

# Automated Physical Design in Database Caches



Tanu Malik, Xiaodan Wang, and Randal Burns  
Johns Hopkins University  
{tmalik,xwang,randal}@cs.jhu.edu



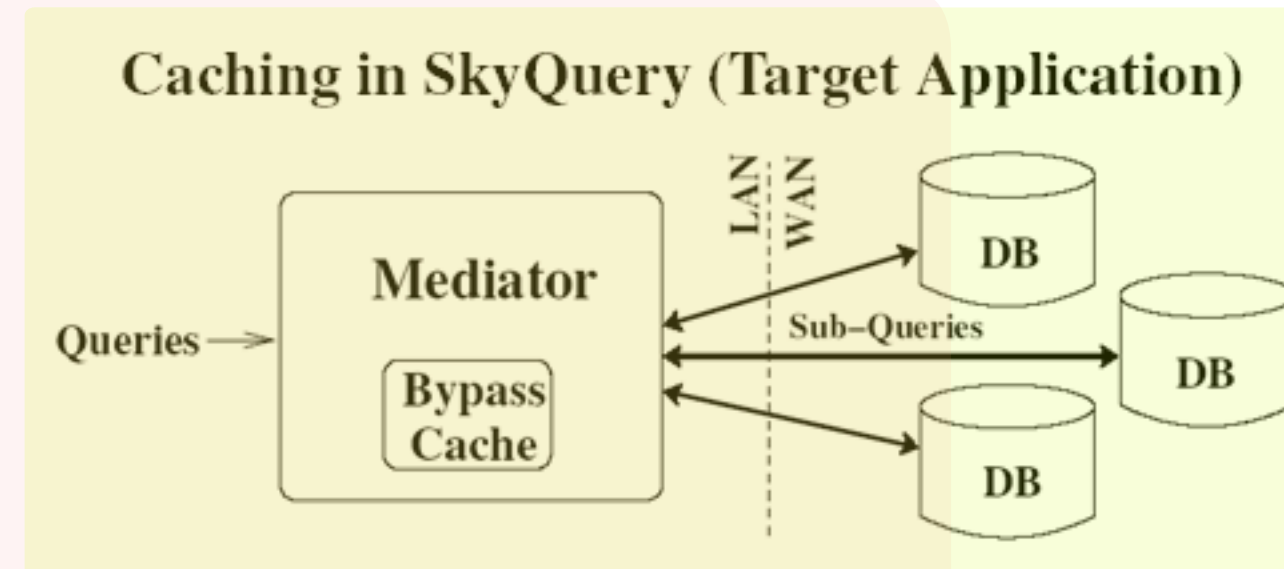
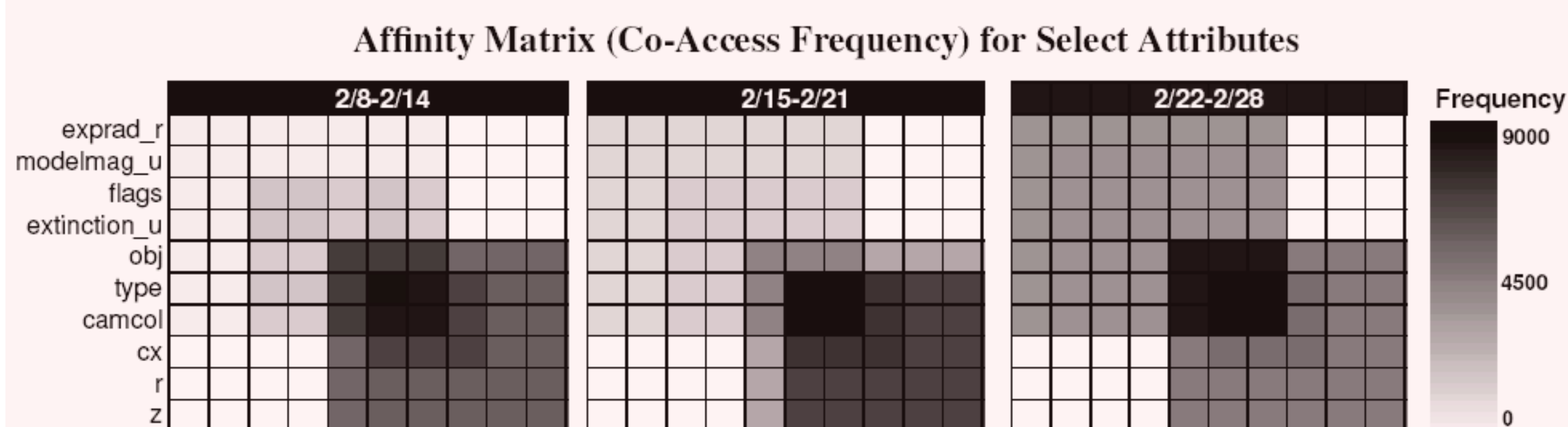
Debabrata Dash and Anastasia Ailamaki  
Carnegie Mellon University  
{ddash,natassa}@cs.cmu.edu

## Introduction

Applications, such as proxy caches and content distribution networks, serve a continuous stream of queries from tens of thousands of users in which a representative workload may not be available. As such, current research emphasizes the need for design tools that are always on and can continuously adapt the physical design to changes in the workload.

We study an online approach to vertical partitioning that detects incremental changes in the workload and adapts the physical design automatically. Our target application is physical design of the Bypass cache, a proxy database cache for the SkyQuery federation of Astronomy databases.

## Why Online Vertical Partitioning?

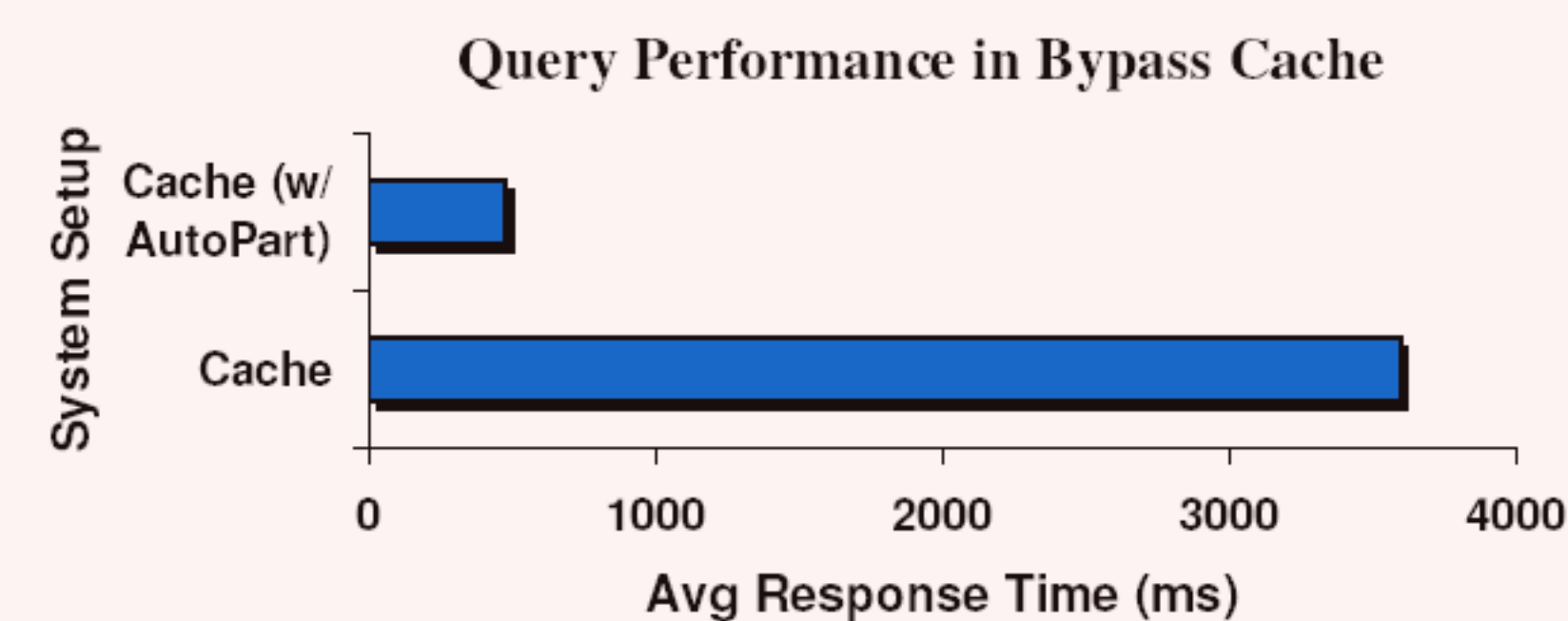


### Workload Evolution in SkyQuery

The schema in SkyQuery is fixed after the initial public release of the data and remains static even as the workload changes over time. The figure above illustrates change on a weekly basis.

### Concrete Performance Benefits

Without indices, partitioning improves query performance by up to six folds.

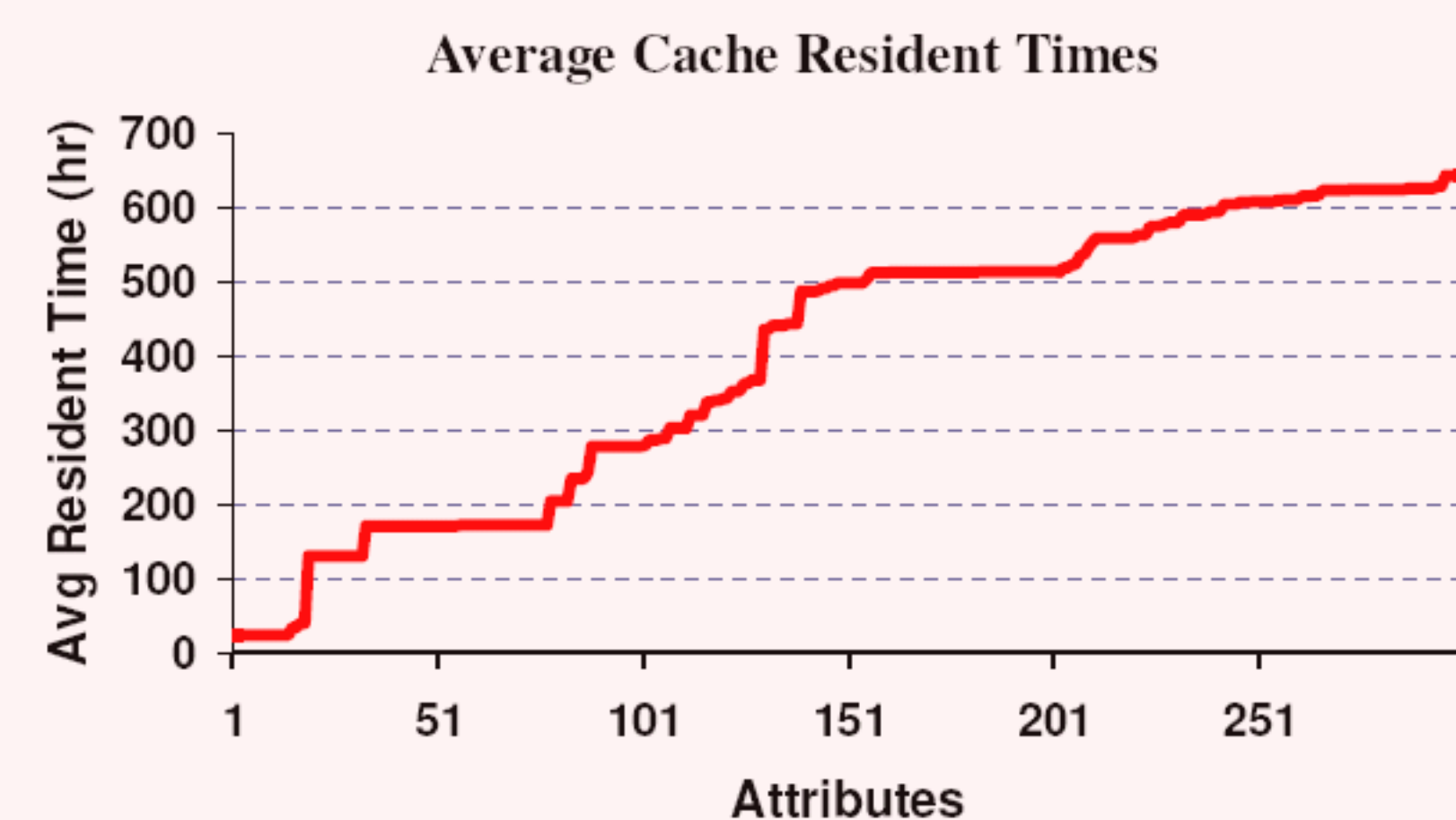


### No Redundant Data

Unlike partitioning, construction of auxiliary data structures such as indices and materialized views introduces redundant data that compete for cache space.

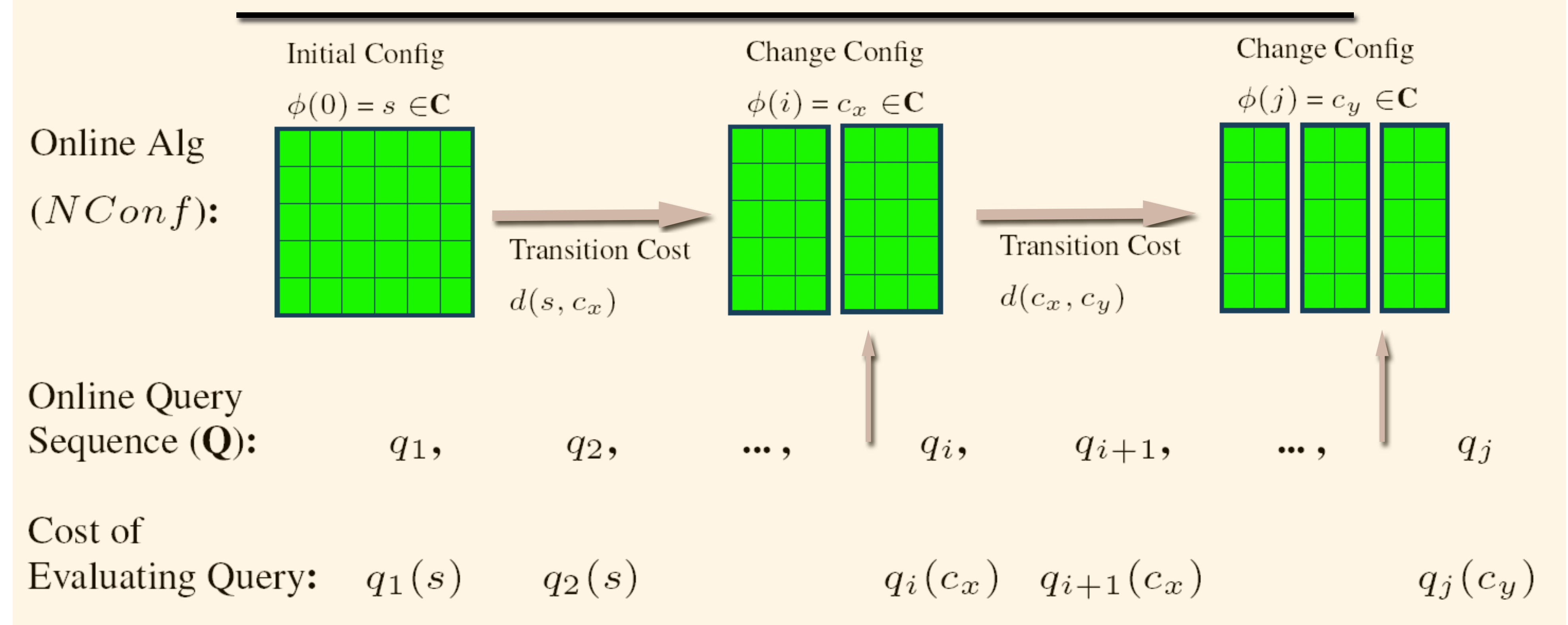
### Captures Transition Cost

We account for the overhead of frequent transitions (cost of partitioning columns). This ensures the partitioning of long-lived columns in Bypass caches such that costs can be amortized over more queries.



## Problem Abstraction

$C = \{c_1, \dots, c_n\}$ : set of partitions (configurations)  
 $\phi$ : function  $[1, m] \rightarrow C$



The goal is to find a  $\phi$  that minimizes:

$$\text{cost}(\phi, \mathbf{Q}) = \sum_{i=1}^n q_i(\phi(i)) + d(\phi(i-1), \phi(i))$$

We approximate  $\phi$  by transitioning at  $q_i$  if remaining in the current configuration incurs a substantial amount of “extra cost” for queries  $q_1, \dots, q_i$

## Preliminary Results

We evaluate our solution on a 500MB configuration of the TPC-D benchmark by partitioning the *Orders* table. There are two workloads.  $Wkld_{Const}$ , in which the access pattern remains static, and  $Wkld_{Sky}$ , which approximates SkyQuery in that the query sequence exhibits three significant changes in its access pattern.

