Scaling the IO Wall with Declarative IO

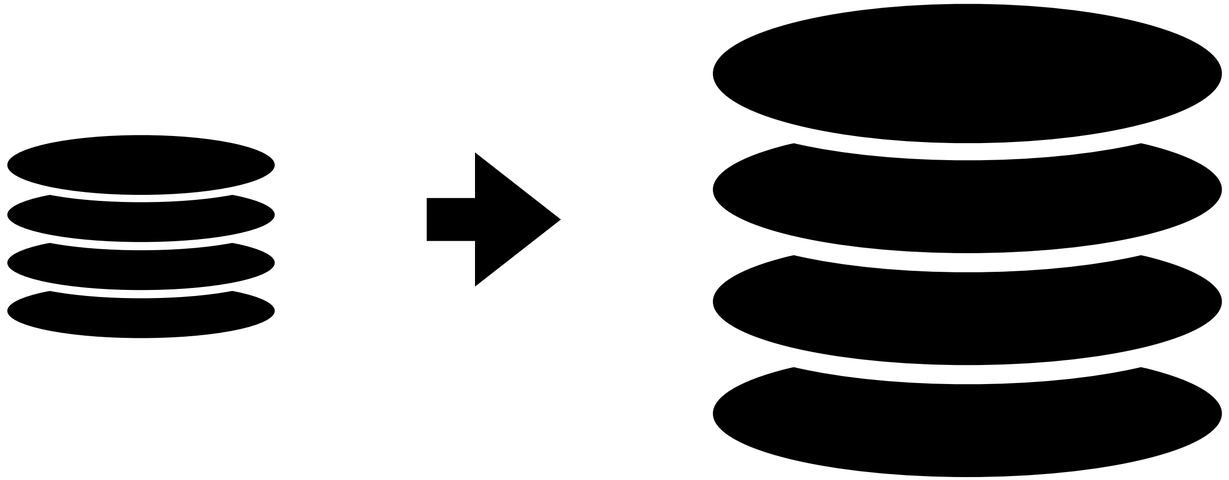
Sanjith Athlur*, **Sara McAllister*,** Theo Gregersen, Timothy Kim,
Yiwei Chen, Sarvesh Tandon, Lucy Wang,
Daniel S. Berger, Saurabh Kadekodi, Arif Merchant, Benjamin Berg,
Nathan Beckmann, Rashmi Vinayak, George Amvrosiadis, Gregory R. Ganger

Carnegie Mellon University · University of Wisconsin, Madison · Google
Microsoft Azure and University of Washington · UNC Chapel Hill

Future of HDDs: More capacity but same IO

HDDs provide low cost/byte storage capacity for exabyte-scale clusters

24-40 TB[1]
2026



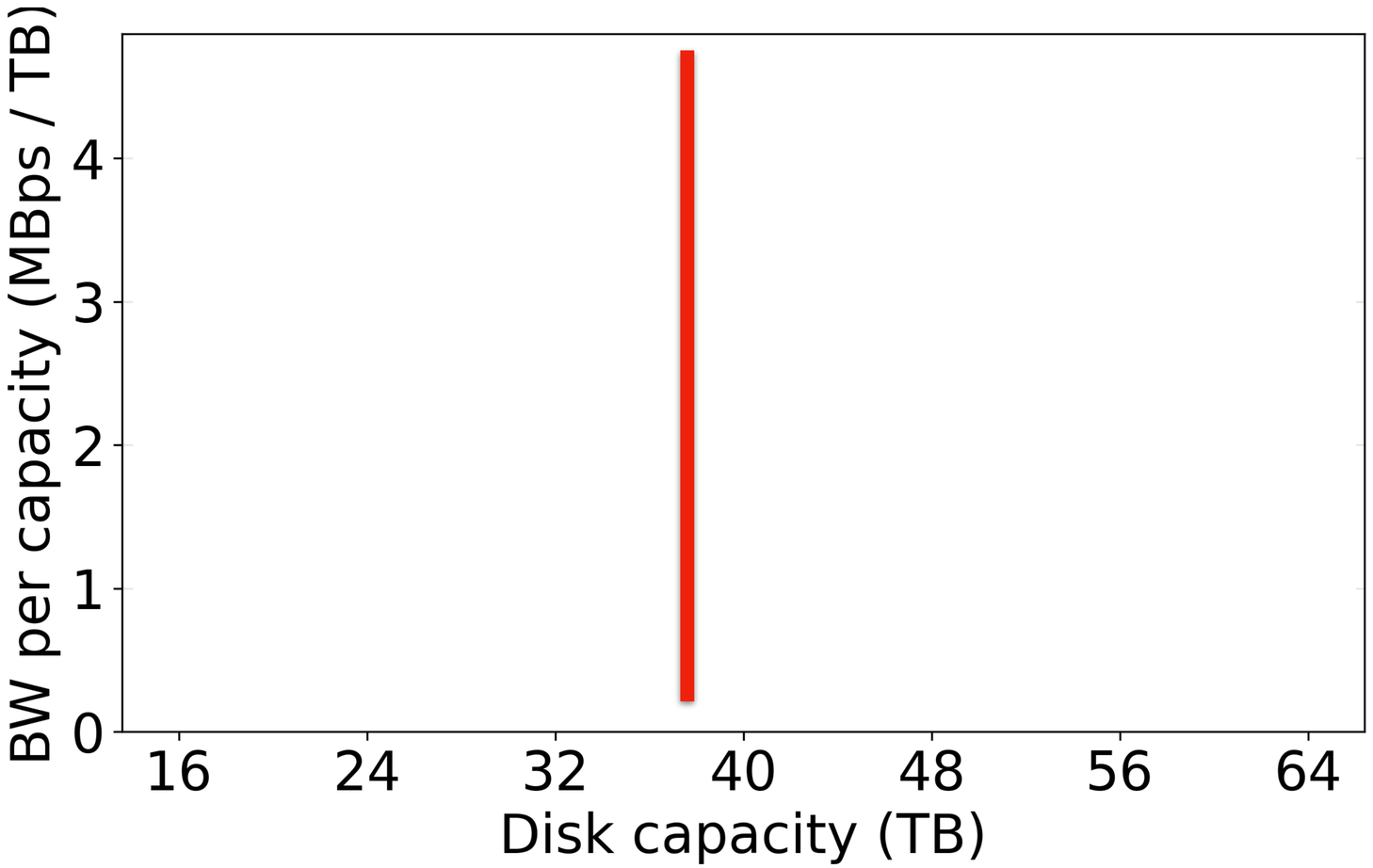
55-100 TB[1]
2031

Capacity doubling over 5 years for each HDD manufacturer

IO supply (IOPS and Bandwidth) **will not significantly increase**

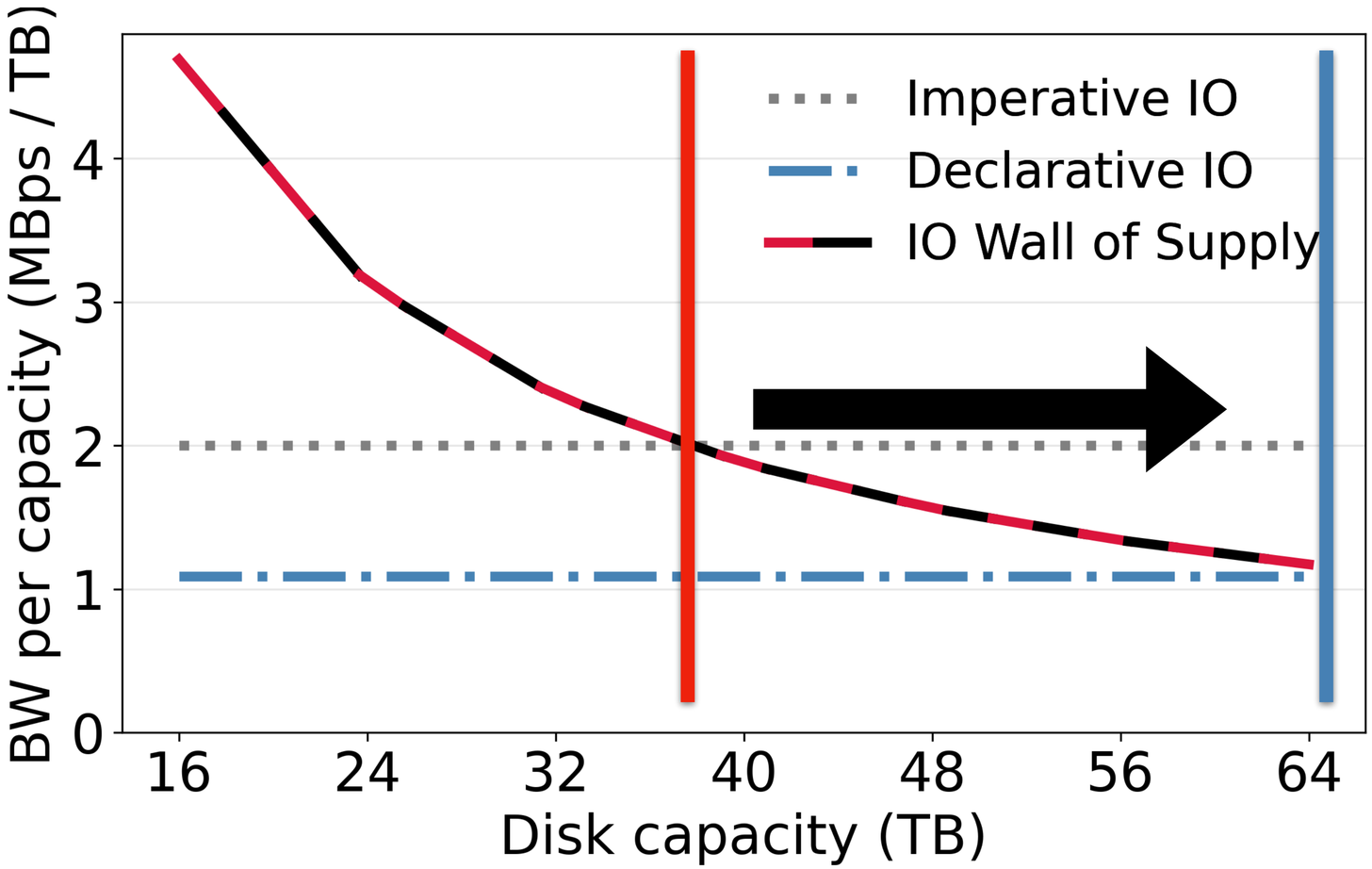
[1] https://www.forbes.com/sites/tomcoughlin/2025/05/23/seagate-shows-path-to-over-100tb-hard-disk-drives/
https://www.techradar.com/pro/toshiba-wants-to-launch-a-55tb-hard-drive-by-2030-40tb-model-set-to-appear-in-2026-new-slides-show
https://investor.wdc.com/news-releases/news-release-details/western-digital-accelerates-storage-innovation-ai-era

HDDs are crashing into an IO wall



IO demand outpaces supply after 36 TB HDDs

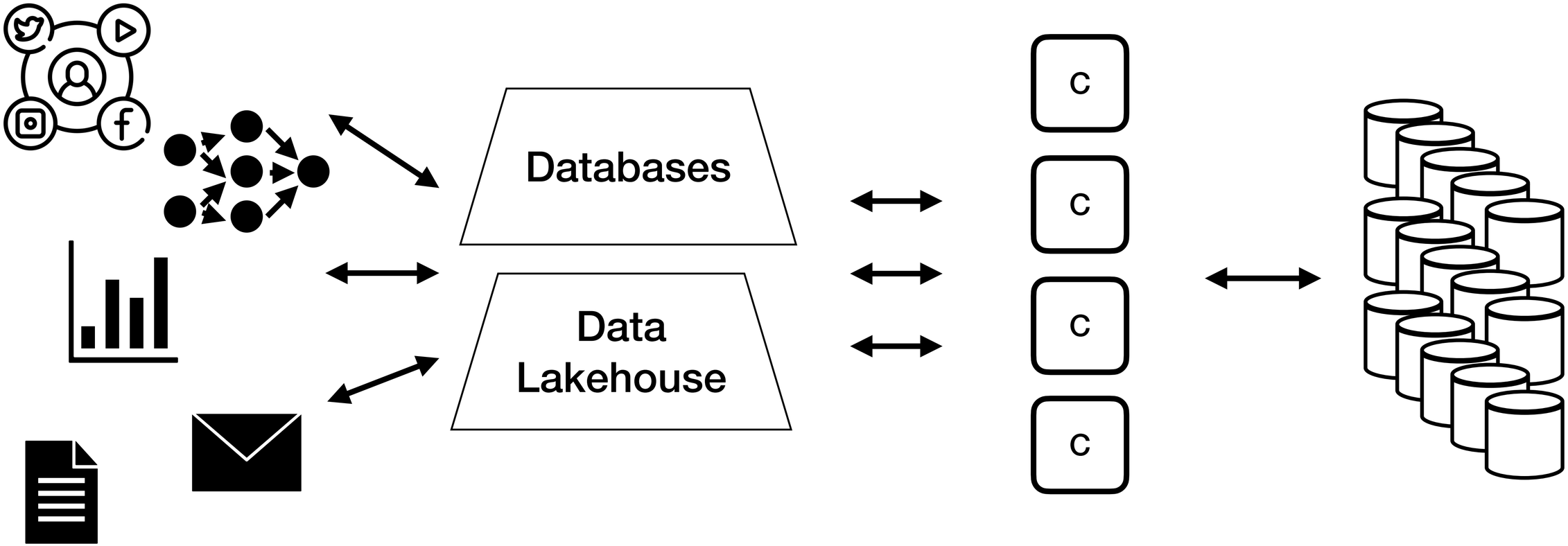
Scaling the IO Wall with Declarative IO



Declarative IO uncovers IO flexibility, enabling DINGO to combine IO

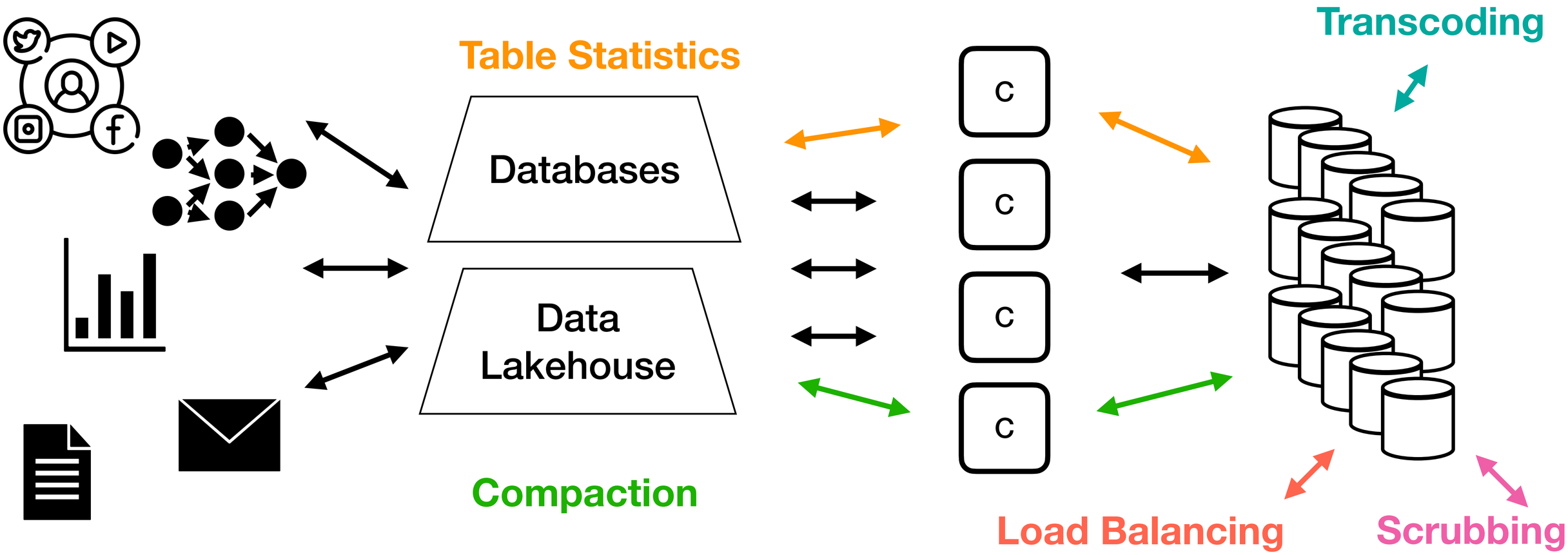
DINGO has 26-51% IO savings → allowing **1.7x larger HDDs**

Most application IO is handled by caches



Application **Data Management** **Caches** **Bulk Storage**

45-70% of disk IO is maintenance for 6 hyperscalers



Application **Data Management** **Caches** **Bulk Storage**

45-70% of disk IO is maintenance for 6 hyperscalers

Maintenance IO is split across many different tasks

See paper for more details:

- Challenges with caching maintenance IO
- Details about different maintenance tasks
- Breakdown of maintenance IO for 3 of the hyperscalers (Google, Microsoft Azure, Meta)

Transcoding

Lakehous

Paper

Application

Data

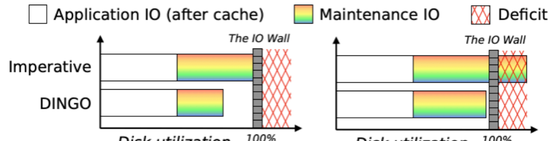
ARTIFACT EVALUATED
USENIX
AVAILABLE

Scaling the IO wall with Declarative IO

Sanjith Athlur[#*] Sara McAllister[#†‡*] Theo Gregersen[#] Timothy Kim[#]
Yiwei Chen[#] Sarvesh Tandon[#] Lucy Wang[#]
Daniel S. Berger[#§] Saurabh Kadekodi[‡] Arif Merchant[‡] Benjamin Berg[¶]
Nathan Beckmann[#] Rashmi Vinayak[#] George Amvrosiadis[#] Gregory R. Ganger[#]
[#]*Carnegie Mellon University* [†]*University of Wisconsin, Madison* [‡]*Google*
[§]*Microsoft Azure and University of Washington* [¶]*UNC Chapel Hill**

Abstract

HDD capacities will greatly increase over the next ten years, lowering cost-per-TB in large-scale storage systems. Unfor-tunately, device bandwidth will not grow proportionally to



026

Imperative IO → Arbitrary timing + ordering

Scrubbing: Check that every block is still valid every month

```
def scrubbing():
  while True:
    for block_id in ALL_BLOCKS:
      scrub(block_id)
      sleep(RATE_LIMITER)
    sleep(ONE_MONTH)
```

Defines an order to read blocks

Uses data from IO

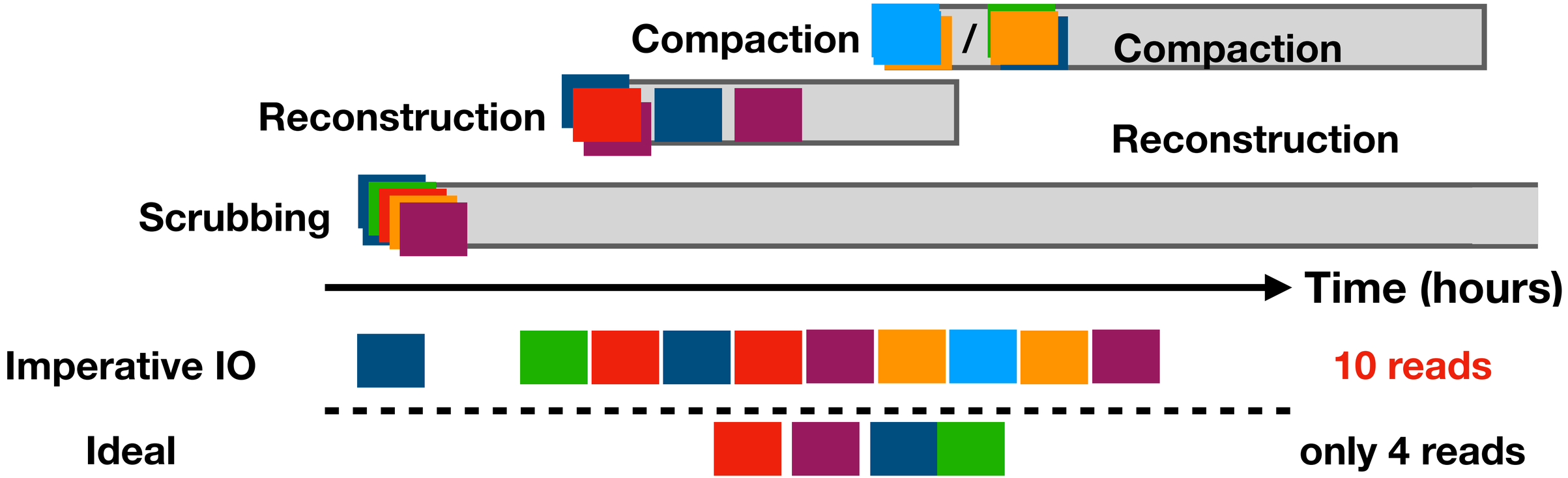
Controls time of execution

Imperative IO unnecessarily imposes order and timing

Imperative interfaces necessitate inefficiency

Imperative interfaces force uncachable, uncoordinated, and often repeated IO

Exposing time, order, & data flexibility



Time, order, & data flexibility w/in tasks allows IO reuse

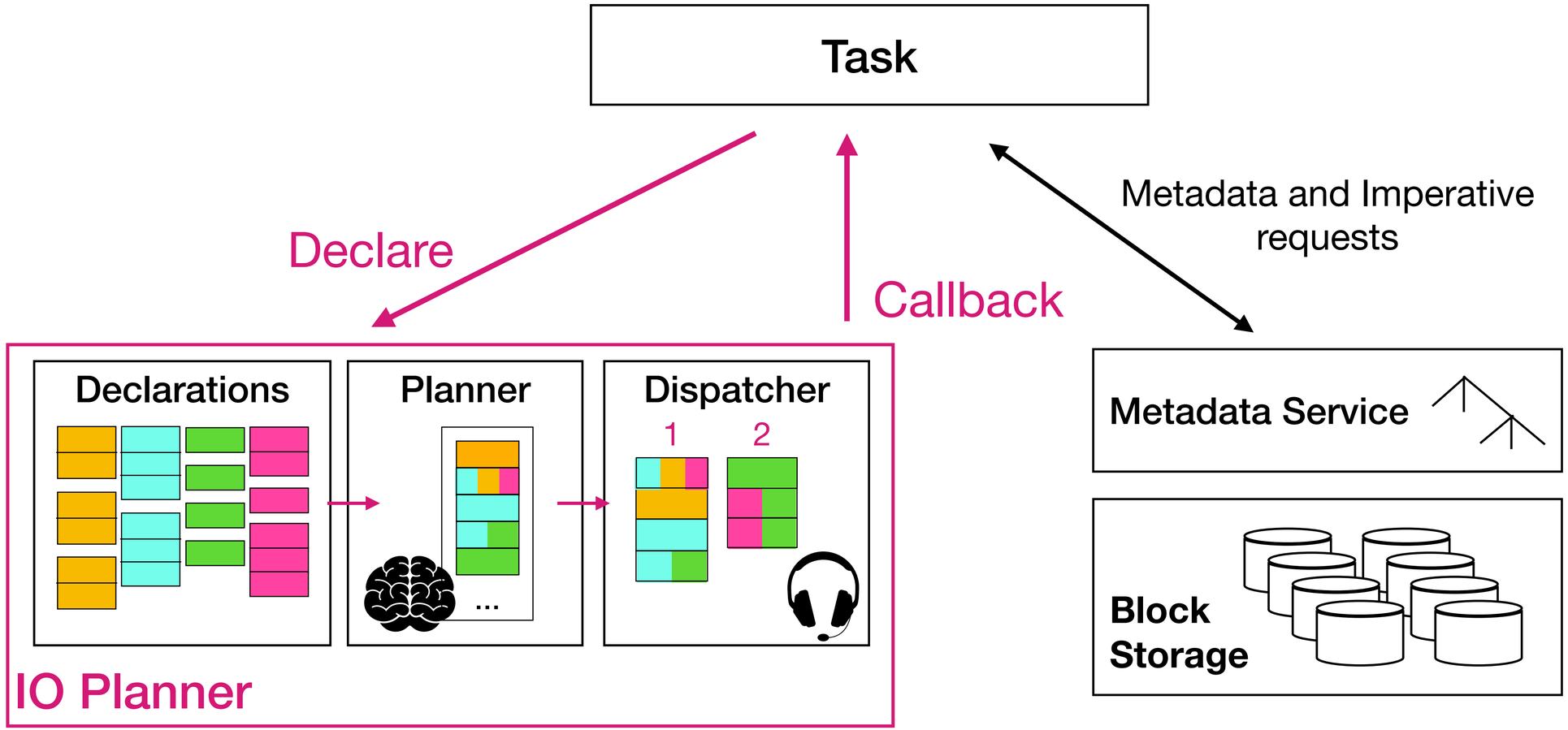
Declarative IO enables storage to see IO flexibility

Scrubbing: Check that every block is still valid every month

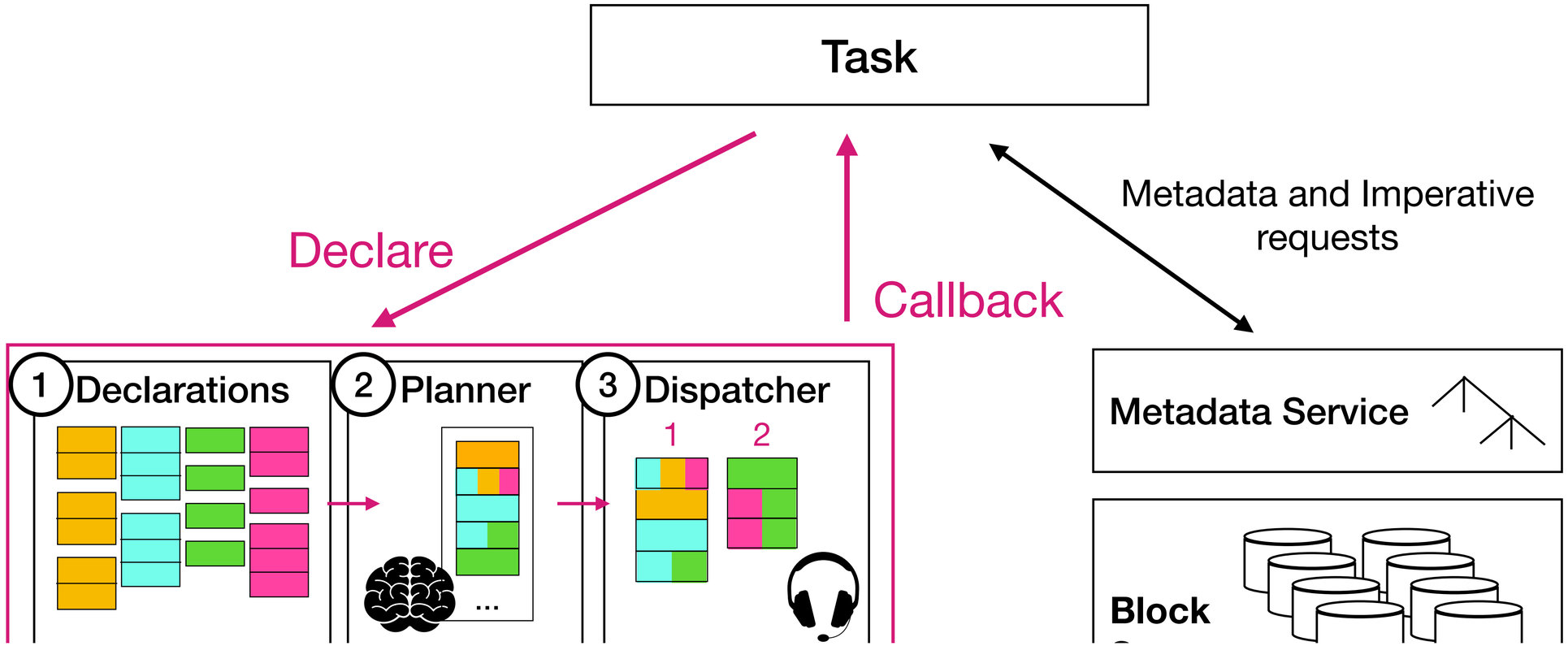
```
def scrubbing():
  while True:
    for block_id in ALL_BLOCKS:
      scrub(block_id)
      sleep(RATE_LIMITER)
    sleep(ONE_MONTH)
```

```
def scrubbing_declarative():
  while True:
    declare(
      read_sets=[
        BlockSet(block_id)
        for block_id
        in ALL_BLOCKS],
      deadline=ONE_MONTH,
      callback=scrub,
    )
    sleep(ONE_MONTH)
```

DINGO: A storage system with Declarative IO



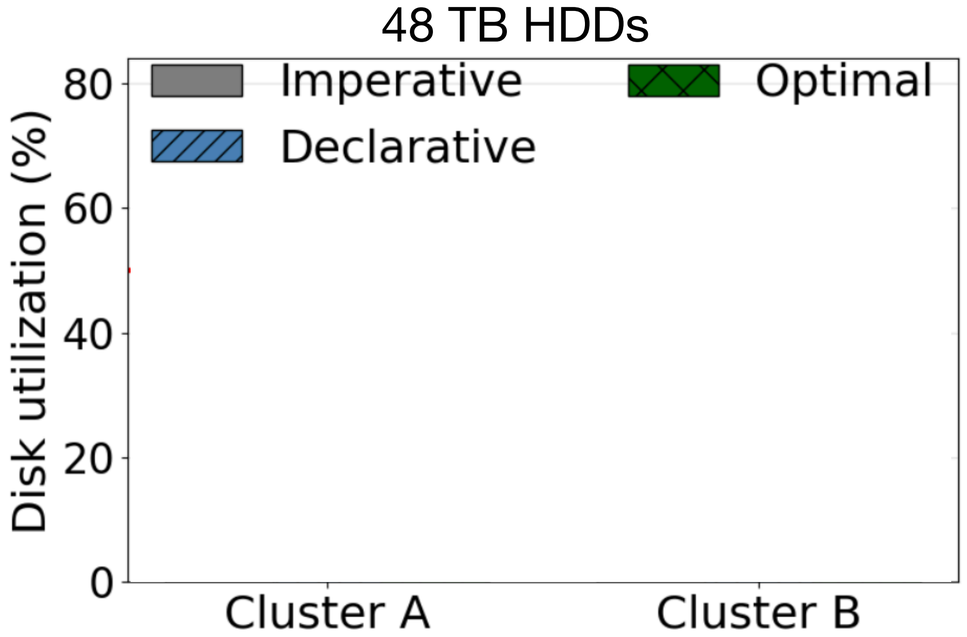
Challenges in designing DINGO



1. How to keep declaration state to **scale to exabyte clusters**
2. How to plan IO to **maximize IO reuse while meeting deadlines**?
3. How to **manage caches** to keep data for callbacks with limited cache space?

DINGO achieves near optimal disk utilization

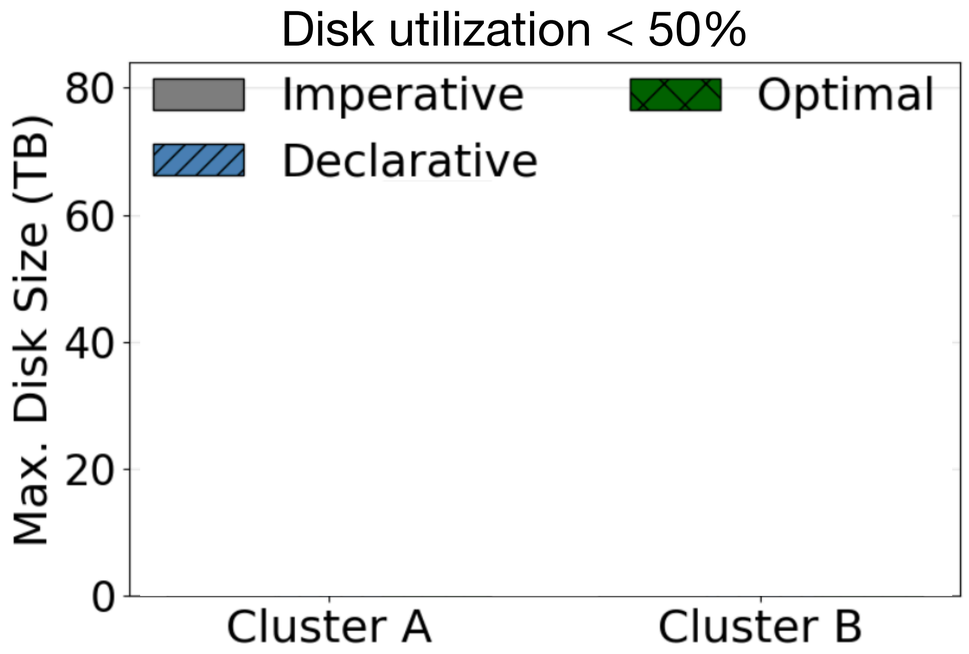
Simulated based on maintenance distributions from multiple hyperscalars



Reduces IO on 48 TB drives → Enables increasing load on existing drives

DINGO enables 1.7x larger HDDs

Simulated based on maintenance distributions from multiple hyperscalars



Enables >60 TB HDDs

Scaling the IO Wall with Declarative IO

Maintenance tasks constitute of 45-70% HDD IO

- No maintenance tasks dominates

Maintenance is time, order, and data flexible

- And tasks access overlapping sets of bytes far apart in time

Declarative IO: Distributed storage interface to express IO flexibility

- Tasks declare IO needs (what to read, how much to read, deadline)

DINGO: IO planner to coordinate declarations

Evaluation: DINGO saves **26-51%** of maintenance IO

- DINGO enables deployment of **1.7x larger HDDs**

Paper

Code