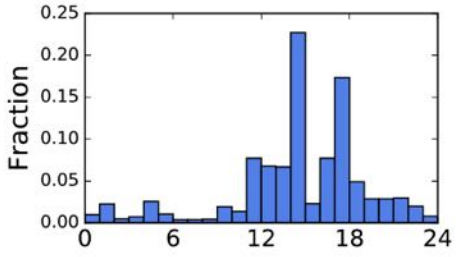
(d) RAID card



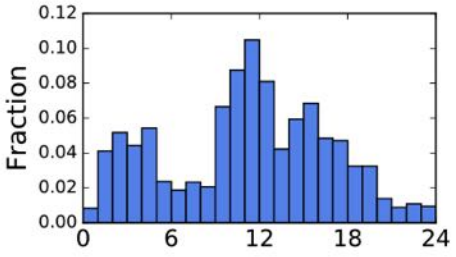
(e) SSD



(f) Power



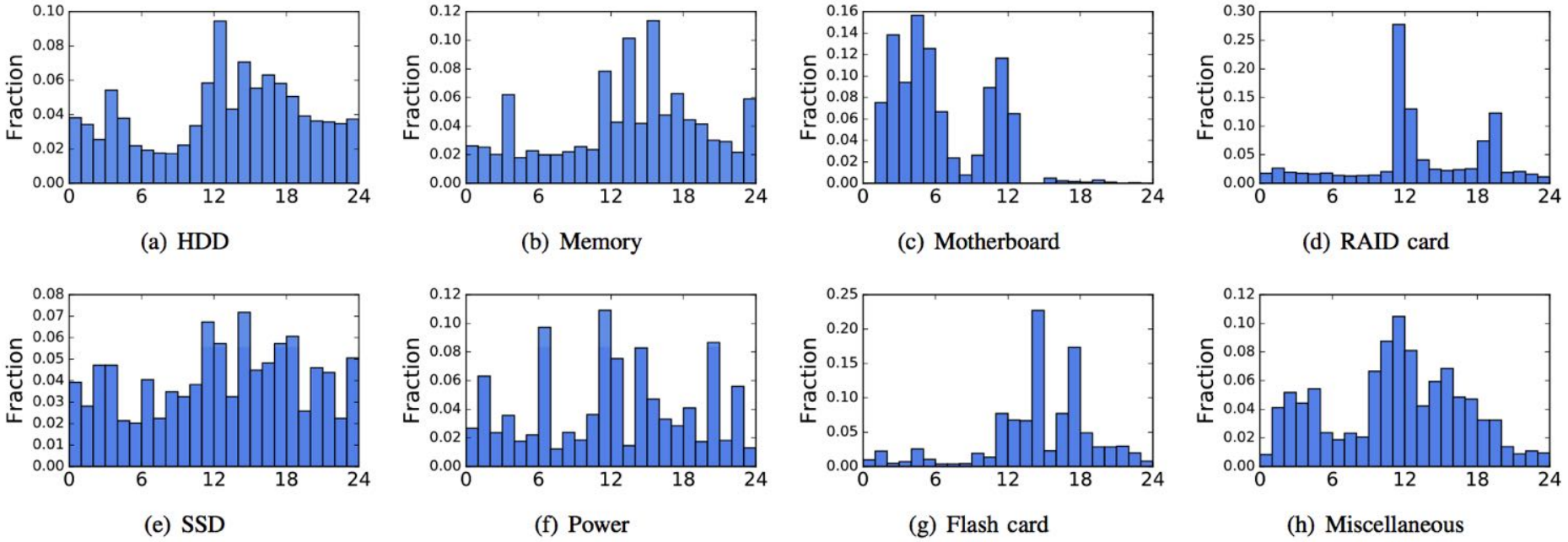
(g) Flash card



(h) Miscellaneous

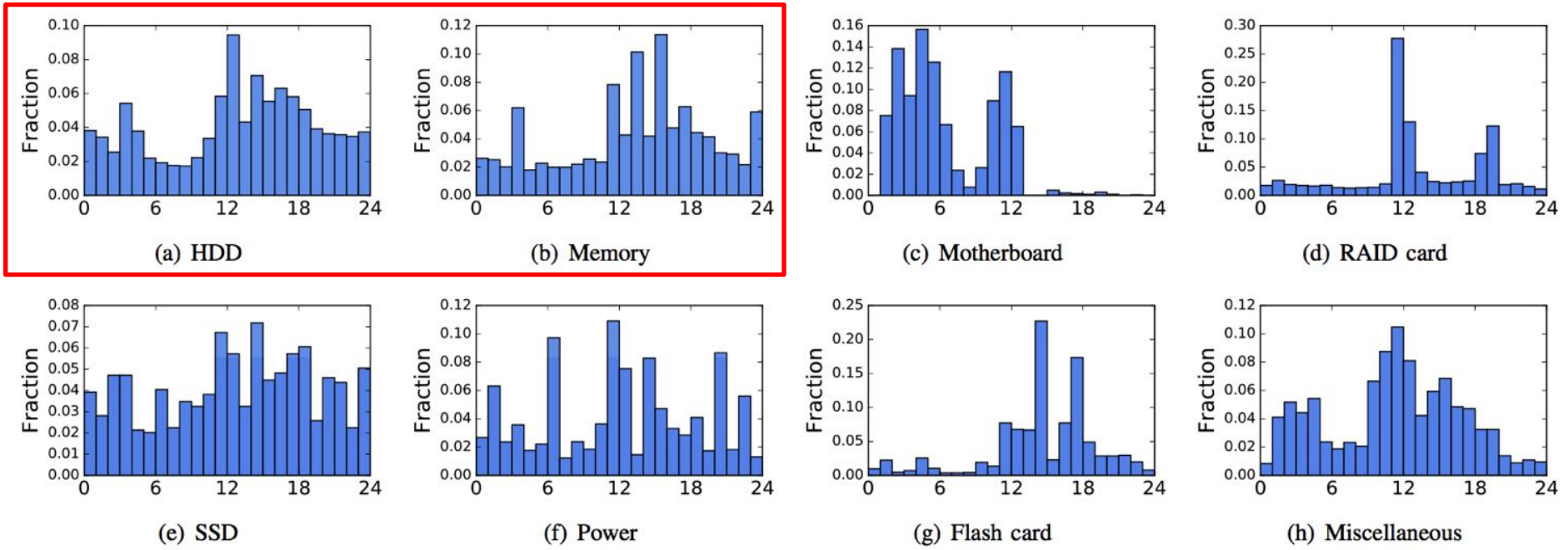
FR is NOT Uniformly Random over Hours of the Day

- Possible Reasons
  - High workload results in more failures
  - Human factors
  - Components fail in large batches



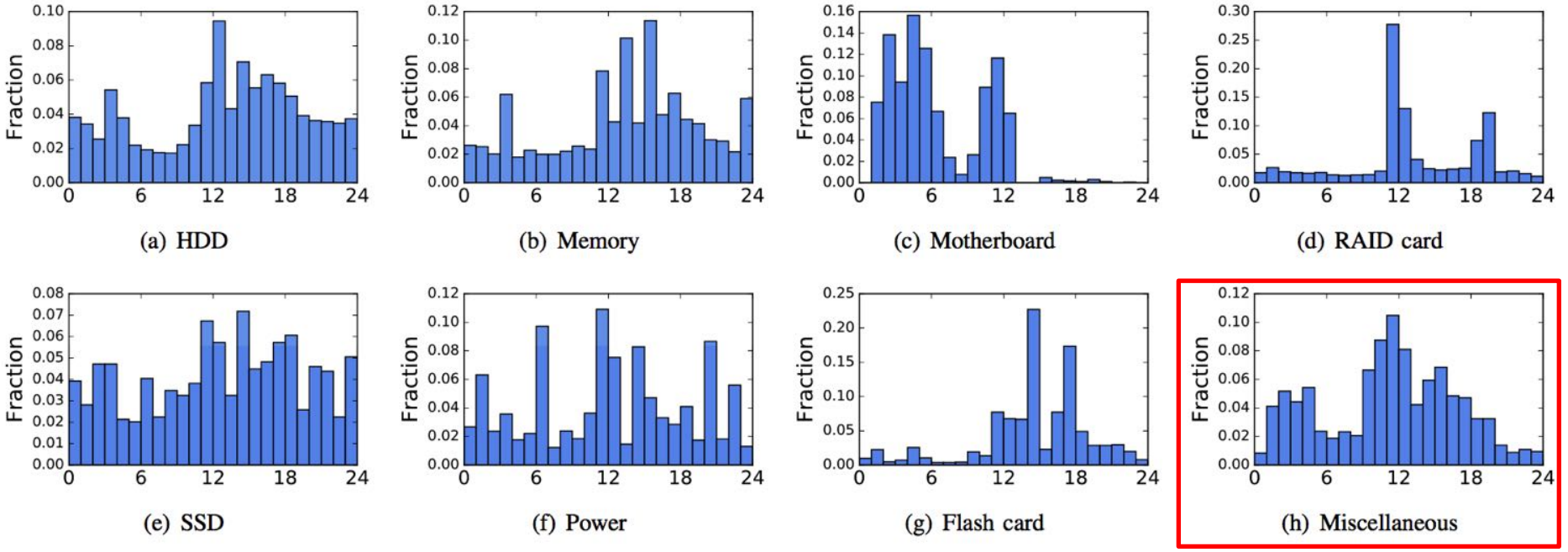
FR is NOT Uniformly Random over Hours of the Day

- Possible Reasons
  - High workload results in more failures
  - Human factors
  - Components fail in large batches



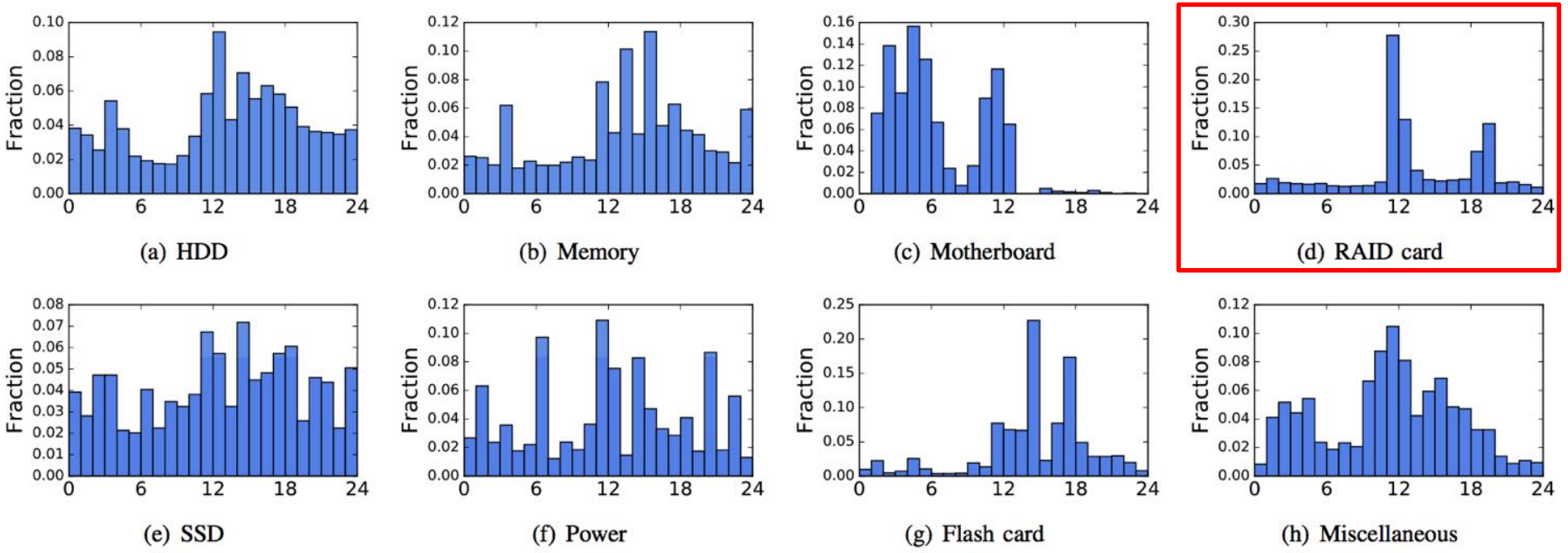
FR is NOT Uniformly Random over Hours of the Day

- Possible Reasons
  - High workload results in more failures
  - Human factors
  - Components fail in large batches



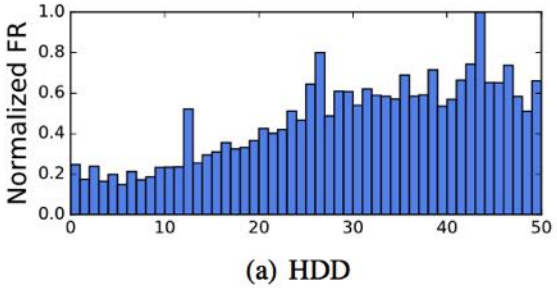
FR is NOT Uniformly Random over Hours of the Day

- Possible Reasons
  - High workload results in more failures
  - Human factors
  - Components fail in large batches

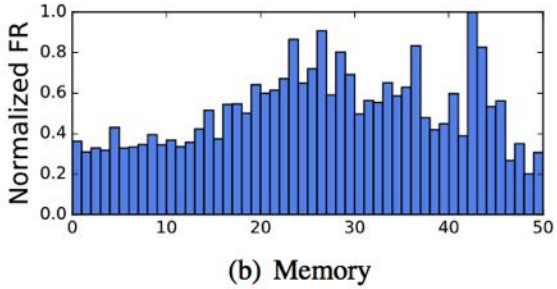


FR of each Component Changes During its Life Cycle

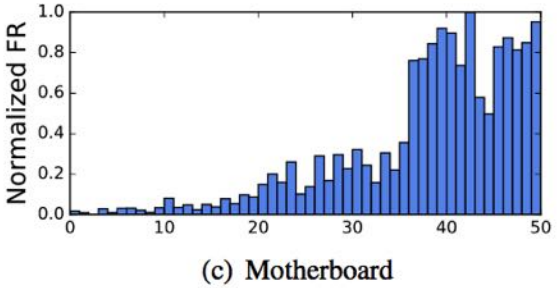
- Different component classes exhibit different FR patterns.



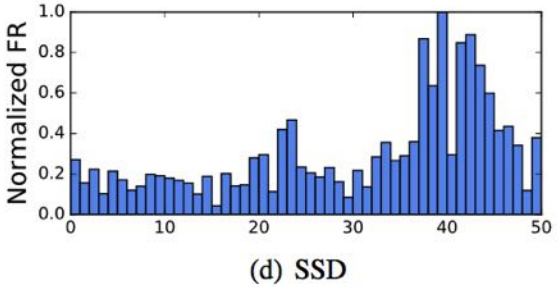
(a) HDD



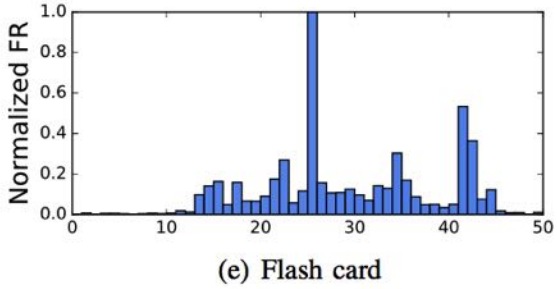
(b) Memory



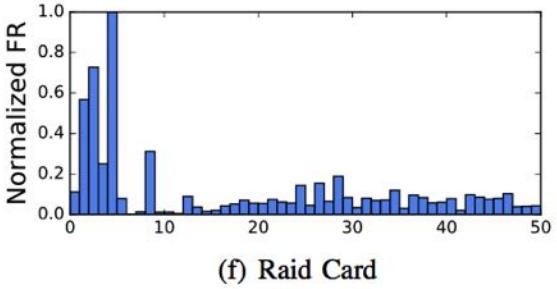
(c) Motherboard



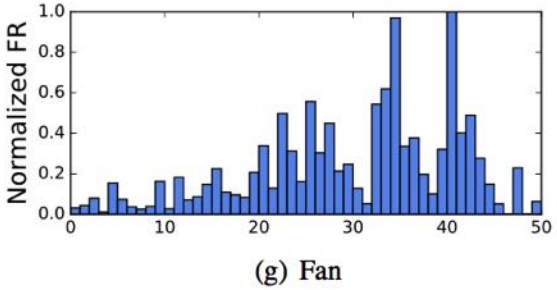
(d) SSD



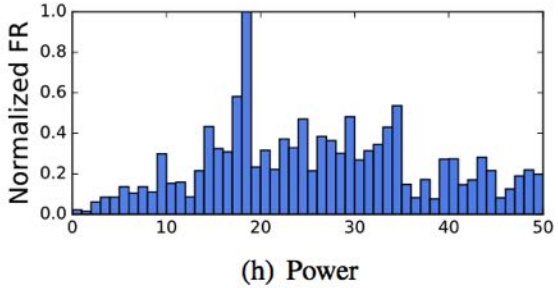
(e) Flash card



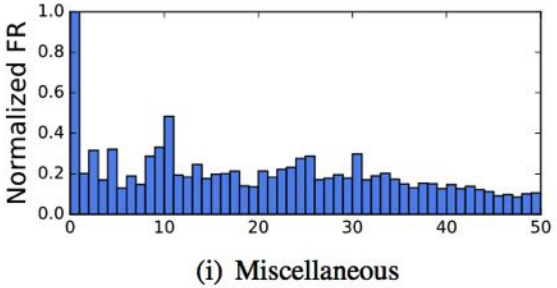
(f) Raid Card



(g) Fan



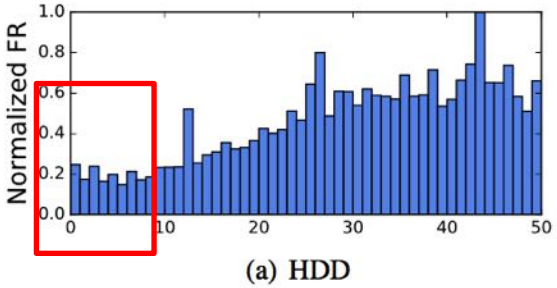
(h) Power



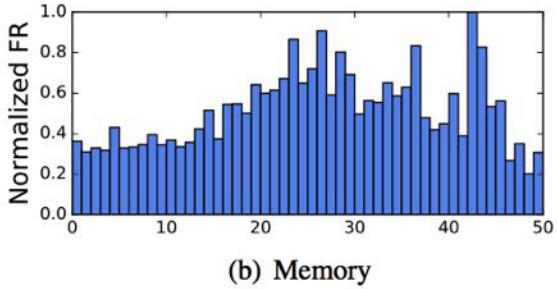
(i) Miscellaneous

FR of each Component Changes During its Life Cycle

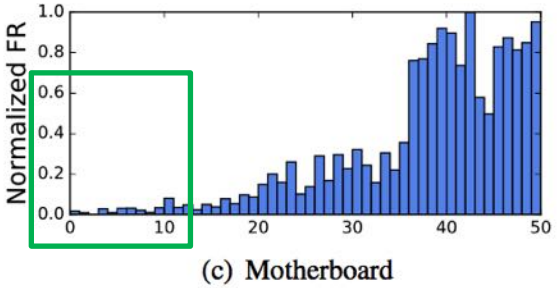
- Infant mortalities:



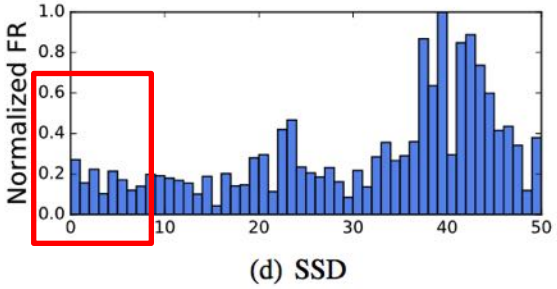
(a) HDD



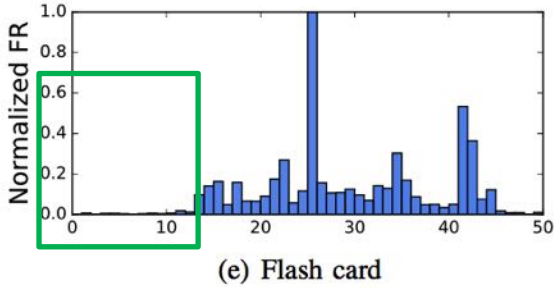
(b) Memory



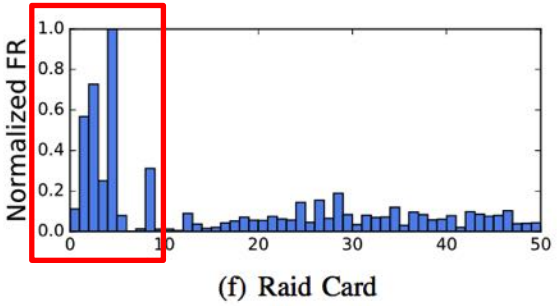
(c) Motherboard



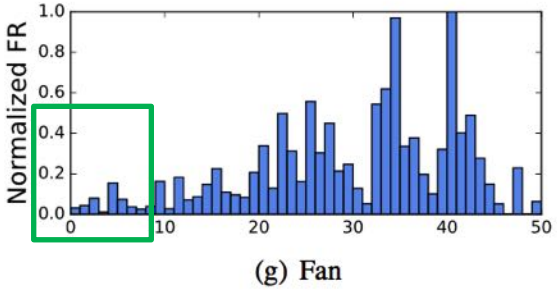
(d) SSD



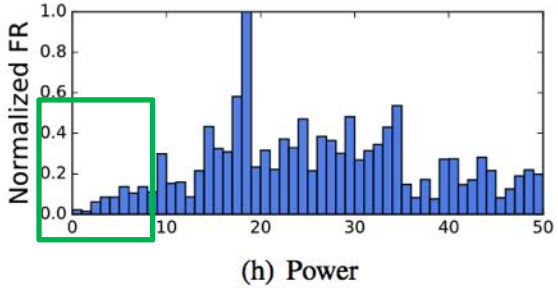
(e) Flash card



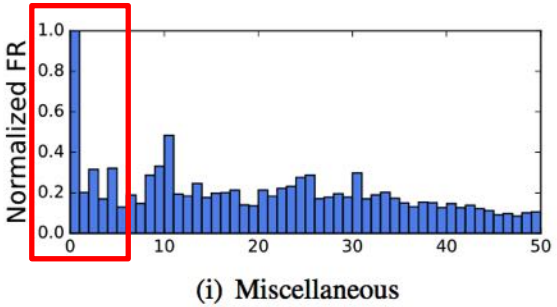
(f) Raid Card



(g) Fan



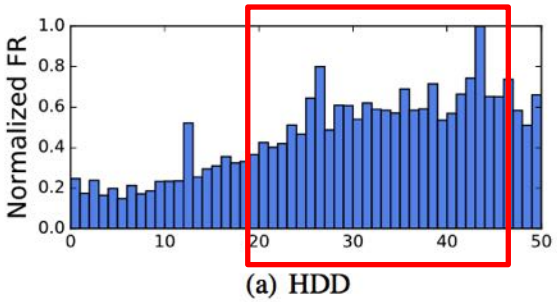
(h) Power



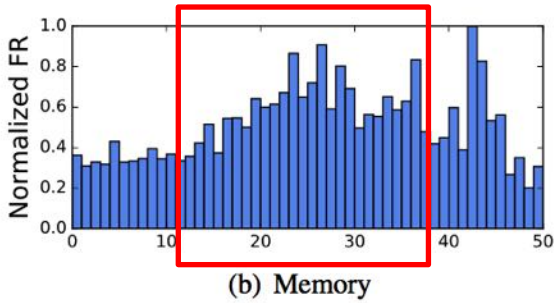
(i) Miscellaneous

FR of each Component Changes During its Life Cycle

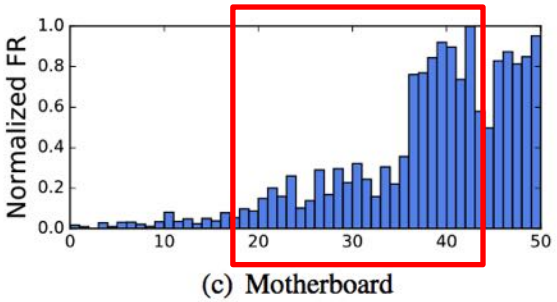
- Wear out



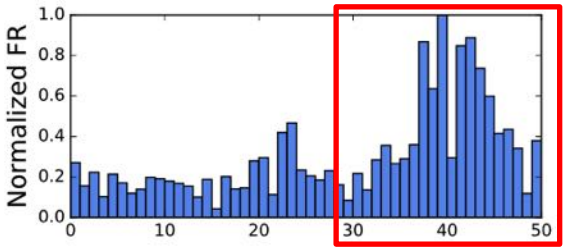
(a) HDD



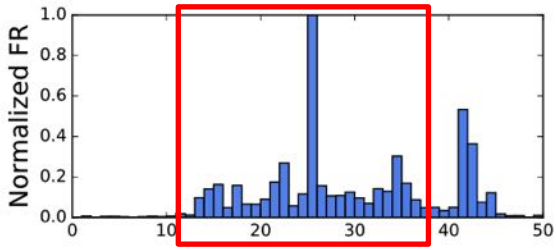
(b) Memory



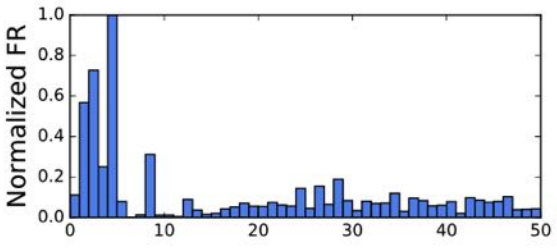
(c) Motherboard



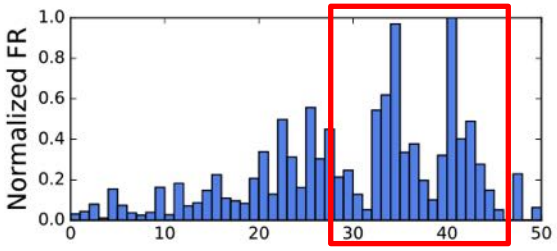
(d) SSD



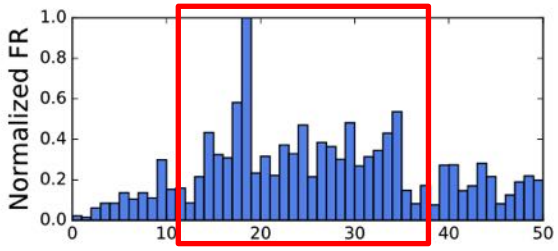
(e) Flash card



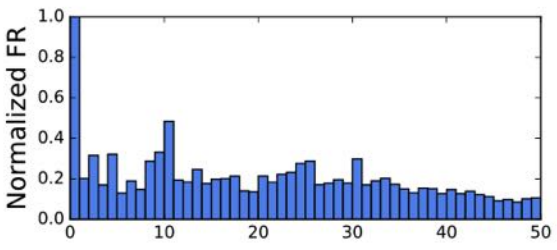
(f) Raid Card



(g) Fan



(h) Power

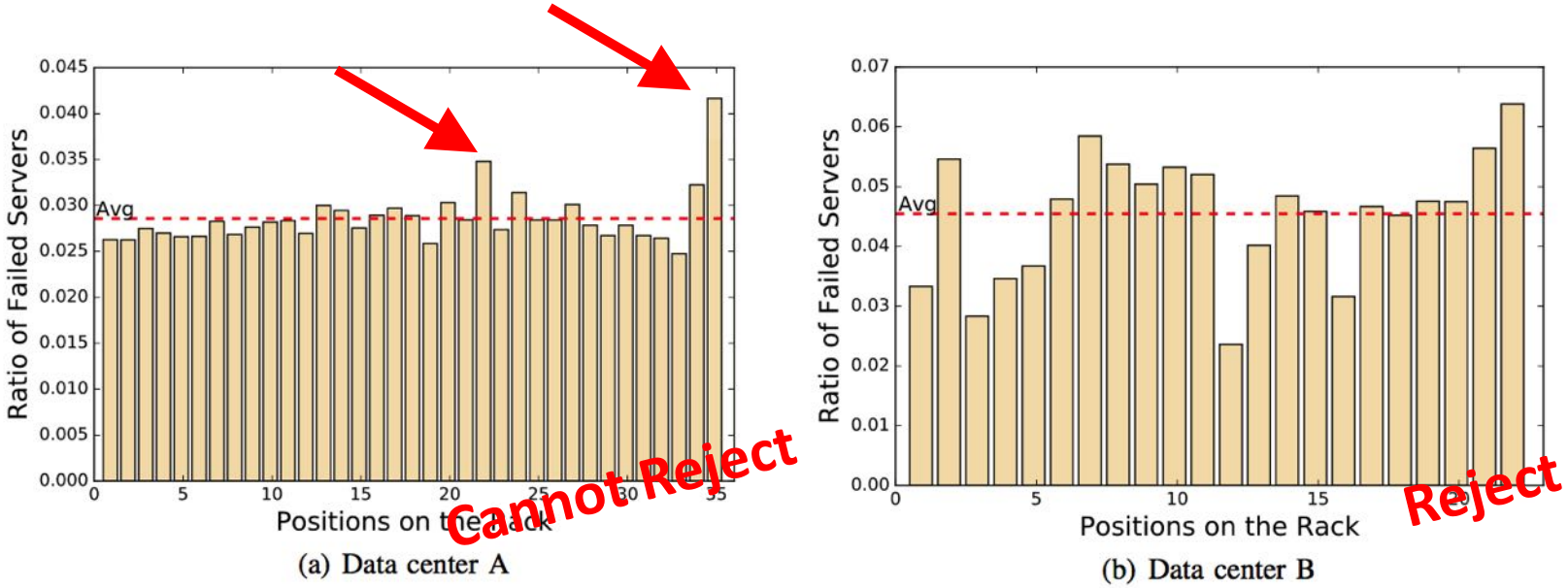


(i) Miscellaneous

SPACIAL DISTRIBUTION OF THE FAILURES

Physical Locations Might Affect the FR Distribution

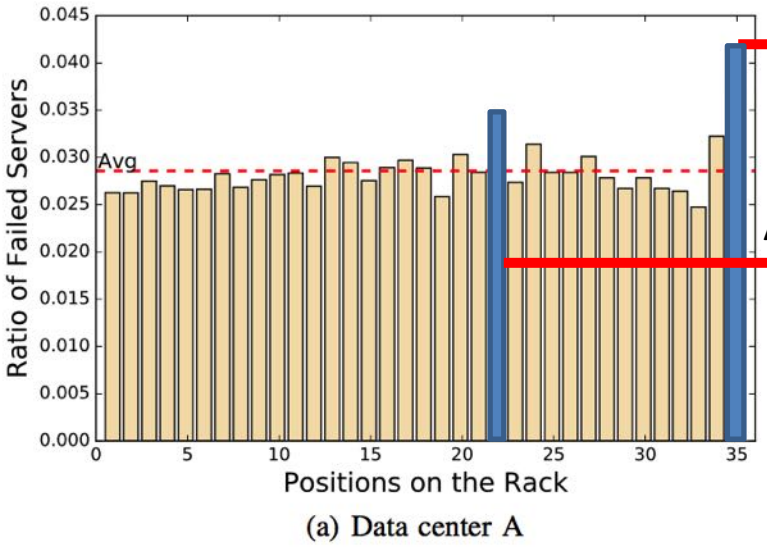
- **Hypothesis 3.** *The failure rate on each rack position is independent of the rack position.*



- In general, at 0.05 significance level:
  - can not reject the hypothesis in 40% of the data centers
  - can reject it in the other 60%

FR Can be Affected by the Cooling Design

- FRs are higher at rack position 22 and 35



(a) Data center A

At the top

Above the PSU

A typical Scorpion rack

- Possible reasons
  - Design of IDC cooling and physical structure of the racks

CORRELATED FAILURES

Correlated Failures are Common

- Correlated failures: *batch failures, correlated component failures, repeating synchronous failures*
- Fact: 200+ HDD failures on each of 22.5% of the days
- Case study
  - Nov. 16th and 17th, 2015
  - 5,000+ servers, or 32% of all the servers of the product line, reporting hard drive *SMARTFail* failures
  - 99% of these failures were detected between 21:00 on the 16th and 3:00 on the 17th.
  - Operators replaced about 1,600, decommissioned the remaining 4000+ out-of-warranty drives
  - Failure reason not clear yet

Causes of Correlated Failures

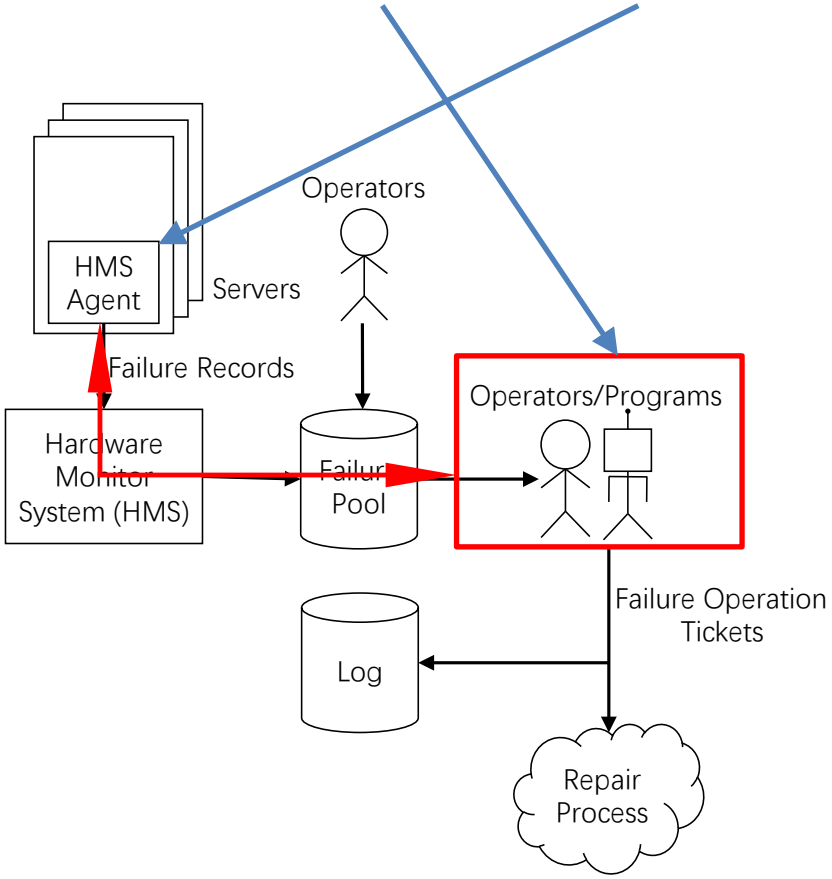
All the following have happened before😭

- Environmental factors (e.g., humidity)
- Firmware bugs
- Single point of failure (e.g., power module failures)
- Human operator mistakes
- …

OPERATORS' RESPONSE TO FAILURES

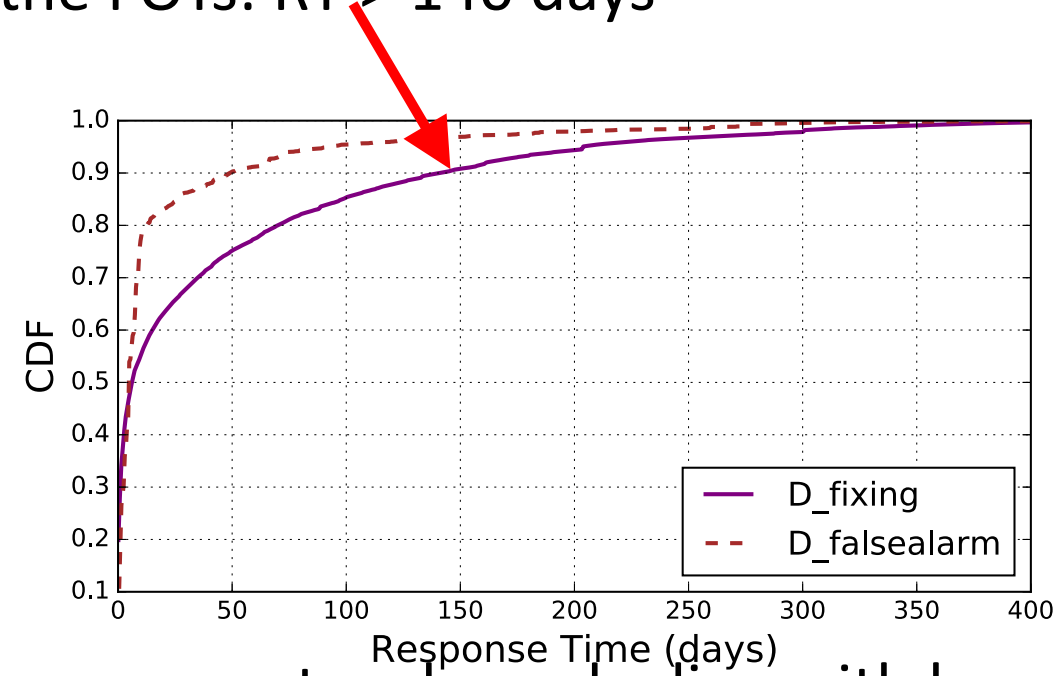
Operators' Response to Failures

- Response time: *RT = op_time – err_time*



RT is Very High in General

- RT for *D_fixing*: Avg. 42.2 days, median 6.1 days
- 10% of the FOTs: RT > 140 days

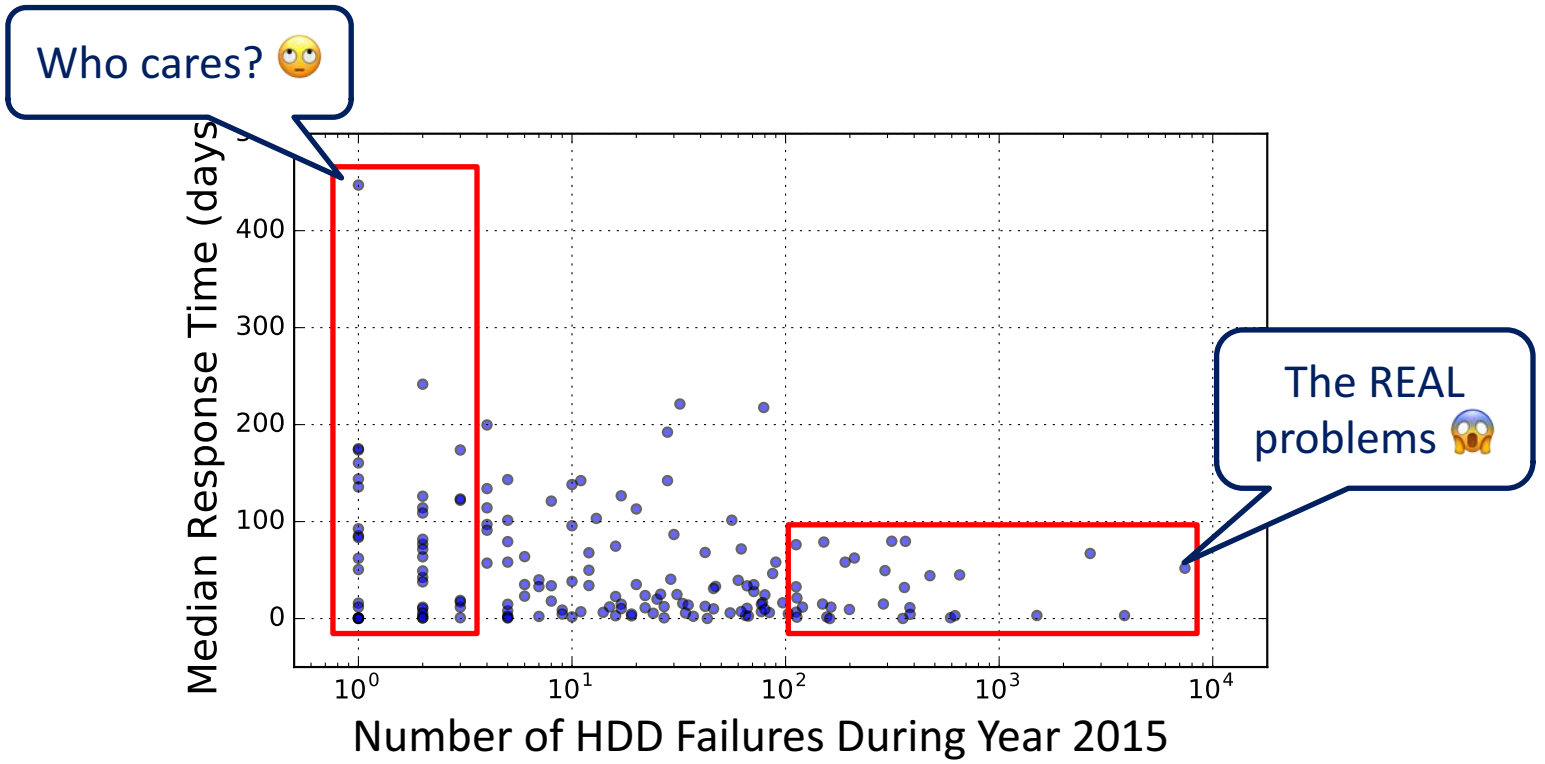


- Is it because operators busy dealing with large number of failures?

- No!

RT in Different Product Lines Varies

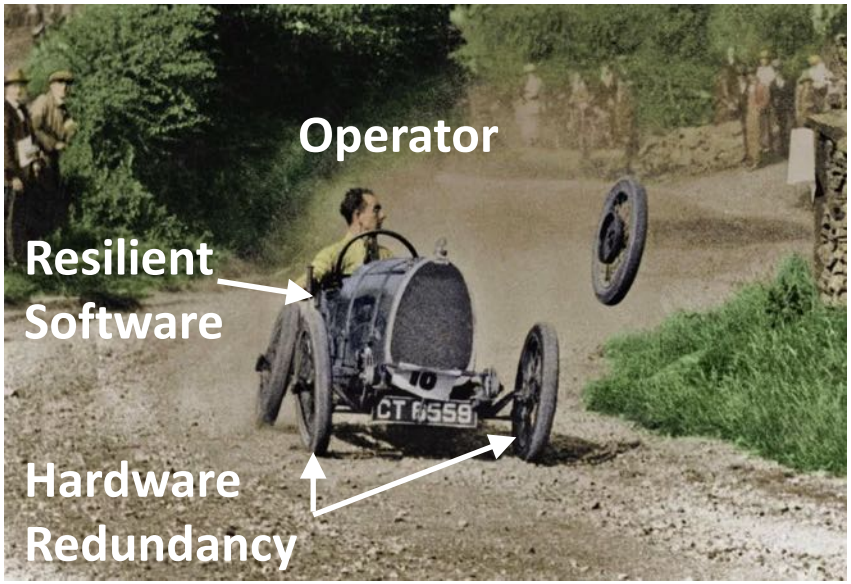
- Observation 1: Variation of *RT* in different product lines is large
- Observation 2: Operators respond to large number of failure more quickly



OPs are Less Motivated to Respond to HW Failures

Possible reasons

- Software redundancy design
  - Delayed Responding, process failures in batches
- Many hardware failures are no longer urgent
  - E.g., SMART failures may not be fatal
- Repair operation can be costly
  - E.g., Task migration

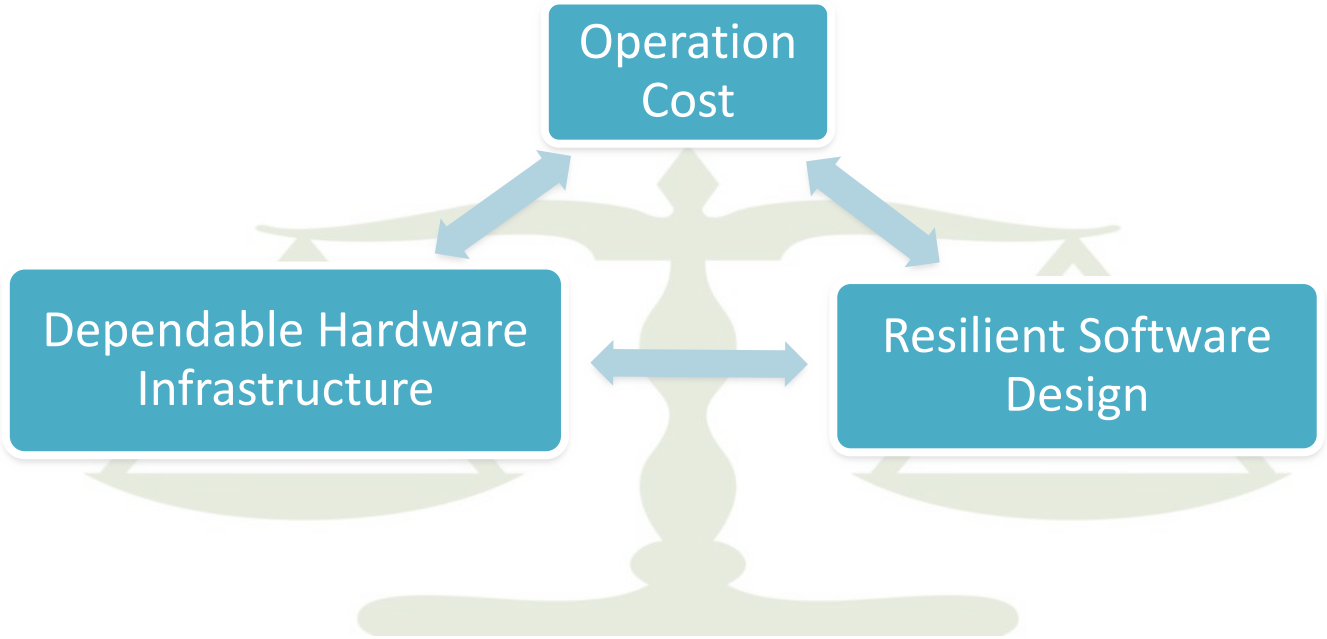


Lessons Learned I

- Much old wisdom still holds.
- More correlated failures ⇒ software design challenge
- Automatic hardware failure detection & handling: 😁
- Data center design: avoid “bat spot”

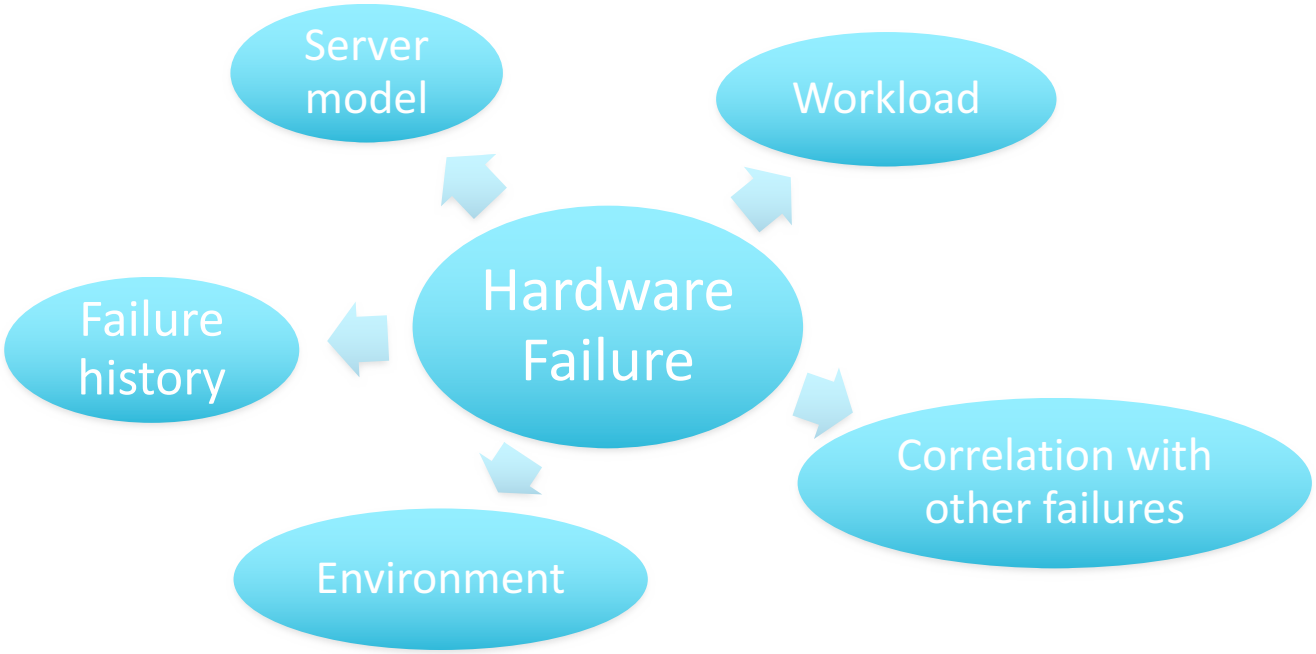
Lessons Learned II

- Strike the right balance among *software stack complexity*, *hardware dependability*, and *operation cost*.
- Data center dependability needs joint optimization effort that crosses layers.



Lessons Learned III

- *Stateful* failure handling system
  - Data mining tool: discover correlation among failures
  - Provide operators with extra information



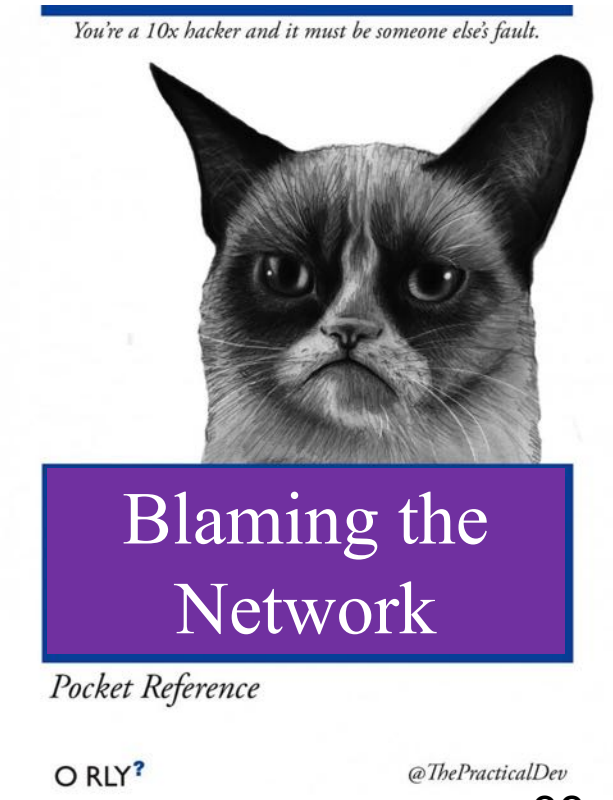
Outline

- The new trends of data center failures
  - G. Wang *et. al.* What Can We Learn from Four Years of Data Center Hardware Failures? DSN'2017 [Best Paper]
- The new trends of infrastructure technologies
  - SD* : Ops -> DevOps
    - Jin *et. al.*, Optimizing Bulk Transfers with Software-Defined Optical WAN. Sigcomm'2016
    - Wei et. al., A 12-Rack, 180-Server Datacenter Network (DCN) Using Multi-wavelength Optical Switching and Full Stack Optimization. In OFC (PDP) 2016
  - Scripts / commands -> AI and natural language
    - Xiang *et. al.,* Debugging OpenStack Problems Using a State Graph Approach. APSys'16 [BEST PAPER]

A view from a software engineer: Traditional networking

- Network is a pipe
- It should be an infinite pipe
- It is managed by someone we never met
- Nothing to do on the network
- When everything works, no one cares about the network (good)

- And a great bonus application too –



A view from a software engineer: Software defined networking (SDN)

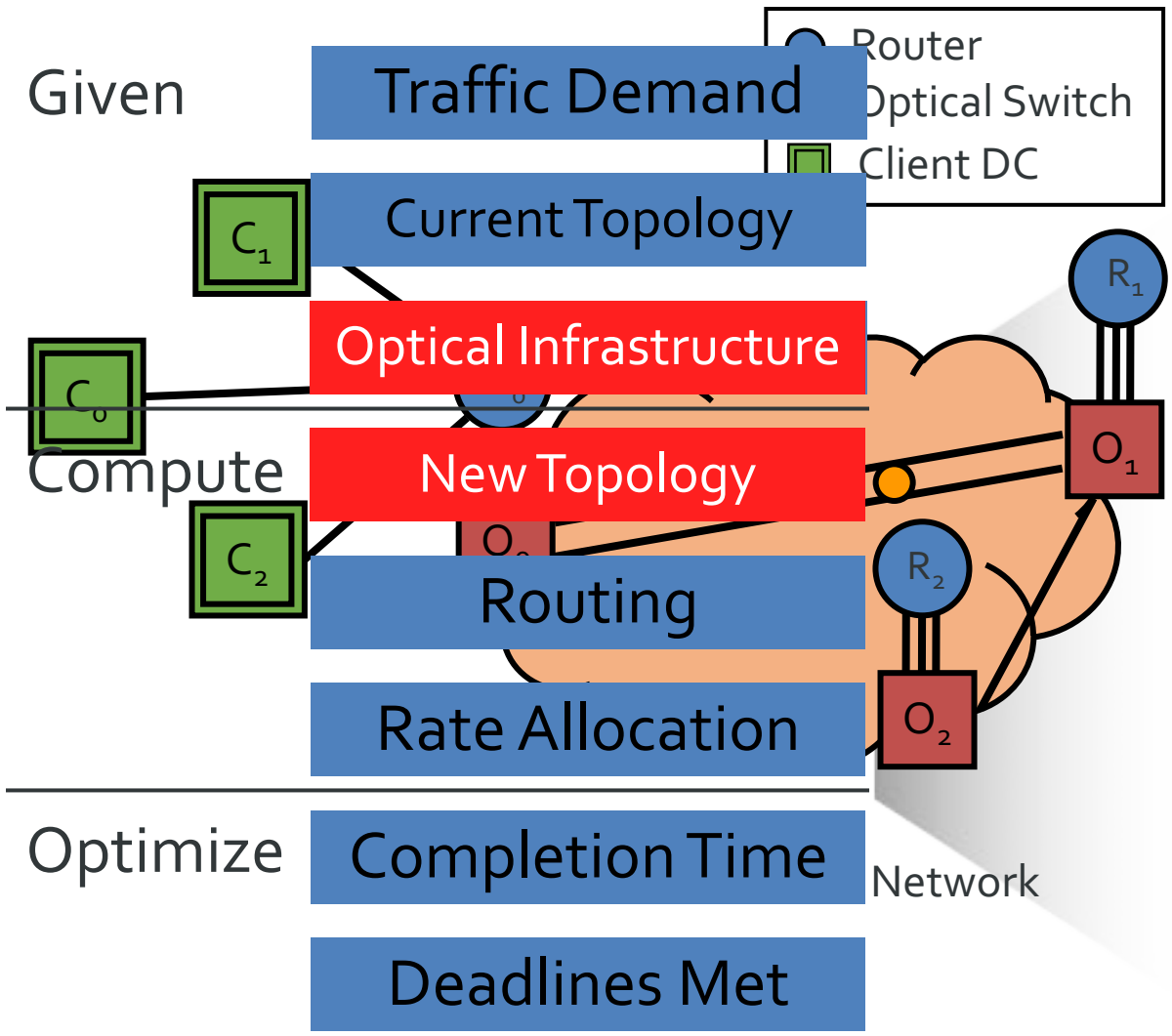
- A huge opportunity to understand and solve many problems
  - E.g. long tail problem in data centers
  - E.g. Actionable

- Challenges:
  - How much flexibility we can get?
  - How ``software defined'' can the network be?
  - Are we making the system even more complex (as there are many more parameters to tune)?

Our key ideas

- Joint optimization over the physical layer and the network layer
  - More flexibility
  - A flexible optical network and its optimizations
  - Bulk transfer over wide-area-networks: Owan [Sigcomm 16]
  - Data center networks: Dfabric [OFC- PDP 16]

Joint Optimization: network layer + optical layer

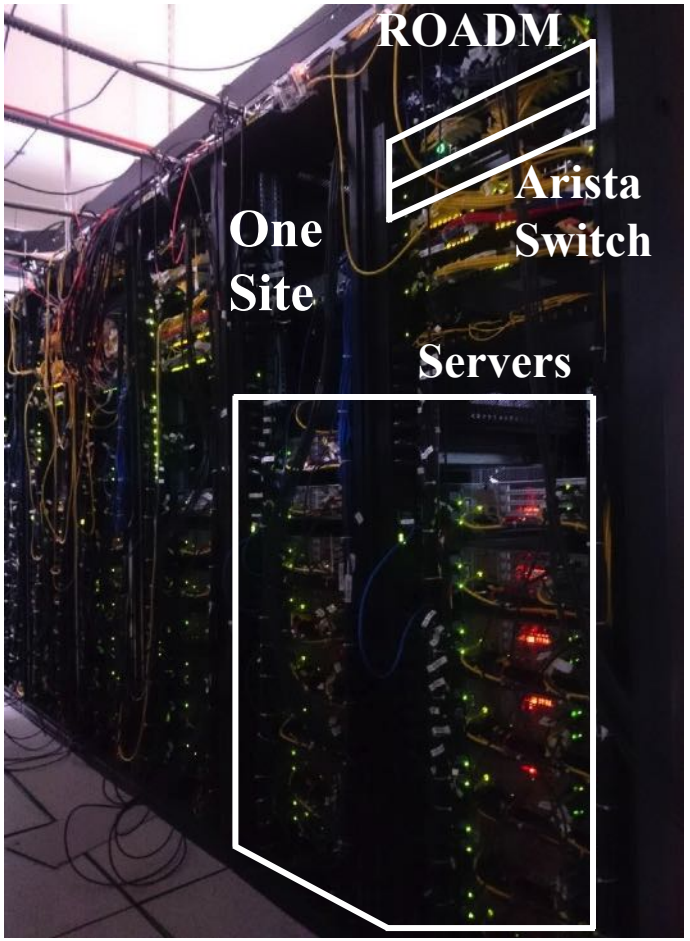


Constraints

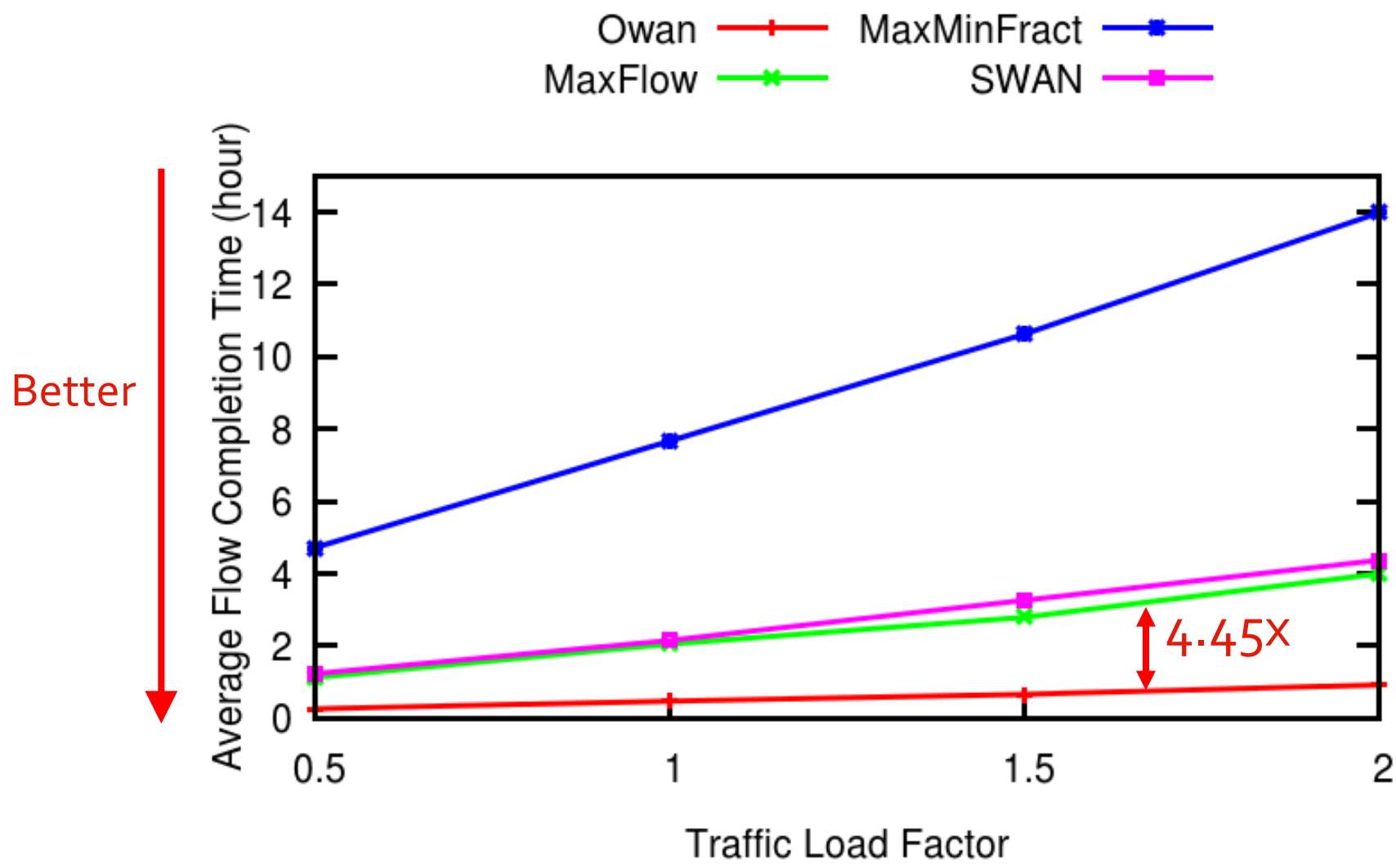
- # of Router Ports
- Optical Reach
- # of Regenerators
- # of Wavelengths
- Link Capacity

Testbed Implementation

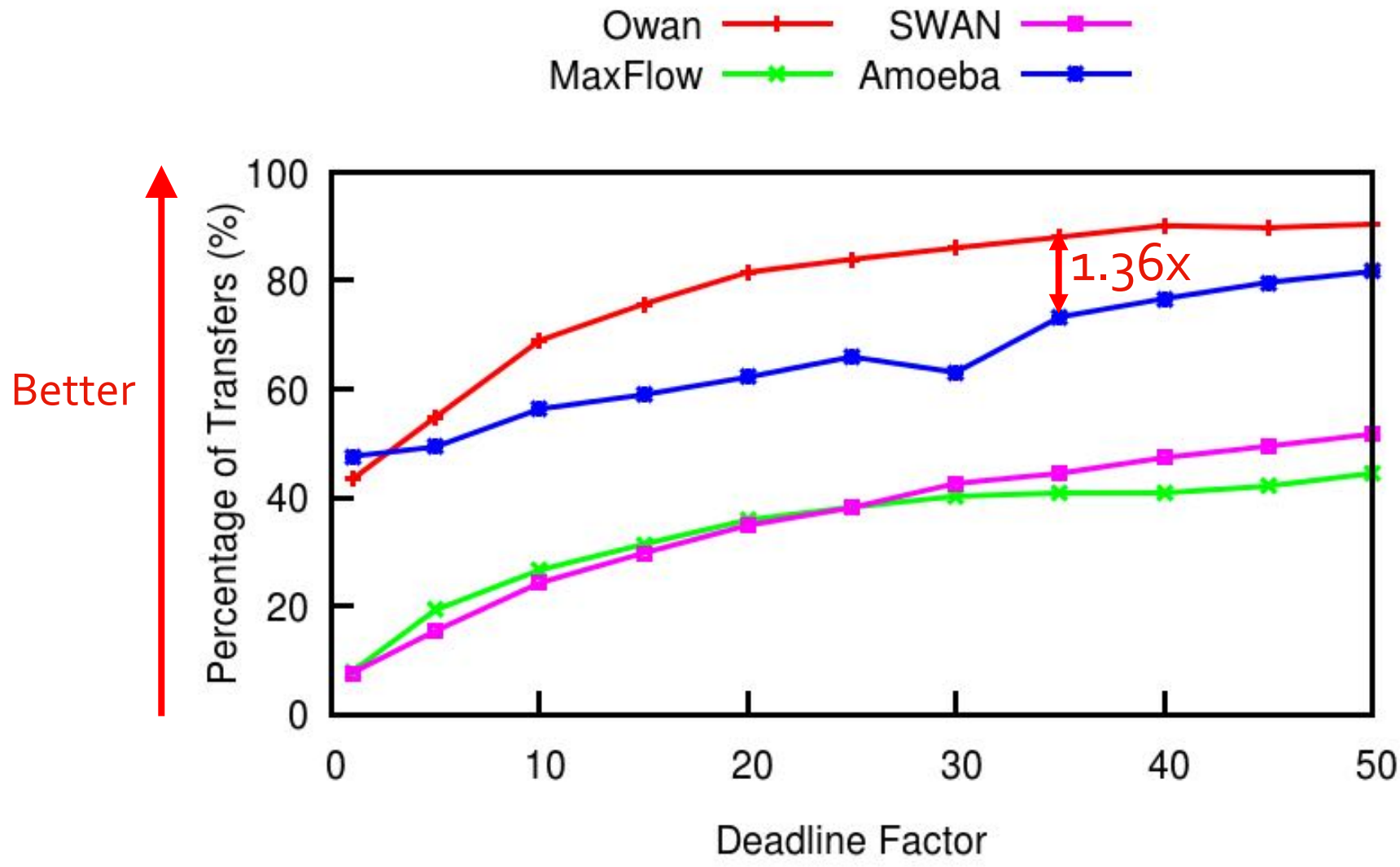
- 9 Sites
- Emulating Internet2 network
- 135 servers
  - Two 6-core Intel E5-2620v2
  - 10GE



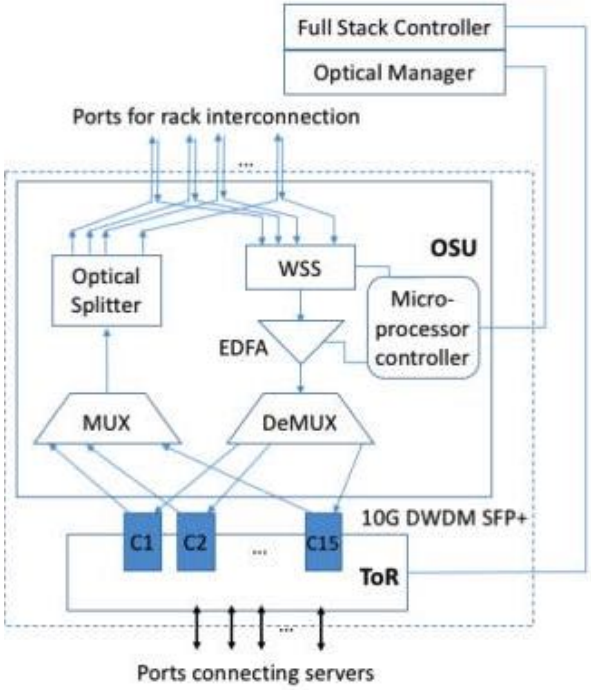
Better Average Completion Time



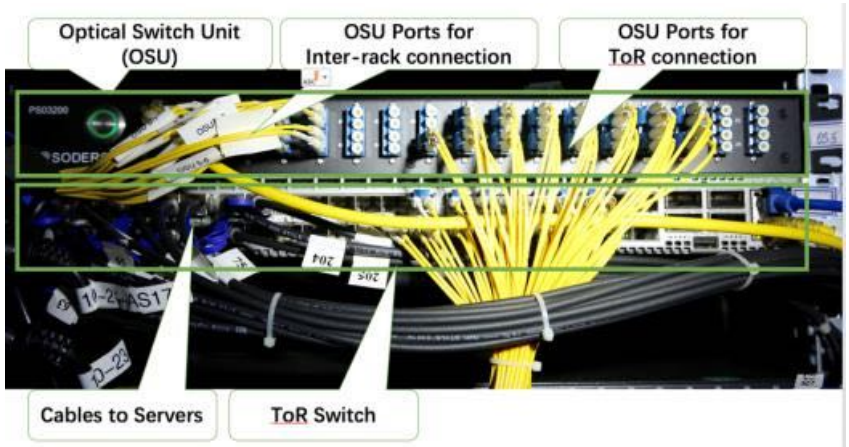
More Transfers Meet Deadlines



Used in data centers: Low-cost Optical Switching Unit (OSU)



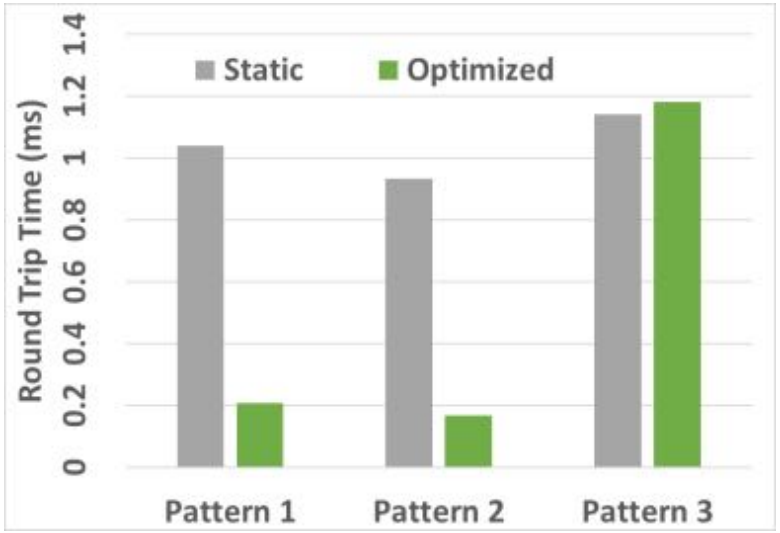
Built from off-the-shelf components



Results: Long Tail Latency Reduction

- Optimized topology vs. static topology
- Subset of 8 racks with three traffic patterns

- Pattern 1: Cross-network bulk data transfer
- Pattern 2: Two separate traffic intensive cliques, with limited traffic in between.
- Pattern 3: All-to-all uniformly distributed traffic



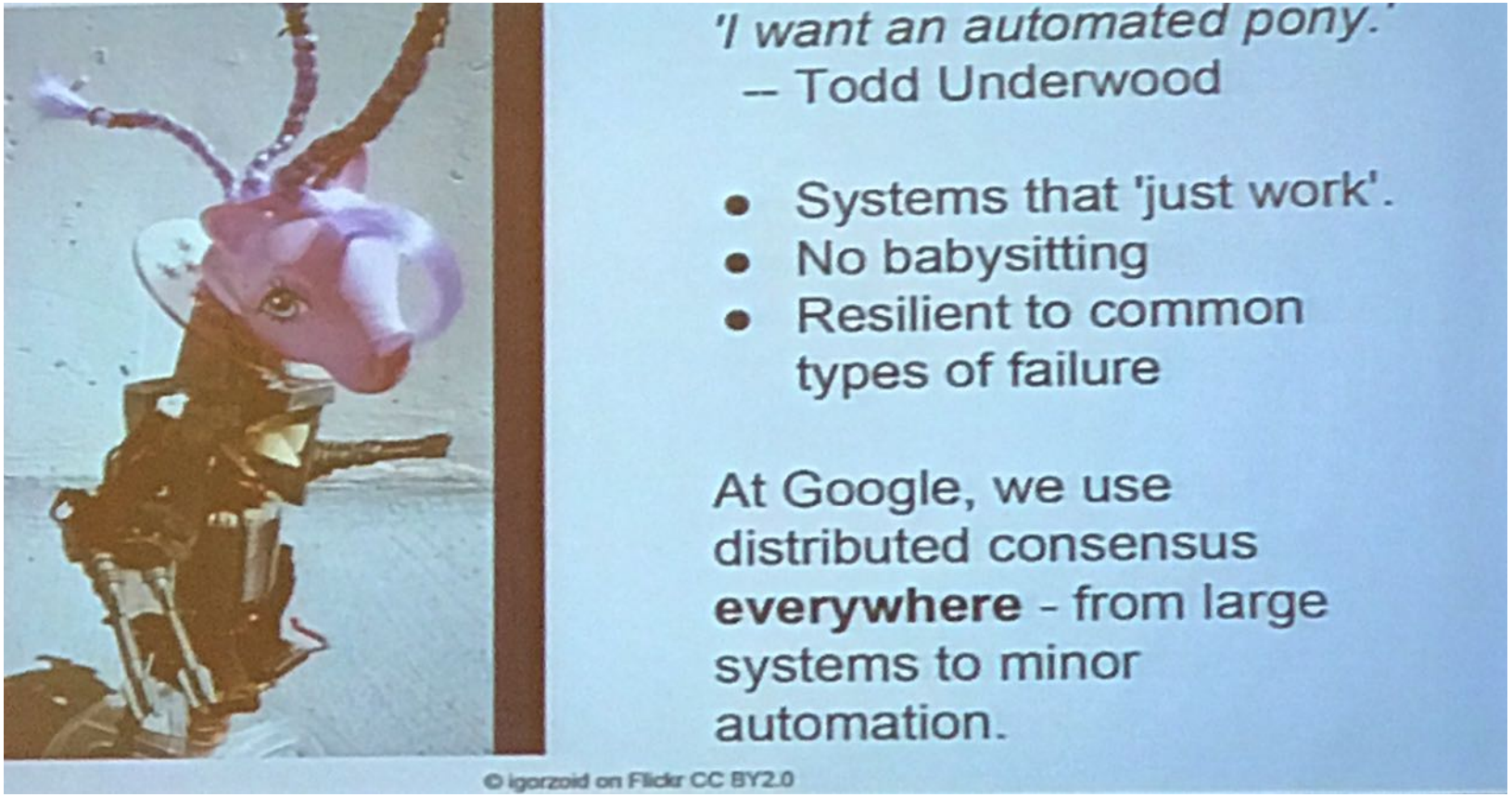
99th percentile of round trip time

46

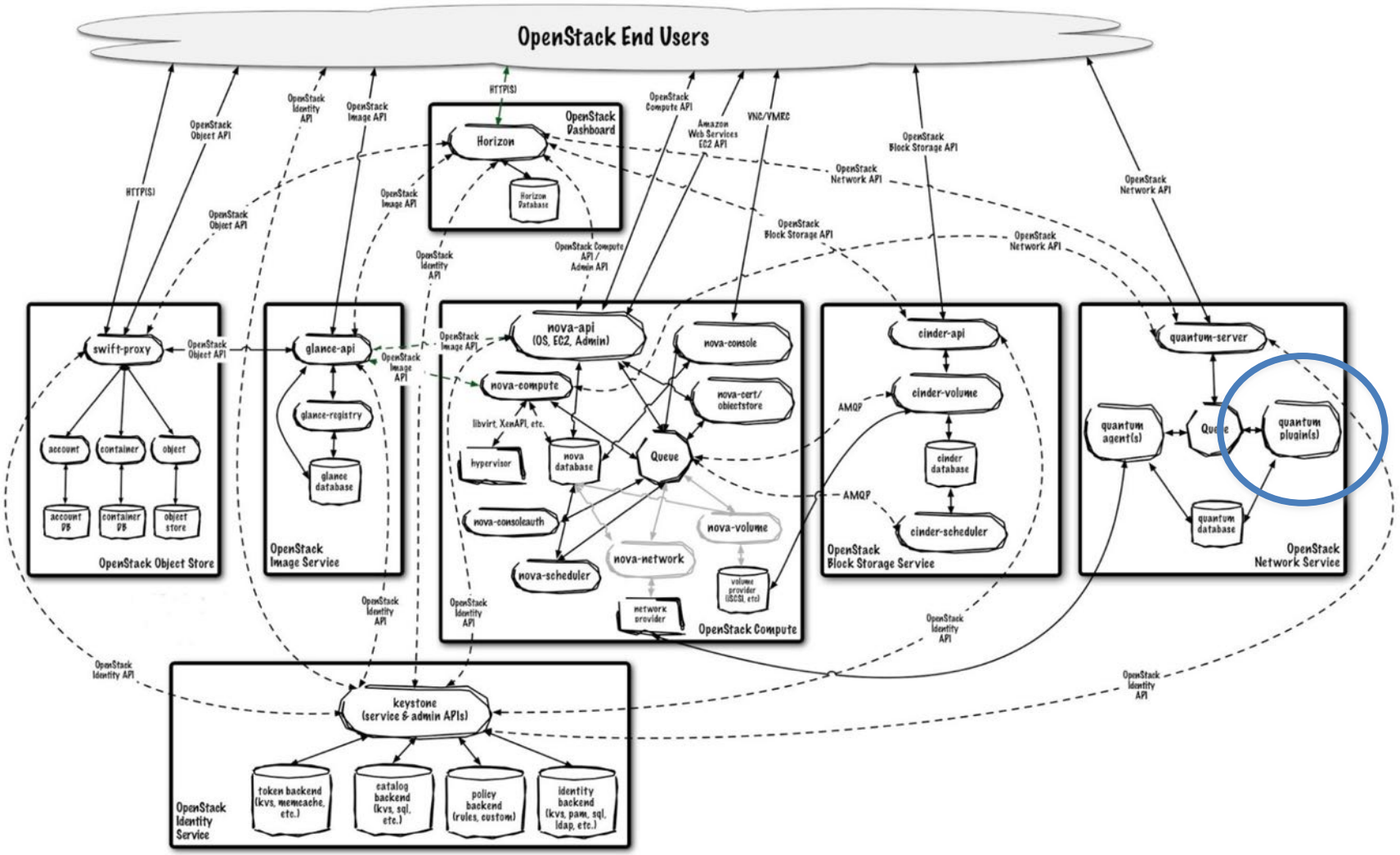
Networking: What is next?

- Networking becomes more like a distributed software system
  - Fabric functionalities provides flexibility and performance optimizations
  - All the jobs done in software
  - P4 even provides a compiler to program networks
- What is next?
  - Even more flexible network fabric
  - Consistency and transactions (like most modern OSes)
  - Software-based, global optimizations
  - Network operations -> dev-ops

Consensus is the key…



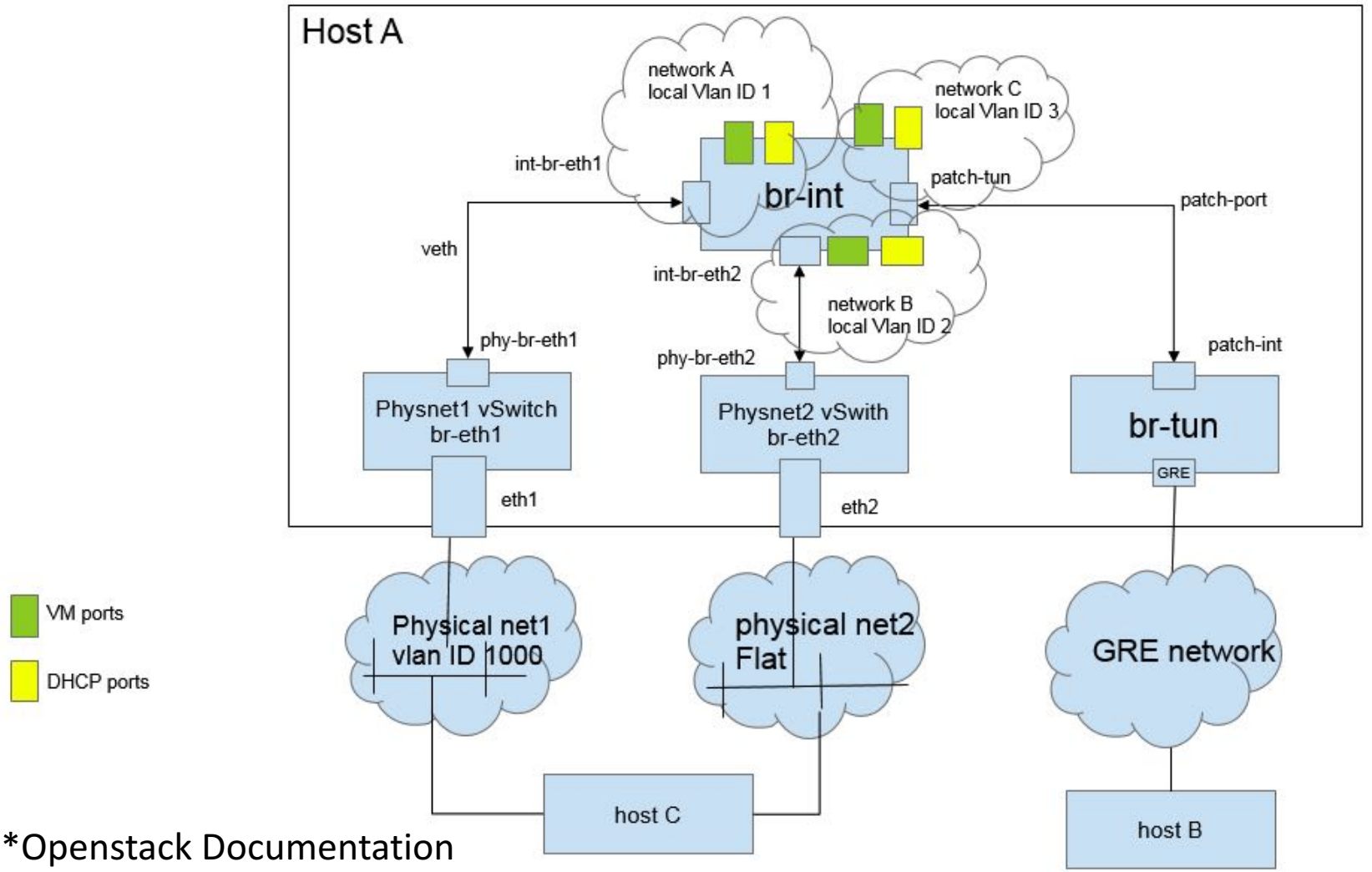
How many rules do we need?



*Openstack Documentation

49

How many rules do we need? (cont'd)

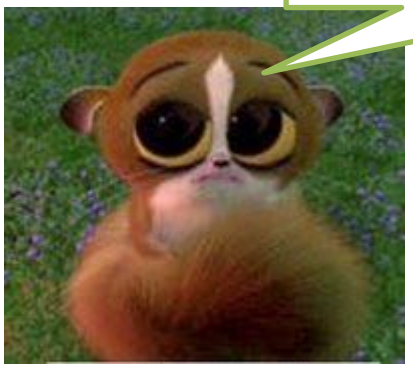


*Openstack Documentation

50

Trouble shooting for end users

My network is down!



User configuration? (attached NIC?)
Connected the Virtual Network to public?
Physical network down?
OVS down?
OVS agent down?
Network node down?
Floating IP not correctly configured?
Security group rules not set up correctly?
……

How many rules needs to know as a professional openstack operator?

The operational knowledge does not transfer!

Good news for IT consulting business.

Key idea 1: Using natural language to query about system states

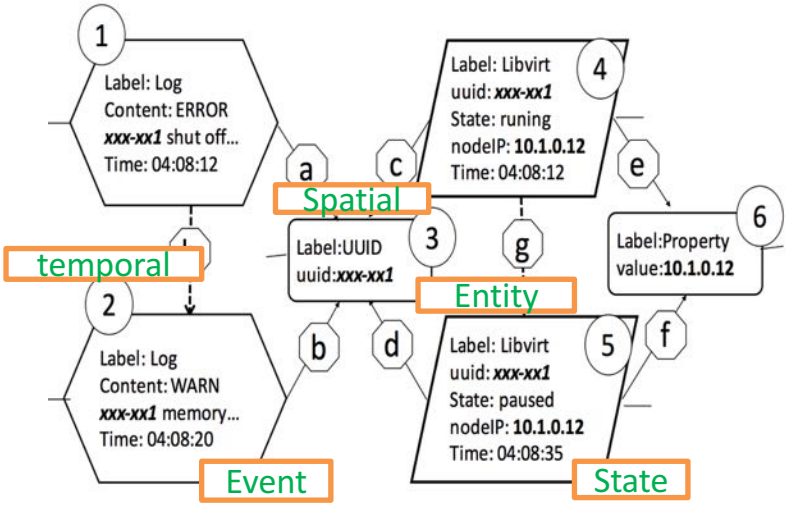
- No longer need to remember inconsistent terminology
- No longer to understand obscure correlations among different components

- Something better than
  - Command line arguments
  - Regular expressions

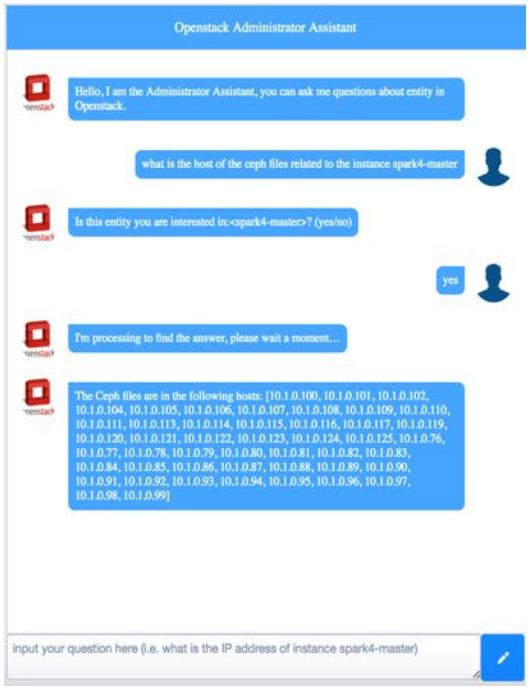
- We can do it as we accumulate a number of FAQs from users and system administrators too …

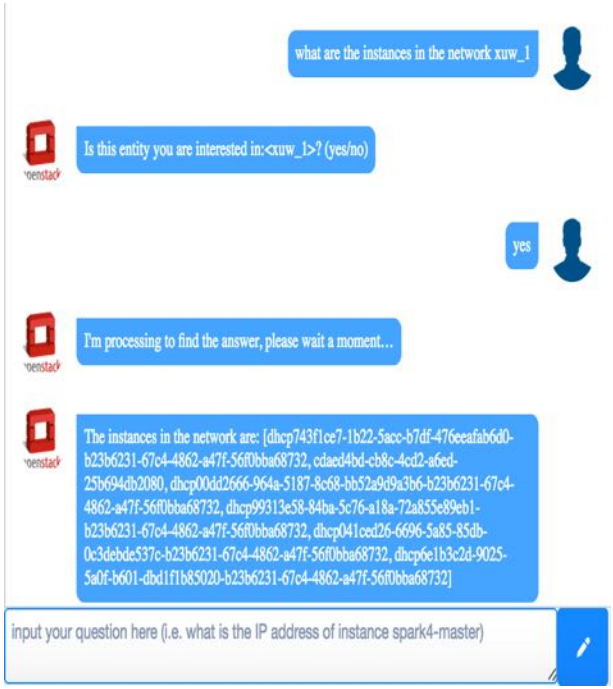
Key idea 2: automatically discover knowledge in systems using most basic rules

- We can capture the knowledge: System Operation State Graph (SOSG)
- Turn ad-hoc system state queries into a uniform graph traversal.
- Anomaly detection to find hidden problems.



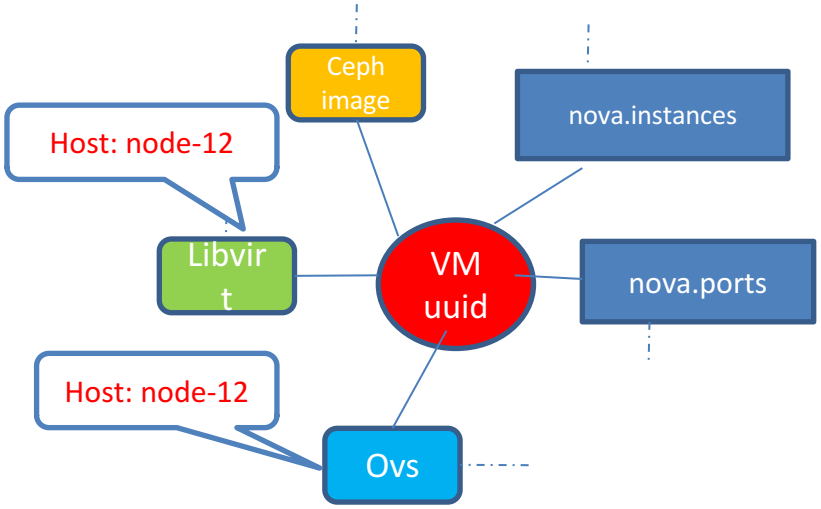
What do we build?



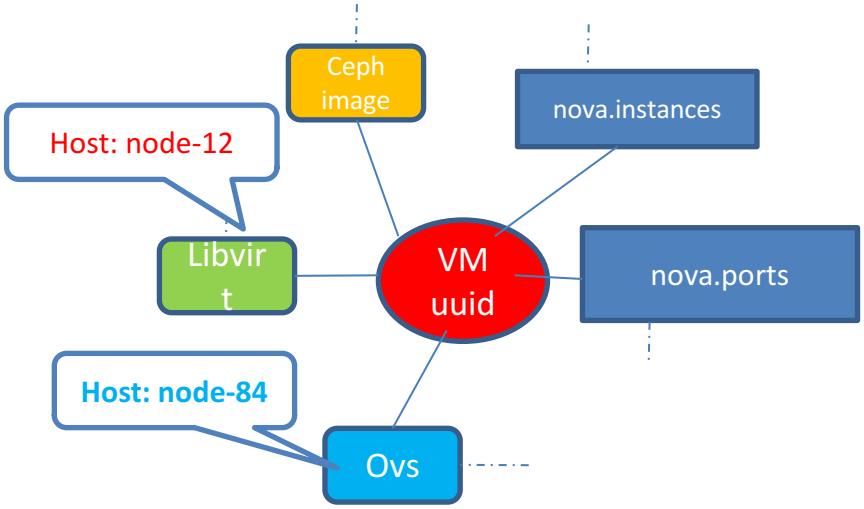


- Responds in ~10s on average

What we can do?
-- Anomaly detection

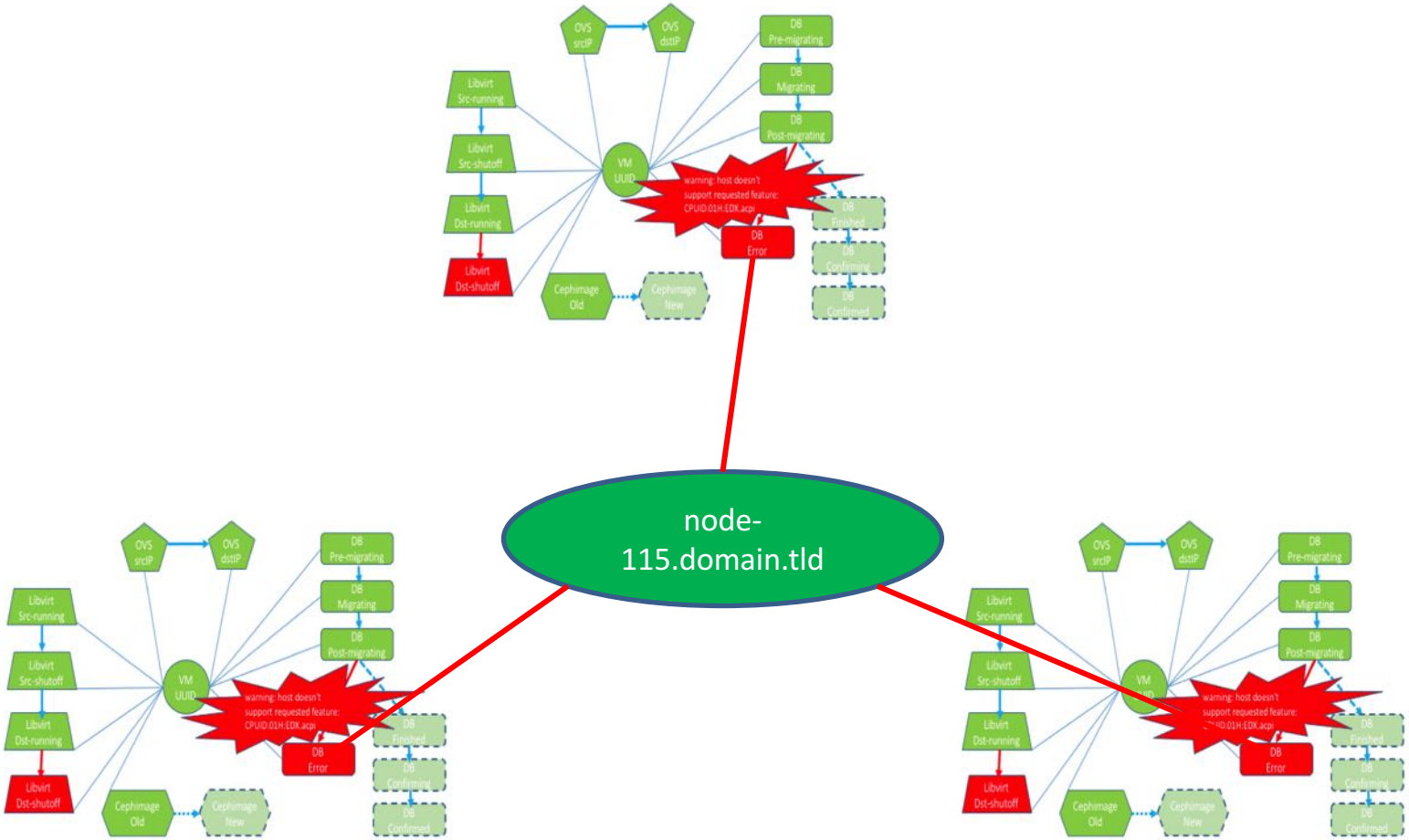


Normal case



Anomalous case

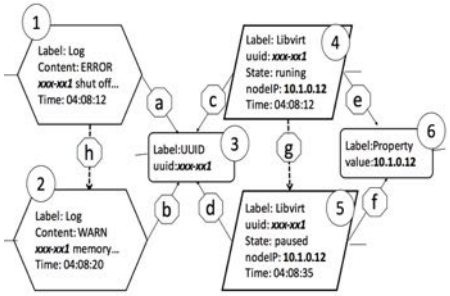
What we can do?
-- Correlation of failures



How we did this?

Raw States

Logs



Automatically generated
“Knowledge graph” for OpenStack

Summary

- Think everything as cross-layer
  - Hardware – software – operation
  - *Do not over emphasis on reliability in a single layer*
- The new trends of infrastructure technologies
  - Ops -> DevOps
  - Scripts / commands -> AI and natural language
- *Consensus* is the key
- Other challenges in systems reliability
  - Trust and privacy over multiple parties
  - Layers above the IaaS