Diagnosing Performance Changes by Comparing Request Flows

Raja Sambasivan, et al.

Presented by
Mingrui Zhang(1110379057)
Hongxu Chen(1110379002)

January 3, 2014

Part I

Introduction

Outline

Difficulties

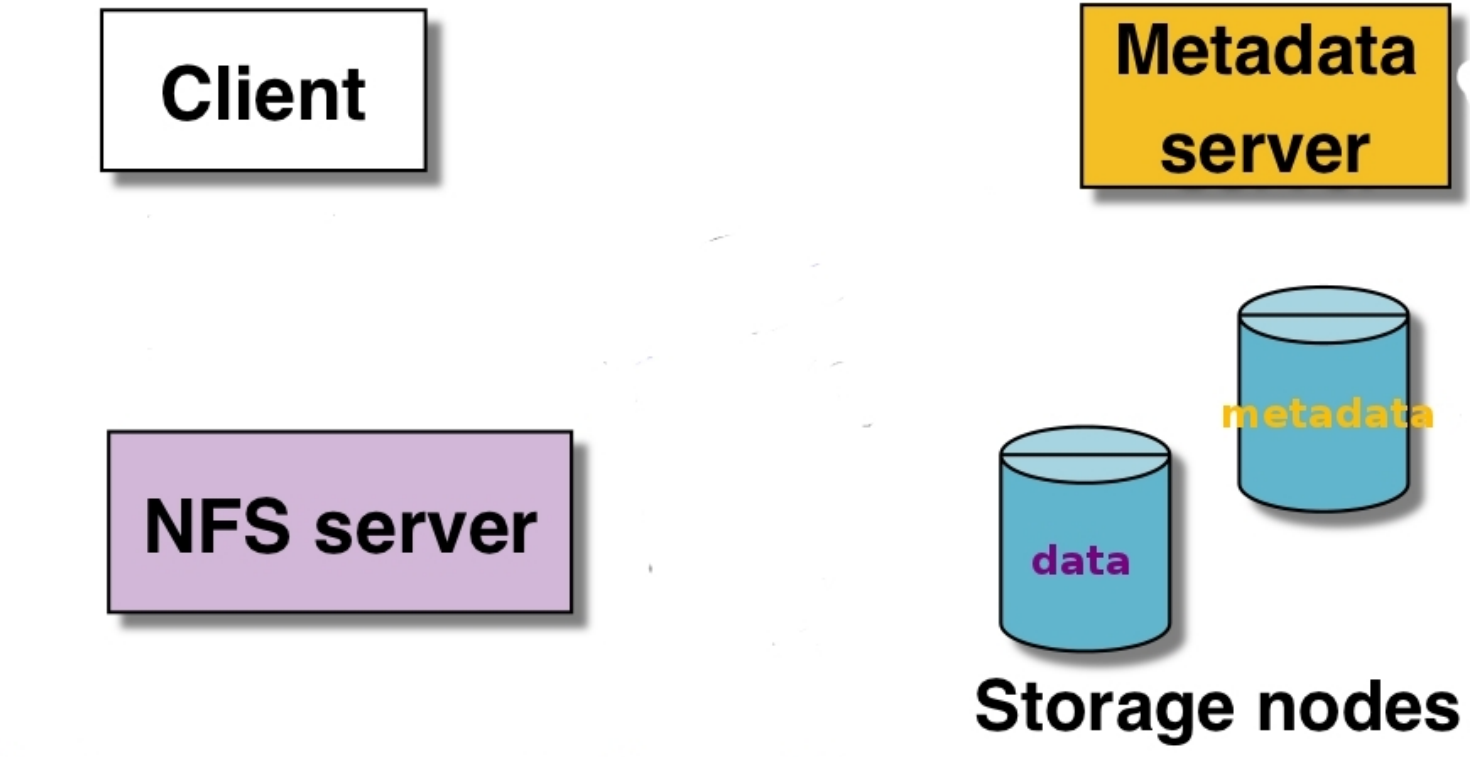
Performance problems in distributed systems

- May have many root causes
- May be contained in any one or more of the component processes
- May emerge from the interactions among component processes

An Example

Example

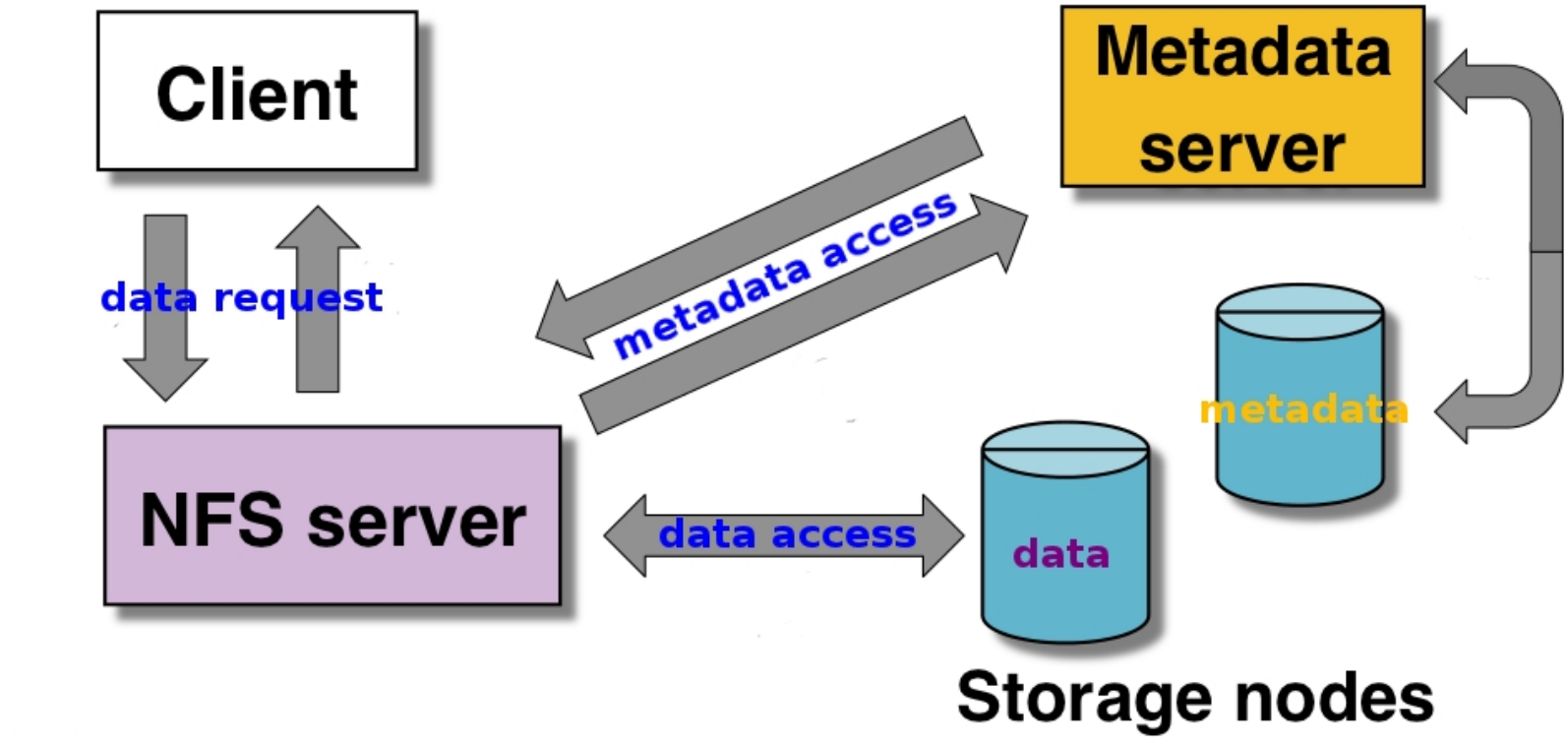
Debugging a feature addition



An Example

Before addition

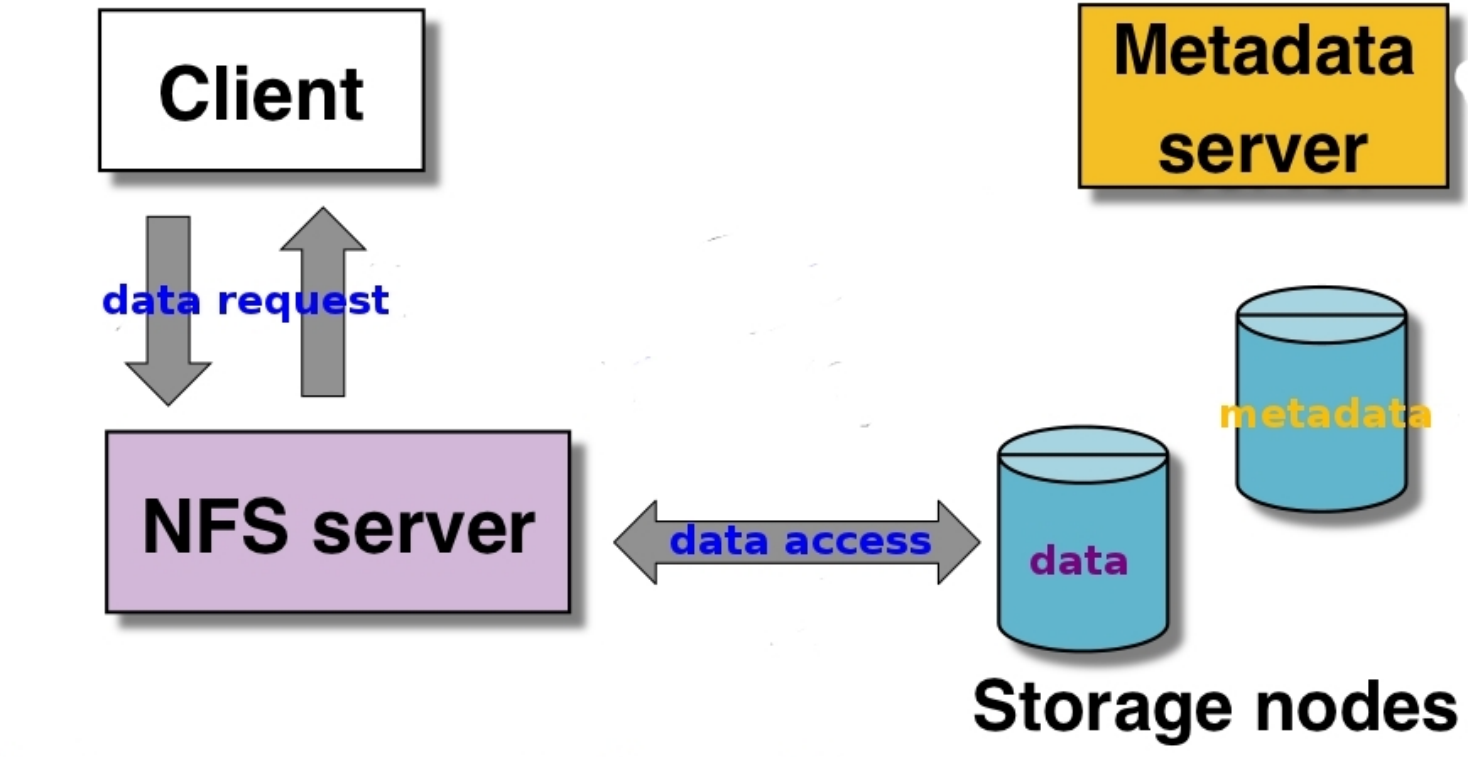
Every file access needs a MDS access



An Example

After: Metadata prefetched to clients

Most requests *don't* need MDS access



Debugging a feature addition

- Adding metadata prefetching reduced performance instead of improving it
- How to efficiently diagnose this?

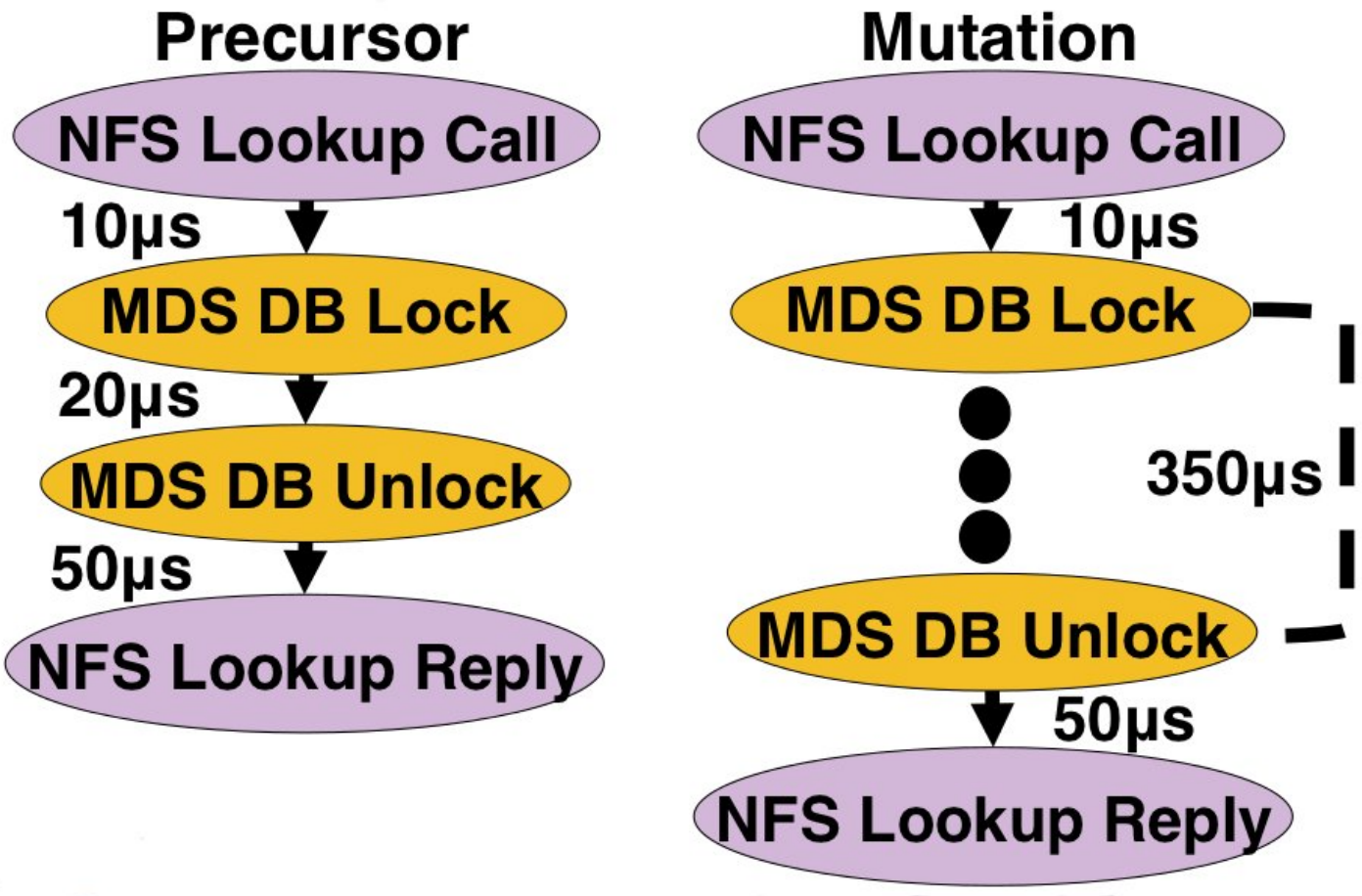
Debugging a feature addition

- Adding metadata prefetching reduced performance instead of improving it
- How to efficiently diagnose this?

Outline

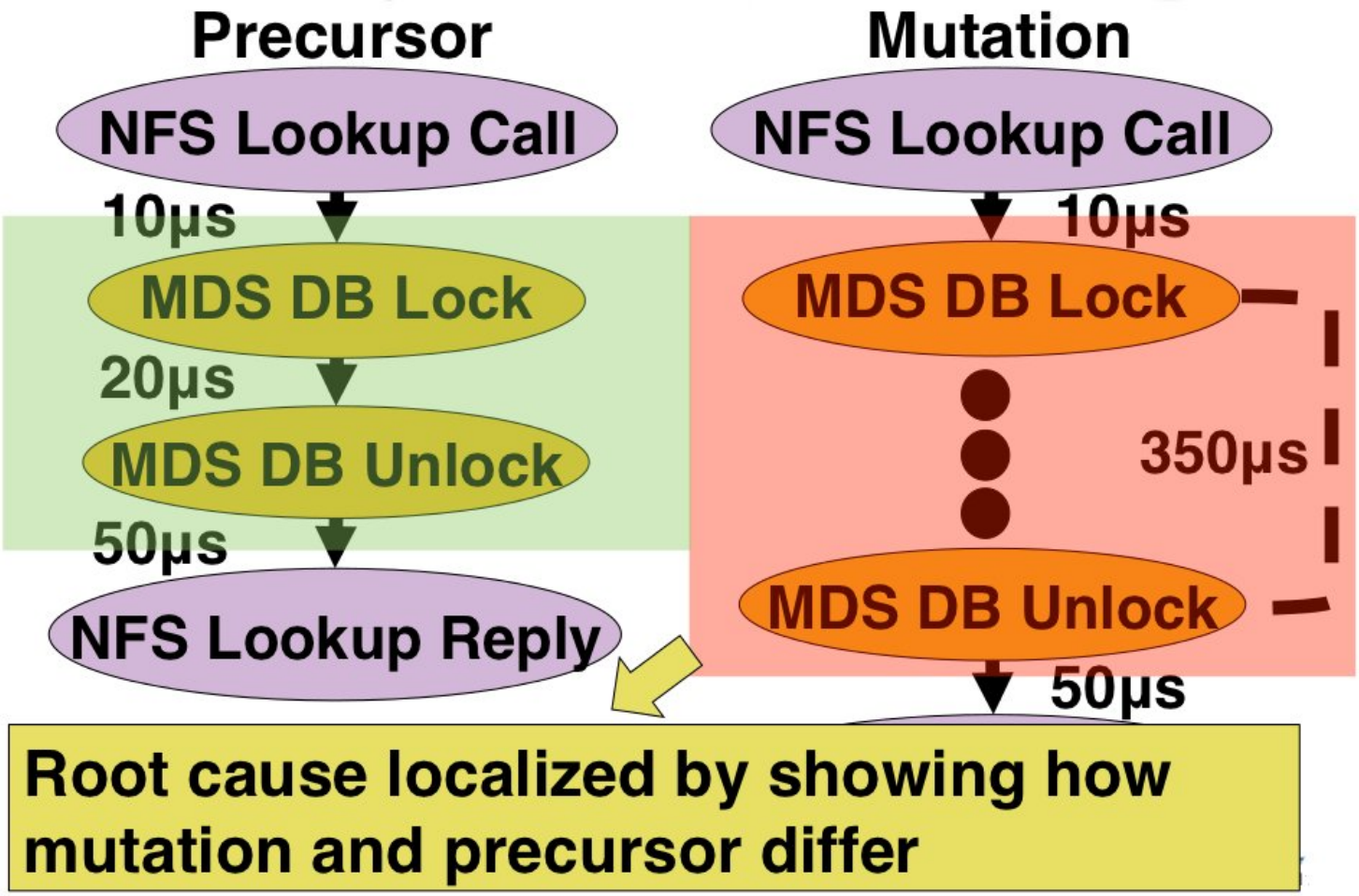
Request-flow comparison will show

Jump to evaluation



Request-flow comparison will show

Jump to evaluation



Request Flow Comparison

- Identifies distribution changes
  - ☞ Distinct from anomaly detection
    - E.g. Magpie, Pinpoint, etc.
- Satisfies many use cases
  - ☞ Performance regressions/degradations
  - ☞ Eliminating the system as the culprit

Request Flow Comparison

- Identifies distribution changes
  - ☞ Distinct from anomaly detection
    - E.g. Magpie, Pinpoint, etc.
- Satisfies many use cases
  - ☞ Performance regressions/degradations
  - ☞ Eliminating the system as the culprit

Contributions

- Heuristics for identifying mutations,precursors, and for ranking them
  - ✍ Implementation in Spectroscope
- Use of Spectroscope to diagnose
  - ✌ Unsolved problems in Ursa Minor
  - ✌ Problems in Google services

Contributions

- Heuristics for identifying mutations,precursors, and for ranking them
  - ✍ Implementation in Spectroscope
- Use of Spectroscope to diagnose
  - ✌ Unsolved problems in Ursa Minor
  - ✌ Problems in Google services

Part II

Spectroscope Workflow

Outline

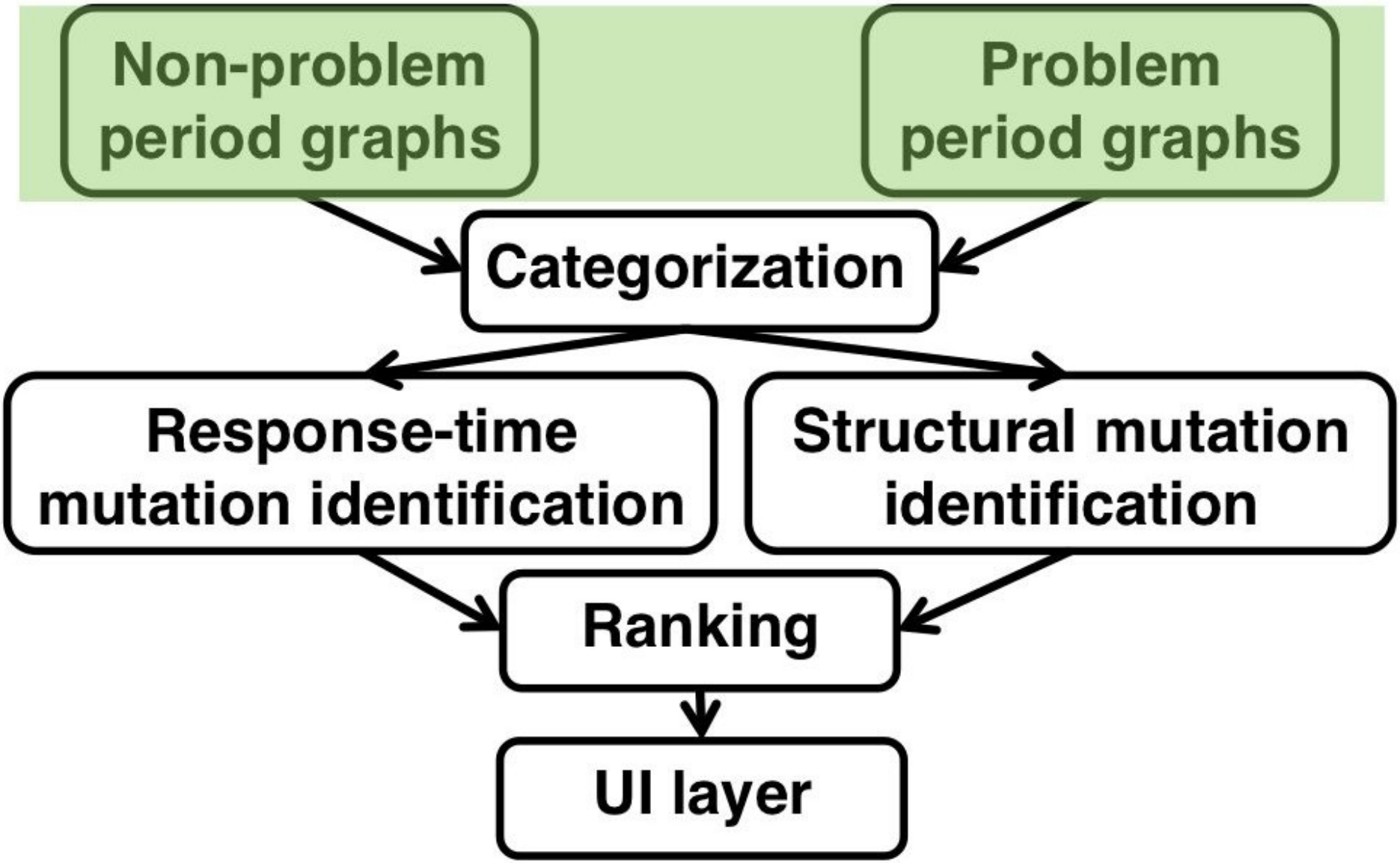
Overview

- Input: two periods of activity
  1. non-problem period graph
  2. a problem period graph
- Categrazation
  1. Response time mutation
  2. Structural mutation and precursor
- Ranking by expected contribution to the performance change
- Visulation

Figure: Spectroscope's workflow for comparing request flows

Outline

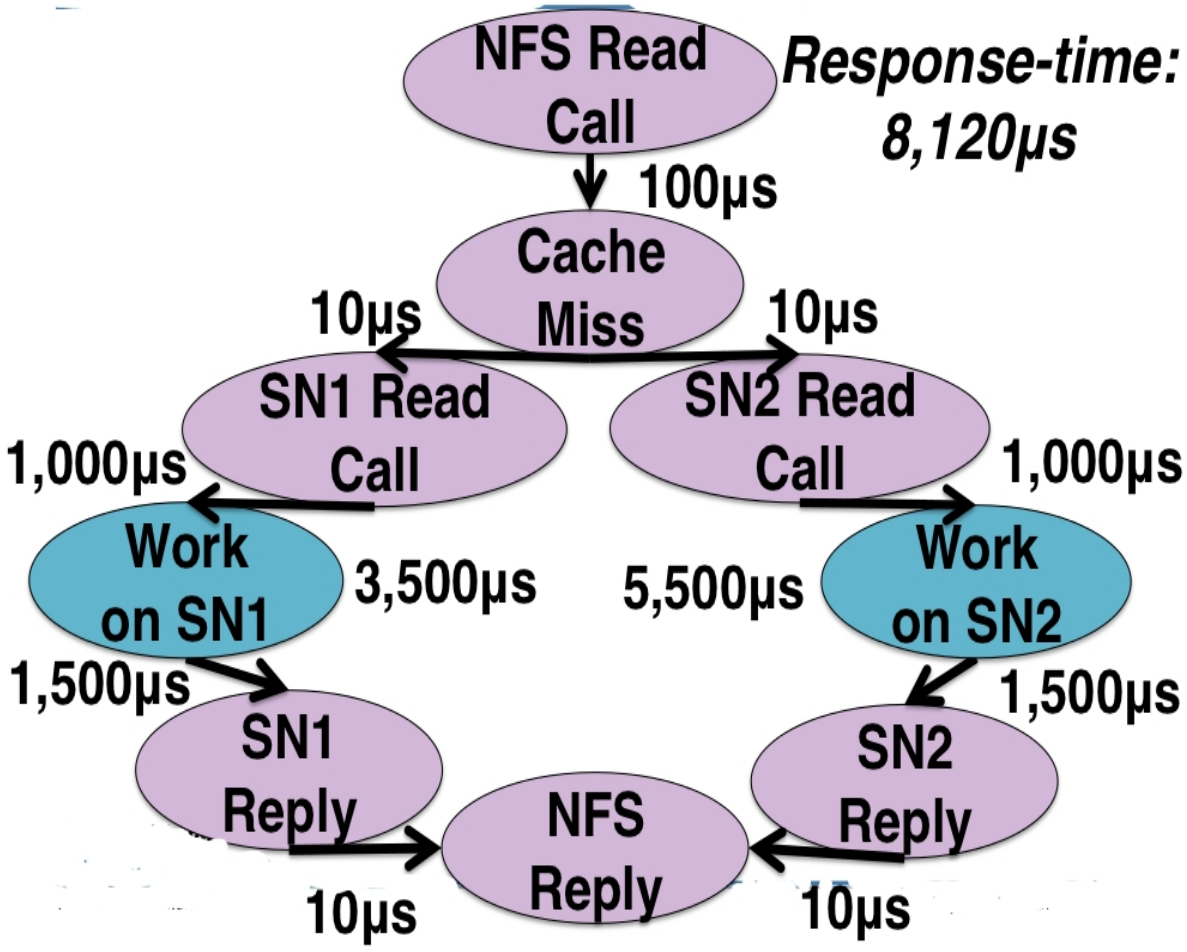
Spectroscope workflow (i)



Graphs via end-to-end tracing

- Used in research & production systems
- Works as follows:
  1. Tracks trace points touched by requests
  2. Request-flow graphs obtained by stitching together trace points accessed
- Yields $< 1\%$ overhead request sampling

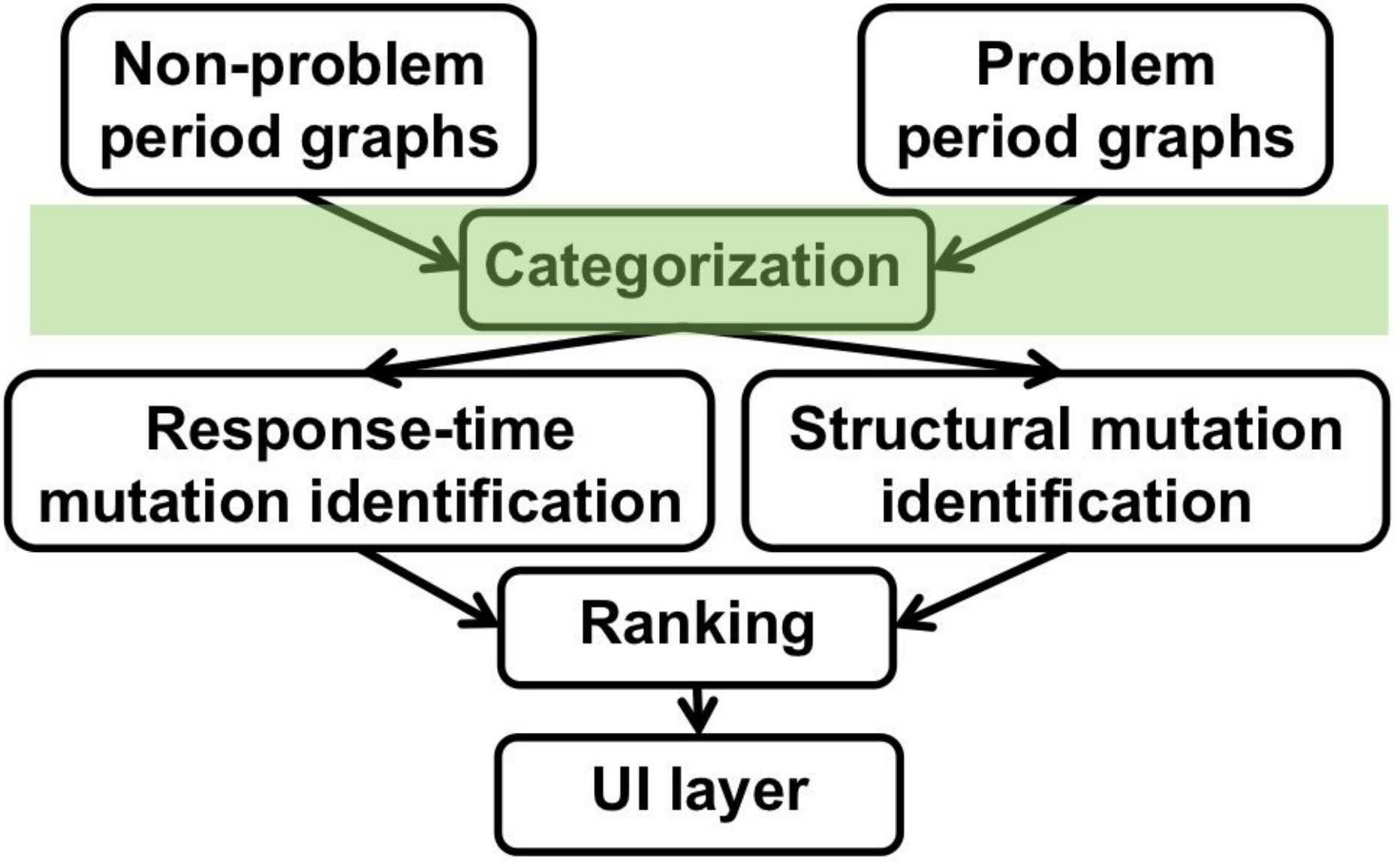
Example: Graph for a striped read



- Node:trace point
- Edge:latency

Outline

Spectroscope workflow (ii)



Categorization

- Problem:Even small distributed systems can service hundreds to thousands of requests per second, so comparing all of them individually is not feasible.
- It is meaningless to compare individual requests flows
- Groups together similar request flows
  1. Categories: basic unit for comparisons
  2. Allows for mutation identification by comparing per-category distributions

Categorization

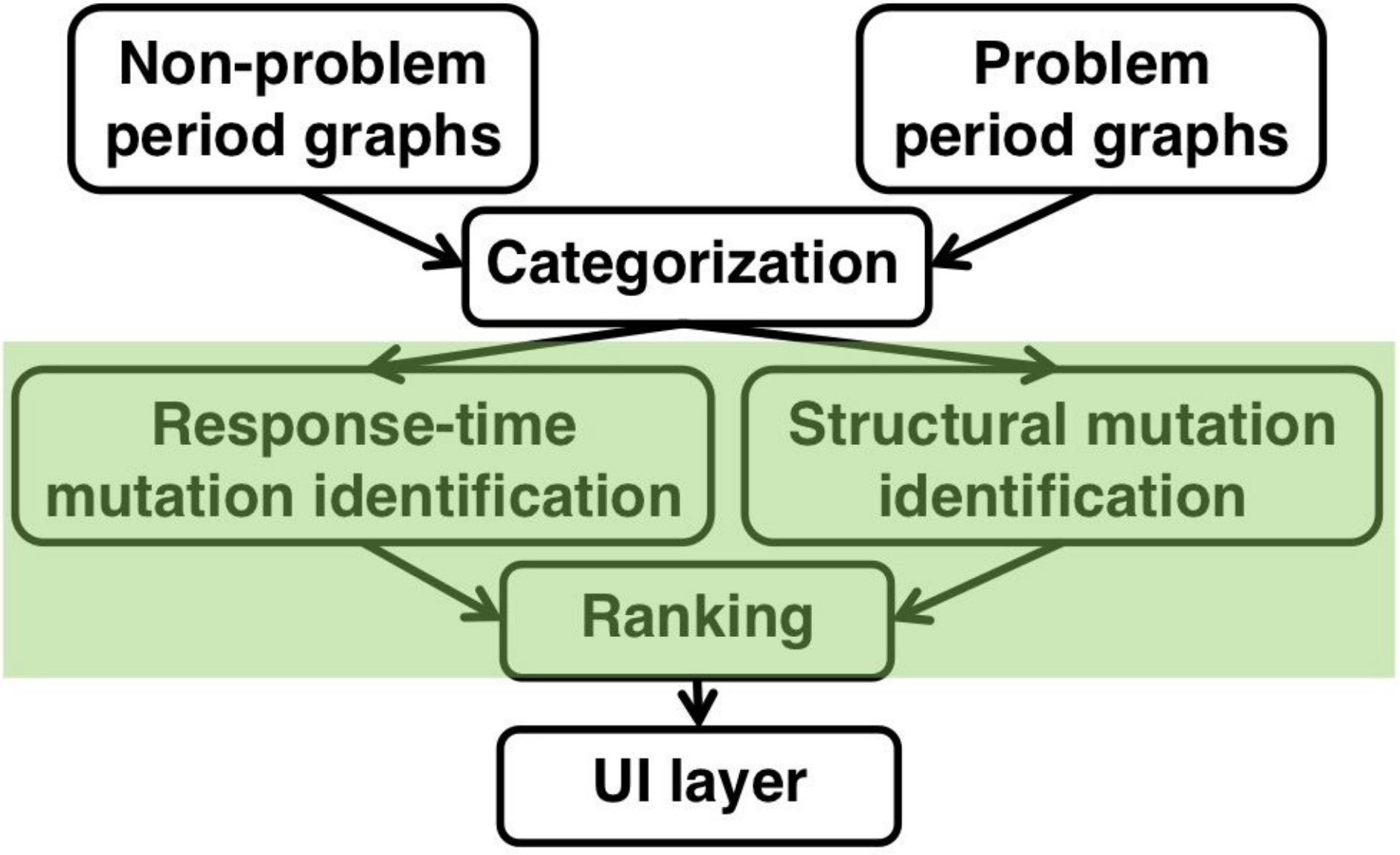
- Problem:Even small distributed systems can service hundreds to thousands of requests per second, so comparing all of them individually is not feasible.
- It is meaningless to compare individual requests flows
- Groups together similar request flows
  1. Categories: basic unit for comparisons
  2. Allows for mutation identification by comparing per-category distributions

What to bin into a category?

- Identically structured requests
  - Uses same path/similar cost expectation
- Same path/similar costs notion is valid
  - For $88 - 99\%$ of Ursa Minor categories
  - For $47 - 69\%$ of Bigtable categories
    1. Lower value due to sparser trace points
    2. Lower value also due to contention

Outline

Spectroscope workflow (iii)



Types of Mutations

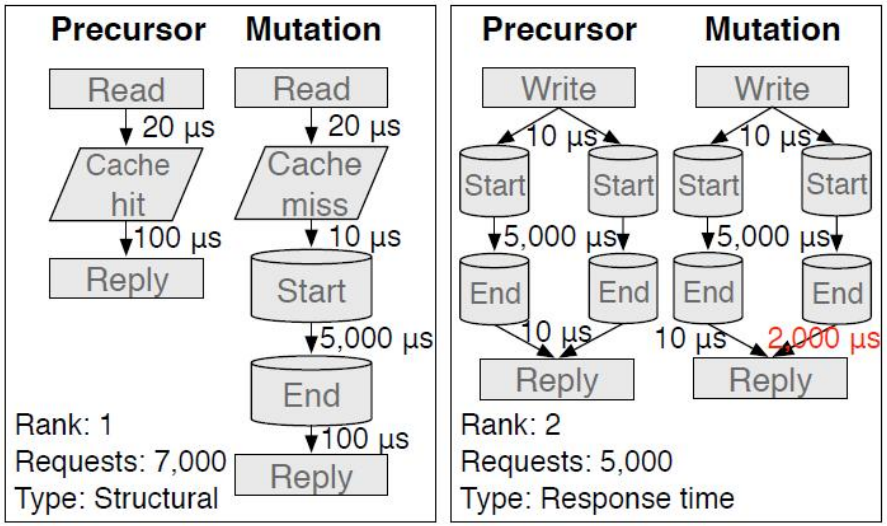


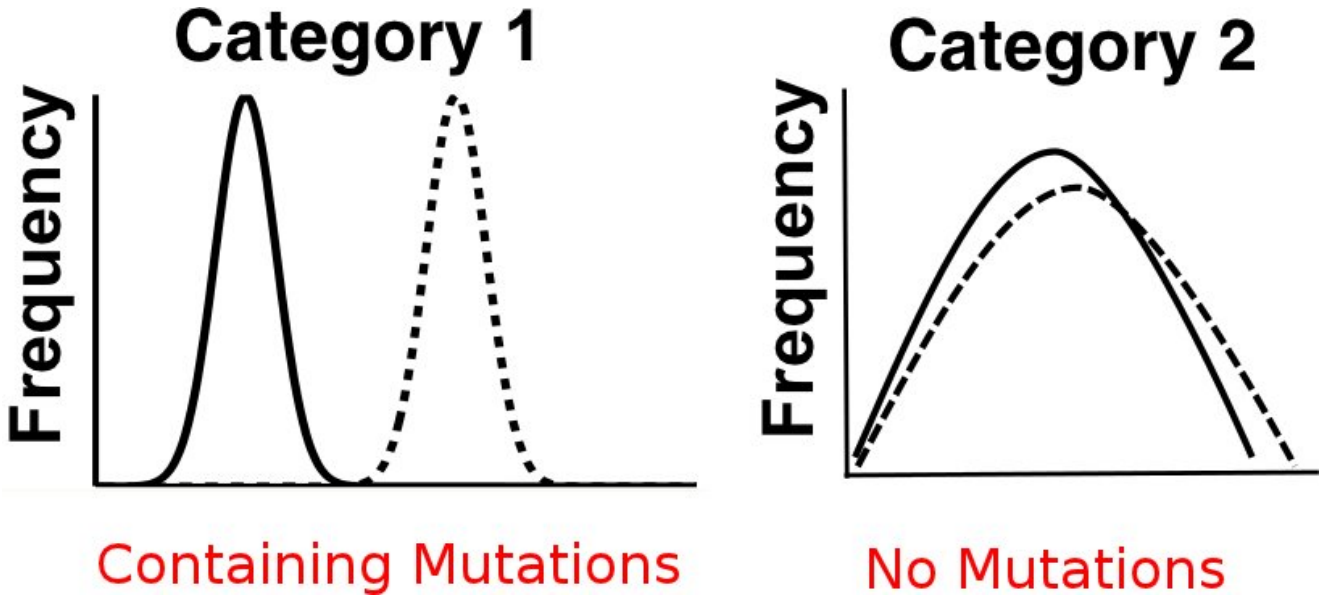
Figure: Types of Mutations

- response-time mutations
  - ☞ increased cost between the periods
  - ☞ same structure,different response time
  - — Root cause localized by identifying interactions responsible
- structural mutations
  - ☞ different paths through the system in the problem period
  - — Root cause localized by:
    1. identifying their precursors
    2. identifying how mutation & precursor differ

Identifying response-time mutations

use of Kolmogorov-Smirnov two-sample, non-parametric hypothesis test

- Sets apart natural variance from mutations
- Also used to find interactions responsible



Identifying structural mutations

- Assume similar workloads executed
  1. Categories with more problem period requests contain mutations
  2. Reverse true for precursor categories
- Threshold used to differentiate natural variance from categories mutations

Mapping mutations to precursors

- eliminate different root nodes in precursor categories
- remove precursor categories having decreased in request count less than the increase in request count of the structure-mutation category
- rank remaining precursor categories according to likelihood of having donated requests

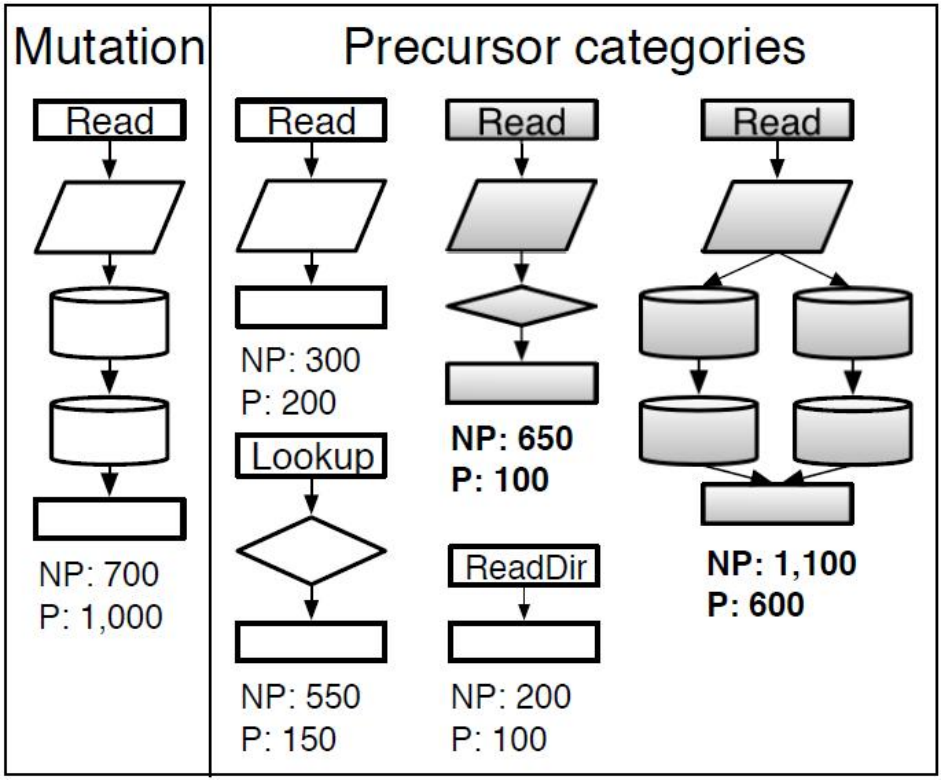
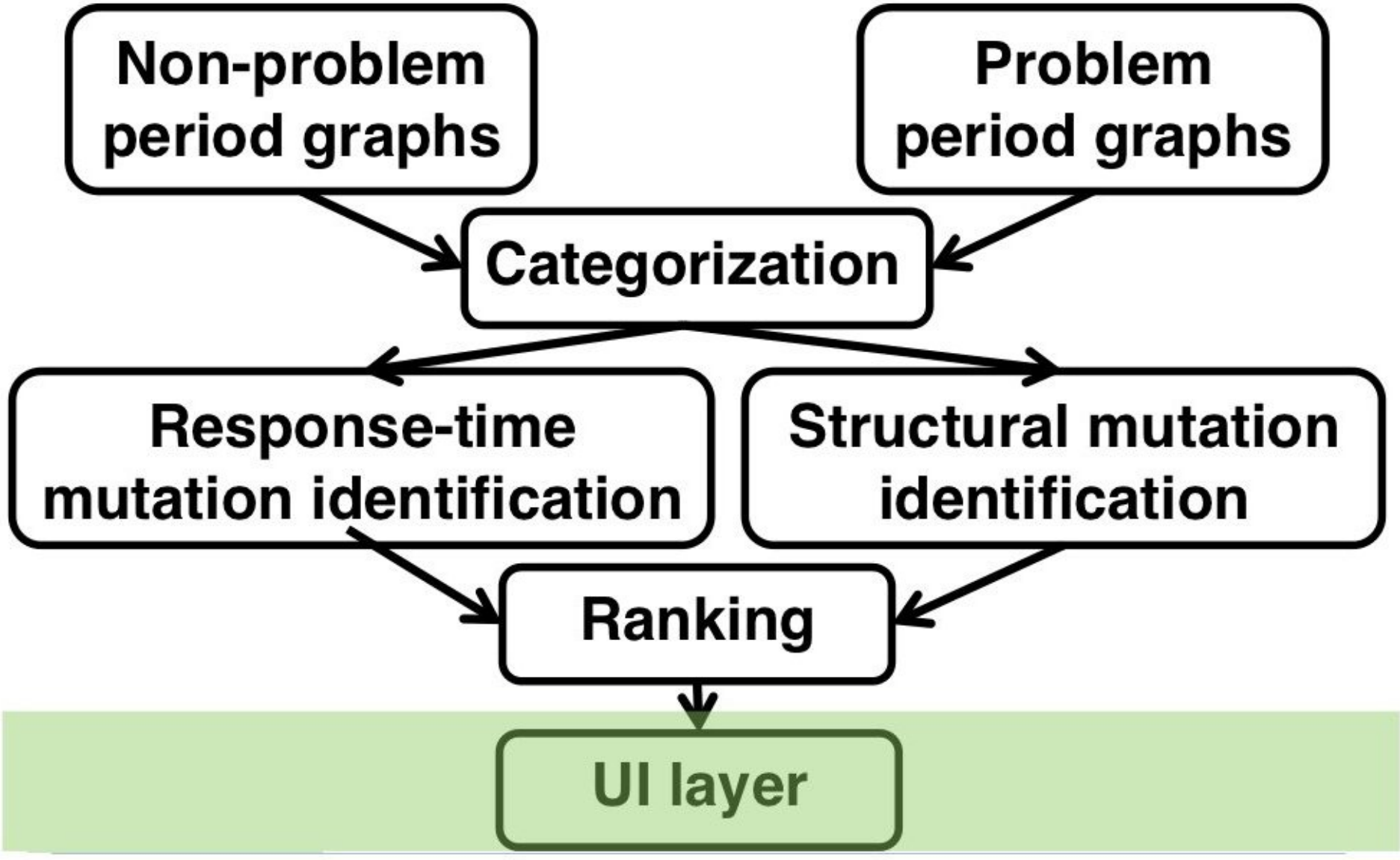


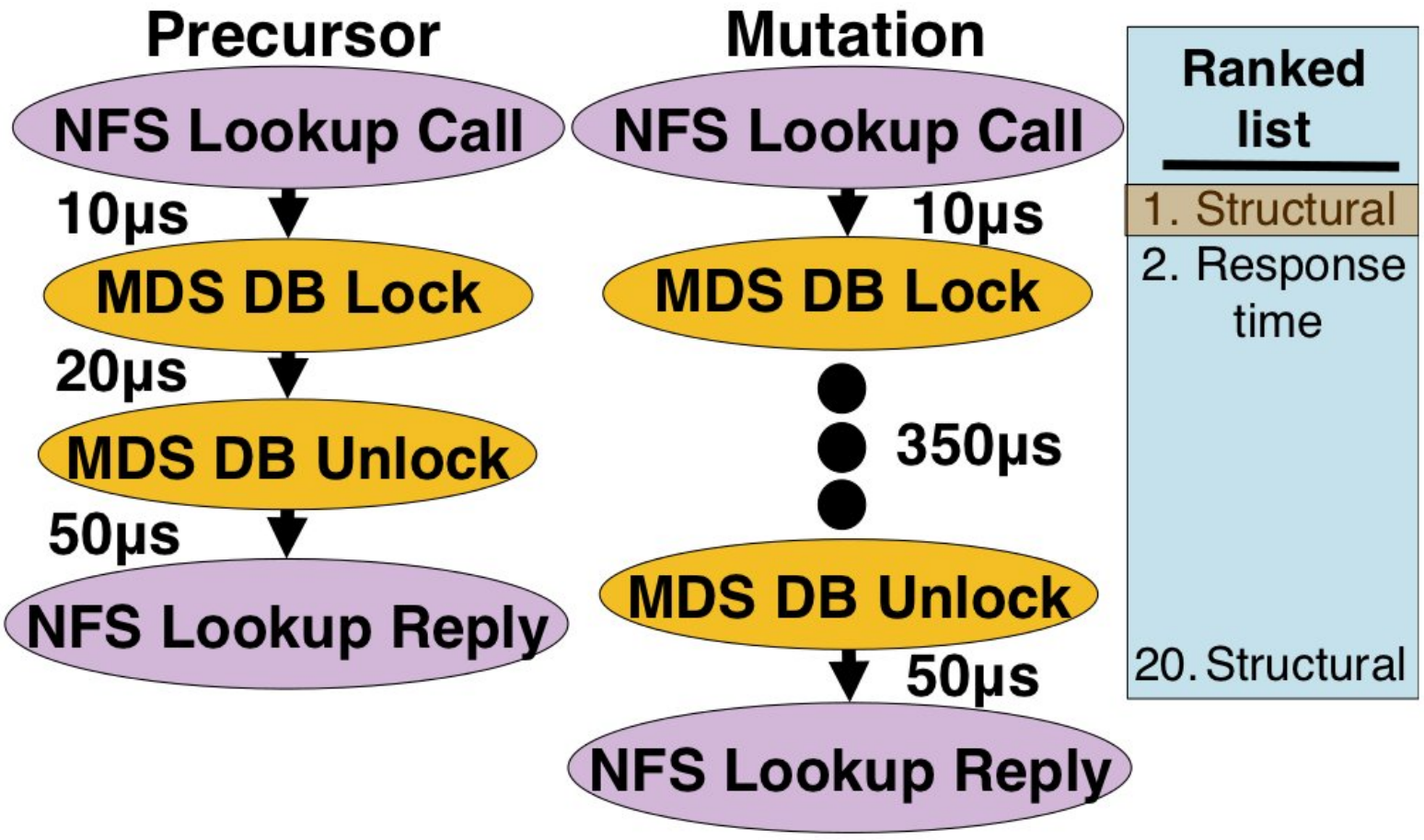
Figure: Only two categories are reserved

Outline

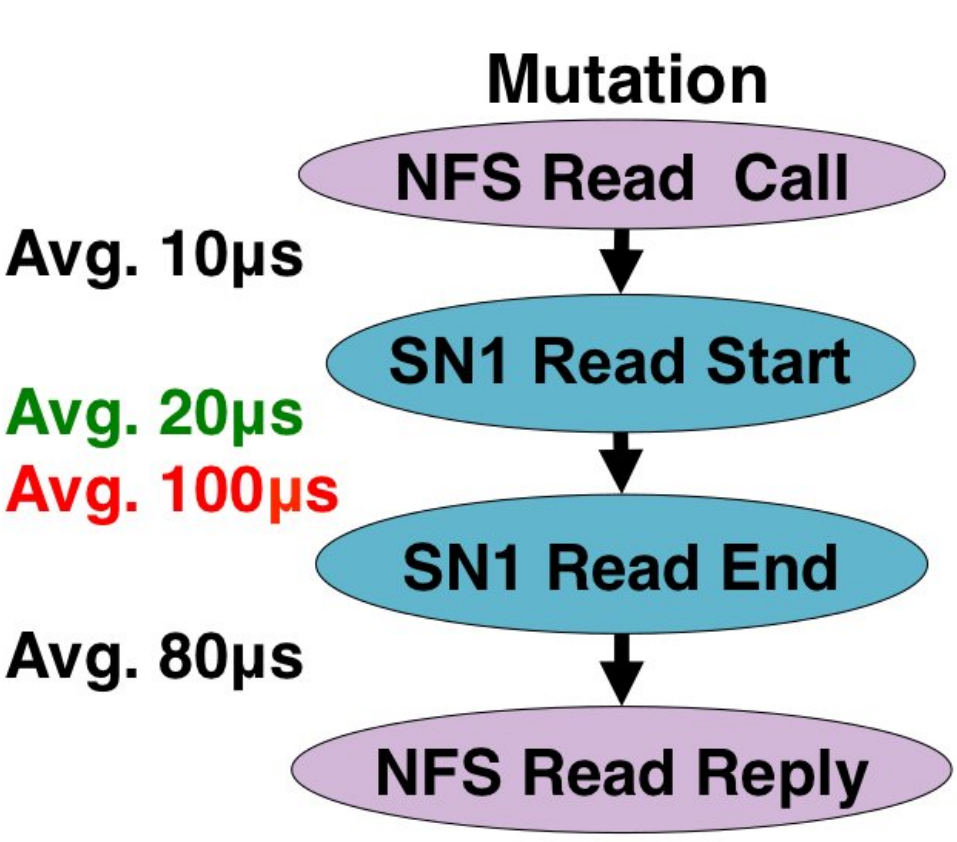
Spectroscope workflow (iv)

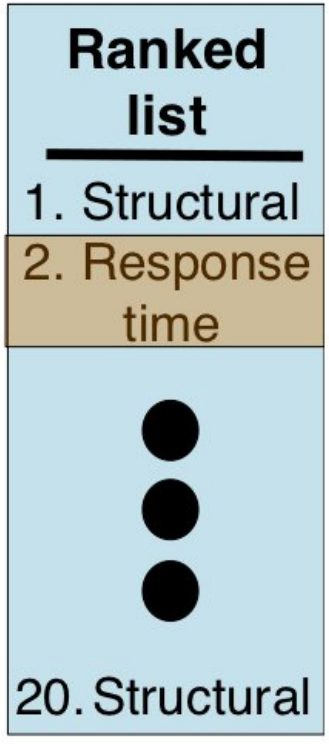


UI layer


Precursor
NFS Lookup Call
10µs
MDS DB Lock
20µs
MDS DB Unlock
50µs
NFS Lookup Reply
Mutation
NFS Lookup Call
10µs
MDS DB Lock
350µs
MDS DB Unlock
50µs
NFS Lookup Reply
Ranked list
1. Structural
2. Response time
20. Structural

UI layer


Mutation
NFS Read Call
Avg. 10µs
SN1 Read Start
Avg. 20µs
Avg. 100µs
SN1 Read End
Avg. 80µs
NFS Read Reply


Ranked list
1. Structural
2. Response time
20. Structural

Part III

Experiemental appratus

Outline

Methodoly

Three complementary metrics provided for evaluation

- The percentage of the 10 highest-ranked categories that are relevant($Top\ 10rel.$)
- The percentage of false-positive categories(FPs)
- Request coverage($Cov.$)

Overview of the Ursa Minor case studies

Name	Manifestation	Root cause	# of results	*Top 10 rel.* %	*FPs* %	*Cov.*%
MDS config	Structural	Config. change	128	100	2	2
RMWs	Structural	Env. change	3	100	0	100
MDS prefetch. 50	Structural	Internal change	7	29	71	93
MDS prefetch. 10	Structural	Internal change	16	70	56	96
Create behaviour	Structural	Design problem	11	40	64	N/A
$100\mu s$ delay	Response time	Internal change	17	0	100	0
$500\mu s$ delay	Response time	Internal change	166	100	6	92
$1ms$ delay	Response time	Internal change	178	100	7	93
Periodic spikes	No change	Env. change	N/A	N/A	N/A	N/A

Overview of the Ursa Minor case studies

Name	Manifestation	Root cause	# of results	*Top 10 rel.* %	*FPs* %	*Cov.*%
MDS config	Structural	Config. change	128	100	2	2
RMWs	Structural	Env. change	3	100	0	100
MDS prefetch. 50	Structural	Internal change	7	29	71	93
MDS prefetch. 10	Structural	Internal change	16	70	56	96
Create behaviour	Structural	Design problem	11	40	64	N/A
$100\mu s$ delay	Response time	Internal change	17	0	100	0
$500\mu s$ delay	Response time	Internal change	166	100	6	92
$1ms$ delay	Response time	Internal change	178	100	7	93
Periodic spikes	No change	Env. change	N/A	N/A	N/A	N/A

MDS [1] configuration change

- used Spectroscope to compare request flows between two runs of *postmark-large*, one from before the check-in and one from after.

[1] Metadata Server

Overview of the Ursa Minor case studies

Name	Manifestation	Root cause	# of results	*Top 10 rel.* %	*FPs* %	*Cov.*%
MDS config	Structural	Config. change	128	100	2	2
RMWs	Structural	Env. change	3	100	0	100
MDS prefetch. 50	Structural	Internal change	7	29	71	93
MDS prefetch. 10	Structural	Internal change	16	70	56	96
Create behaviour	Structural	Design problem	11	40	64	N/A
$100\mu s$ delay	Response time	Internal change	17	0	100	0
$500\mu s$ delay	Response time	Internal change	166	100	6	92
$1ms$ delay	Response time	Internal change	178	100	7	93
Periodic spikes	No change	Env. change	N/A	N/A	N/A	N/A

Read-modify-writes

Compare request flows between a run of *IoZone* in which

- the Linux client's I/O size was set to 16KB and
- the Linux client's I/O size was set to 4KB

Overview of the Ursa Minor case studies

Name	Manifestation	Root cause	# of results	*Top 10 rel.* %	*FPs* %	*Cov.*%
MDS config	Structural	Config. change	128	100	2	2
RMWs	Structural	Env. change	3	100	0	100
MDS prefetch. 50	Structural	Internal change	7	29	71	93
MDS prefetch. 10	Structural	Internal change	16	70	56	96
Create behaviour	Structural	Design problem	11	40	64	N/A
$100\mu s$ delay	Response time	Internal change	17	0	100	0
$500\mu s$ delay	Response time	Internal change	166	100	6	92
$1ms$ delay	Response time	Internal change	178	100	7	93
Periodic spikes	No change	Env. change	N/A	N/A	N/A	N/A

MDS prefetching ▸ Jump to Graph

- Compared two runs of *linux-build*, one with prefetching disabled and another with it enabled.
- SM_THRESHOLD set to be 50,10 respectively.

Overview of the Ursa Minor case studies

Name	Manifestation	Root cause	# of results	*Top 10 rel.* %	*FPs* %	*Cov.*%
MDS config	Structural	Config. change	128	100	2	2
RMWs	Structural	Env. change	3	100	0	100
MDS prefetch. 50	Structural	Internal change	7	29	71	93
MDS prefetch. 10	Structural	Internal change	16	70	56	96
Create behaviour	Structural	Design problem	11	40	64	N/A
$100\mu s$ delay	Response time	Internal change	17	0	100	0
$500\mu s$ delay	Response time	Internal change	166	100	6	92
$1ms$ delay	Response time	Internal change	178	100	7	93
Periodic spikes	No change	Env. change	N/A	N/A	N/A	N/A

Create behaviour

- To serve a CREATE, the metadata server executed a tight inter-component loop with a storage node
- Categories containing *structural mutations* executed this loop more times than their precursor categories

Overview of the Ursa Minor case studies

Name	Manifestation	Root cause	# of results	*Top 10 rel.* %	*FPs* %	*Cov.*%
MDS config	Structural	Config. change	128	100	2	2
RMWs	Structural	Env. change	3	100	0	100
MDS prefetch. 50	Structural	Internal change	7	29	71	93
MDS prefetch. 10	Structural	Internal change	16	70	56	96
Create behaviour	Structural	Design problem	11	40	64	N/A
$100\mu s$ delay	Response time	Internal change	17	0	100	0
$500\mu s$ delay	Response time	Internal change	166	100	6	92
$1ms$ delay	Response time	Internal change	178	100	7	93
Periodic spikes	No change	Env. change	N/A	N/A	N/A	N/A

Slowdown due to code changes

- Compare request flows between two runs of *SFS97*.
- Injecting $100\mu s$, $500\mu s$, and $1ms$ spin loops

Overview of the Ursa Minor case studies

Name	Manifestation	Root cause	# of results	*Top 10 rel.* %	*FPs* %	*Cov.*%
MDS config	Structural	Config. change	128	100	2	2
RMWs	Structural	Env. change	3	100	0	100
MDS prefetch. 50	Structural	Internal change	7	29	71	93
MDS prefetch. 10	Structural	Internal change	16	70	56	96
Create behaviour	Structural	Design problem	11	40	64	N/A
$100\mu s$ delay	Response time	Internal change	17	0	100	0
$500\mu s$ delay	Response time	Internal change	166	100	6	92
$1ms$ delay	Response time	Internal change	178	100	7	93
Periodic spikes	No change	Env. change	N/A	N/A	N/A	N/A

Periodic spikes

Currently suspect the problem to be backup activity initiated from the facilities department.

Outline

Experiences at Google

- Inter-cluster performance
- Performance change in a large service

✎ Challenges remain in scaling request-flow comparison techniques to large distributed services

Summary

- Introduced request-flow comparison as a new way to diagnose perform changes
- Presented algorithms for localizing problems by identifying mutations
- Showed utility of our approach by using it to diagnose real, unsolved problems

Thank You for your attention!

Thank You for your attention!

Thank You for your attention!

Thank You for your attention!

Thank You for your attention!