Diagnosing performance changes by comparing request flows

Raja Sambasivan
Alice Zheng, Michael De Rosa, Elie Krevat,
Spencer Whitman, Michael Stroucken,
William Wang, Ilari Shafer, Lianghong Xu,
Greg Ganger

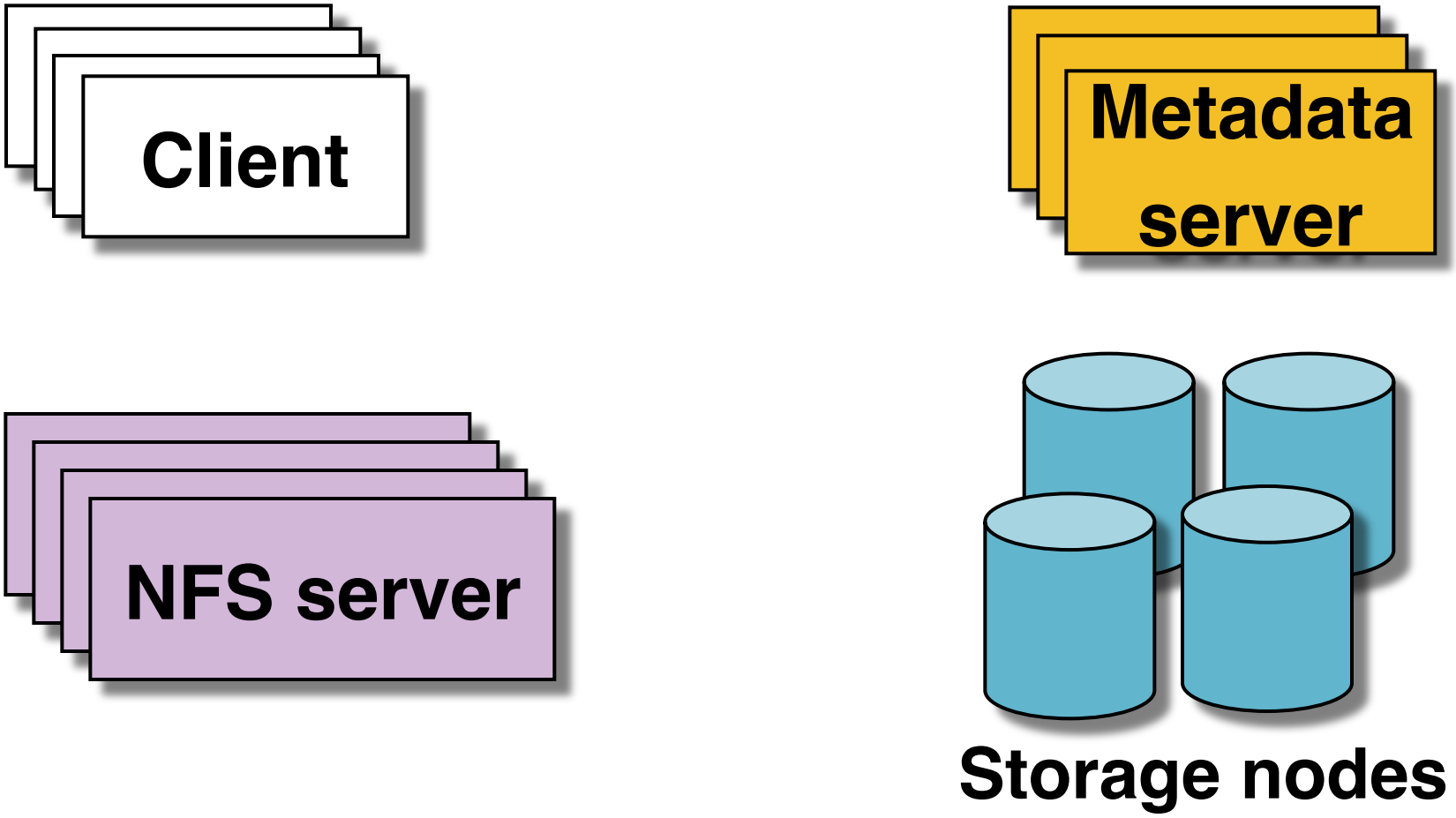
http://www.istc-cc.cmu.edu/

Perf. diagnosis in distributed systems

- Very time consuming and difficult
  - Root cause could be in any component

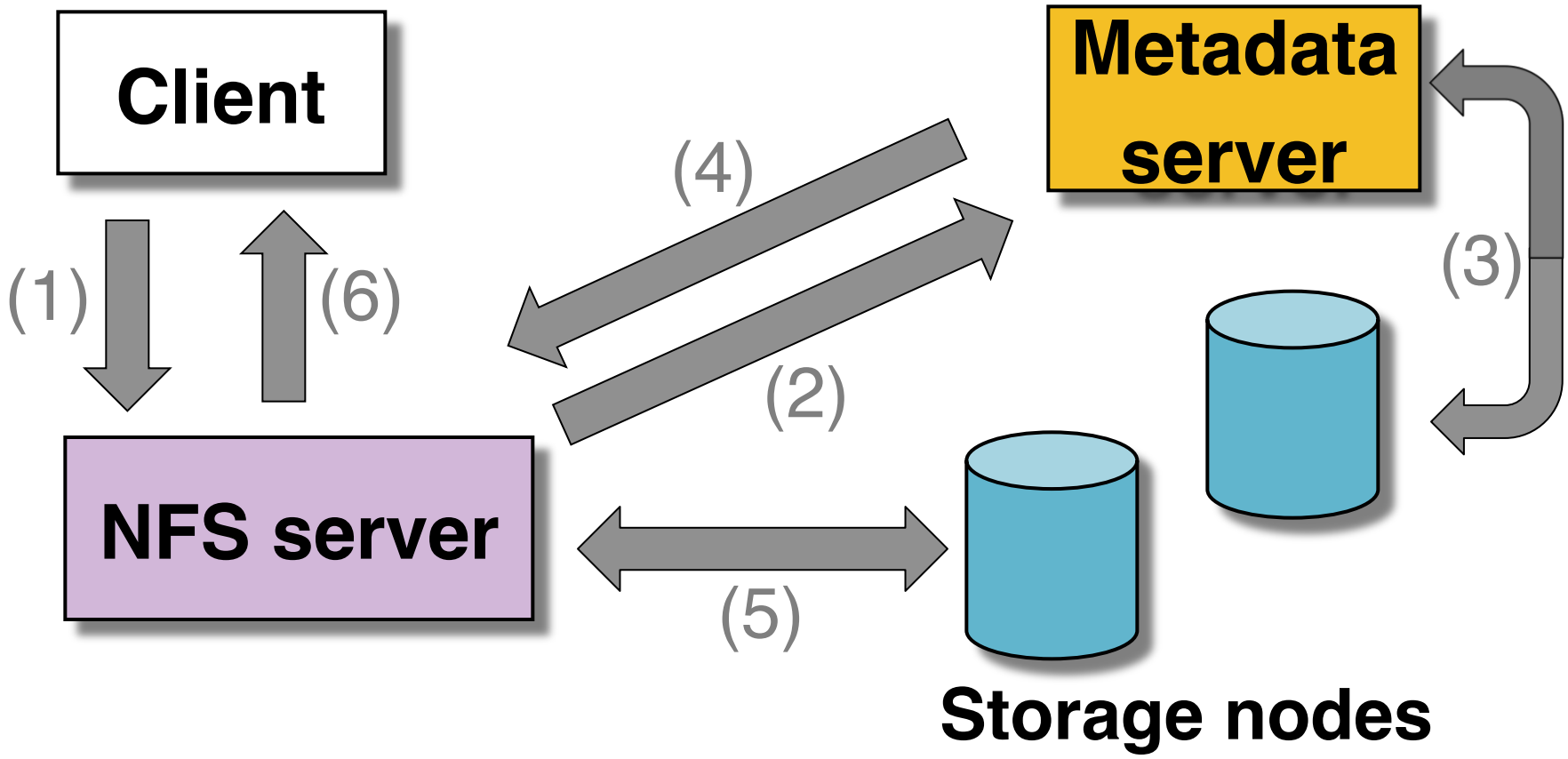
- **Request-flow comparison** [Sambasivan11a]
  - Helps localize performance changes
  - Key insight: Changes manifest as mutations in request timing/structure

Perf. debugging a feature addition



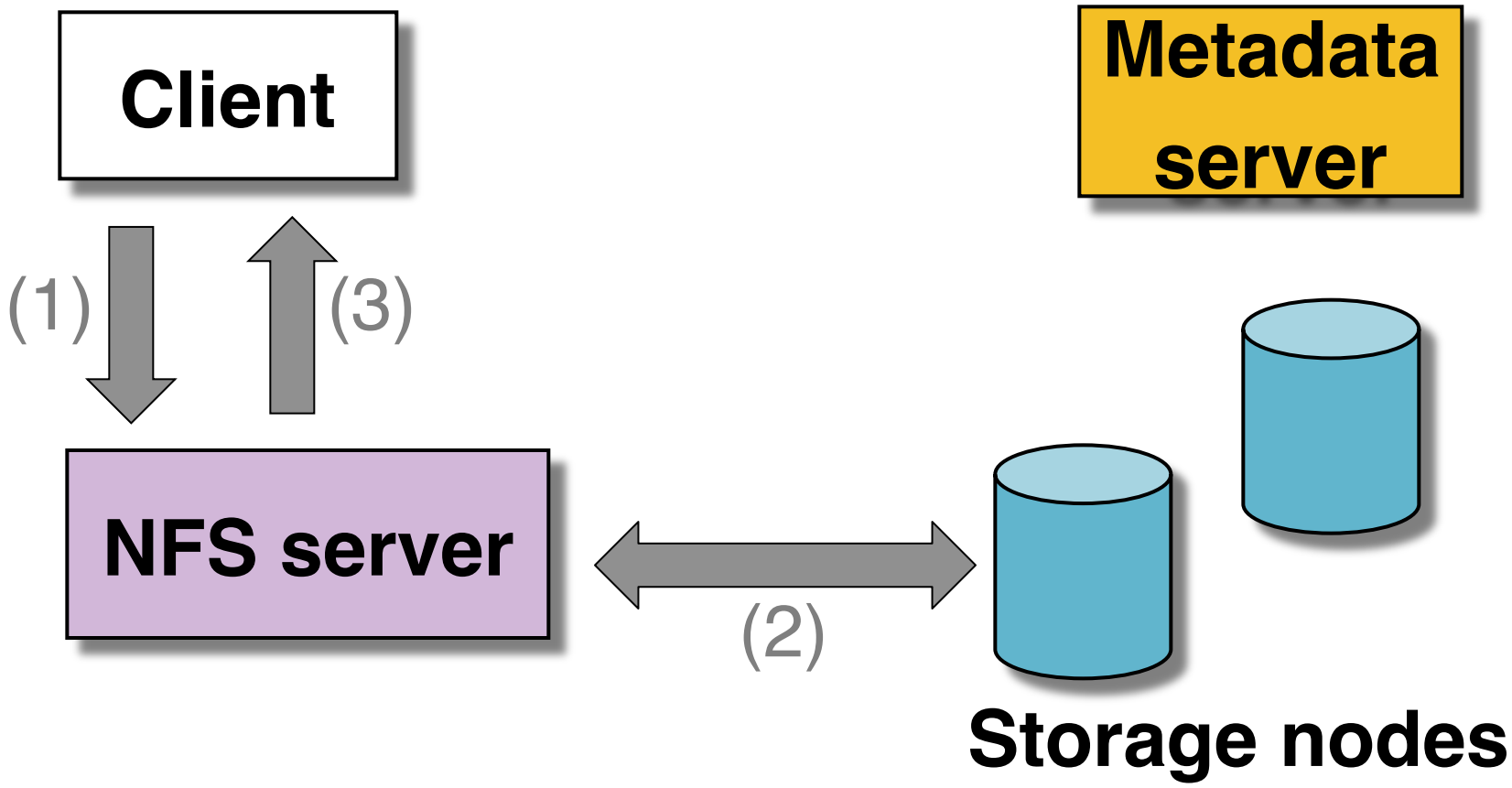
Perf. debugging a feature addition

- Before addition:
  - Every file access needs a MDS access



Perf. debugging a feature addition

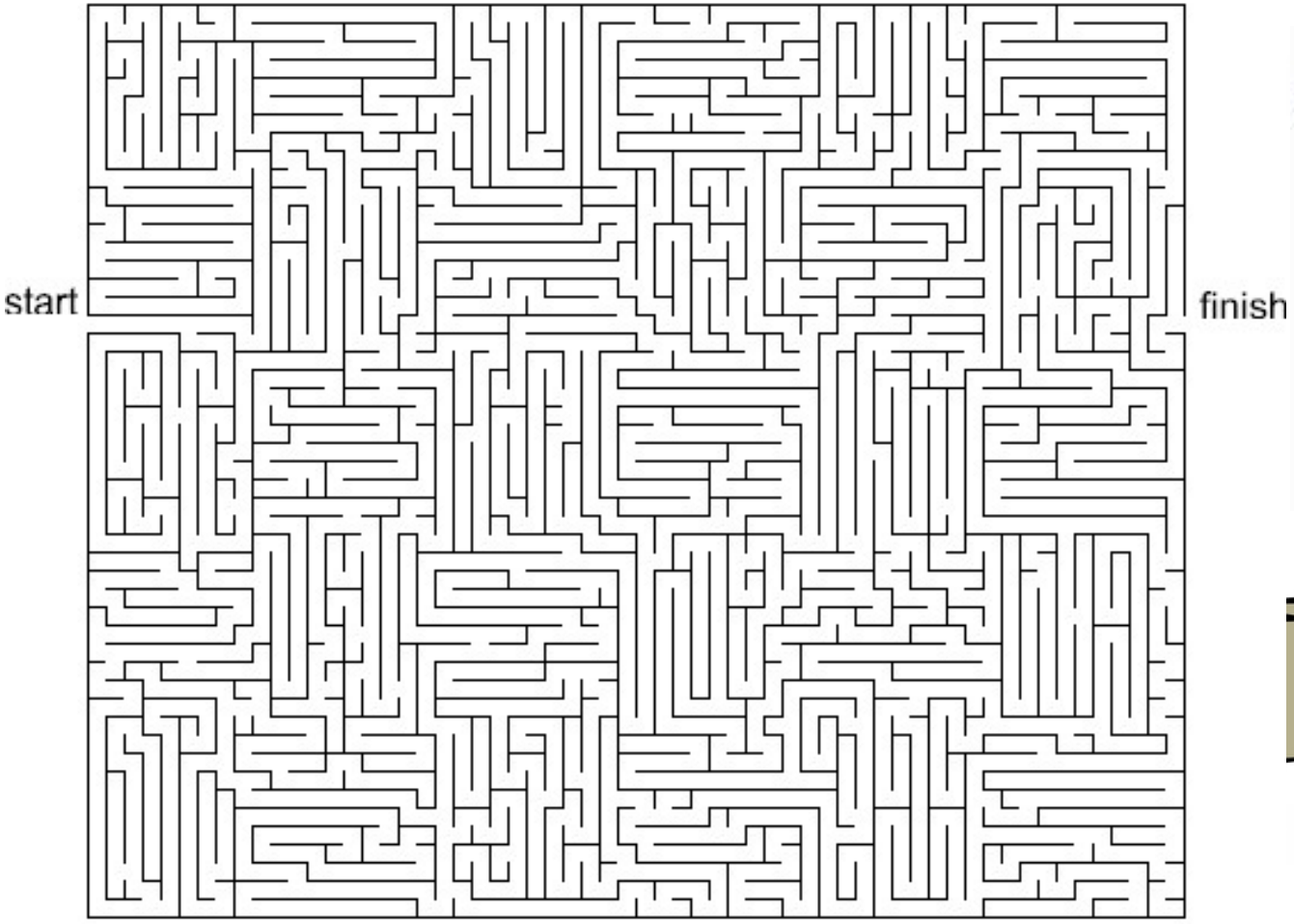
- After: Metadata prefetched to clients
  - Most requests **don't** need MDS access



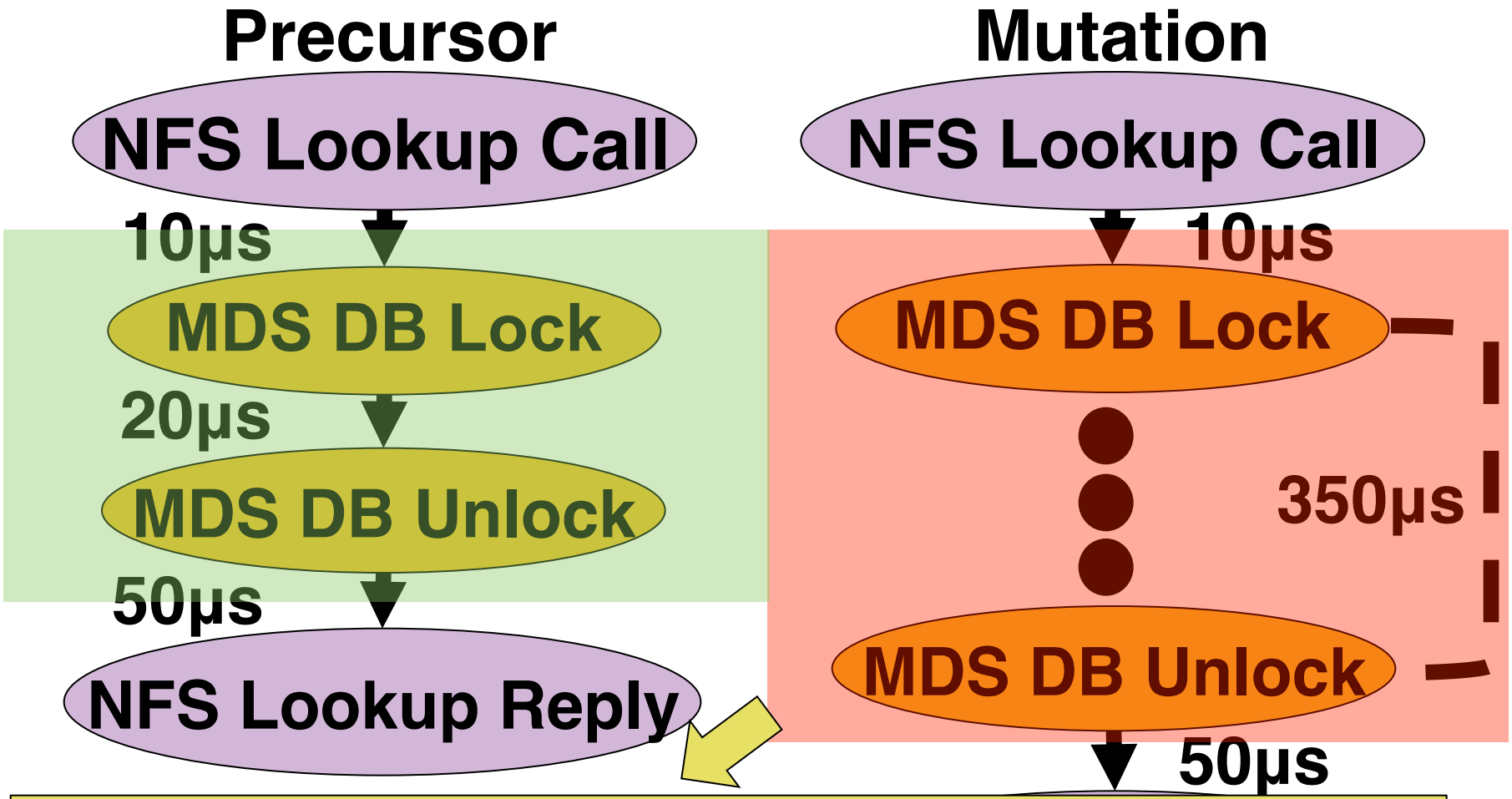
Perf. debugging a feature addition

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

My initial solution



Request-flow comparison will show



Root cause localized by showing how mutation and precursor differ

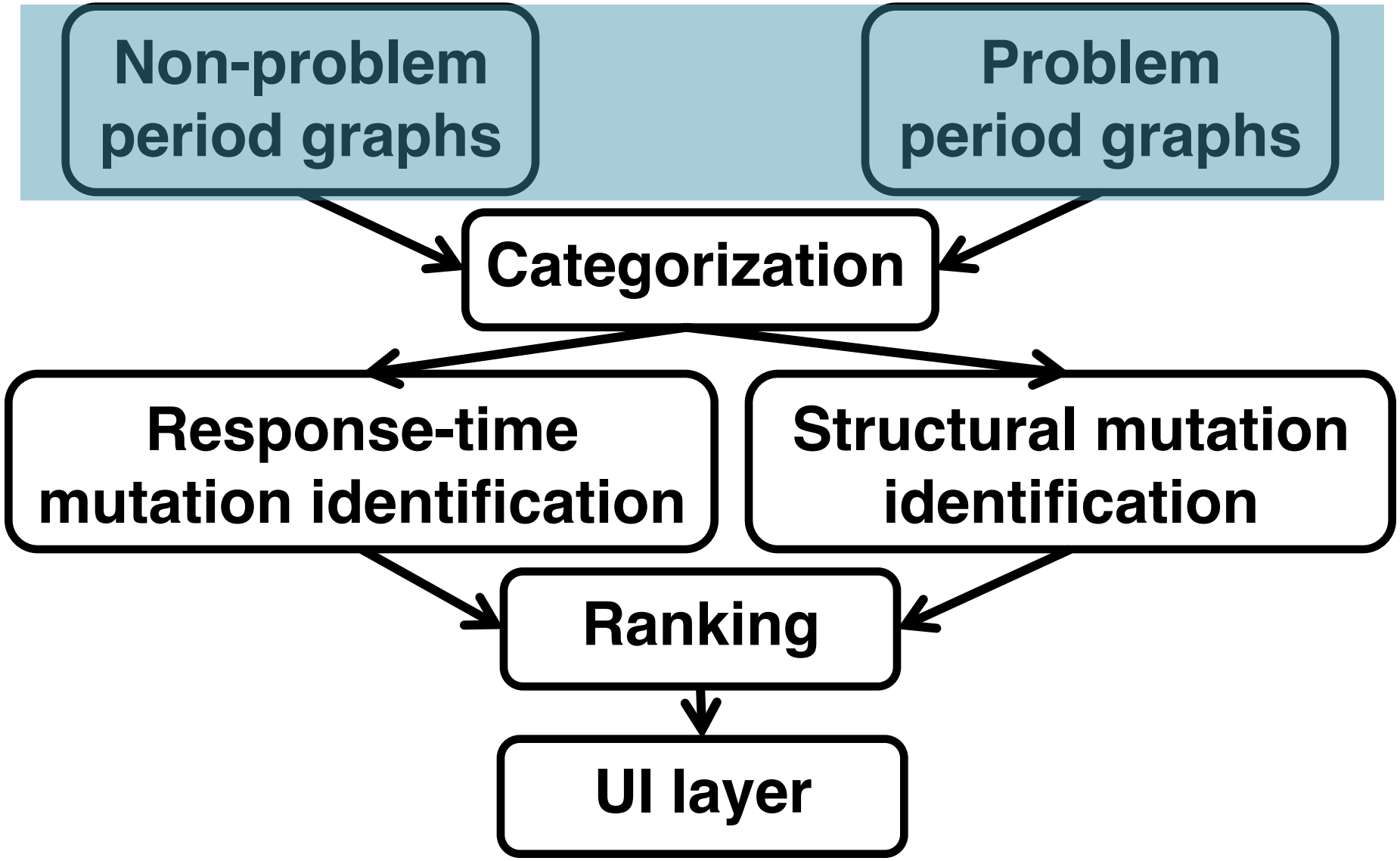
Request-flow comparison

- Identifies distribution changes in request-flow timing and structure
- Does not rely on explicit SLAs or labels
- Satisfies many use cases
  - Diagnosing regressions/degradations
  - Eliminating the system as the culprit

This talk

- Overview of implementation of request-flow comparison in Spectroscope
- Results of diagnosing real problems in Ursa Minor & certain Google services
- Future work on generalizing this technique

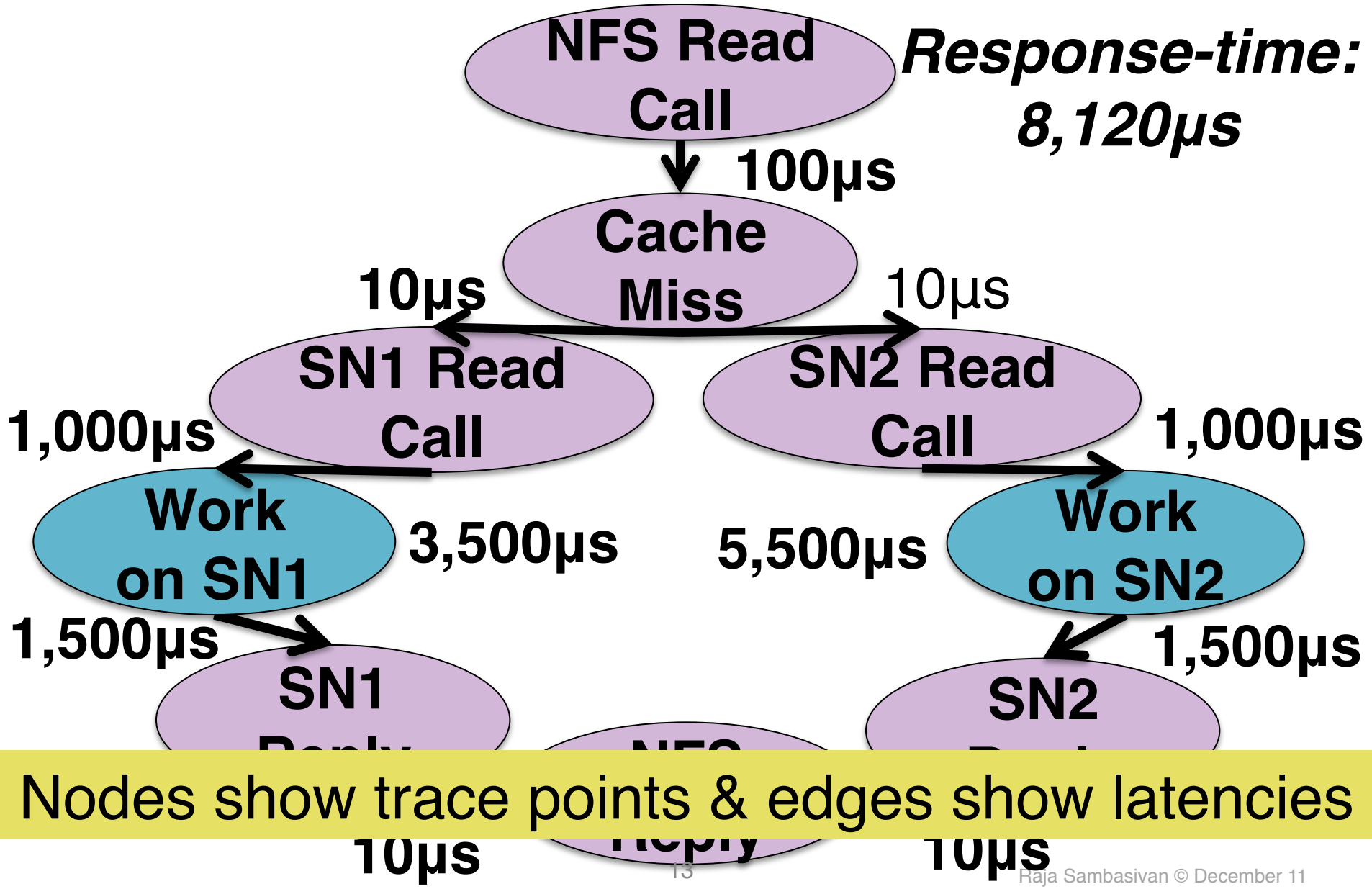
Spectroscope workflow



Graphs via end-to-end tracing

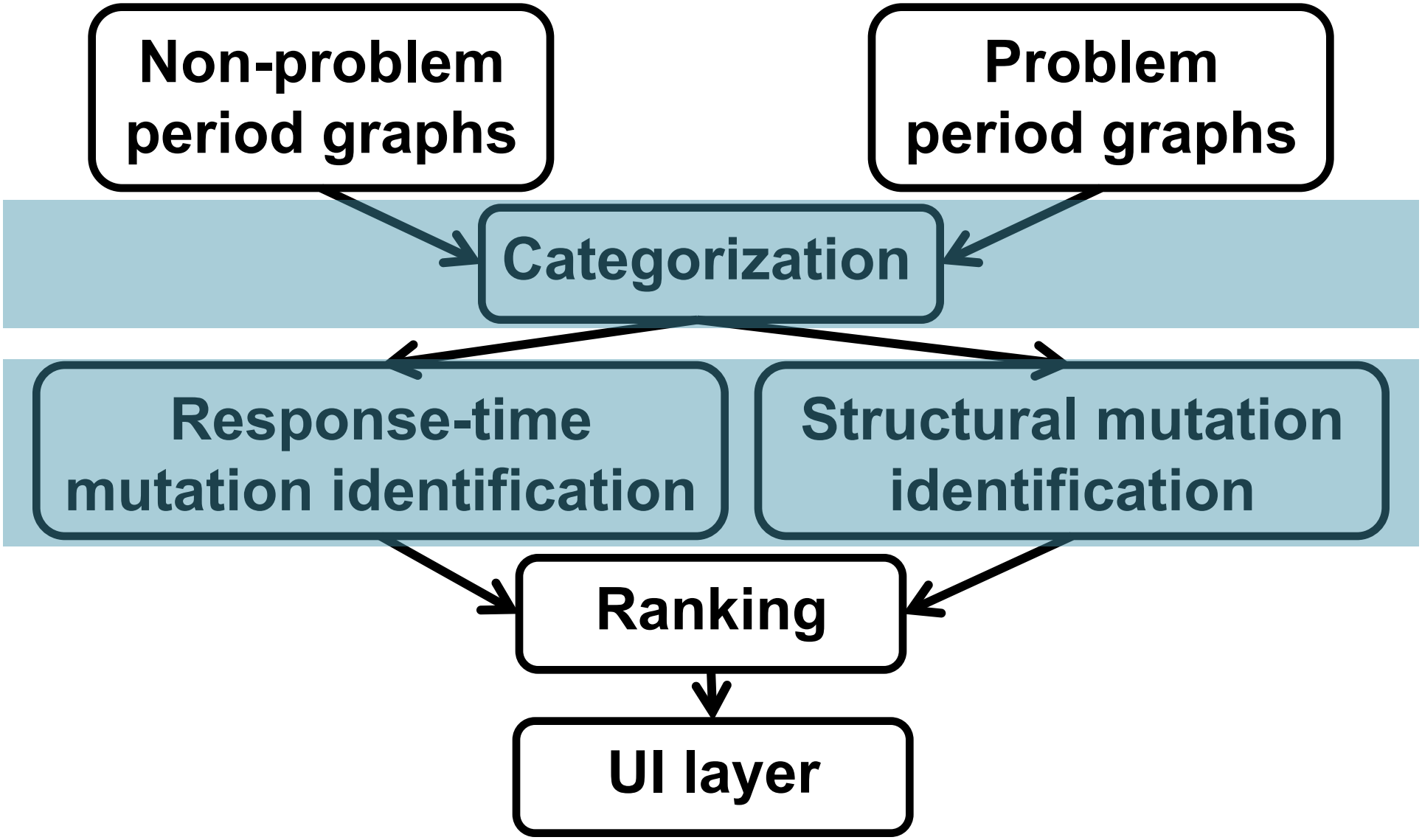
- Used in research & production systems
- Works as follows:
  - Tracks trace points touched by requests
  - Request-flow graphs obtained by stitching together trace points accessed
- Yields < 1% overhead w/req. sampling

Example: Graph for a striped read



Nodes show trace points & edges show latencies

Spectroscope workflow



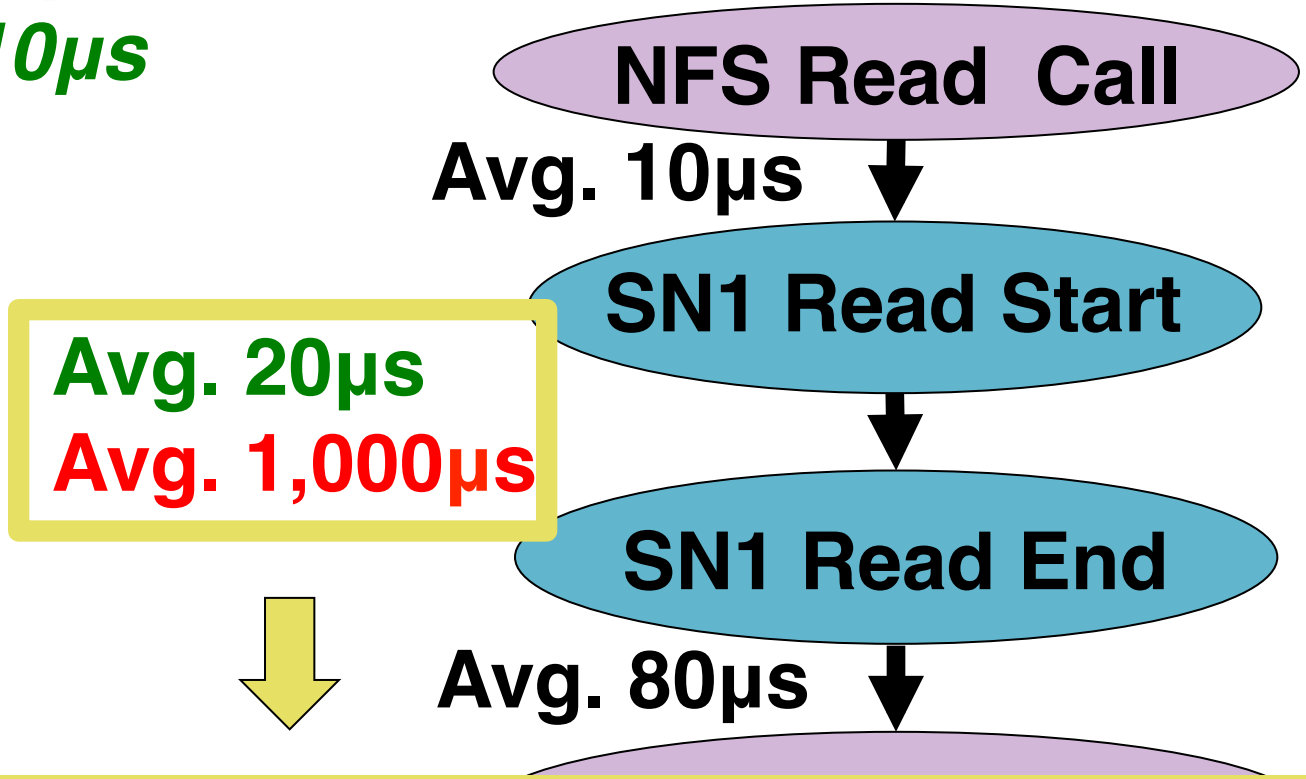
Type 1: Response-time mutations

- Requests that:
  - are structurally identical in both periods
  - have larger problem period latencies
- Root cause localized by…
  - identifying interactions responsible
- Identified via statistical hypothesis testing

Response-time mutation example

Avg. non-problem response time: 110µs

Avg. problem response time: 1,090µs



Problem localized by ID'ing responsible interaction

Type 2: Structural mutations

- Requests that:
  - take different paths in the problem period
- Root caused localized by…
  - identifying their precursors
    - Likely path during the non-problem period
  - id'ing how mutation & precursor differ

ID'ing categories w/structural mutations

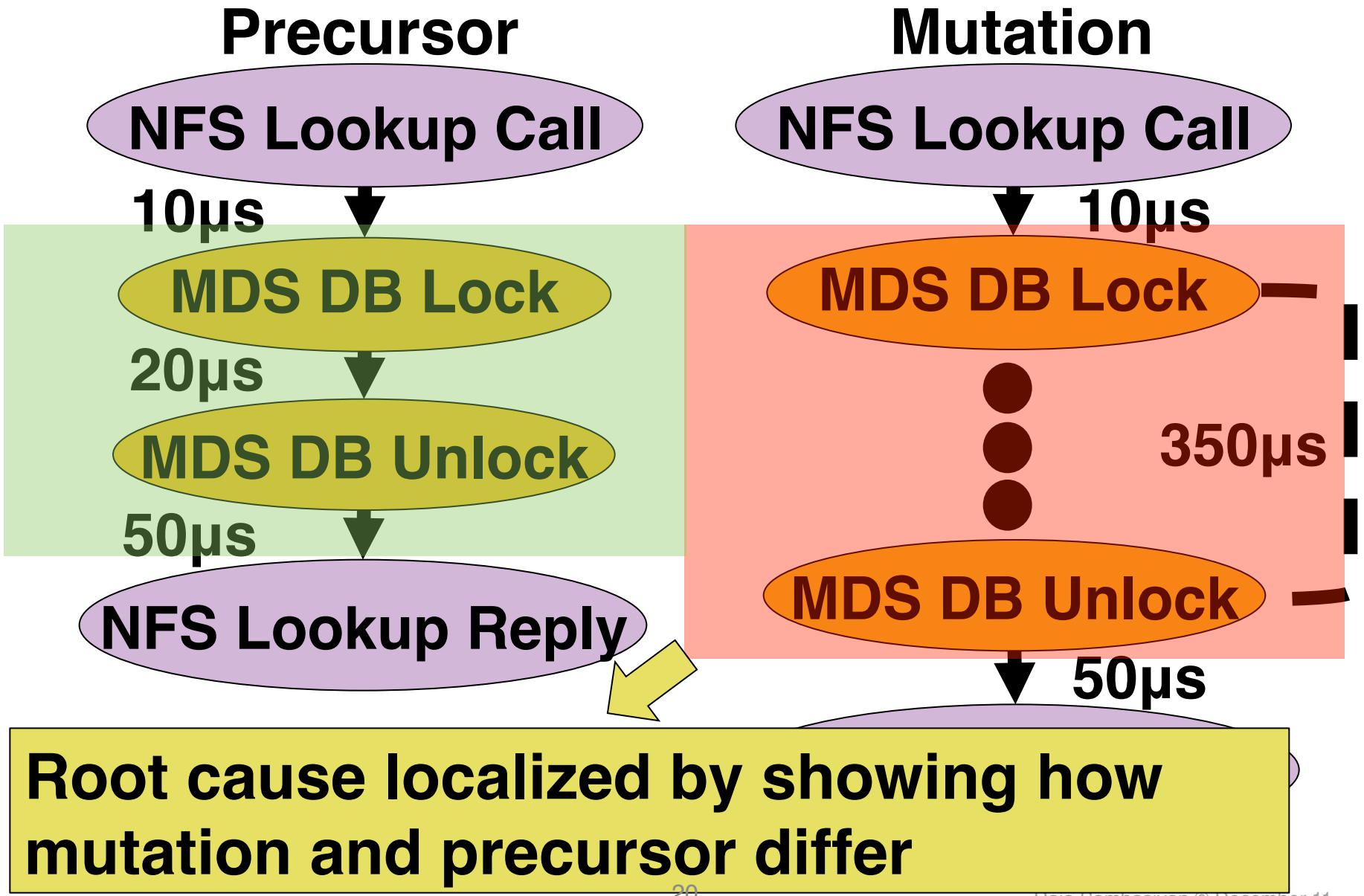
- Assume similar workloads executed
  - Categories with more problem period requests contain mutations
  - Reverse true for precursor categories
- Threshold used to differentiate natural variance from categories w/mutations

Mapping mutations to precursors

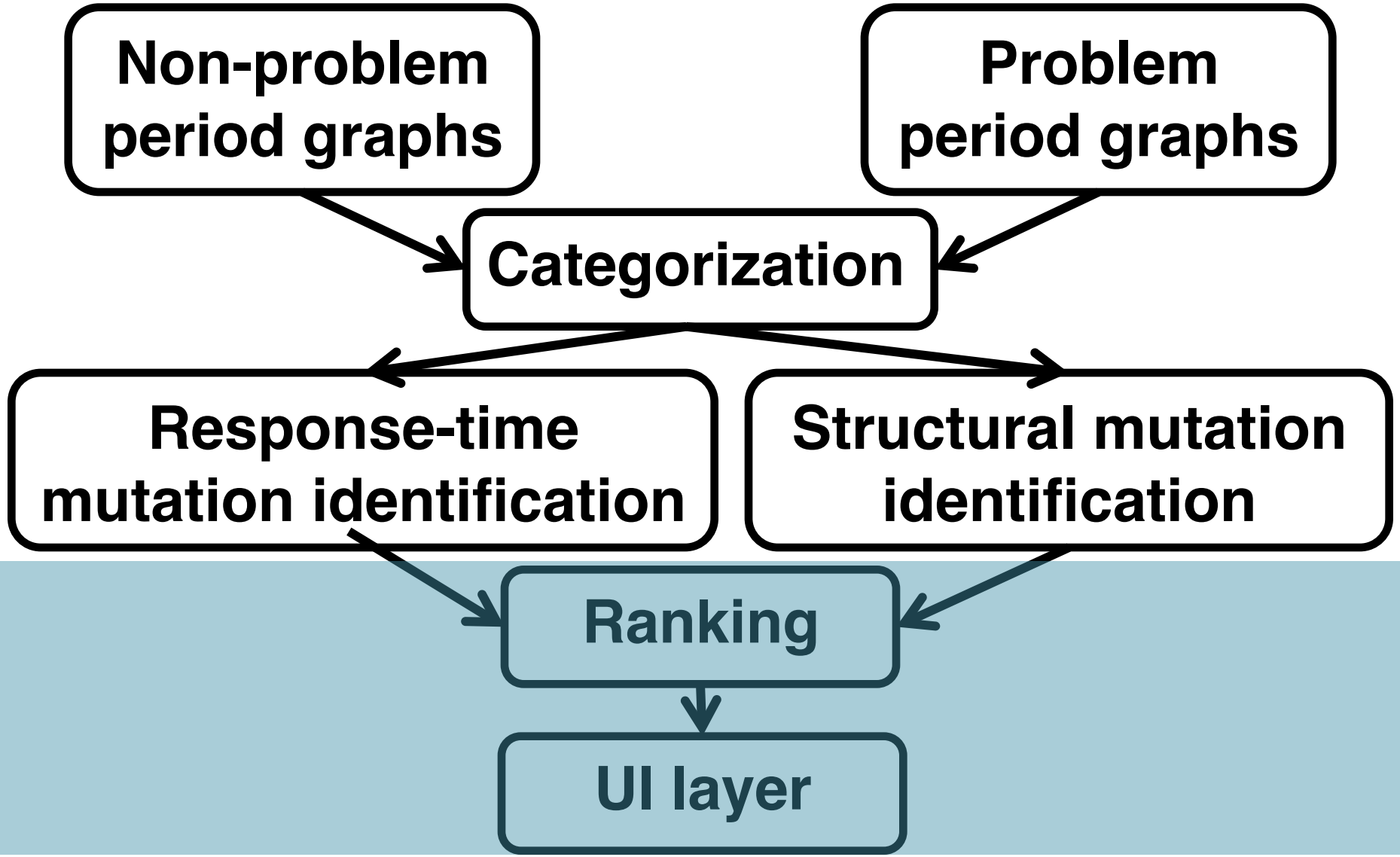
- IDs which precursors donated requests to which structural mutation category

- Accomplished using three heuristics
  - See [Sambasivan11a] for details

Example structural mutation



Spectroscope workflow



Outline

- Introduction
- Spectroscope
- Results of case studies
- Future work & summary

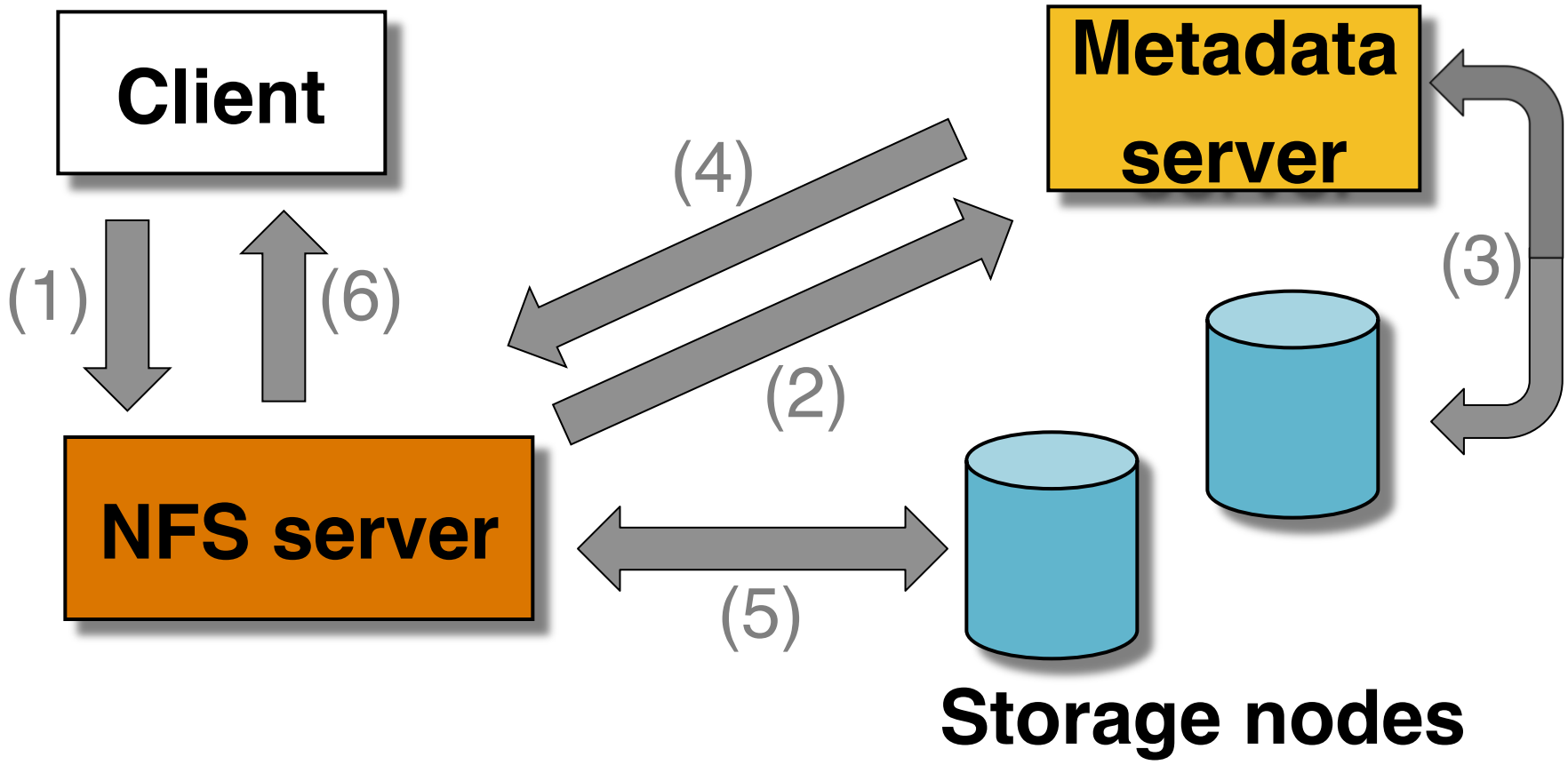
Ursa Minor & Google case studies

- Used Spectroscope to diagnose eight ***real*** & synthetic performance changes
  - Five were previously unsolved

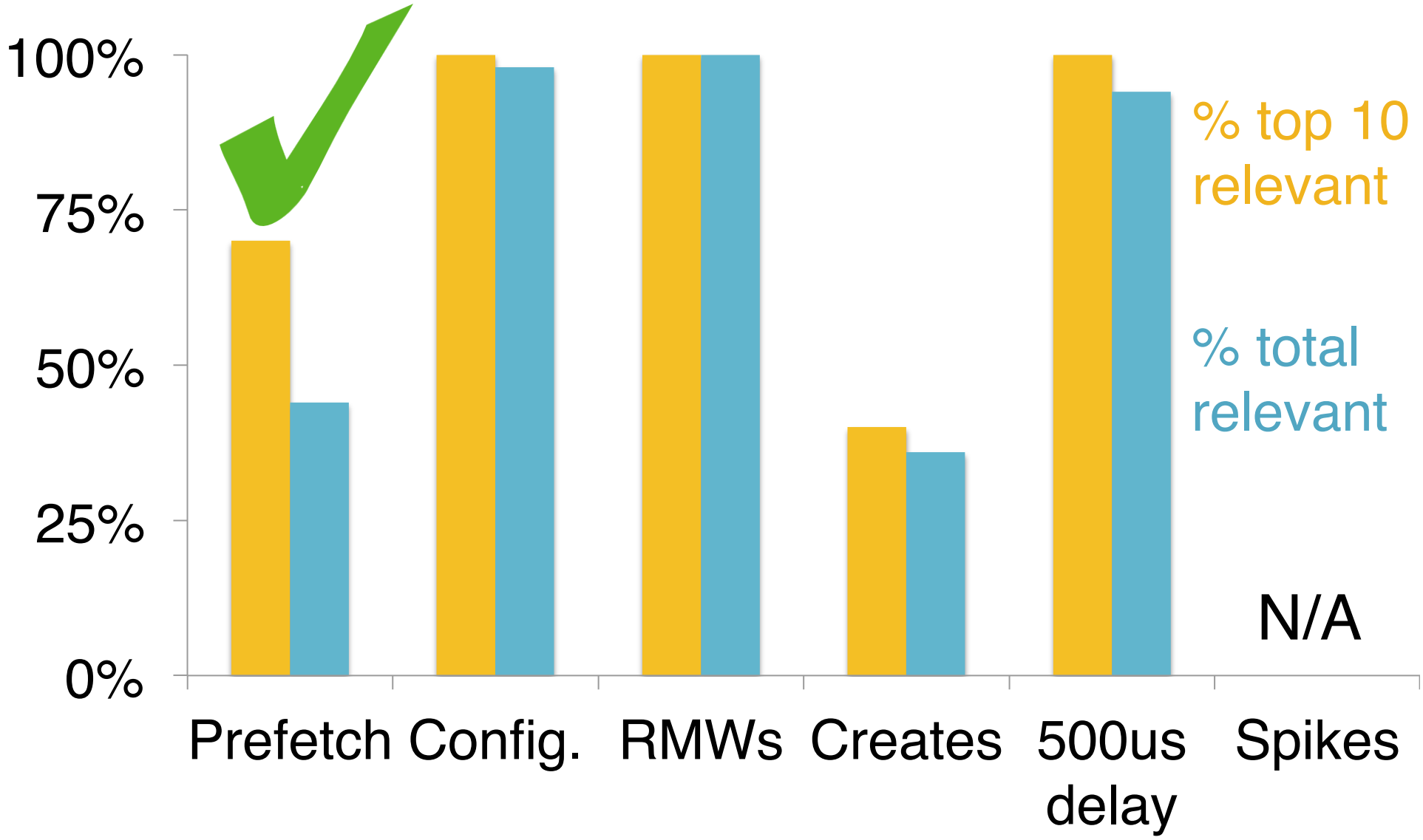
- For Ursa Minor, evaluated ranked list
  - % of top 10 results that were relevant
  - % of total results that were relevant

Ursa Minor 5-component config

- All case studies use this configuration



Ursa Minor's quantitative results



Outline

- Introduction
- Spectroscope
- Results of case studies
- Future work & summary

Future work

- Goal: Extend request-flow comparison so that it is applicable to more systems
- Requires additions to handle:
  - Large request flows
  - Diagnosing resource contention
- Test generality by diagnosing HDFS

Summary

- Introduced *request-flow comparison* as a new way to diagnose perf. changes
- Presented results of using this technique to diagnose real problems
- Described how we'll show it is a general approach useful in many systems

Related work (I)

[Abd-El-Malek05b]: **Ursa Minor: versatile cluster-based storage**. Michael Abd-El-Malek, William V. Courtright II, Chuck Cranor, Gregory R. Ganger, James Hendricks, Andrew J. Klosterman, Michael Mesnier, Manish Prasad, Brandon Salmon, Raja R. Sambasivan, Shafeeq Sinnamohideen, John D. Strunk, Eno Thereska, Matthew Wachs, Jay J. Wylie. FAST, 2005.

[Barham04]:**Using Magpie for request extraction and workload modelling**. Paul Barham, Austin Donnelly, Rebecca Isaacs, Richard Mortier. OSDI, 2004.

[Chen04]: **Path-based failure and evolution management**. Mike Y. Chen, Anthony Accardi, Emre Kiciman, Jim Lloyd, Dave Patterson, Armando Fox, Eric Brewer. NSDI, 2004.

Related work (II)

[Cohen05]: **Capturing, indexing, and retrieving system history**. Ira Cohen, Steve Zhang, Moises Goldszmidt, Terrence Kelly, Armando Fox. SOSP, 2005.

[Fonseca07]: **X-Trace: a pervasive network tracing framework**. Rodrigo Fonseca, George Porter, Randy H. Katz, Scott Shenker, Ion Stoica. NSDI, 2007.

[Hendricks06]: **Improving small file performance in object-based storage**. James Hendricks, Raja R. Sambasivan, Shafeeq Sinnamohideen, Gregory R. Ganger. Technical Report CMU-PDL-06-104, 2006.

Related work (III)

[Kandula09]: **Detailed diagnosis in enterprise networks**. Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitenda Padhye, Paramvir Bah. SIGCOMM, 2011.

[Kasick09]: **System-call based problem diagnosis for PVFS**. Michael P. Kasick, Keith A. Bare, Eugene E. Marinielli III, Jiaqi Tan, Rajeev Gandhi, Priya Narasimhan. HotDep, 2009.

[Kavulya11]: **Practical experiences with chronics discovery in large telecommunications systems**. Soila P. Kavulya, Kaustubh Joshi, Matti Hiltunen, Scott Daniels, Rajeev Gandhi, Priya Narasimhan. SLAML, 2011.

Related work (IV)

[Mann11]**: Modeling the execution of parallel black-box services**. Gideon Mann, Mark Sandler, Darja Krushevskaja, Sudipto Guha, Eyal Even-dar. HotCloud, 2011.

[Reynolds06]: **Detecting the unexpected in distributed systems**. Patrick Reynolds, Charles Killian, Janet L. Wiener, Jeffrey C. Mogul, Mehul A. Shah, Amin Vahdat. NSDI, 2006.

[Sambasivan07]: **Categorizing and differencing system behaviours**. Raja R. Sambasivan, Alice X. Zheng, Eno Thereska, Gregory R. Ganger. HotAC II, 2007.

Related work (V)

[Sambasivan11]: **Automation without predictability is a recipe for failure**. Raja R. Sambasivan and Gregory R. Ganger. Technical Report CMU-PDL-11-101, 2011.

[Sambasivan11a]: **Diagnosing performance changes by comparing request flows**. Raja R. Sambasivan, Alice X. Zheng, Michael De Rosa, Elie Krevat, Spencer Whitman, Michael Stroucken, William Wang, Lianghong Xu, Gregory R. Ganger. NSDI, 2011.

[Sandler11]: **Modeling the parallel execution of black-box services**. Gideon Mann, Mark Sandler, Darja Krushevskaja, Sudipto Guha, Eyal Even-Dar. HotCloud, 2011.

Related work (VI)

[Sigelman10]: **Dapper, a large-scale distributed tracing infrastructure**. Benjamin H. Sigelman, Luiz André Barroso, Mike Burrows, Pat Stephenson, Manoj Plakal, Donald Beaver, Saul Jaspan, Chandan Shanbhag. Google Technical report 2010-1, 2008.

[Tan10]: **Visual, log-based causal tracing for performance debugging of map reduce systems**. Jiaqi Tan, Soila Kavulya, Rajeev Gandhi, Priya Narasimhan. ICDCS, 2010.

[Thereska06b]: **Stardust: tracking activity in a distributed storage system**. Eno Thereska, Brandon Salmon, John Strunk, Matthew Wachs, Michael Abd-El-Malek, Julio Lopez, Gregory R. Ganger. SIGMETRICS, 2006.

Related work (VII)

[Xu09]: **Detecting large-scale systems problems by mining console logs**. Wei Xu, Ling Huang, David Patterson, Michael I. Jordan. SOSP, 2009.