HotStorage 2021

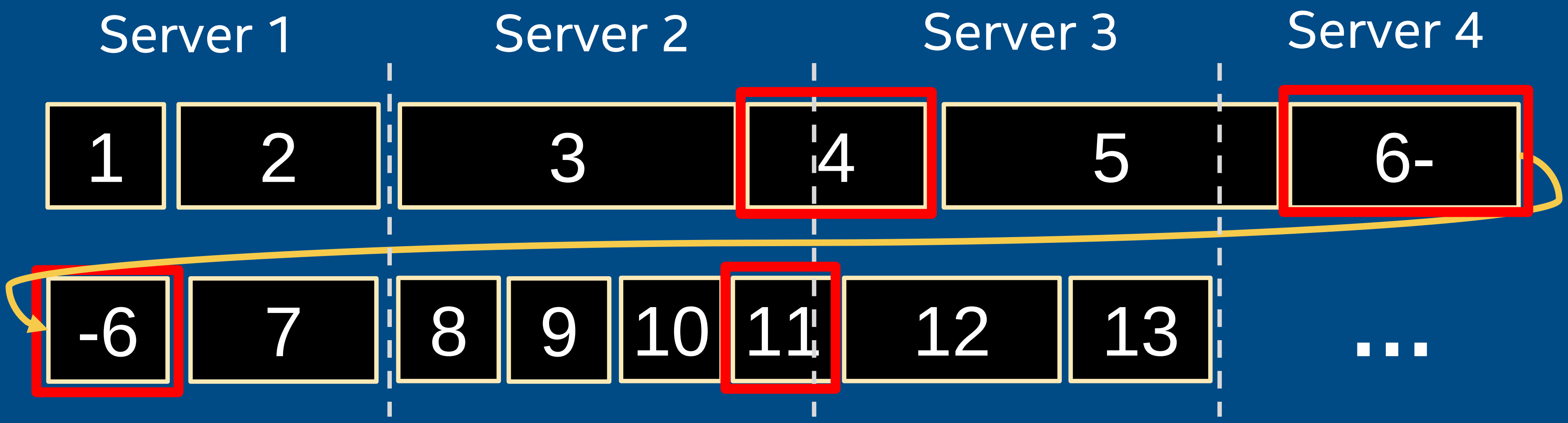
Enabling Near-Data Processing in Distributed Object Storage Systems

Ian F. Adams, Neha Agrawal*, Michael P. Mesnier

Intel Labs, *Portland State University (work done while at Intel)

intel®

NDP's Data Distribution Problem (1/2)



- Near Data-Processing (NDP) is a simple concept
  - Move compute to the data to reduce IO traffic
- But, modern distributed storage make this tricky
  - Sharding/striping systems don't respect semantic boundaries

NDP's Data Distribution Problem (2/2)

S3 Select

Ceph Skyhook

MINIO +

- Some systems punt on the issue…
  - Assume replicas, or data collation…
  - *Erodes NDP goodness, makes life hard for computational storage devices*

- Others go for application specific approaches
  - Ceph Skyhook application has awareness for data placement of tables
  - Some MapReduce apps have record boundary awareness for inter-node reads
  - *Can be complex to realize in practice, difficult to retrofit or extend general purpose systems*

Can we just… you know….. not do that?

More concretely:

1) Can we lay out data in such a way to avoid ***boundary conditions***?

2) Can we do so without ***application specific changes*** to the underlying systems?

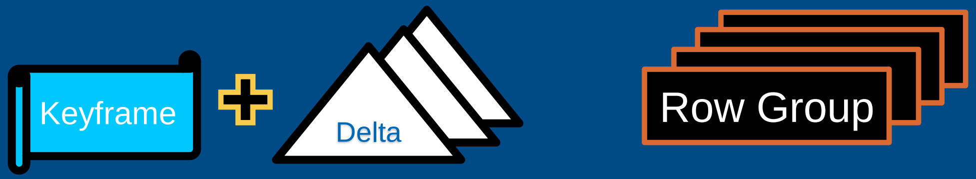
3) Can we do so in a way that ***enables simple NDP*** without a lot of storage system awareness?

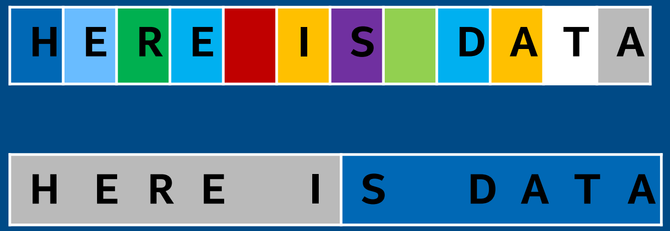
Solution: Data Alignment

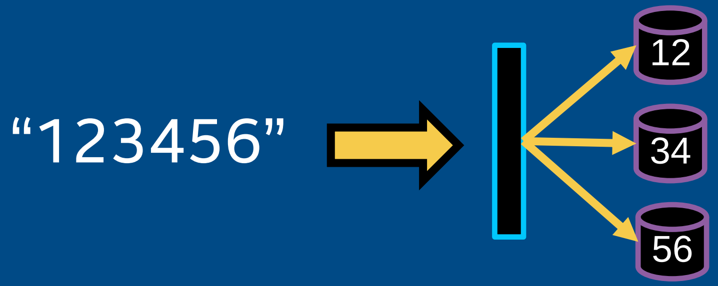
Adjust data locations elements within shards and stripes

Background and Observations

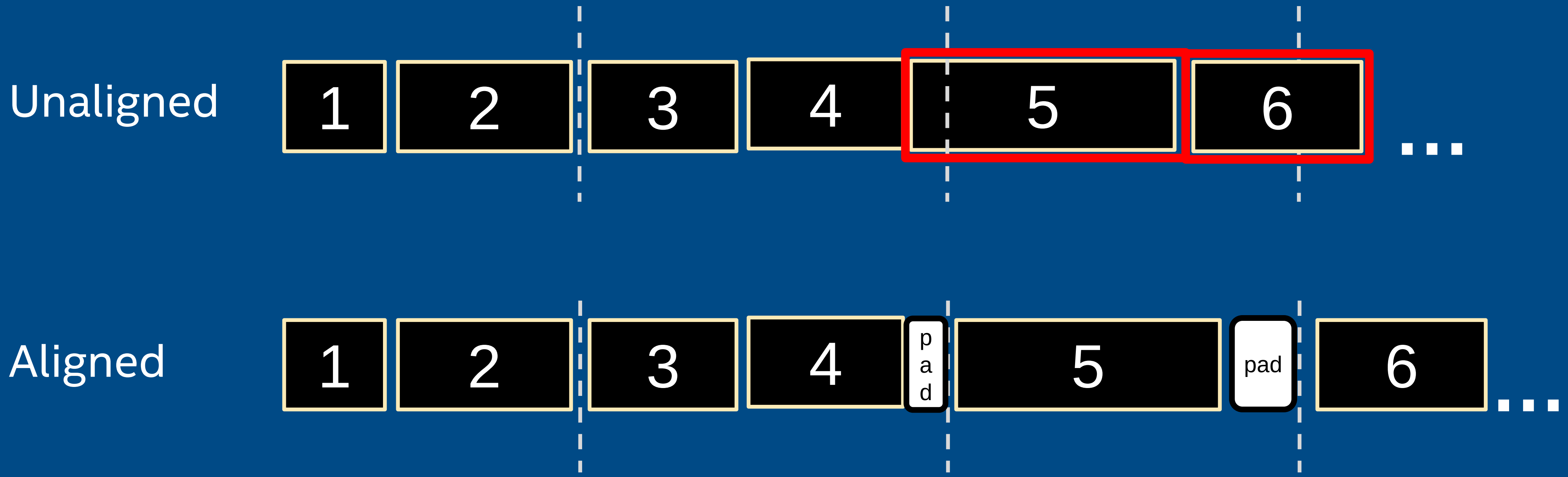
- Lots of useful operations can be done on data elements
  - Keyframes in video, row groups of columnar data…
- Distributed storage shards data at relatively coarse granularity
  - In contrast to traditional RAID, MiB ranges are common
  - Fit many data elements in a shard
- Storage systems lay out data in a predictable manner
  - Shard and stripe boundaries are predictable



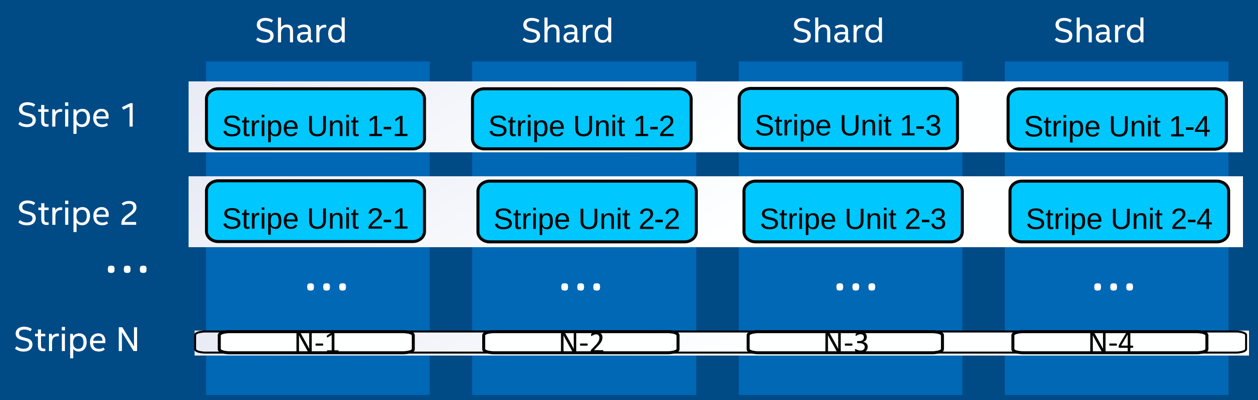




Data Alignment Intuition



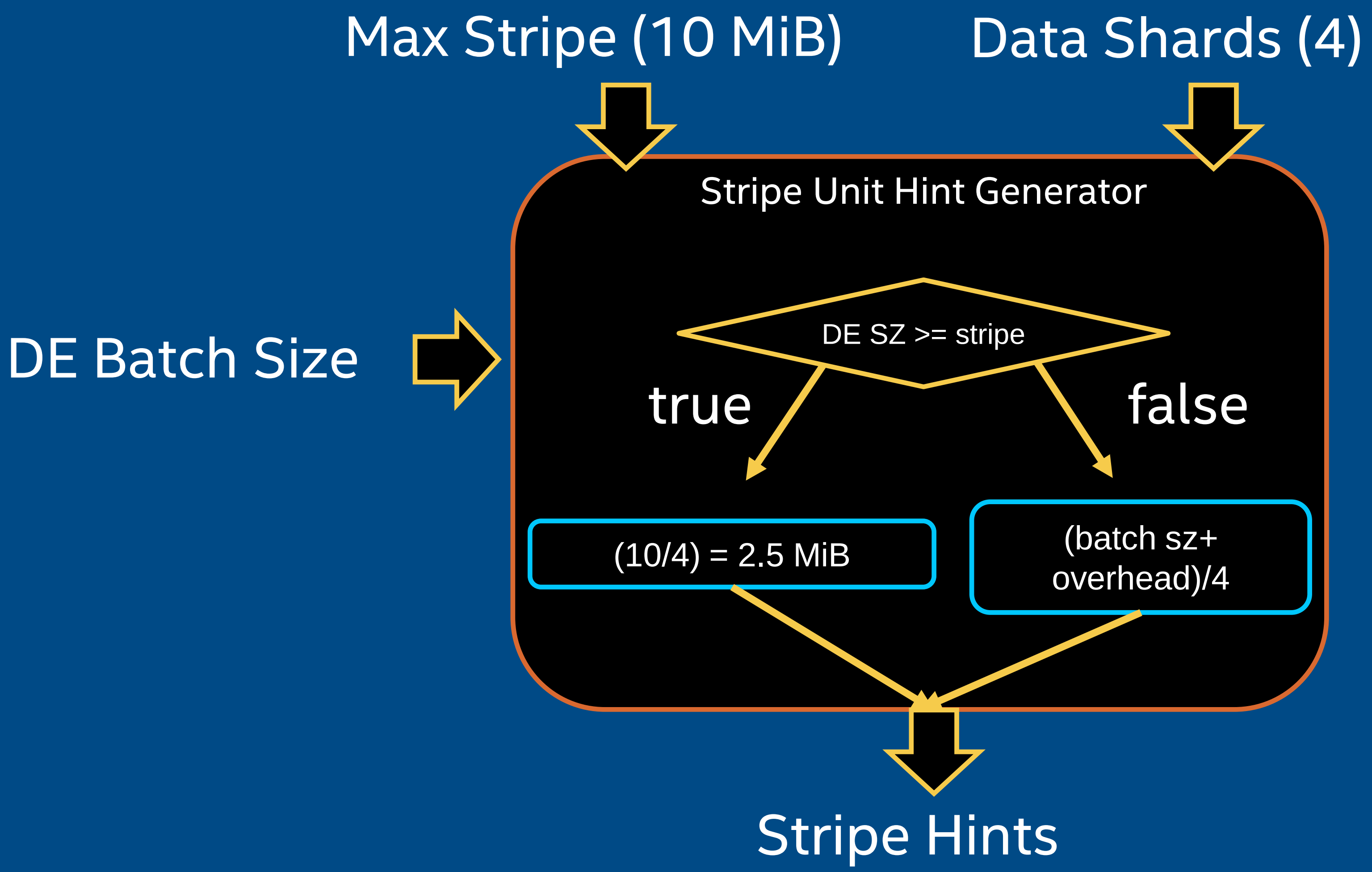
Hints for Striped Environments



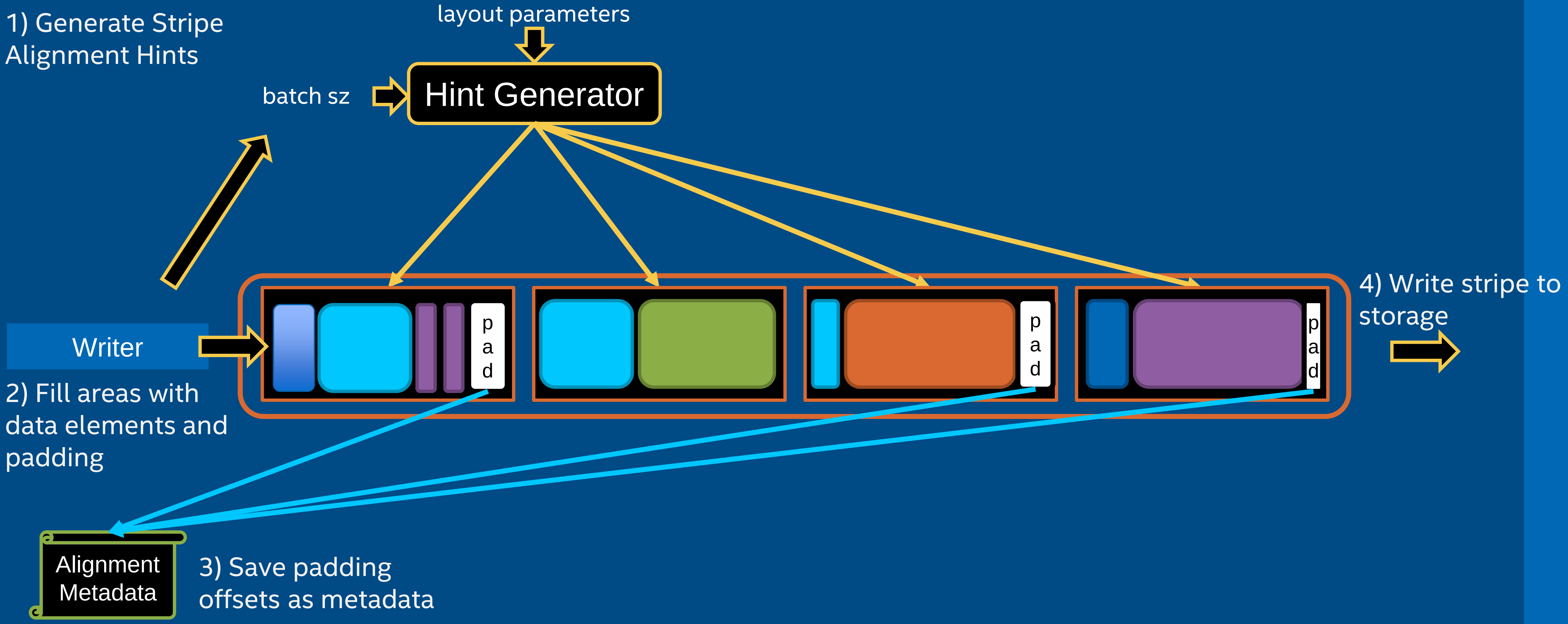
- Stripes are written *across* shards, and are predictable in their sizes
  - Stripe units reside on one shard
  - We can use stripe units sizes to generate *alignment hints.*
- Alignment hints are simply byte offsets that delimit a stripe unit border
  - With these hints, we can tell when a data element will be in a boundary condition

Generating Alignment Hints

- Hint generator inputs
  - Maximum stripe size
  - Number of data shards
  - Estimated size of next batch of data elements (DE)
- Data elements are laid out within the bounds of the hints
  - Unused space is padded out
  - If the batch size is less than a stripe, we assume a dynamic stripe resizing.
    - Overhead to ensure space for padding



Layout Example



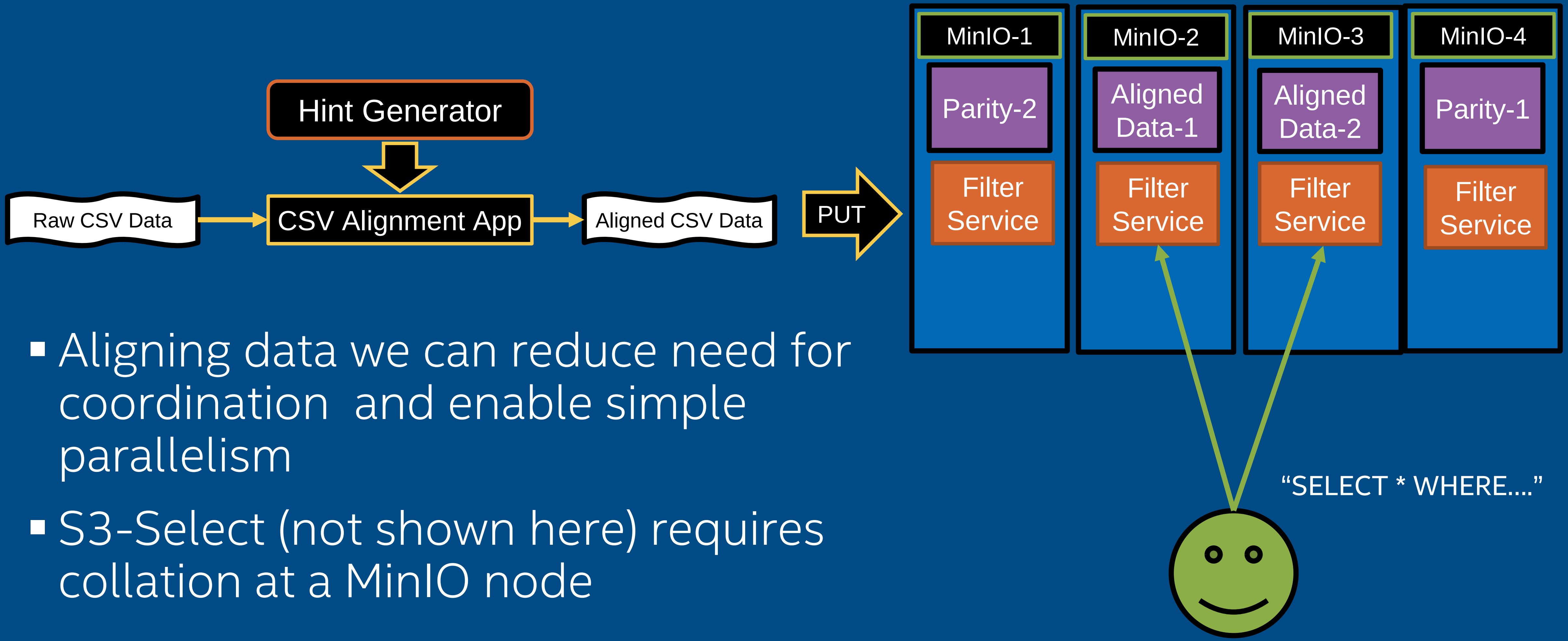
Proof of Concept: CSV Data Filtering

Aligned data stored in MinIO, then processed via parallel containers

Experimental Overview

- 4 Node (2+2) MinIO Deployment
- 1.7 GiB CitiBike dataset
  - Used alignment hints to pad data
- Simple filter query that selects ~25k records
  - Container service offers SQLite functionality
- Compare to built in S3-Select
  - Using unaligned data
- Queries issued from separate node from MinIO

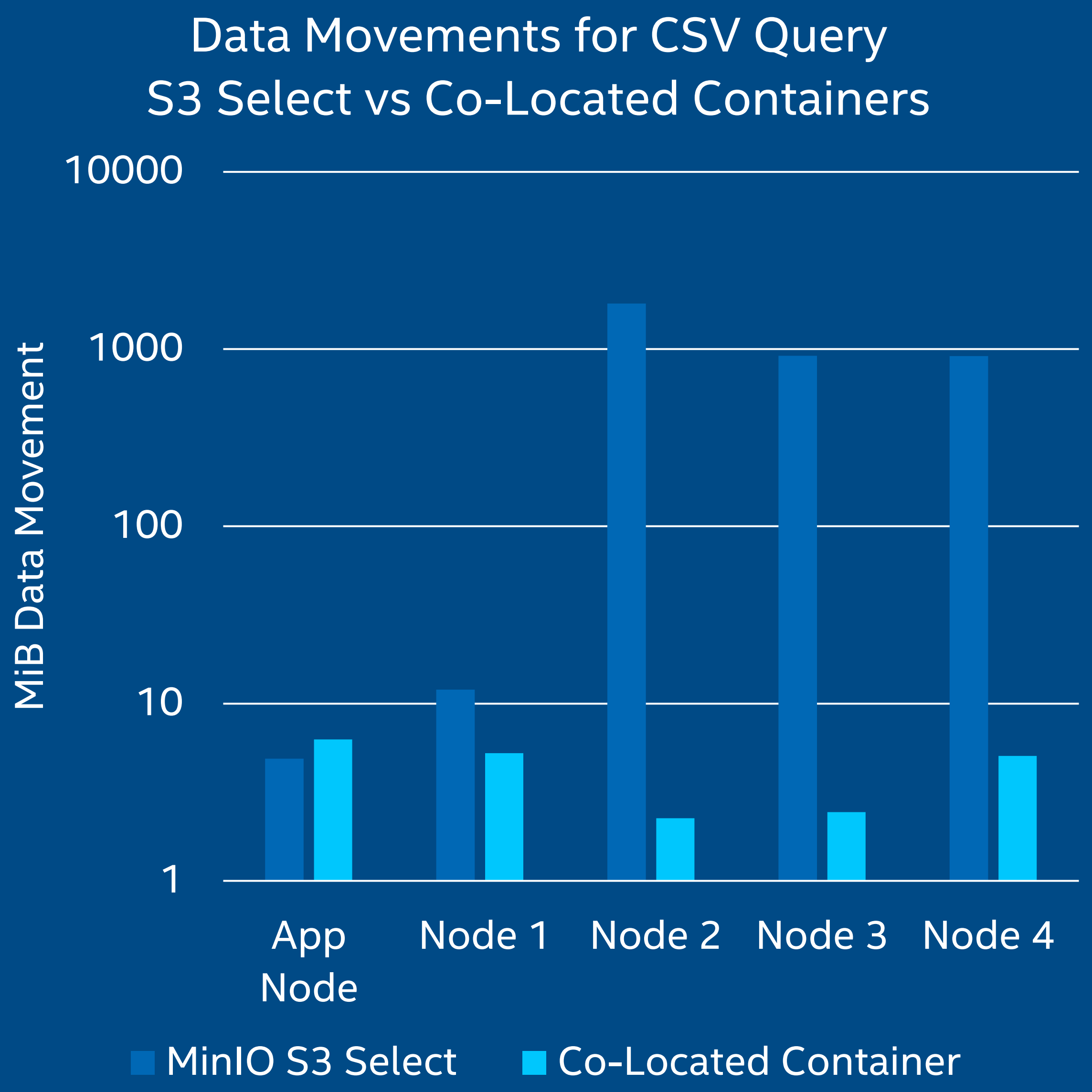
CSV Data Filtering + MinIO



- Aligning data we can reduce need for coordination and enable simple parallelism
- S3-Select (not shown here) requires collation at a MinIO node

Results Overview

- It works!
  - We can trivially parallelize many filtering operations
- Compared to built in filtering, significantly reduces data movement
  - Will vary on selectivity
  - Some collation may still be needed, e.g. a SUM
- Low overhead (for CSV)
  - ~8 KiB for padding and extra metadata



Lots More To Do!

- We started with a very simple example, a good start, but…
  - What about more sophisticated data types?
- More native support for NDP
- Quality of service (QOS)
  - We have a complex, distributed scheduling problem

Thanks for your time! Please reach out with questions or comments!