SOPHIA: Online Reconfiguration of Clustered NoSQL Databases for Time-Varying Workloads

Ashraf Mahgoub[1], Paul Wood[2], Alexander Medoff[1], Subrata Mitra[3], *Folker Meyer*[4], Somali Chaterji[1], Saurabh Bagchi[1]

1: Purdue University; 2: Johns Hopkins University; 3: Adobe Research; 4: Argonne National Laboratory

Supported by NIH R01 AI123037-01 (2016-21), WHIN center (2018-22)

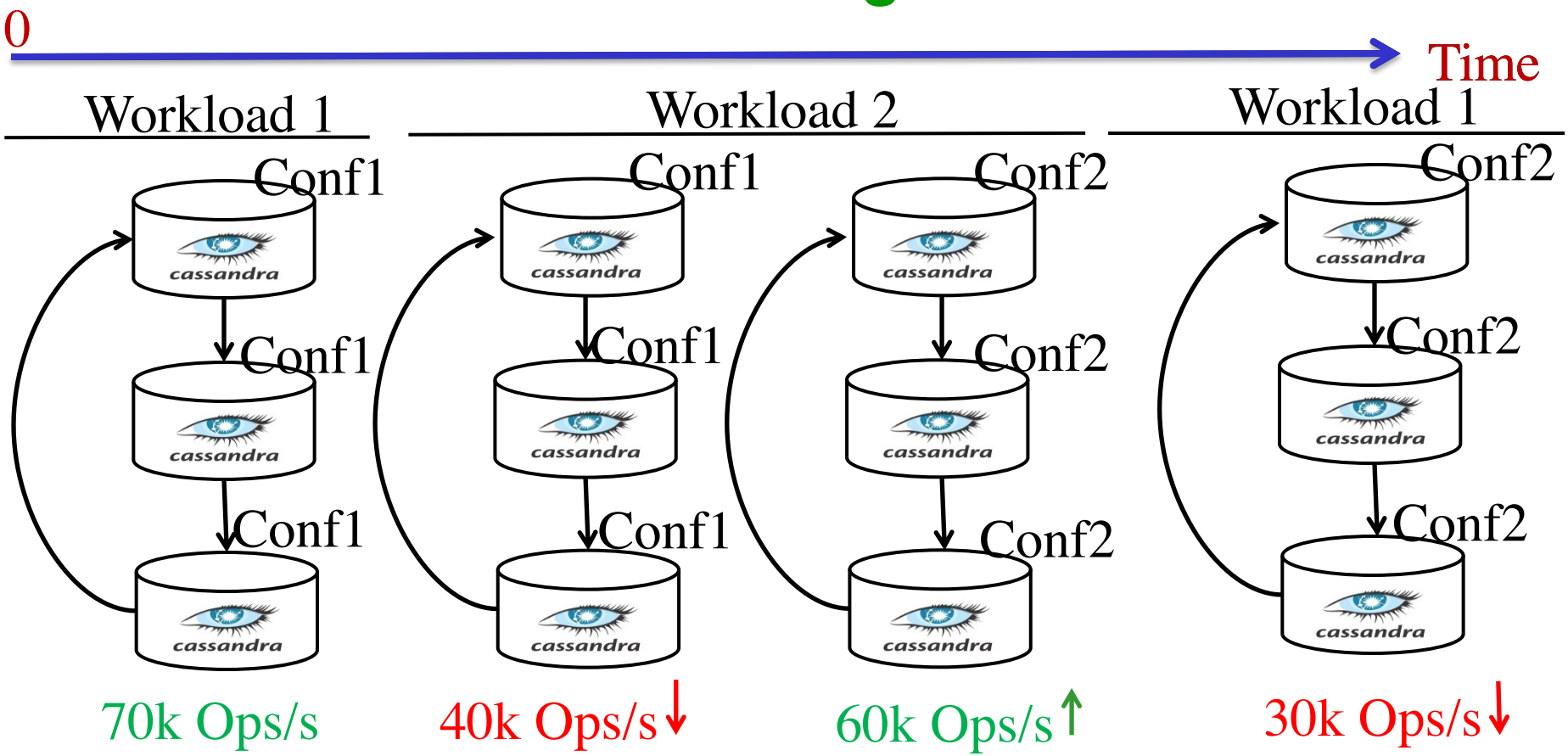
Agenda

- Online Tuning Challenges
- Dynamic Workloads
- Prior work
- Proposed Approach
- Use cases and Evaluation
- Conclusion

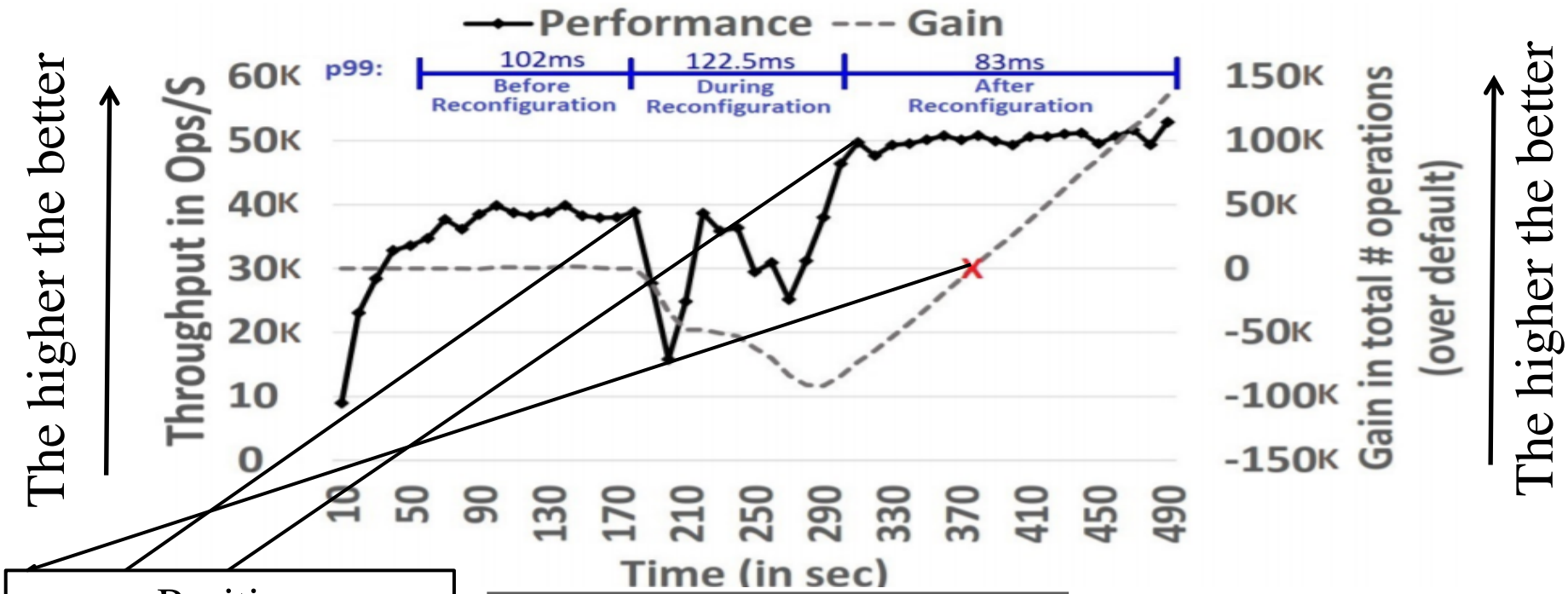
Online NoSQL Tuning Challenges

1. Numerous configuration parameters that control and impact the performance
2. Workload characteristics change over time
3. Accordingly, reconfiguration is needed as the optimal parameters change
4. Reconfiguration has a cost
    1. Often a server restart is needed for the new configuration to take effect
    2. During restart data may become unavailable or throughput may be degraded
    3. Workload changes can be transient and therefore cost of reconfiguration may not be recouped

Online Reconfiguration



Look before you ~~leap~~ change



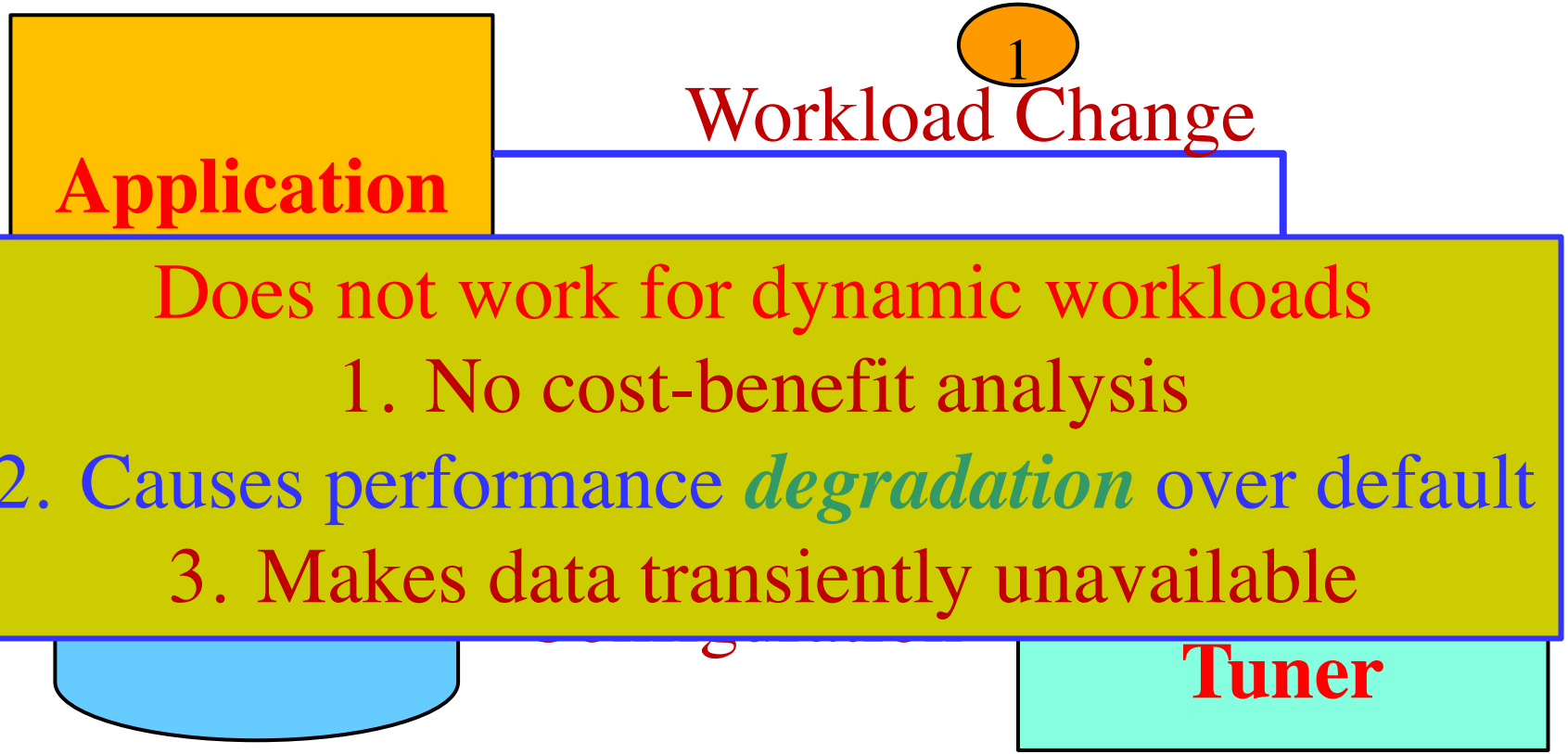
Positive Reconfiguration Gain

Steady State

Cassandra DBMS, MG-RAST production trace
servers = 2,
Replication Factor (RF) = 2
Consistency Level (CL) = 1

PURDUE UNIVERSITY

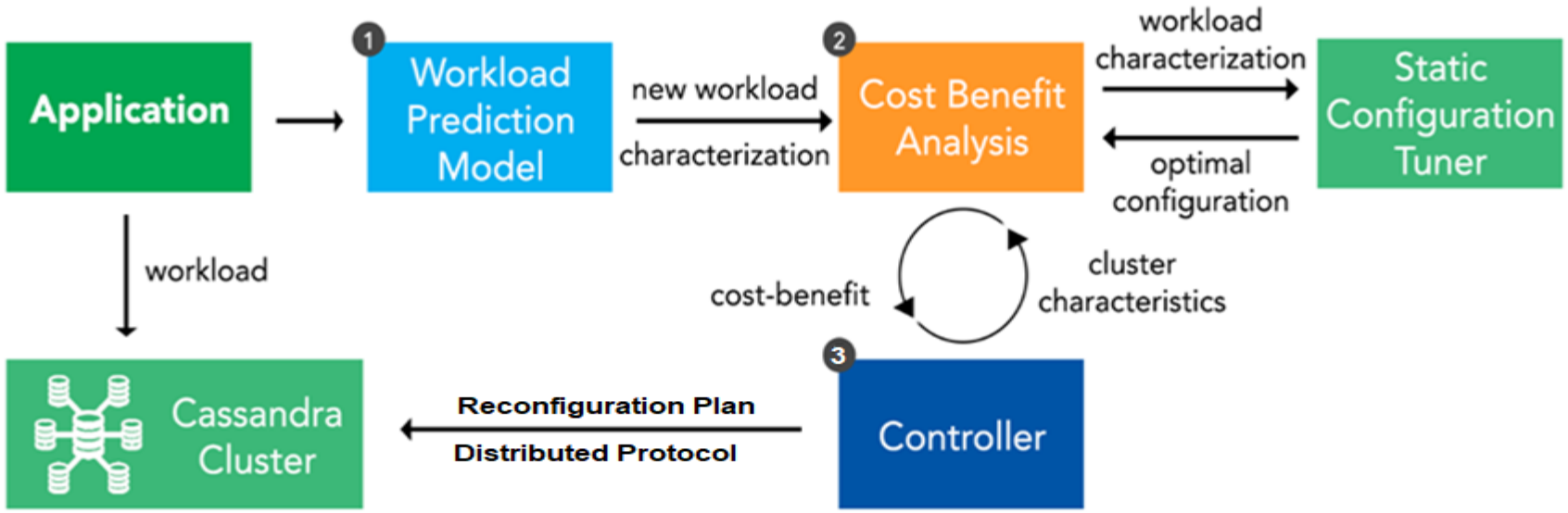
Limitations of Prior Work



Does not work for dynamic workloads

1. No cost-benefit analysis
2. Causes performance *degradation* over default
3. Makes data transiently unavailable

Solution Overview: SOPHIA



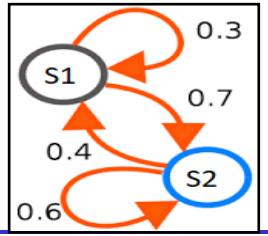
Reconfiguration Plan

Time	Configuration
T_1	P_1=V_{11}, P_2=V_{12} …
T_2	P_1=V_{21}, P_2=V_{22} …
…	…
T_N	P_1=V_{N1}, P_2=V_{N2} …

PURDUE UNIVERSITY

Feature Space Reduction & Workload Prediction

- We use Rafiki (Mahgoub et al., Middleware'17) as a static tuner
  - Identifies the most impactful parameters
  - Quickly finds the optimal configuration for the current phase of the workload
- The set of most impactful parameters identified by the static tuner (Rafiki) require a server restart for their new values to take effect
- For workload prediction, we use n-order Markov-Chain models to represent the different states of the workload and predict the future patterns.



Cost-Benefit Analysis

- We estimate **the cost** of the entire reconfiguration plan as:

$$L = \sum_{k \in [1,M]} H_{\text{sys}}(\boldsymbol{W}(t_k), \boldsymbol{C}_k) \cdot T_r \qquad (1)$$

- H_{sys} is the overall system Ops/S, $W(t_k)$ is the workload at time t_k , C_k is the new configurations in the k^{th} step of the plan, T_r time needed by a single server to restart

Cost-Benefit Analysis (Continued)

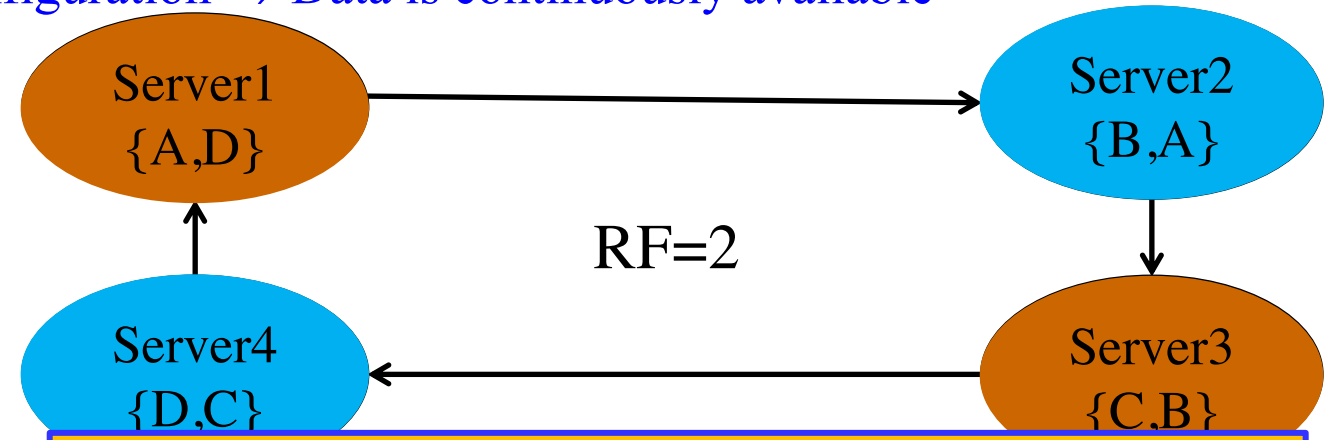
- Benefit *B*: Improvement in the cluster's performance with the new configuration vs. with the old configuration

- We then apply Genetic Algorithms (GA) to search the space of configuration plans space and find the best reconfiguration plan

$$\boldsymbol{C}^{*} = \arg\max_{\boldsymbol{C}^{\Delta}_{\text{sys}}} B(\boldsymbol{C}^{\Delta}_{\text{sys}}, \boldsymbol{W}) - L(\boldsymbol{C}^{\Delta}_{\text{sys}}, \boldsymbol{W}) \qquad (3)$$

Distributed Online Reconfiguration Protocol

- Identify the ***Minimum Availability Subset*** using data placement information, Replication Factor (RF) and Consistency Level (CL)
  - Definition: the minimum subset of servers that cover all data records exactly CL times.
- SOPHIA ensures that at least one ***Minimum Availability Subset*** is up during reconfiguration ⇒ Data is continuously available



For CL=1:
Minimum Availability Subset ={1,3} or {2,4}

Distributed Online Reconfiguration Protocol (Cont.)

Each server performs this distributed protocol to apply new configurations

1. ***Drain:*** Flush all uncommitted data records to disk. This is needed to avoid executing long and expensive data *repair* processes.
2. ***Shutdown:*** The Cassandra process is killed on the node.
3. ***Configuration file:*** Replace the configuration file with new values for all parameters that need changing.
4. ***Restart:*** Restart the Cassandra process on the same node.
5. ***Sync:*** Wait for Cassandra's instance to completely rejoin the cluster by letting a coordinator know where to locate the node and then synchronizing missed updates during the node's downtime.

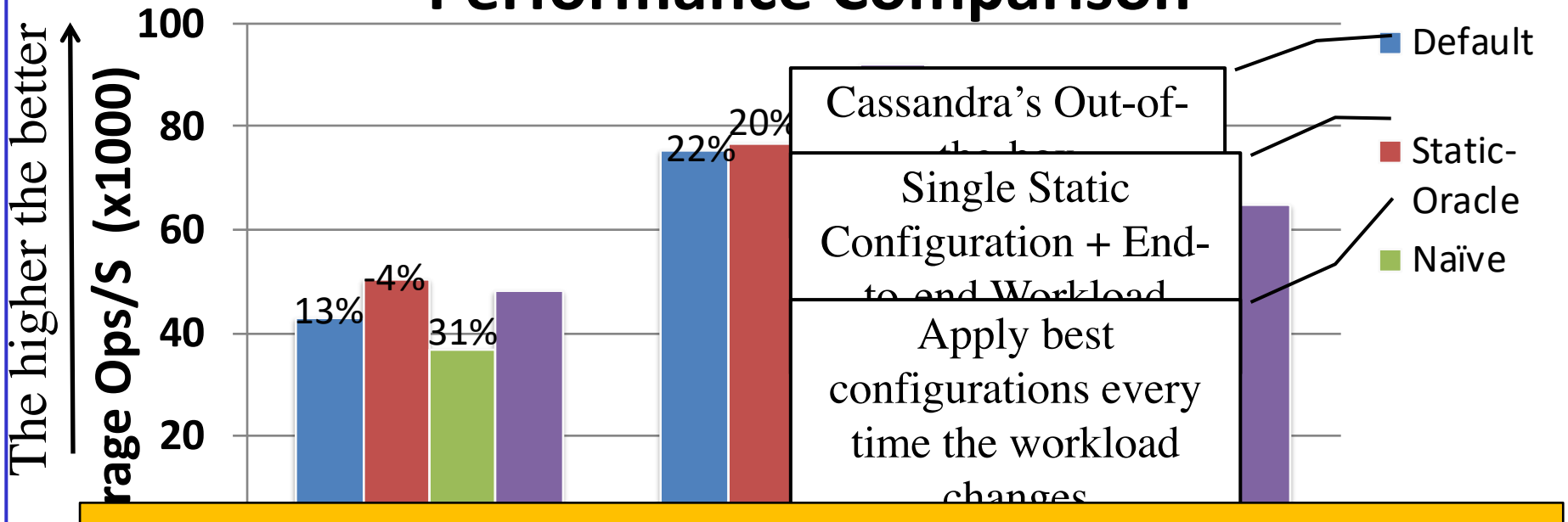
Use Cases and Evaluation

1. MG-RAST:
   - Real workload traces from the largest metagenomics analysis portal
   - Its workload does not have any discernible daily or weekly pattern, as the requests come from all across the globe
   - Workload can change drastically over a few minutes and it is accurately predictable for 5min only

2. Bus-Tracking:
   - Real workload traces from a bus-tracking mobile application
   - Traces show a daily pattern of workload switches.
   - Workload is accurately predictable for longer look-ahead periods (e.g. 2 hours)

3. HPC:
   - Simulated workload traces from batch data analytics jobs submitted to a shared HPC queue.
   - Using profiling techniques, job execution times can be predicted with high accuracy and for longer look-ahead periods.

Performance Comparison



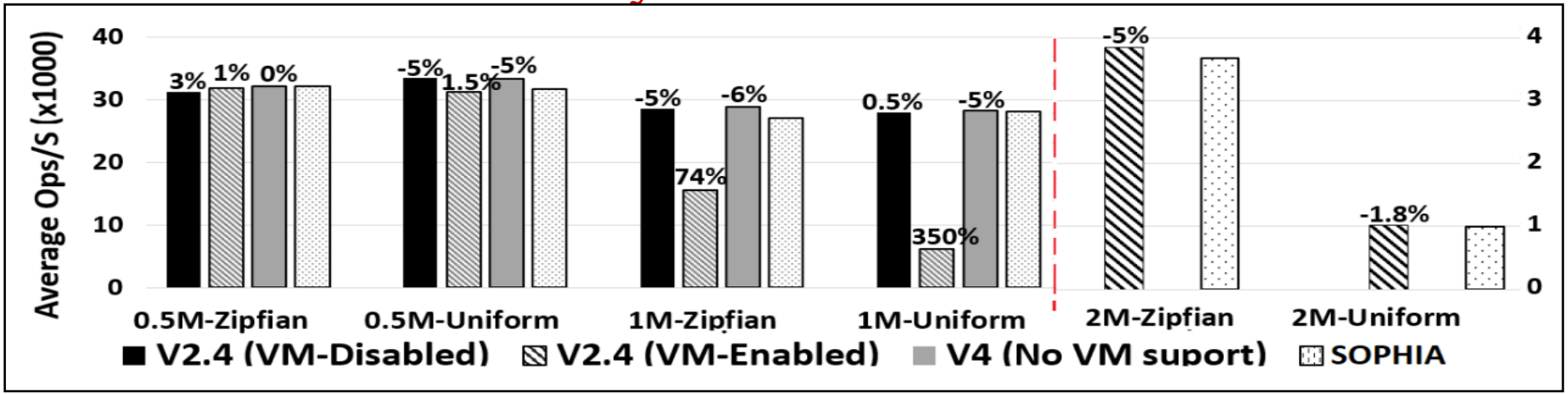
Compared to Static-Oracle, which is a direct application of all

Compared to Naïve: SOPHIA achieves 25%-30% better performance across the three use-cases

and HPC.

Evaluation: Redis

- Redis is an in-memory data store



- By automatically selecting the right parameters for changing workloads, SOPHIA achieves the best of both worlds with jobs that vary in
  - Sizes, access patterns, and request distributions

Insights

- Online tuning of NoSQL databases for dynamic workloads is challenging
- All prior works suffer for dynamic workloads and a straightforward application actually ***degrades*** performance
- SOPHIA addresses all these shortcomings using an optimization technique that combines workload prediction, cost-benefit analysis, and Genetic Algorithms
- Evaluated with real workload traces and two popular NoSQL datastore (Cassandra and Redis)
- SOPHIA achieves globally optimized performance and respects user's data consistency and availability requirements

Thank you