Tracklops : Real-Time NFS Performance Metrics Extractor

CHEOPS 2024

22/04/2024

Théophile Dubuc (ENS de Lyon, Outscale)
Pascale Vicat-Blanc (Inria, ENS de Lyon)
Pierre Olivier (The University of Manchester)
Mar Callau-Zori (Outscale)
Christophe Hubert (Outscale)
Alain Tchana (Grenoble INP)

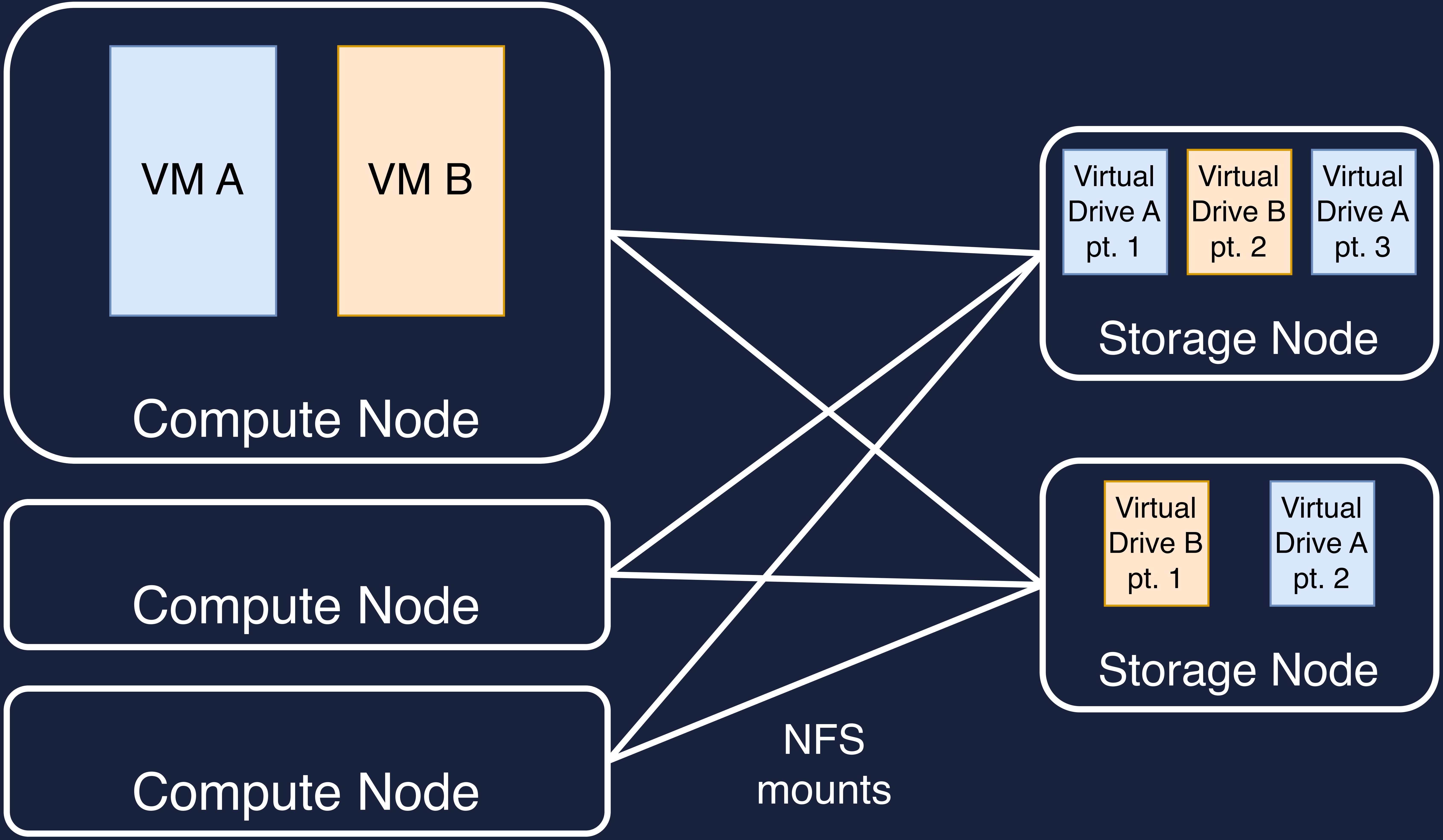
LABORATOIRE D'INFORMATIQUE DE GRENOBLE

ENS DE LYON

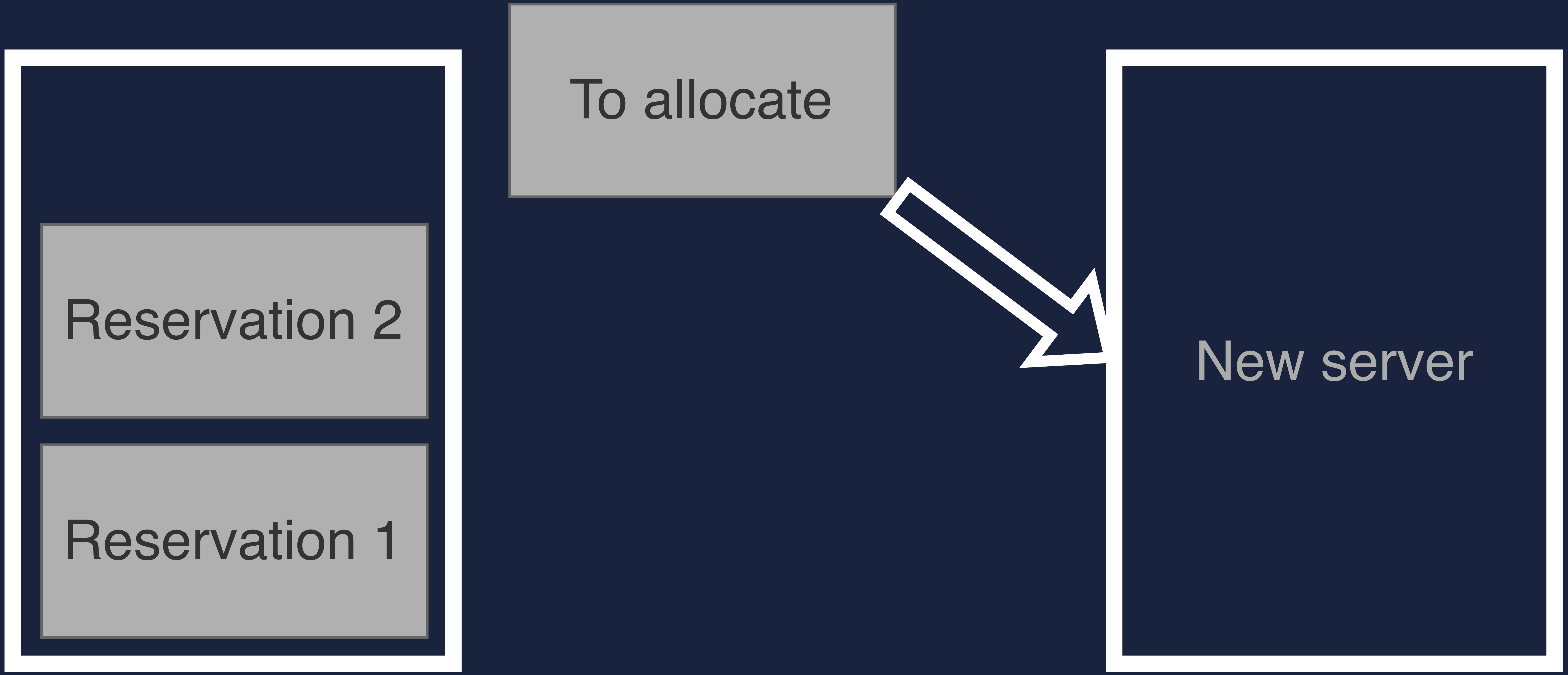
OUTSCALE

Context

Context: cloud services provider architecture based on NFS

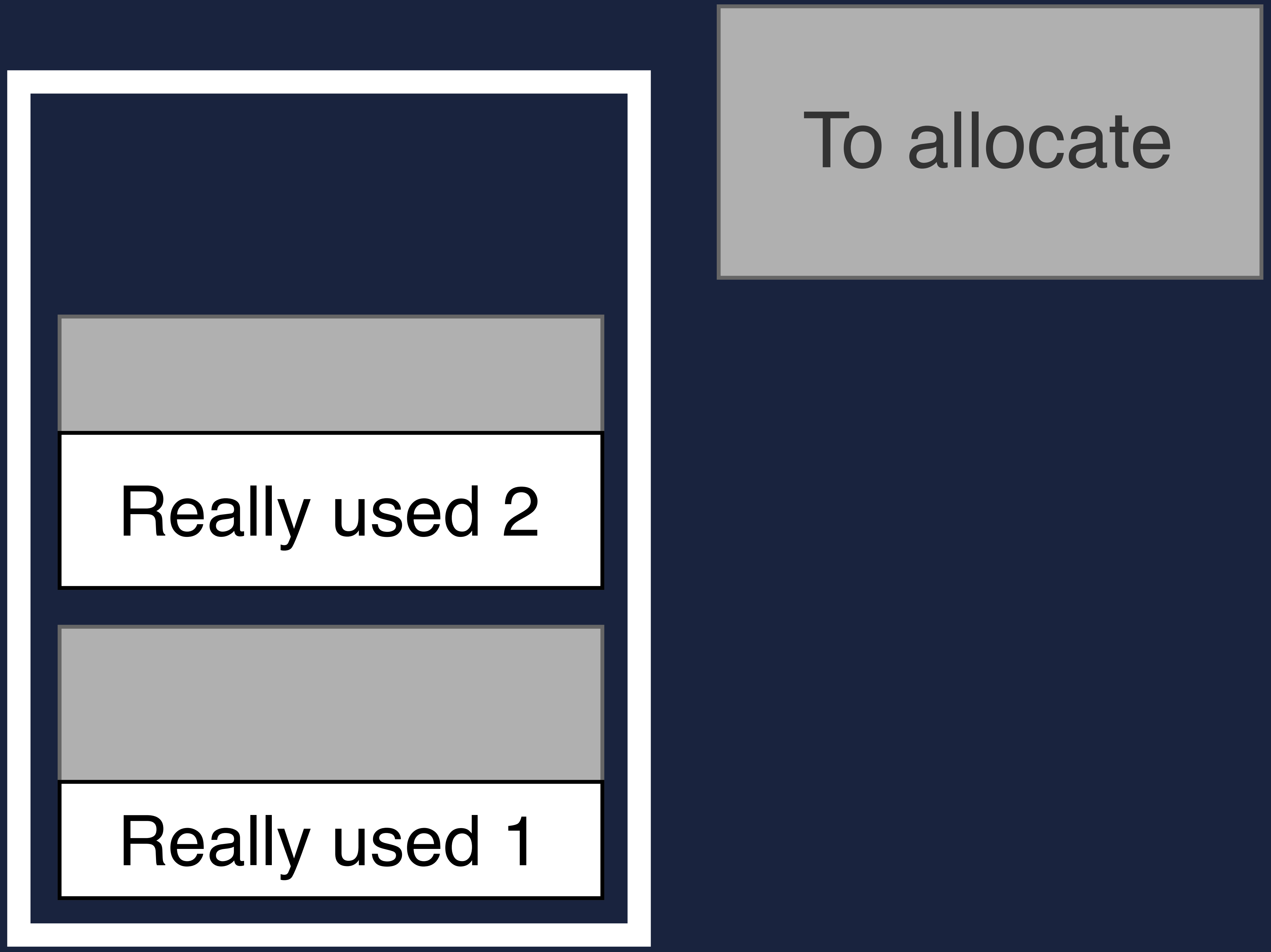


Use-case: volume placement

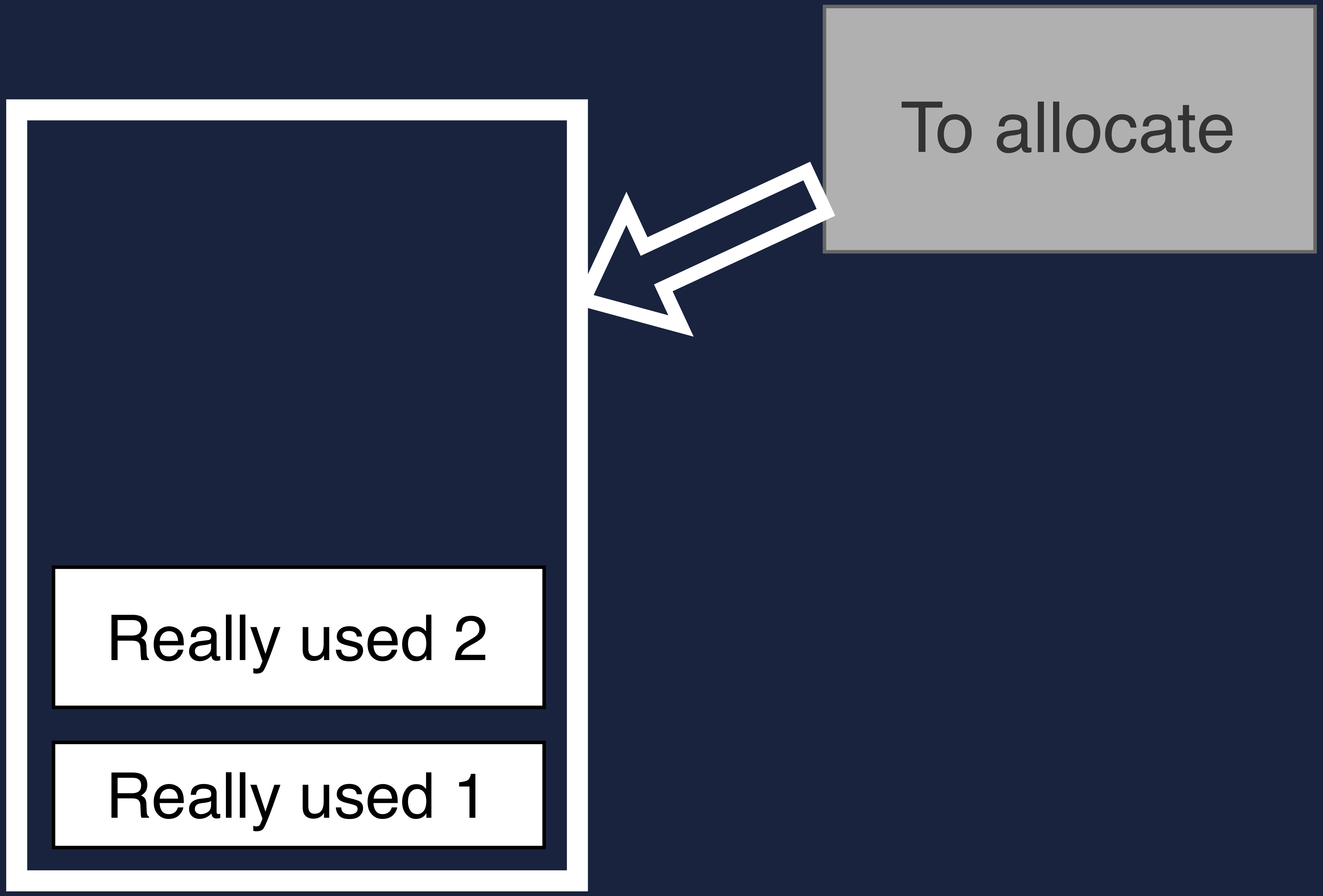


Where to allocate new resource?

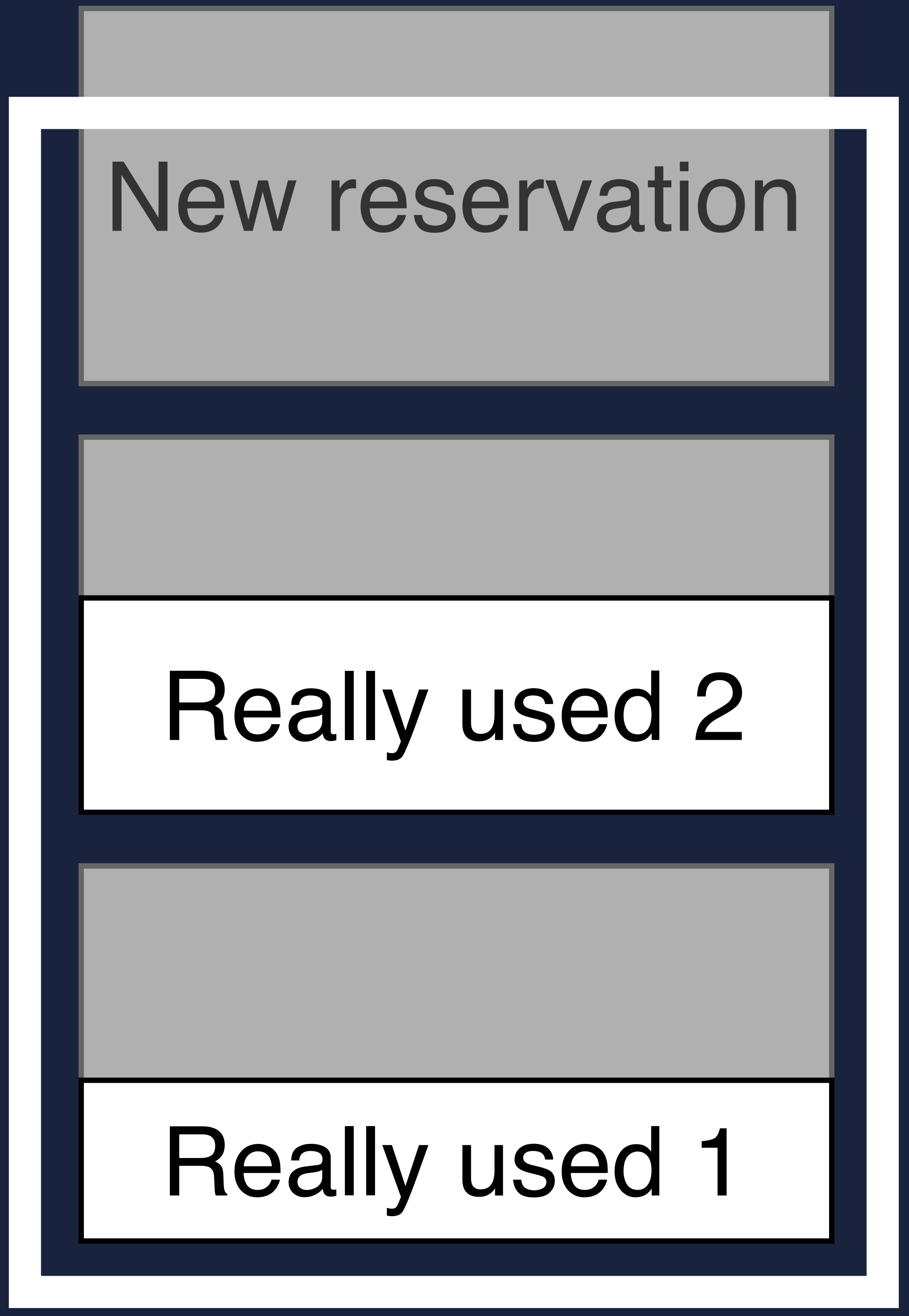
Placement: enable over-commit



Placement: enable over-commit



Load balancing: enable over-commit



Heterogeneous workload -> require per-file NFS usage metrics to predict the overall workload

Relevant metrics for storage placement:

- Size
- Performance: IOps, throughput, latency

Other use-cases for per-file performance metrics

- Volume placement / load balancing (size, iops, throughput, latency)
- Troubleshooting (mostly latency)
- Carbon footprint estimation for customers (size and iops)
- Billing for cloud provider (size and iops)

Théophile Dubuc
theophile.dubuc@ens-lyon.fr

Need for client-side per-file NFS performance metrics

- Storage nodes are usually closed-source: NetApp ONTAP, Dell EMC, …
- 1 - Only provide aggregated metrics (NFS share level)
- 2 - Can’t be instrumented for collecting more

-> Need to infer all the metrics from client-side (compute nodes) only

Constraints summary

Extract NFS performance metrics (iops, throughput, latency):

- In real time
- Per-file
- From the client side
- For production environments:
  - Low overhead
  - Don’t modify the kernel

State of the art

Existing tools

System	IOps	Throughput	Latency	Per-file	Client-side
nfsiostat	🟩	🟩	🟩	🟥	🟩
nfsdist, nfslower	🟥	🟥	🟧	🟥	🟩
blktrace, atop, pidstat	🟩	🟩	🟩	🟥	🟩
inotifywatch	🟩	🟥	🟥	🟩	🟩
Distributed frameworks	🟩	🟩	🟩	🟧	🟥

Design

Base it on eBPF

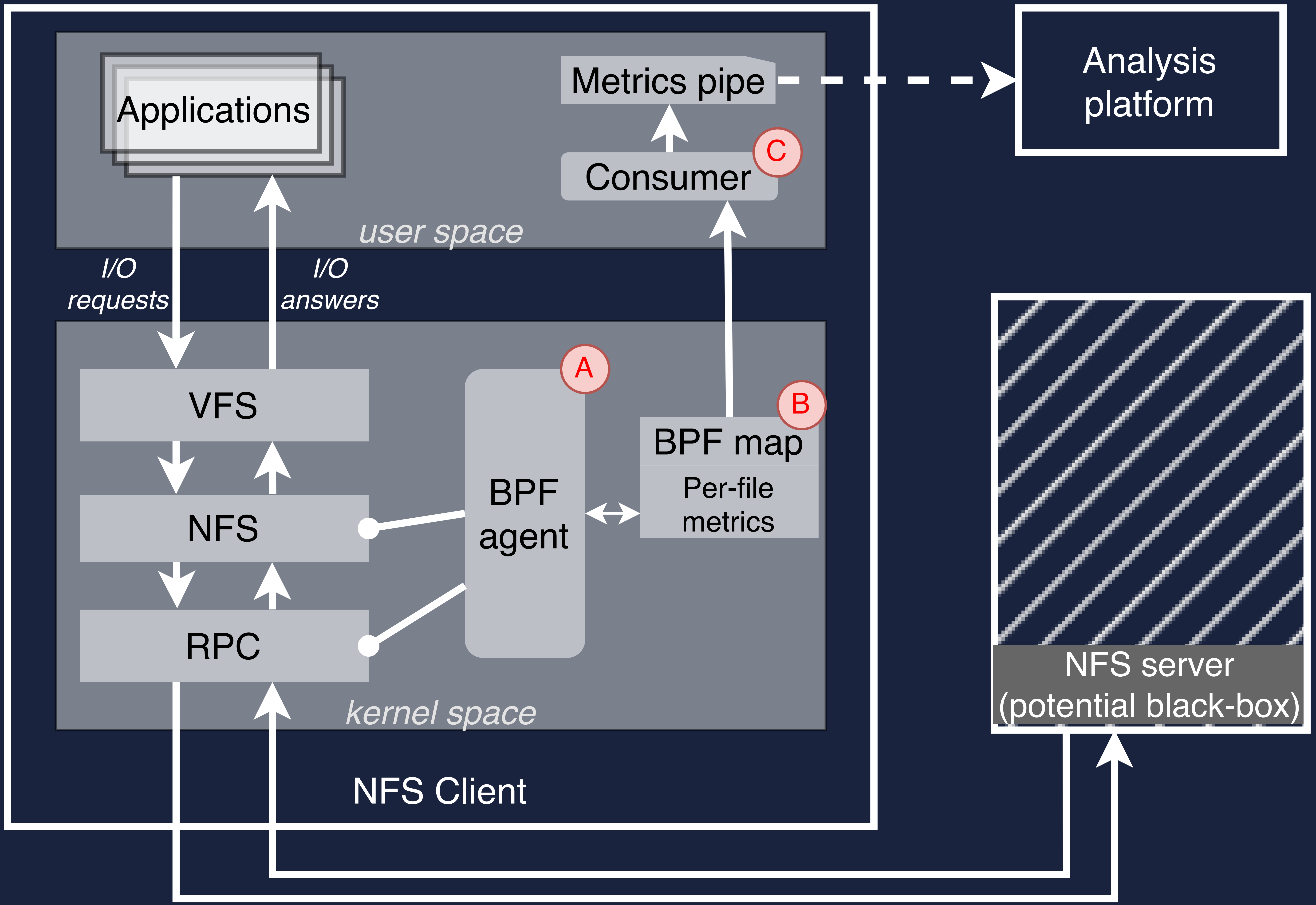
Write programs that hook to kernel events.

Features:

- Low overhead
- Dynamic: nothing to restart or re-compile
- Safe (restricted power + verifier + executed in kernel but isolated)

Design

A. Raw data collection from NFS and RPC tracepoints

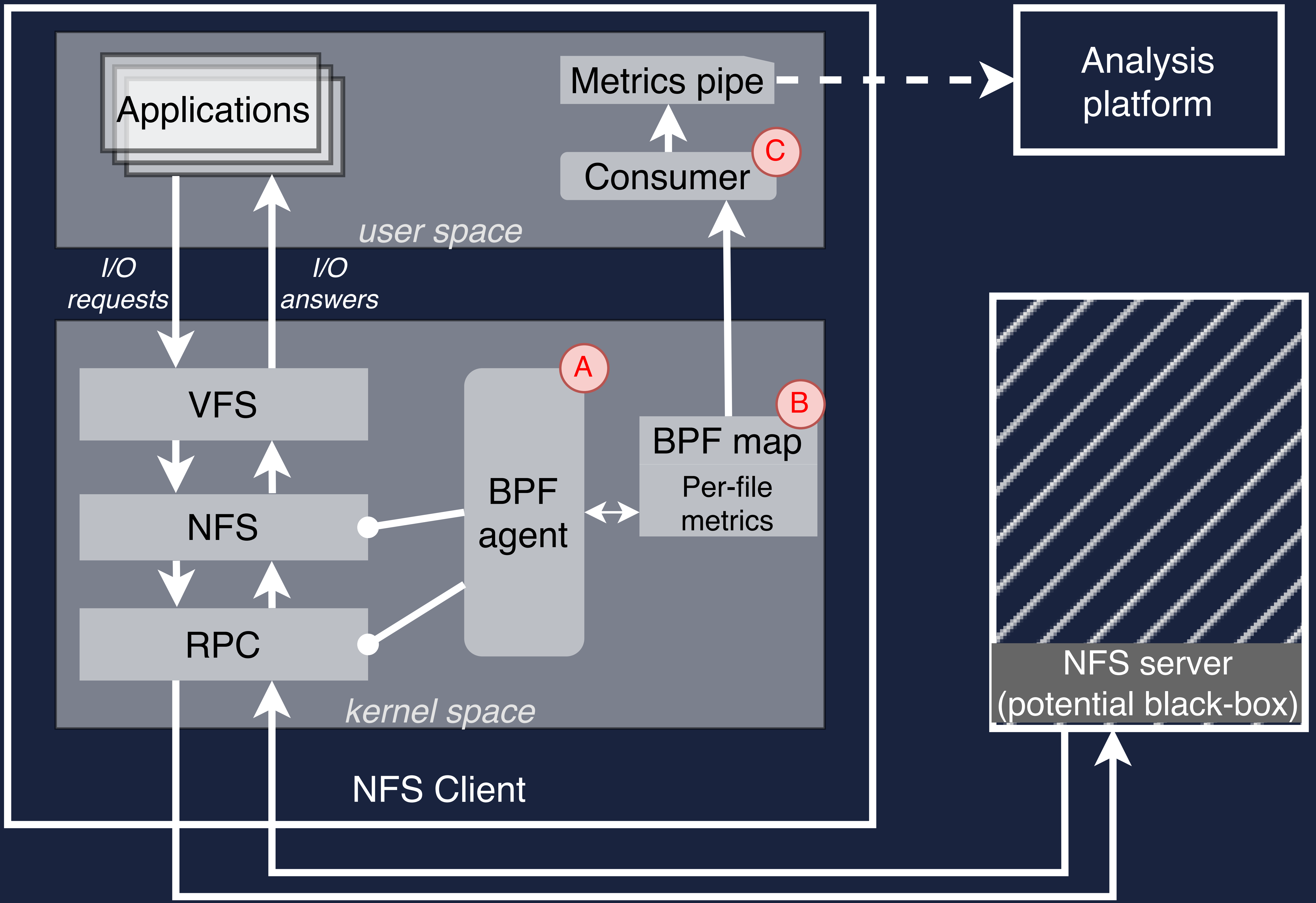


Théophile Dubuc
theophile.dubuc@ens-lyon.fr

Design

A. Raw data collection from NFS and RPC tracepoints

B. In-kernel NFS request reconstruction and storage



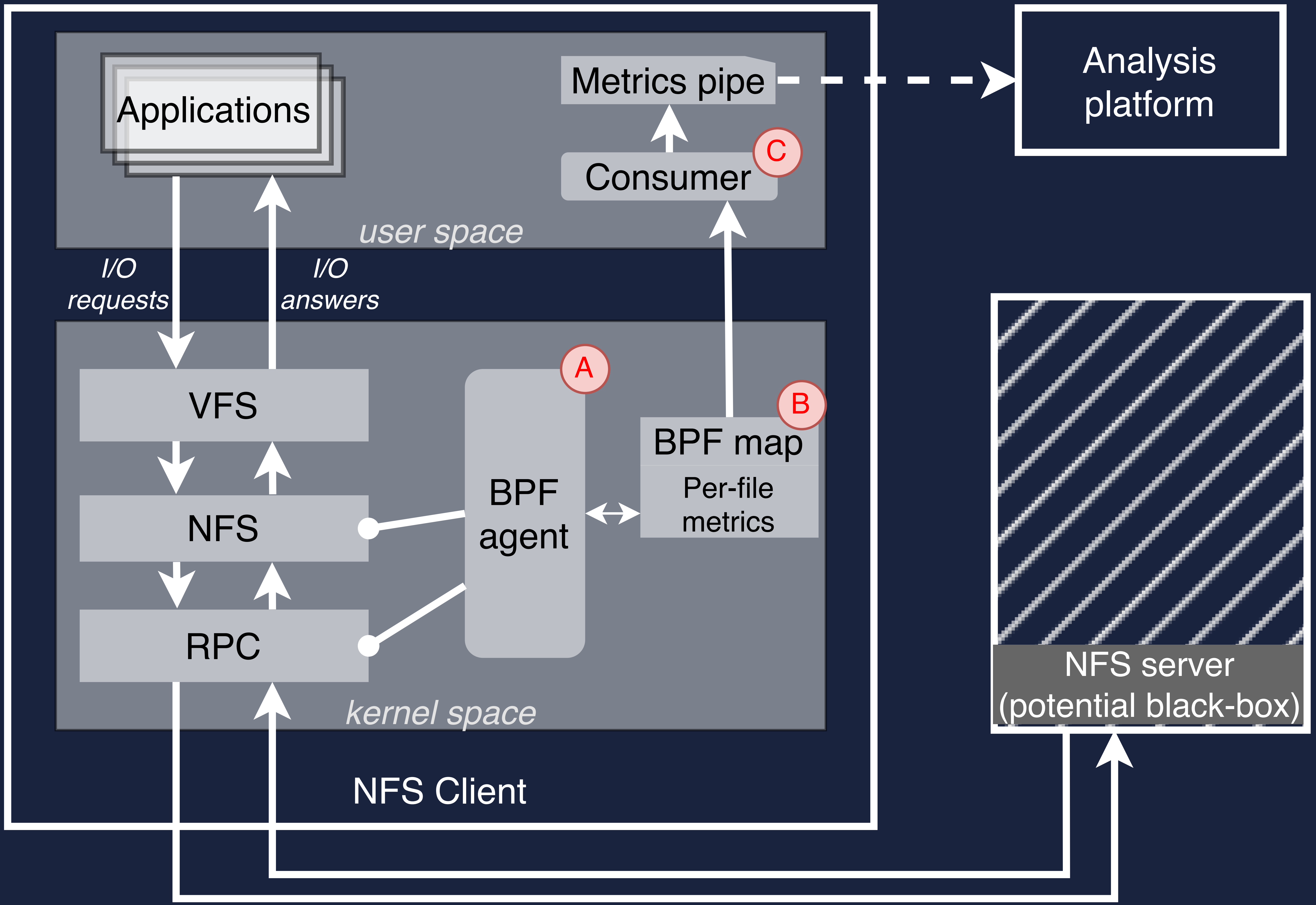
Théophile Dubuc
theophile.dubuc@ens-lyon.fr

Design

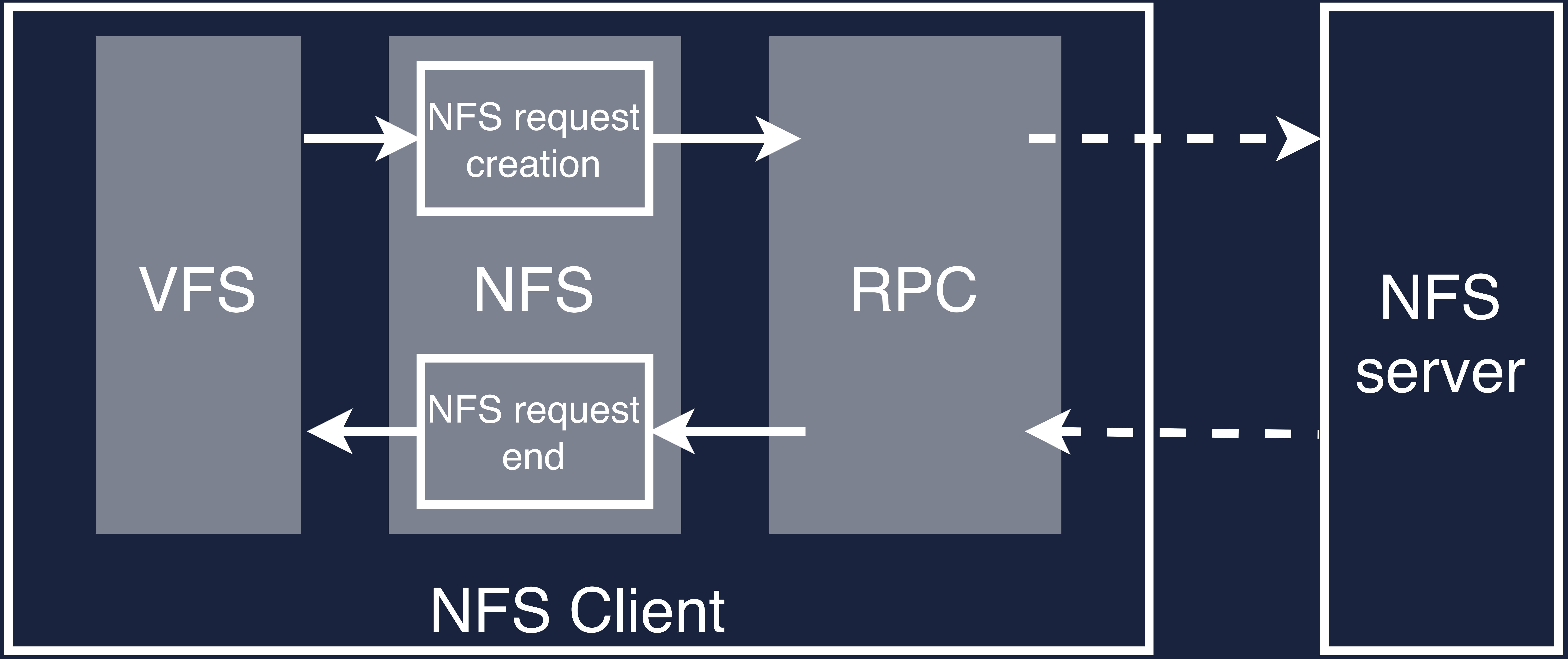
A. Raw data collection from NFS and RPC tracepoints

B. In-kernel NFS request reconstruction and storage

C. User-space polls the map to fetch NFS metrics every g seconds



Request reconstruction



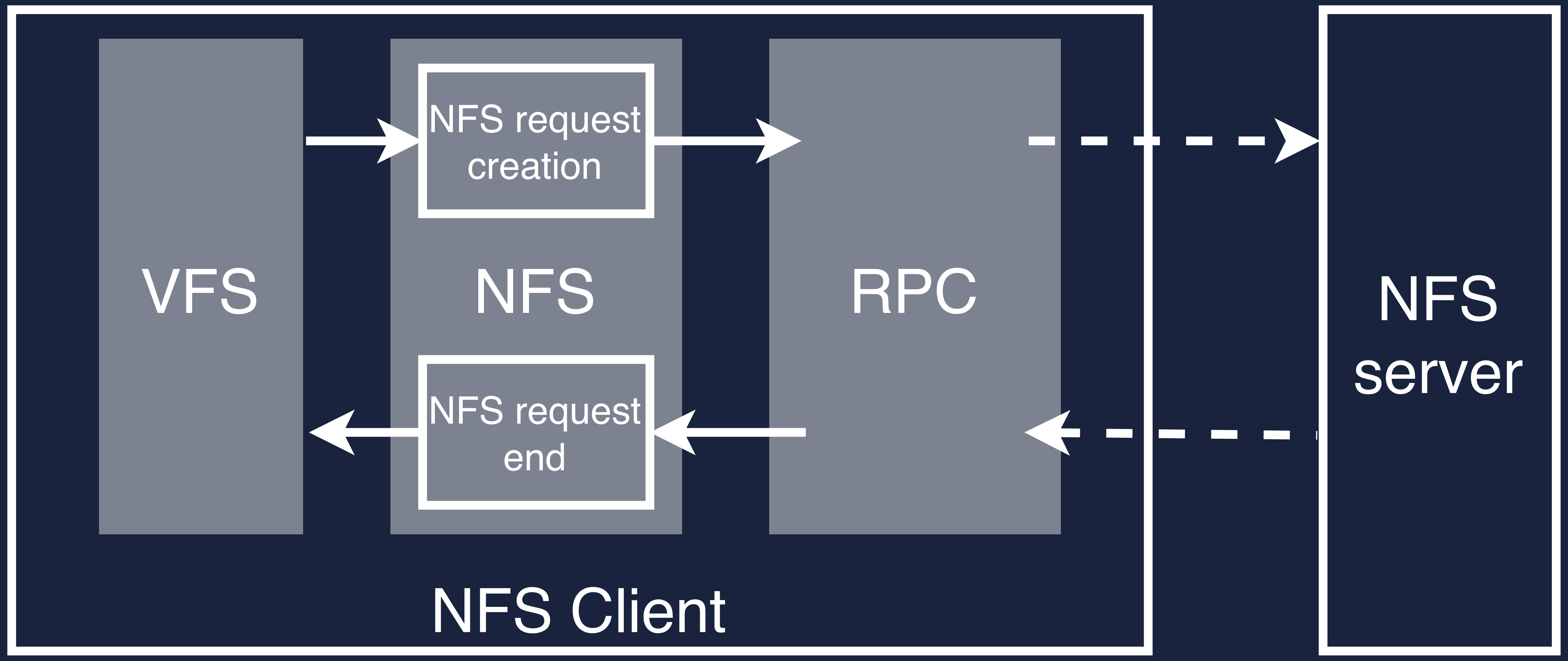
Request reconstruction

How to collect:

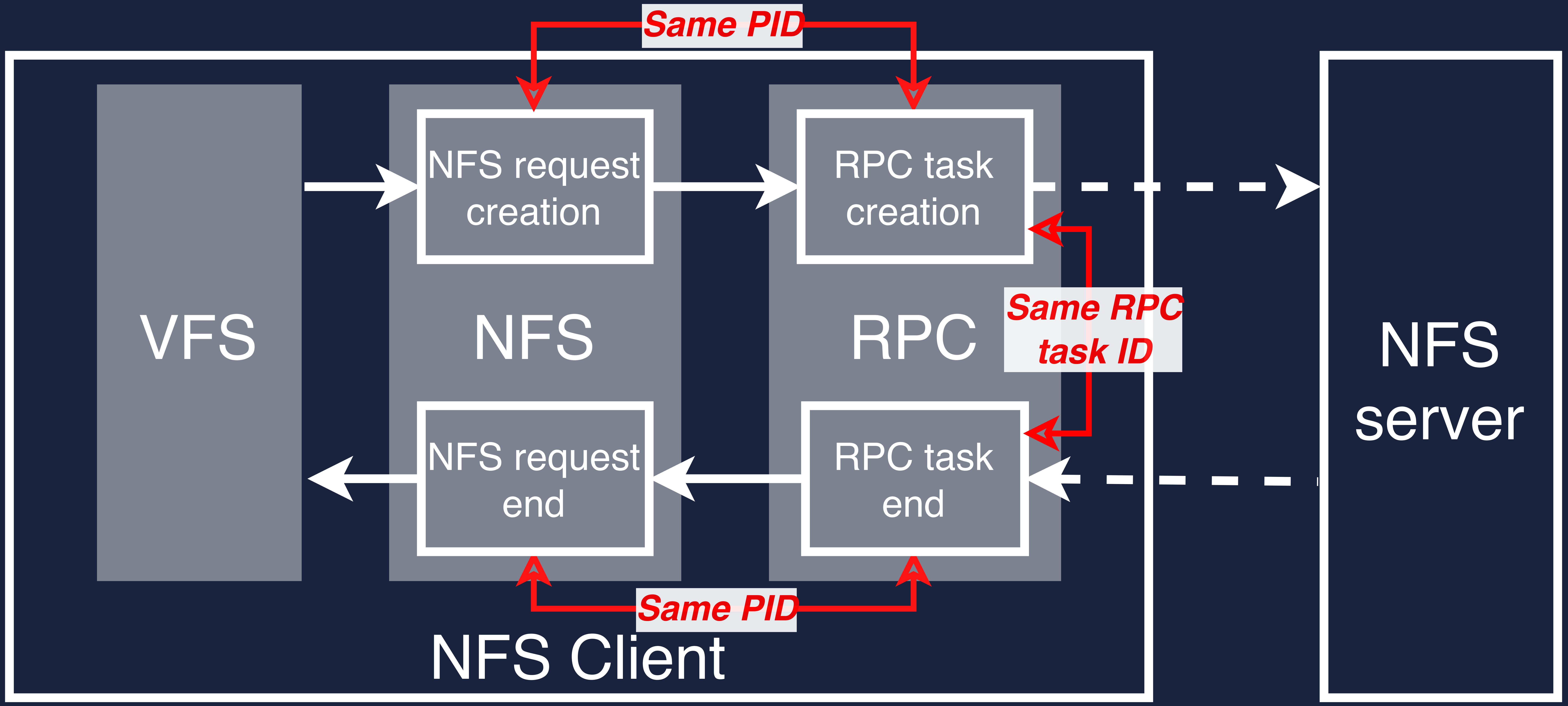
1. IOps? -> count requests

2. Throughput -> collect size of requests

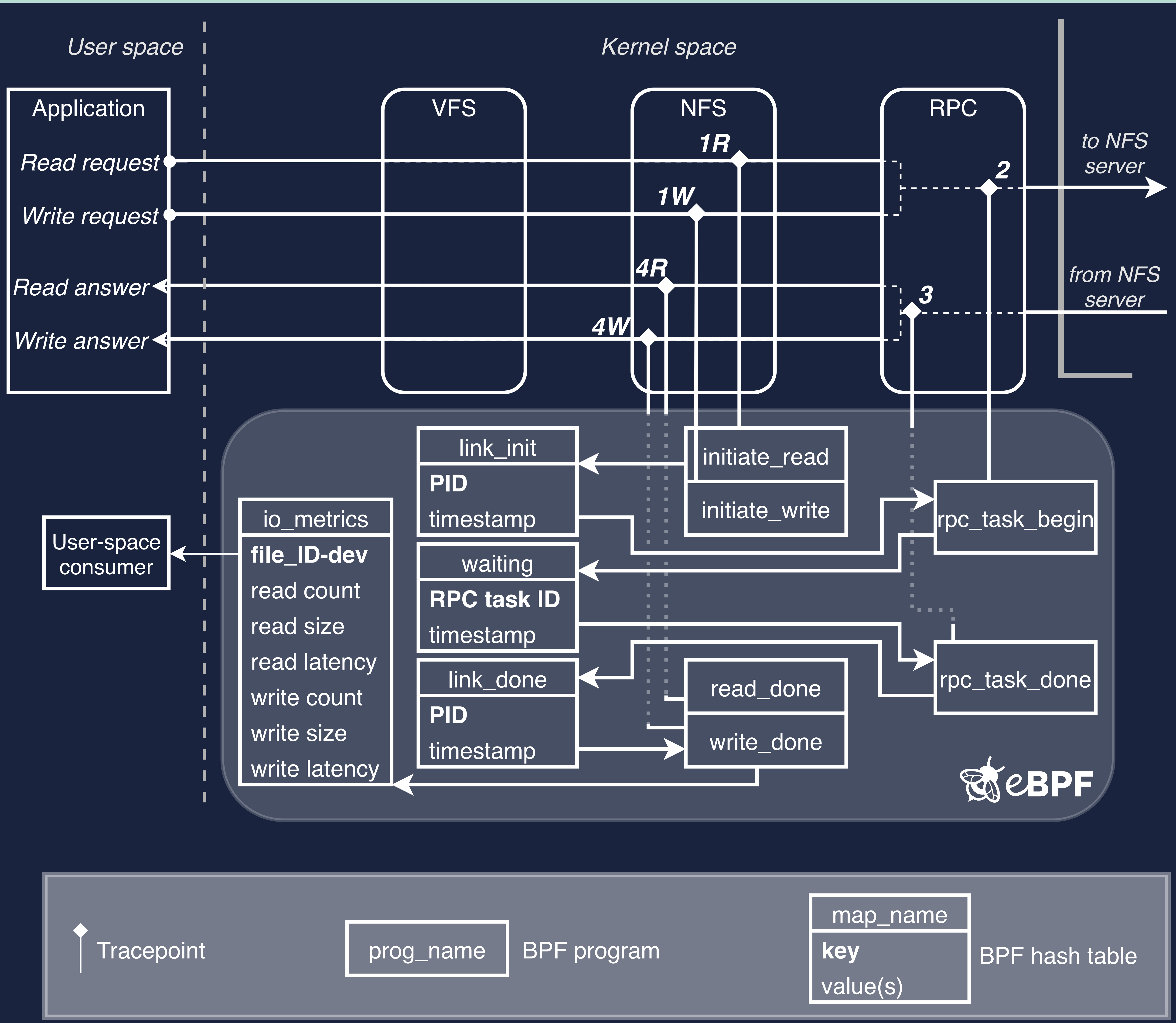
3. Latency -> delta between request end and beginning. But how?



Request reconstruction



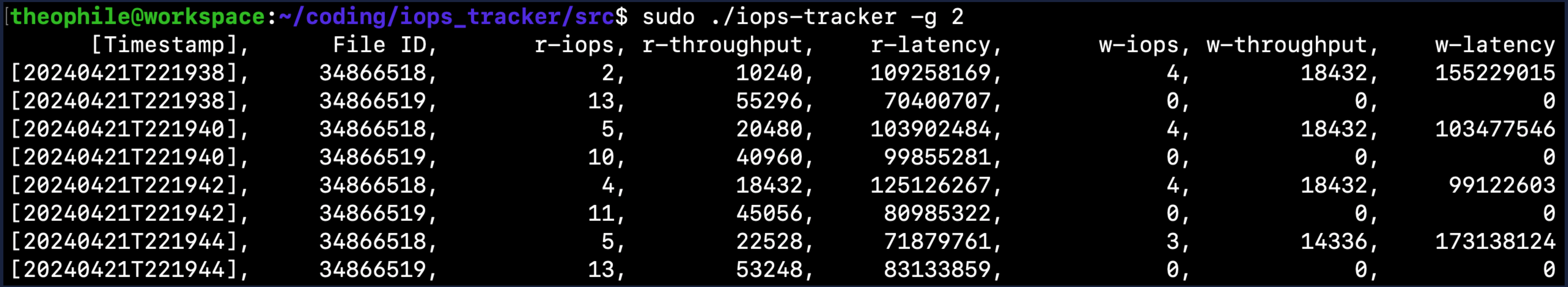
Tracepoints and BPF maps



User-space polling

Every g seconds, the user space fetches the cumulated values and computes:

- IOps = Count / g
- Throughput = Cumulated size / g
- Average latency = Cumulated latency / count



```
theophile@workspace:~/coding/iops_tracker/src$ sudo ./iops-tracker -g 2
      [Timestamp],       File ID,        r-iops, r-throughput,    r-latency,        w-iops, w-throughput,    w-latency
[20240421T221938],     34866518,            2,        10240,    109258169,            4,        18432,    155229015
[20240421T221938],     34866519,           13,        55296,     70400707,            0,            0,            0
[20240421T221940],     34866518,            5,        20480,    103902484,            4,        18432,    103477546
[20240421T221940],     34866519,           10,        40960,     99855281,            0,            0,            0
[20240421T221942],     34866518,            4,        18432,    125126267,            4,        18432,     99122603
[20240421T221942],     34866519,           11,        45056,     80985322,            0,            0,            0
[20240421T221944],     34866518,            5,        22528,     71879761,            3,        14336,    173138124
[20240421T221944],     34866519,           13,        53248,     83133859,            0,            0,            0
```

Evaluation

Overhead evaluation

Claim: the lower the server latency, the higher the impact of the tracer.

Worst-case scenario is a very fast NFS server.

- A single grid’5000 machine
- NFS server is on localhost (low network latency)
- Exported share is in memory (low storage latency)
- Variable granularity and number of fio workers

Overhead evaluation

Claim: the lower the server latency, the higher the impact of the tracer.

Worst-case scenario is a very fast NFS server.

- A single grid’5000 machine
- NFS server is on localhost (low network latency)
- Exported share is in memory (low storage latency)
- Variable granularity and number of fio workers

Result: overhead always < 3.5% (for 4000 workers and 1s granularity)

Volume of generated data

The volume of data generated in a day is:

$$(86400/g) * w * sizeof(log\_entry)$$

With

- g the granularity
- w the number of parallel workers performing I/O operations
- A log entry being 40 bytes long

With g=1 and hundreds of workers, this can be up to a few GBs per machine per day

Conclusion

- Cloud provider (and customer) use-cases require per-file NFS performance metrics
- TrackIOps extracts the metrics in real-time, with very low overhead and from the client only
- Future work: 2 directions
  - Generalize metrics exposition in the kernel to other subsystem: observability by design
  - Extend this work with TCP information to infer latency breakdown between client/network/server

Théophile Dubuc
theophile.dubuc@ens-lyon.fr