Predicting and optimizing the performance of HDF5 applications

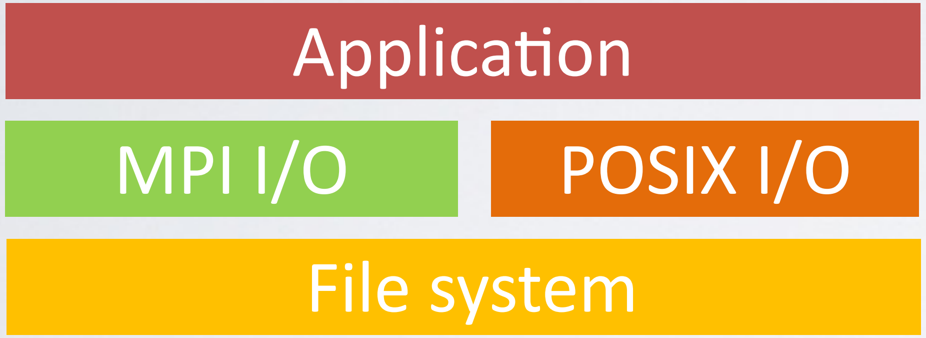
Donghe Kang

kang.1002@osu.edu

THE OHIO STATE UNIVERSITY

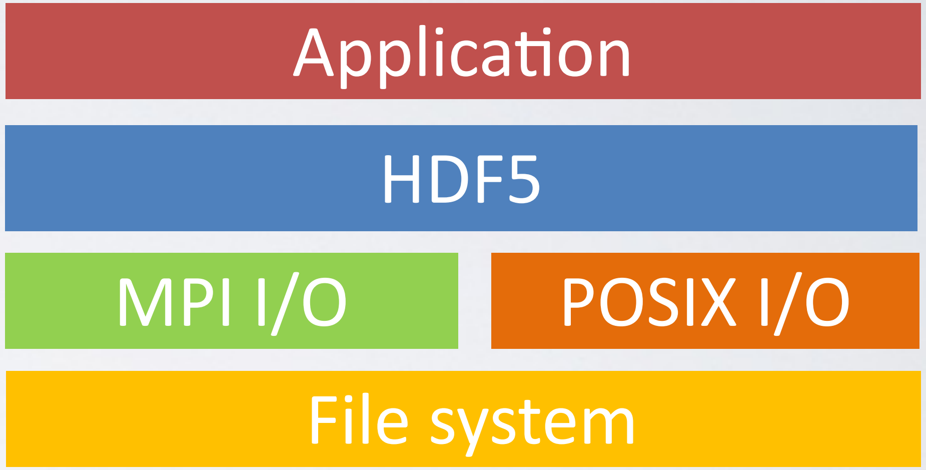
Software stack

- **Previous modeling work**



Applications directly access data. Previous works build models to predict the I/O time.

- **This work**

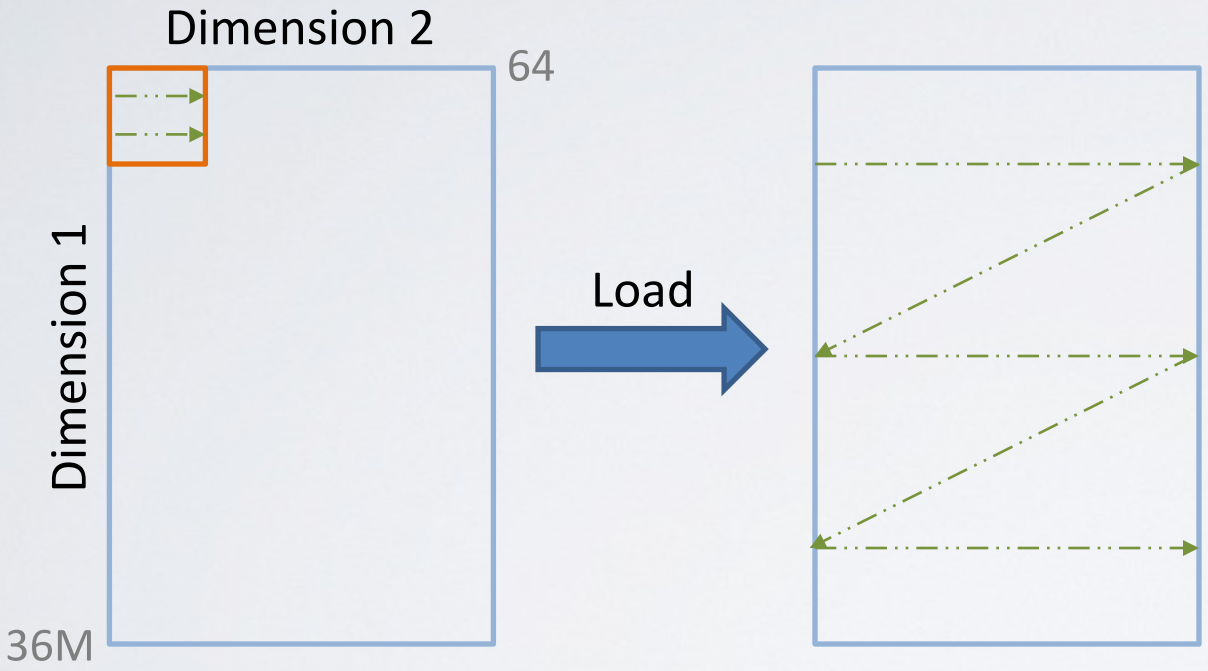


How to predict the end-to-end performance of the array I/O requests?

Predict Optimize

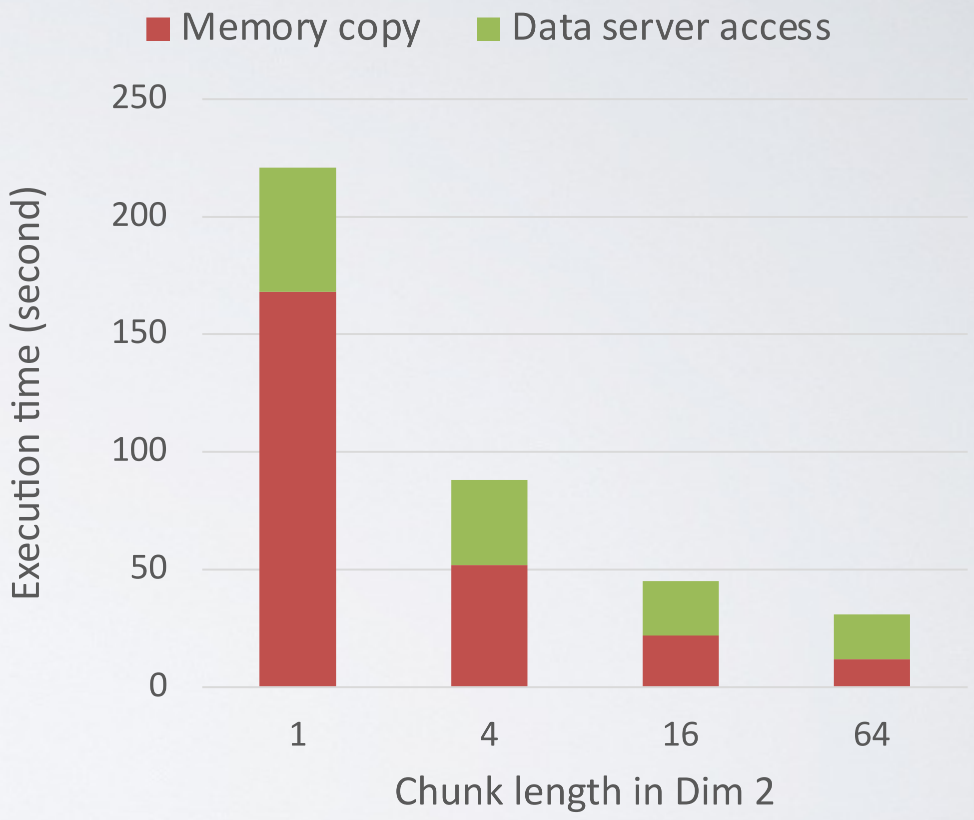
Motivation

- Read an entire array

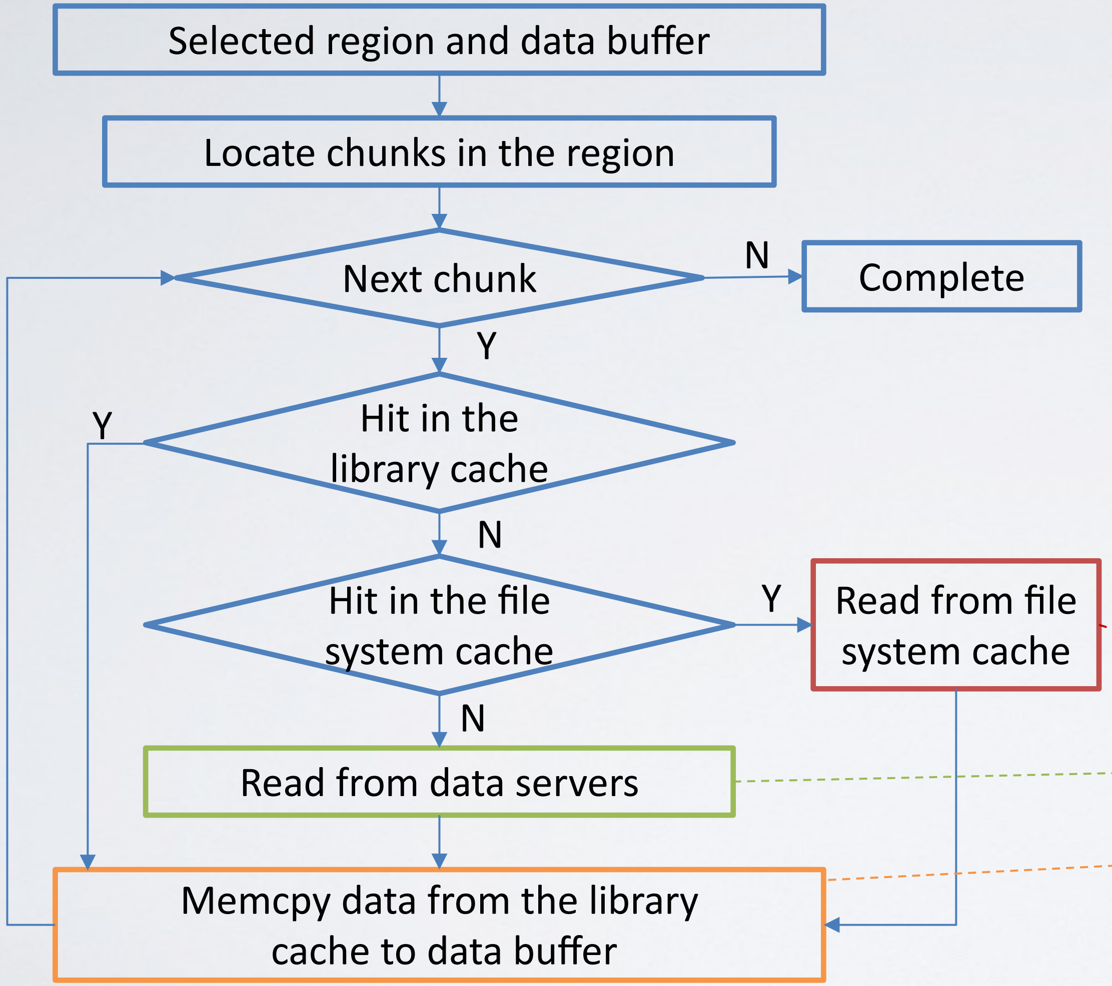


Chunked on-disk storage layout

In-memory serial buffer



Read path in HDF5



The cache read and memory copy operations can spend up to 90% and 87% of the execution time in our experiments.

An end-to-end model should cover all the steps in the I/O path.

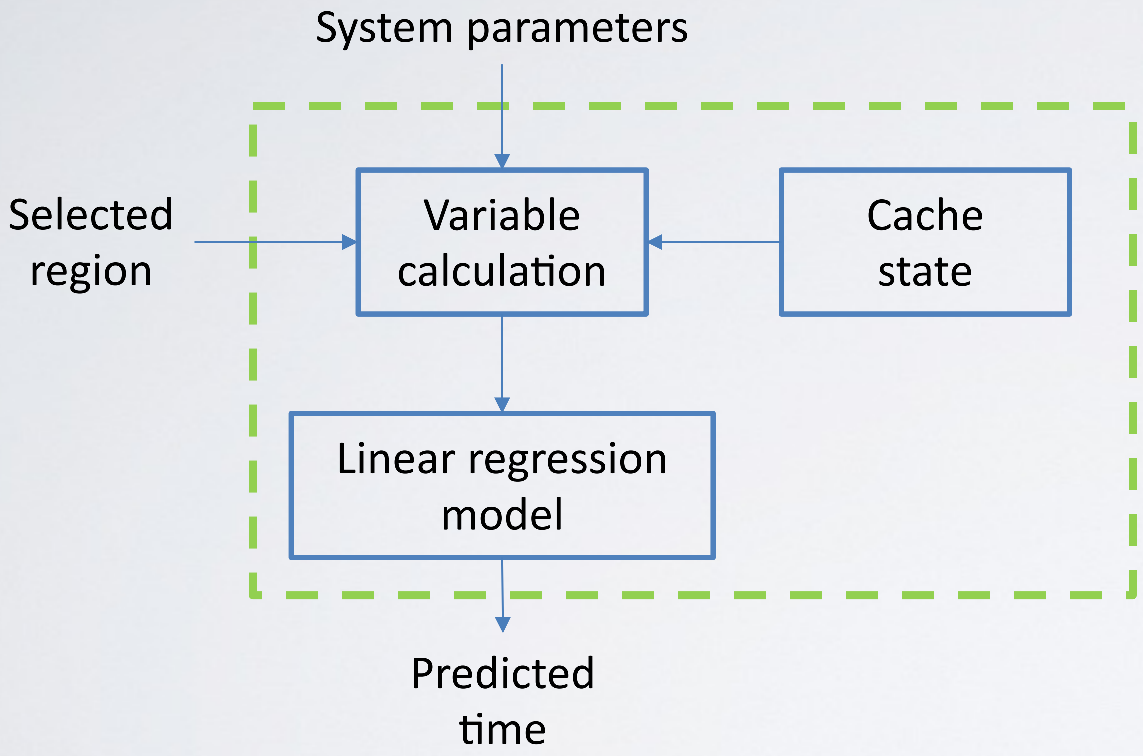
End-to-end time =

cache read time +

data server access time +

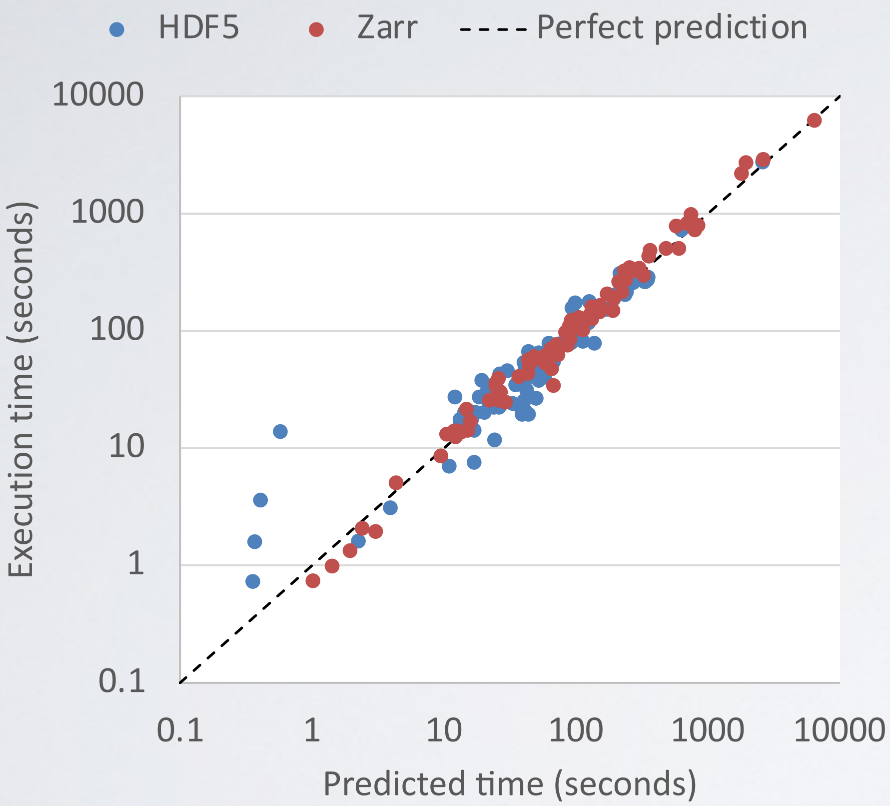
memory copy time

Solution overview



- The system parameters
  - the chunk shape
  - storage format (HDF5 or Zarr)
  - stripe count and size in the file system
  - cache size
- The solution contains three components
  - The cache state component maintains the status of the library cache and the file system cache
  - The variable calculation component computes the variables in the model based on the selected region
  - The model component predicts the cache read time, data server access time and memory copy time

Model accuracy



Real and predicted time

Experiment variables

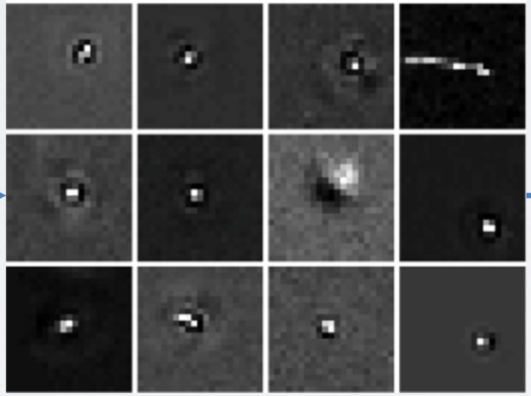
Variable	Value range
Array shape	$36M\times64, 2.4K\times2K\times3K$
Selected region	entire array, 1K rows, 1 column, 1K boxes
Chunk length in Dim 1	70K - 4.5M
Chunk length in Dim 2	1 - 64
Number of processes	1 - 64
Stripe count	1 - 64
File system	Lustre, GPFS

- Takeaways:
  - The RMSE of the model is 0.29
  - The model correctly predicts the fastest library between HDF5 and Zarr 94% of the time.

Small array challenge

- Supernova detection

Segment



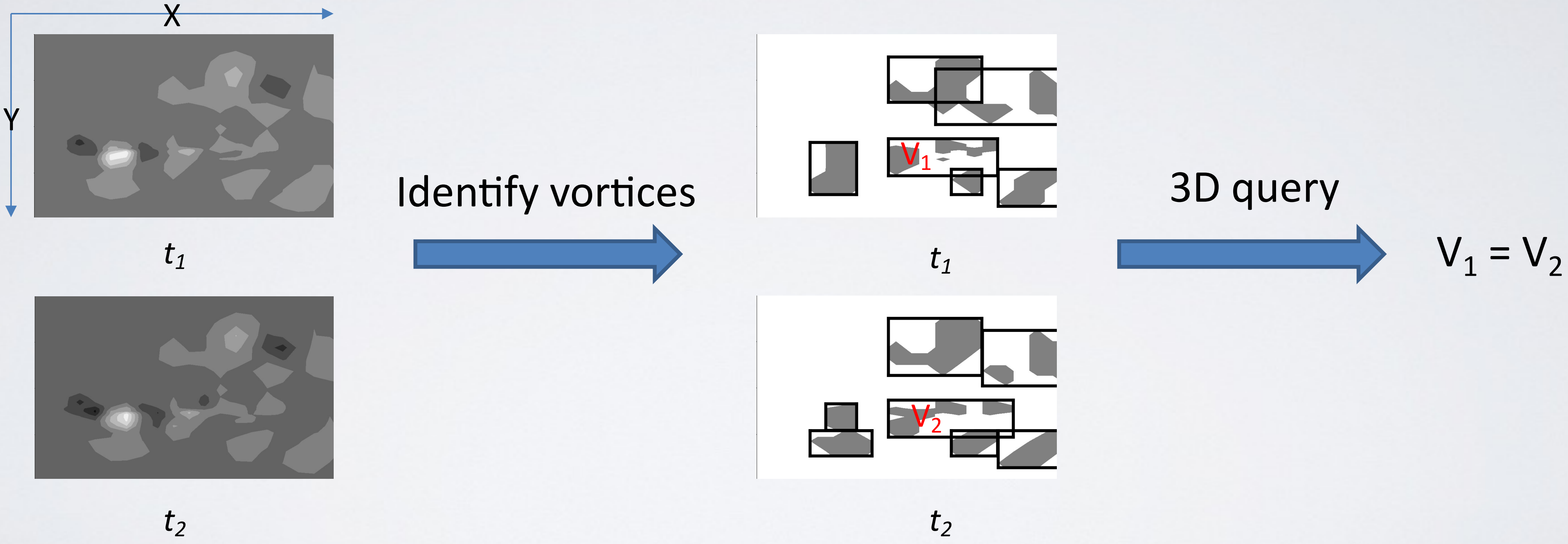
CNN

Real or bogus?

21×21 pixels/image

Small array challenge

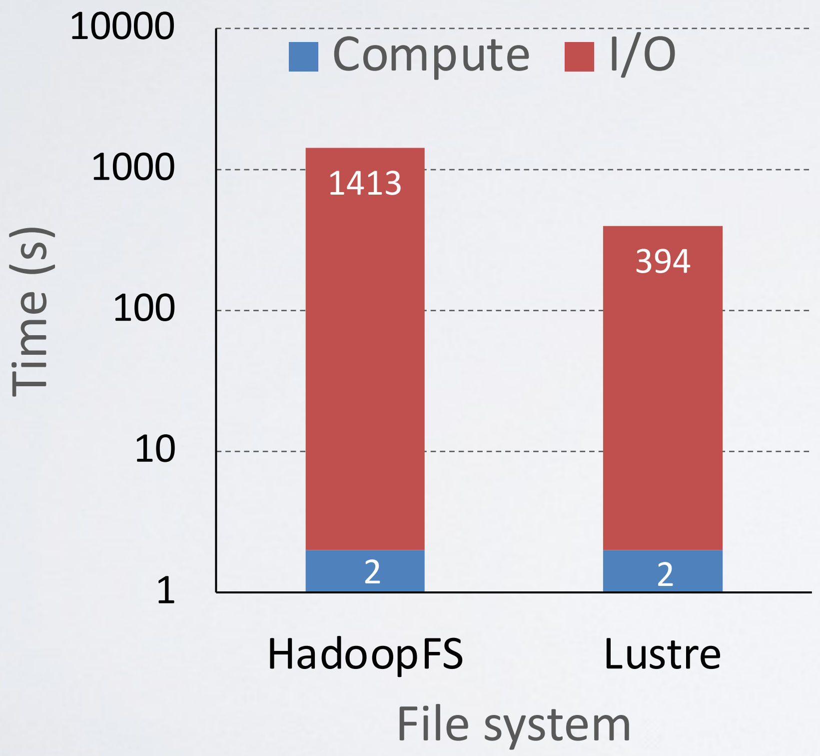
- Vortices prediction
  - 8KB/vortex

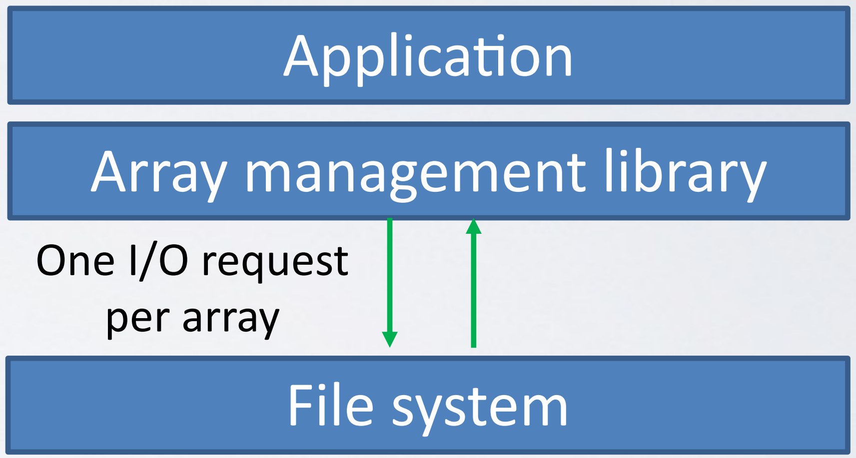


Predict → Optimize

Small array challenge

- One I/O per array
  - I/O takes 200✕ to 700✕ longer than computation

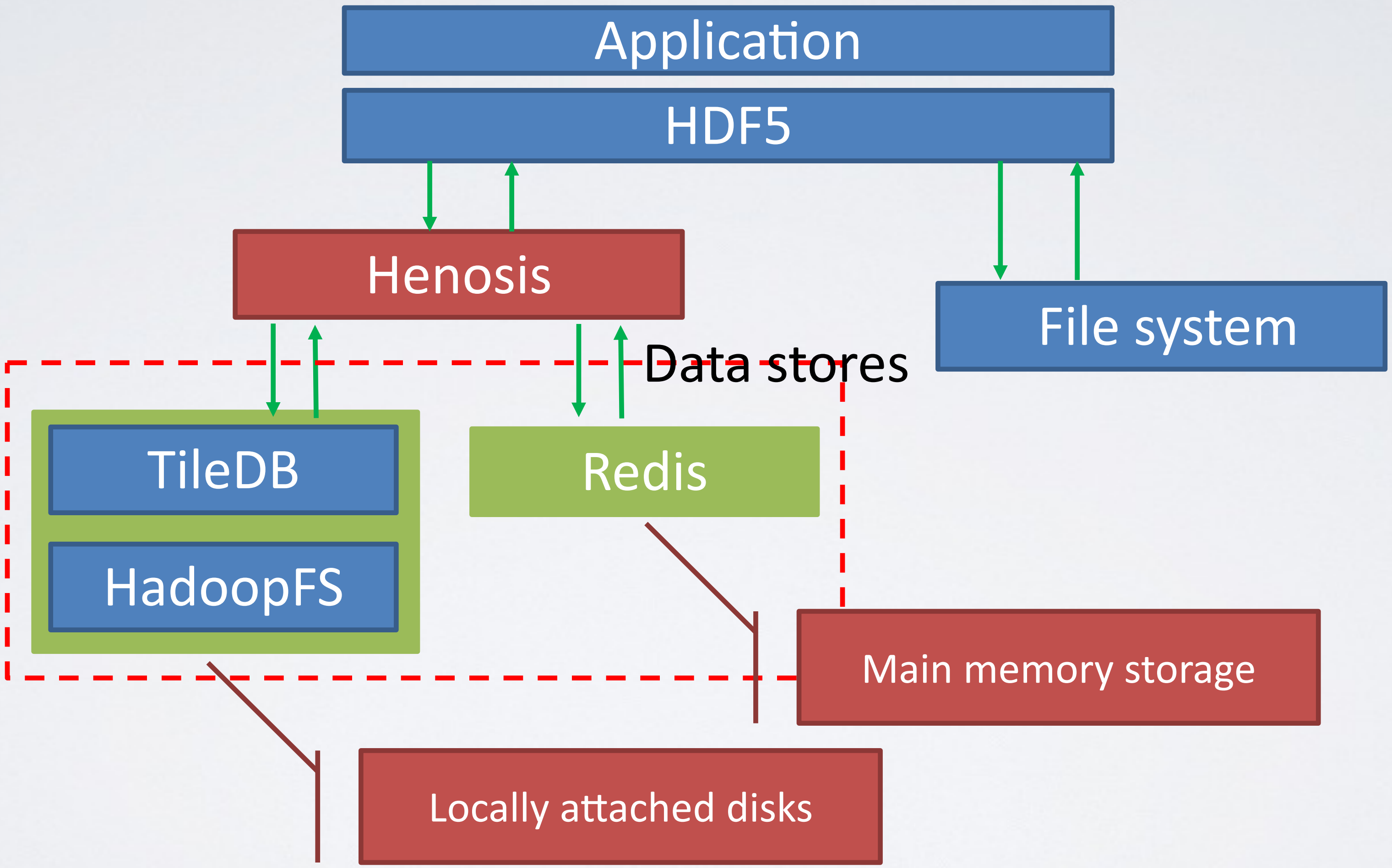




Small array challenge

Too many small I/O requests!

HDF5 on heterogeneous data stores



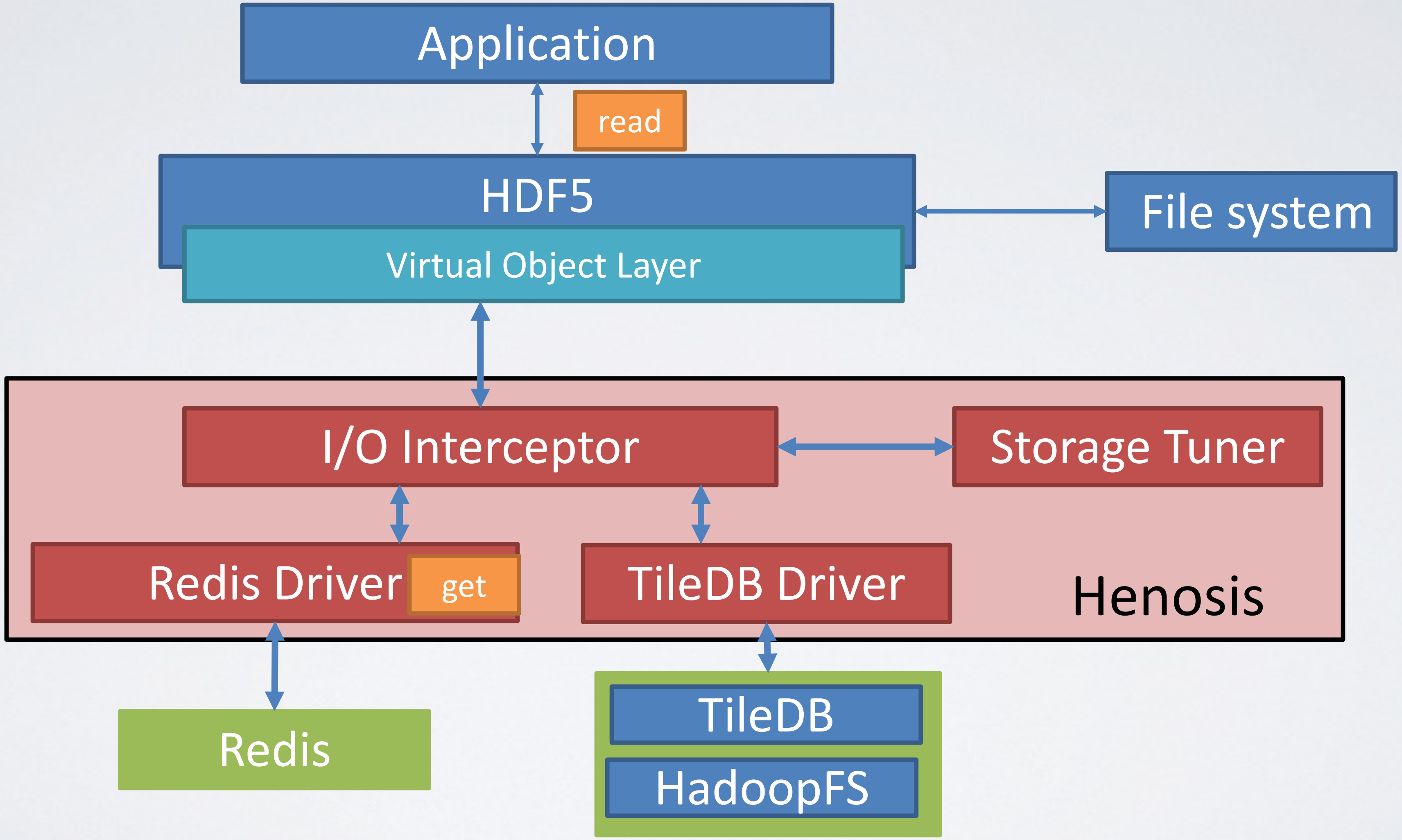
Goals

- Store arrays on heterogeneous data stores
  - Without modifying applications
- Accelerate small I/O requests
  - Placement → improve the performance of one request
  - Consolidation → reduce the number of requests
- Automatically decide the array storage layout
  - Which data store should an array be placed in?
  - How do we store small arrays in chunks?

Goals

- **Store arrays on heterogeneous data stores**
  - **Without modifying applications**
- Accelerate small I/O requests
  - Placement → improve the performance of one request
  - Consolidation → reduce the number of requests
- Automatically decide the array storage layout
  - Which data store should an array be placed in?
  - How do we store small arrays in chunks?

System architecture



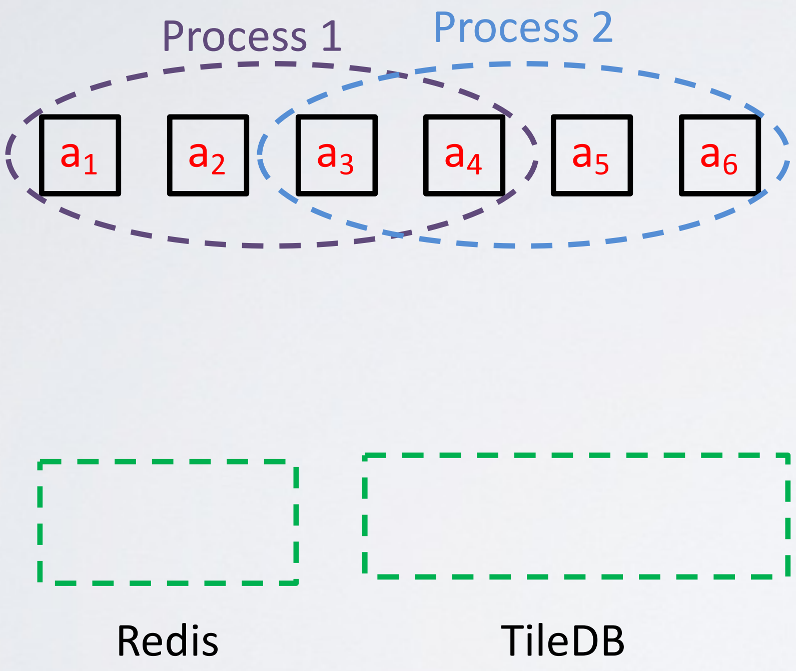
Goals

- Store arrays on heterogeneous data stores
  - Without modifying applications
- **Accelerate small I/O requests**
  - **Placement → improve the performance of one request**
  - **Consolidation → reduce the number of requests**
- Automatically decide the array storage layout
  - Which data store should an array be placed in?
  - How do we store small arrays in chunks?

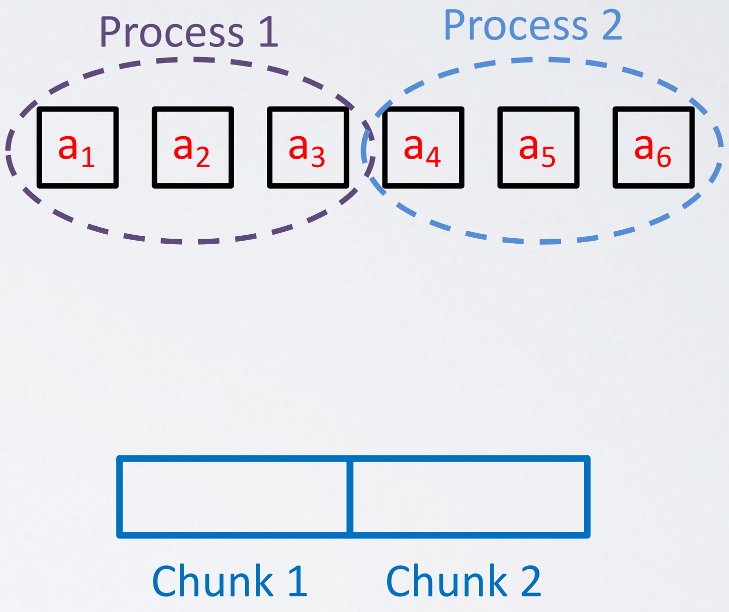
Predict → Optimize

I/O acceleration techniques

- Placement



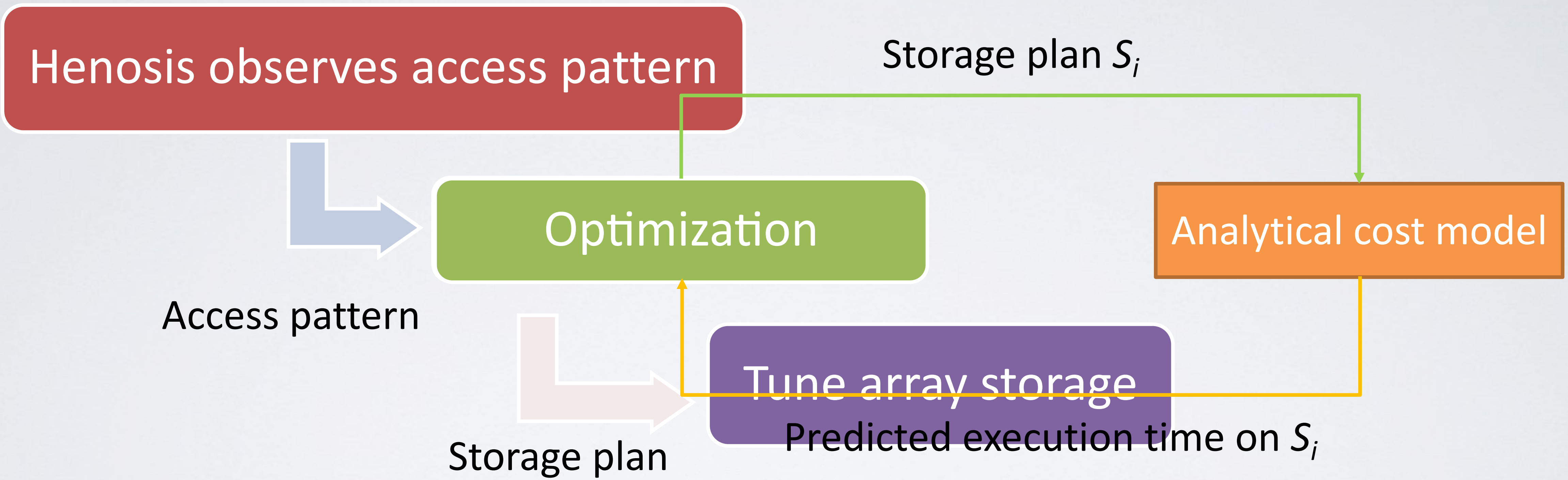
- Consolidation



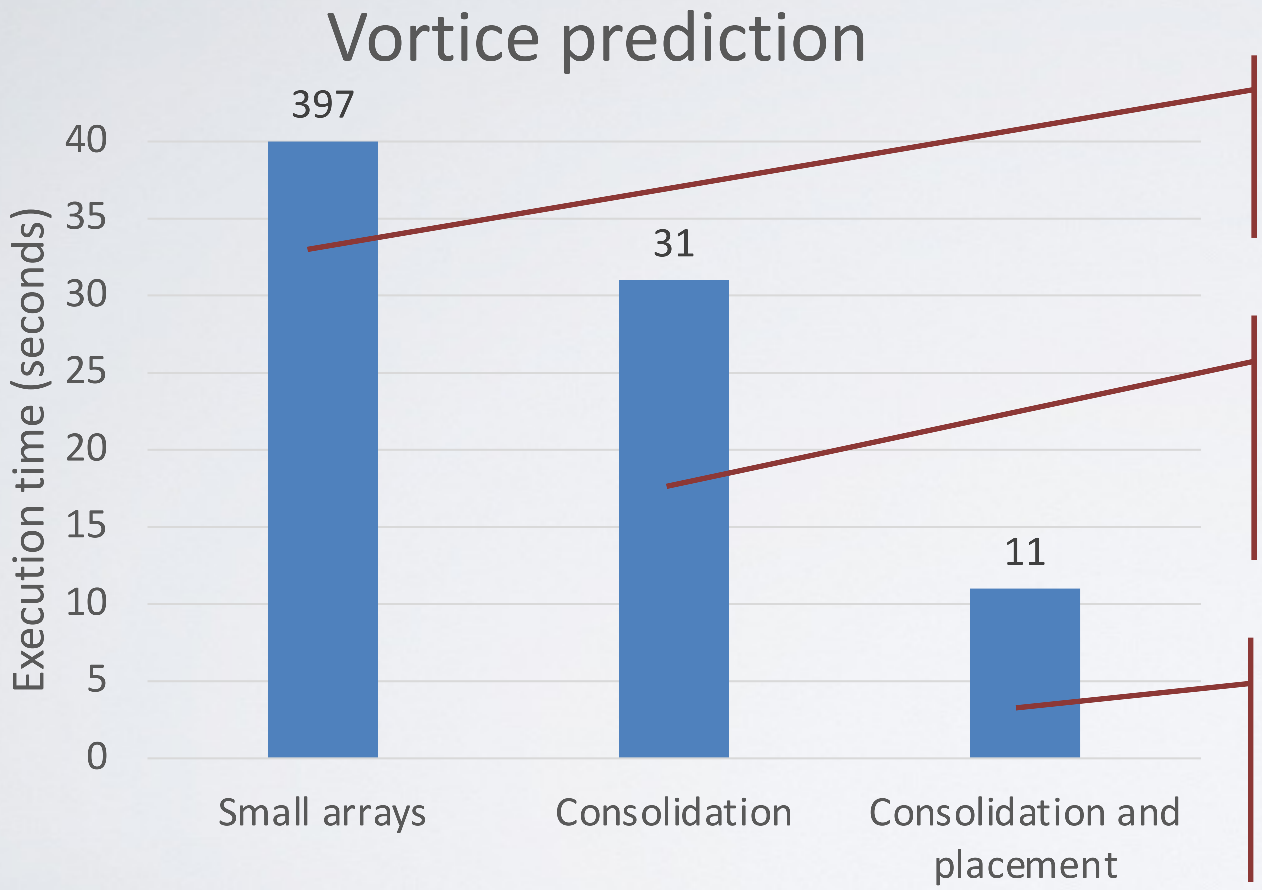
Goals

- Store arrays on heterogeneous data stores
  - Without modifying applications
- Accelerate small I/O requests
  - Placement → improve the performance of one request
  - Consolidation → reduce the number of requests
- **Automatically decide the array storage layout**
  - **Which data store should an array be placed in?**
  - **How do we store small arrays in chunks?**

Optimization workflow



Optimization impact



Directly read many small arrays from HadoopFS

Consolidate small arrays with same access pattern in a chunk. Read in fewer I/O requests.

Place hot arrays and co-accessed arrays in Redis. Read less data from HadoopFS.

Conclusions

- Reading from cache and memory copy to transform layouts can spend 90% of the execution time
  - An end-to-end model to cover the entire I/O path

- Applications spend 99% of the time to read small arrays
  - VOL transparently forward requests to two data stores
  - Placement and consolidation reduce the number of I/O requests to the slow data store
  - An analytical cost model helps to decide the storage layout

Personal webpage: https://web.cse.ohio-state.edu/~kang.1002/