CLUSTER OF EXCELLENCE

SAARLAND UNIVERSITY

Information Systems Group

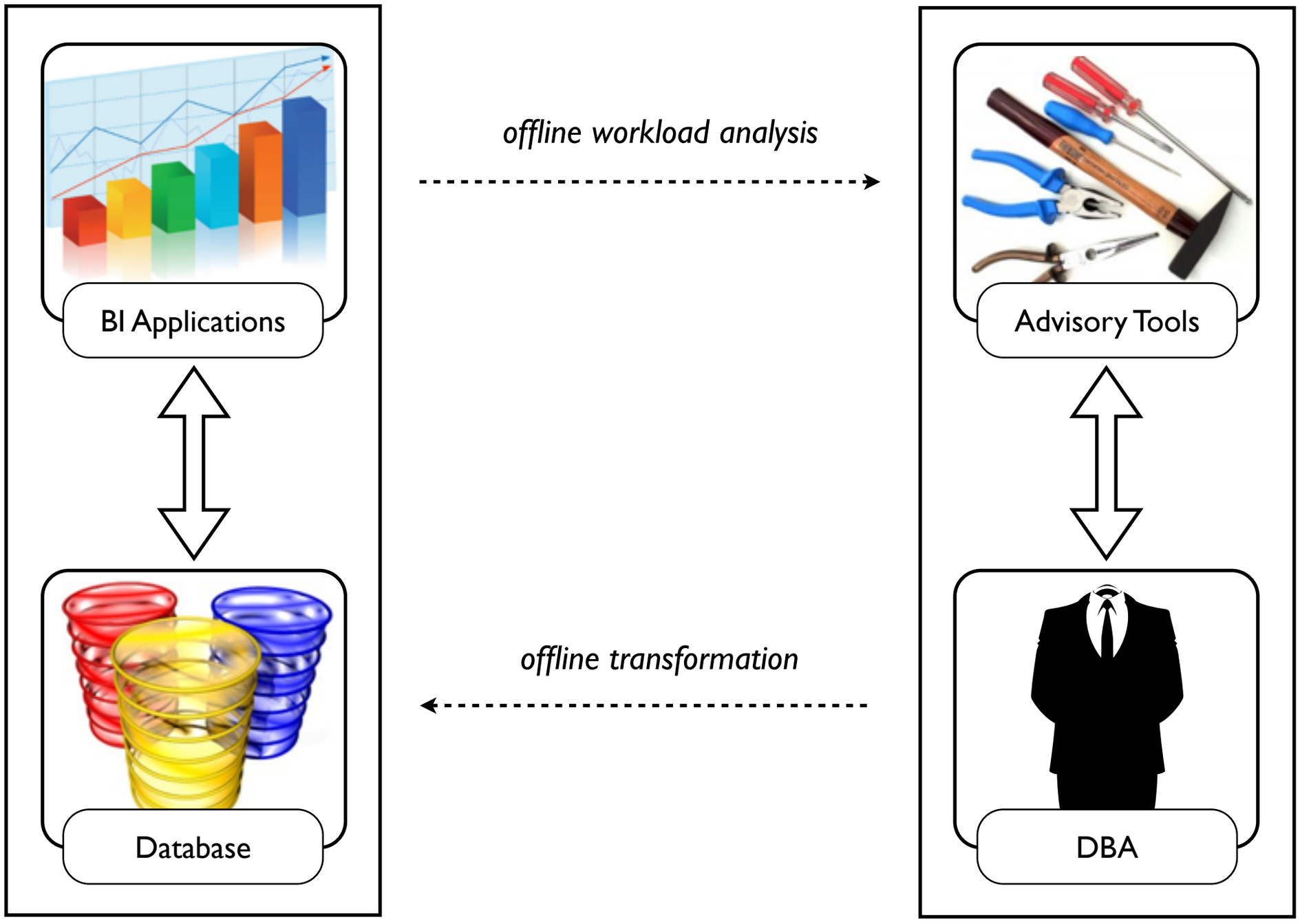
Relax and Let the Database do the Partitioning Online

Alekh Jindal, Jens Dittrich

- presented by Stefan Schuh

VLDB International Workshop on Real-Time Business Intelligence September 2, 2011

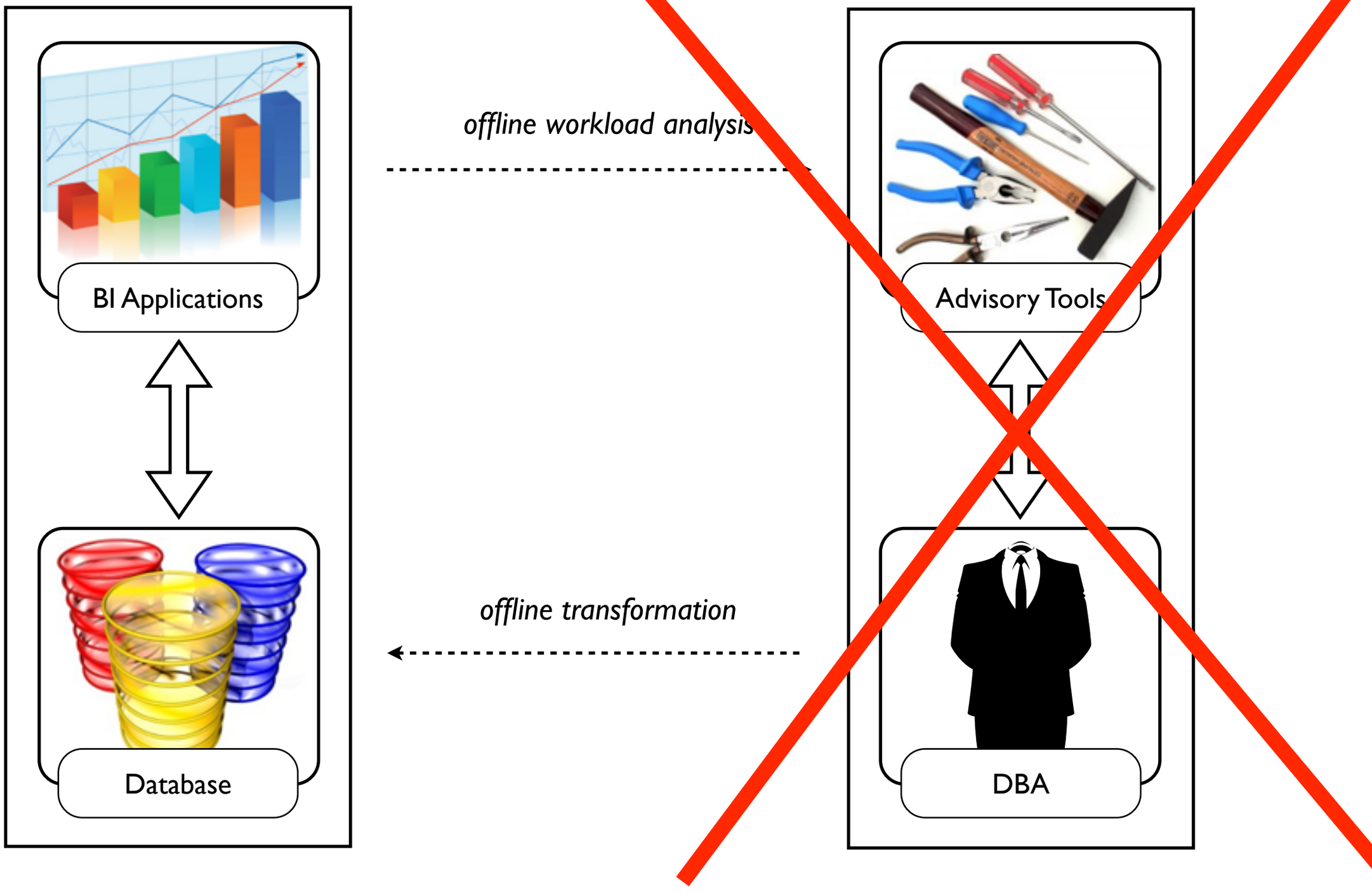
Motivation: Offline Physical Database Design



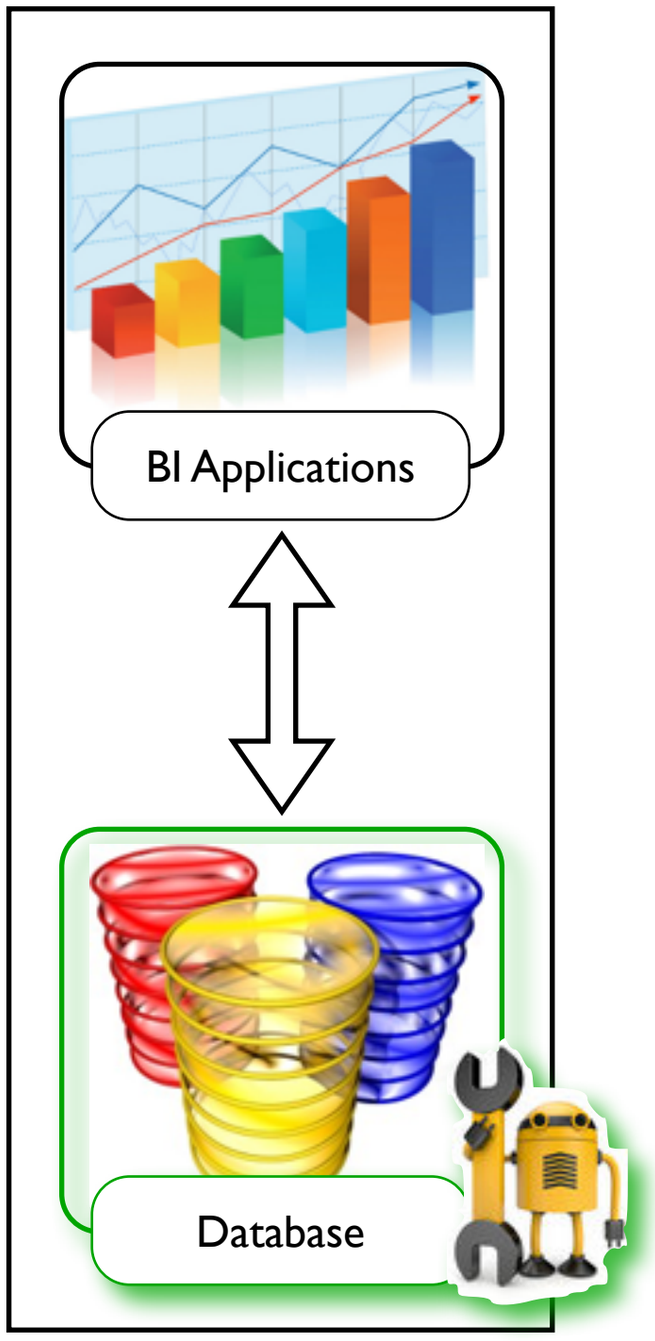
Offline Design Cheats!

- Workloads infrequently change over time
- DBAs always available
- Physical design once-in-a-while process
- DBAs make perfect decisions

Motivation: Offline Physical Database Design



Motivation: Online Physical Database Design



Sub-Problem	Proposed Solution
Indexing	Online Indexing Database Cracking Adaptive Indexing
Materialized Views	Dynamic Materialized Views
Partitioning	WE!

Challenges in Online Partitioning

- Collecting online workload
- Analyzing workload online
- Querying with online workload analysis
- Creating partitions online

Challenges in Online Partitioning

- Collecting online workload
- Analyzing workload online
- Querying with online workload analysis

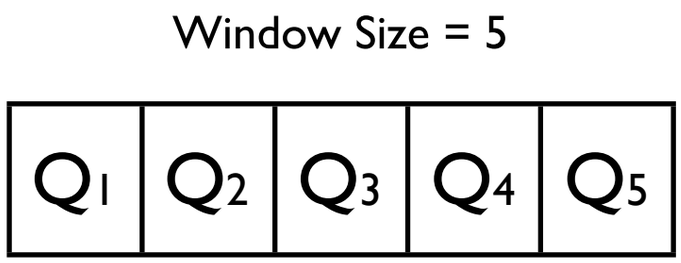
- Creating partitions online

What is the Workload?

- offline approach: take the last query log as workload (static)
- online approach: collect incoming queries in a window and slide it when more queries come (dynamic)

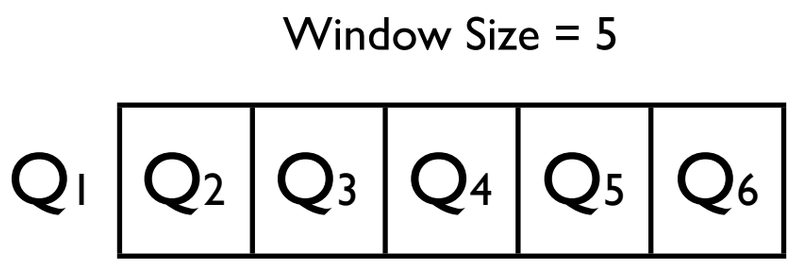
What is the Workload?

- offline approach: take the last query log as workload (static)
- online approach: collect incoming queries in a window and slide it when more queries come (dynamic)



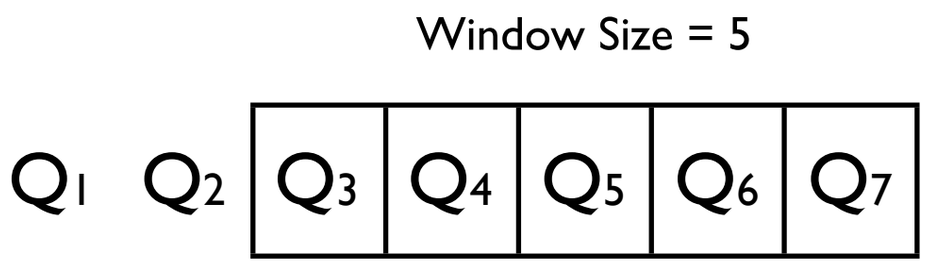
What is the Workload?

- offline approach: take the last query log as workload (static)
- online approach: collect incoming queries in a window and slide it when more queries come (dynamic)



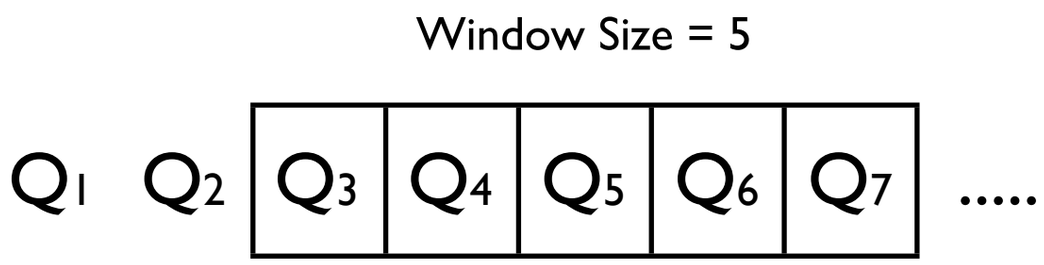
What is the Workload?

- offline approach: take the last query log as workload (static)
- online approach: collect incoming queries in a window and slide it when more queries come (dynamic)



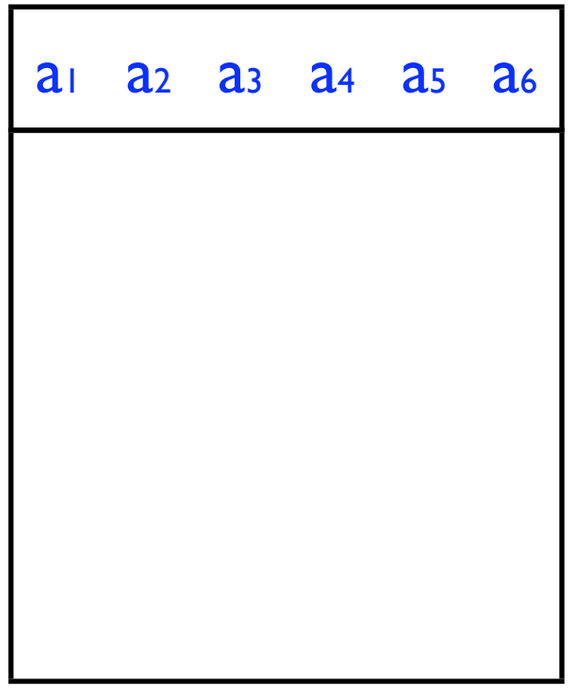
What is the Workload?

- offline approach: take the last query log as workload (static)
- online approach: collect incoming queries in a window and slide it when more queries come (dynamic)



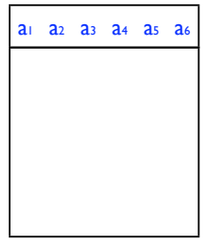
How to Express the Partitioning Problem?

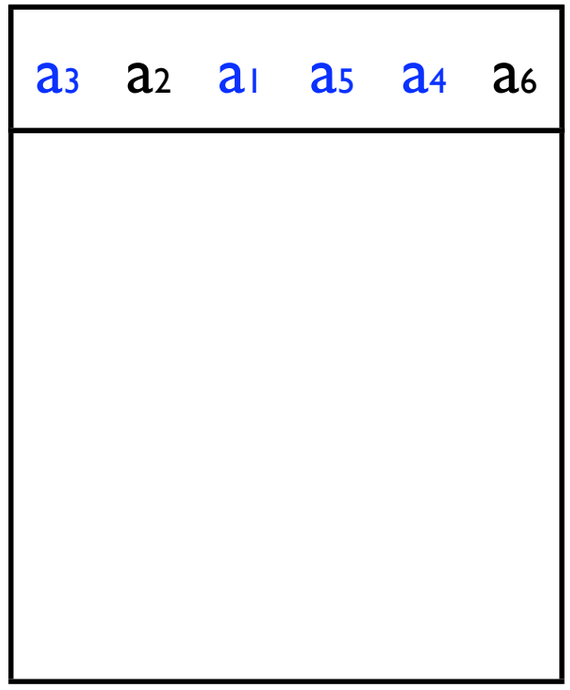
- Partitioning unit P_u e.g. a1, a2, a3, a4, a5, a6



How to Express the Partitioning Problem?

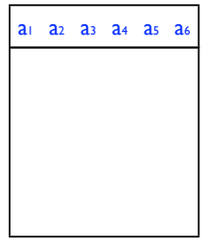
- Partitioning unit P_u e.g. $a_1, a_2, a_3, a_4, a_5, a_6$
- P_u ordering $\preceq$ e.g. $a_3 \preceq a_2 \preceq a_1 \preceq a_5 \preceq a_4 \preceq a_6$

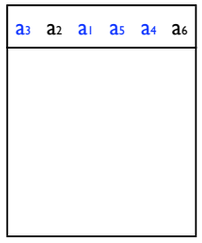


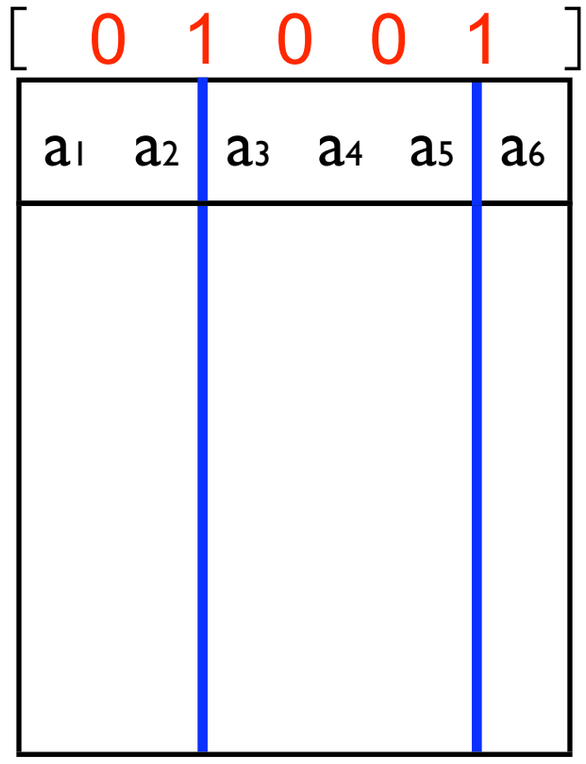


How to Express the Partitioning Problem?

- Partitioning unit P_u e.g. $a_1, a_2, a_3, a_4, a_5, a_6$
- P_u ordering $\preceq$ e.g. $a_3 \preceq a_2 \preceq a_1 \preceq a_5 \preceq a_4 \preceq a_6$
- Split line, Split vector S e.g. [01001]

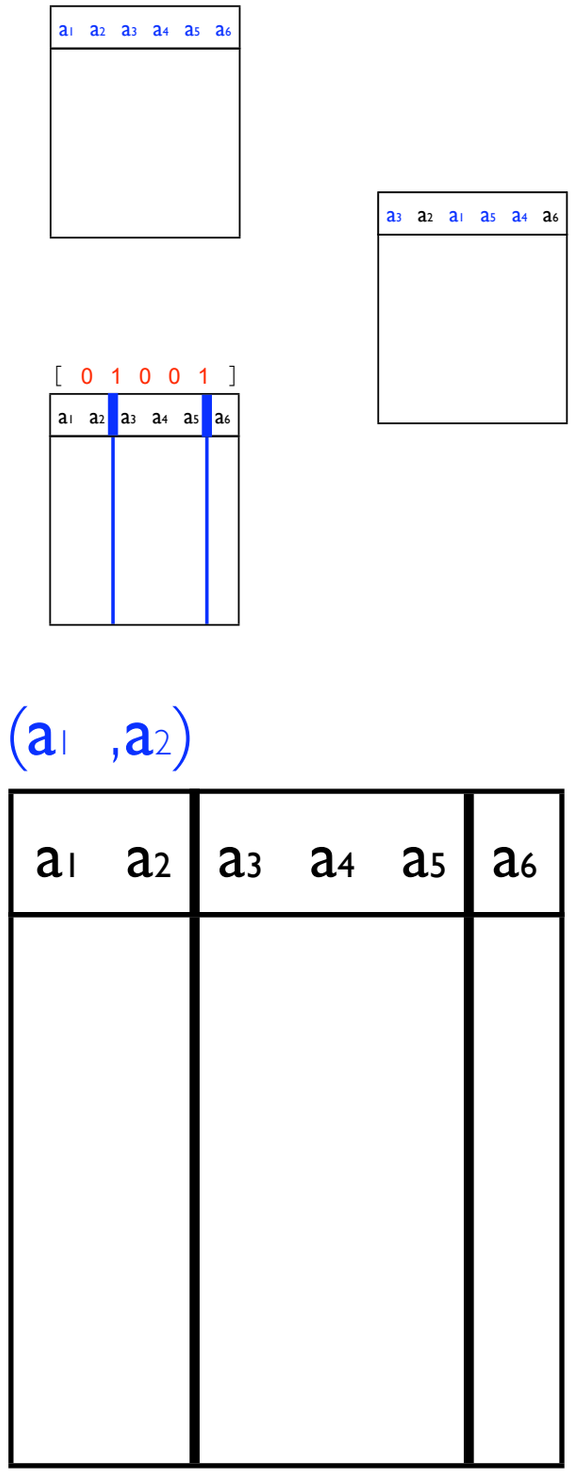






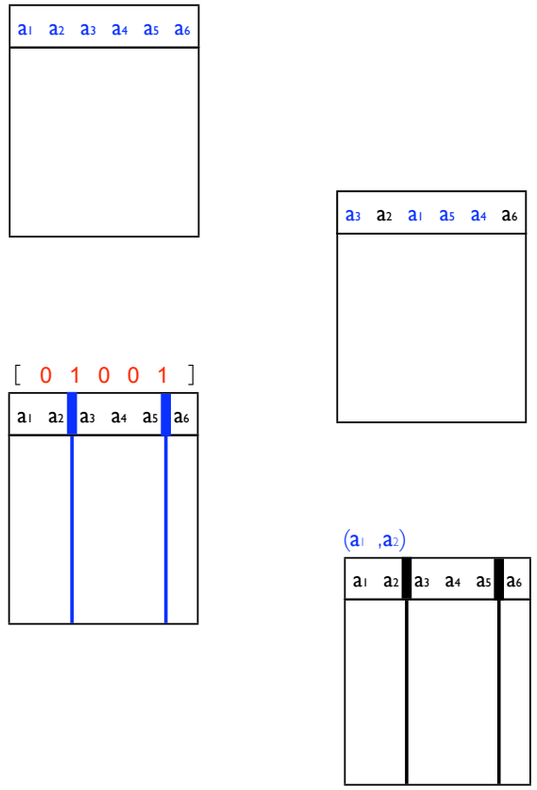
How to Express the Partitioning Problem?

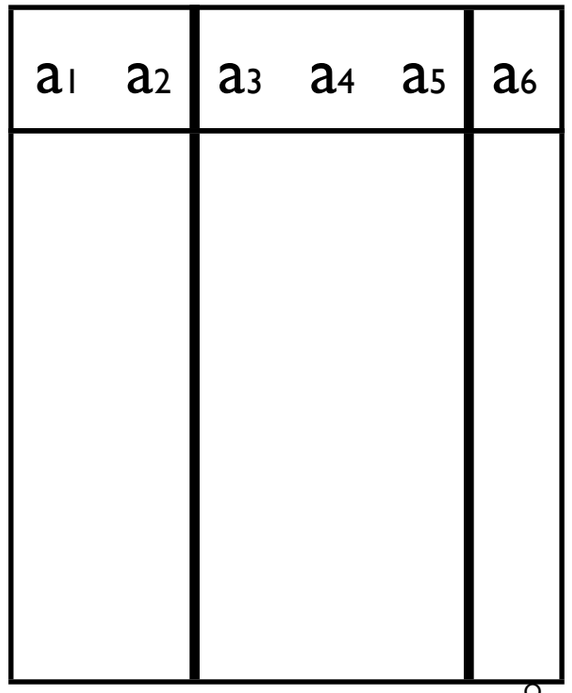
- Partitioning unit P_u e.g. $a_1, a_2, a_3, a_4, a_5, a_6$
- P_u ordering $\preceq$ e.g. $a_3 \preceq a_2 \preceq a_1 \preceq a_5 \preceq a_4 \preceq a_6$
- Split line, Split vector S e.g. [01001]
- Partition $p_{m,r}(S, \preceq)$ e.g. (a_1, a_2)



How to Express the Partitioning Problem?

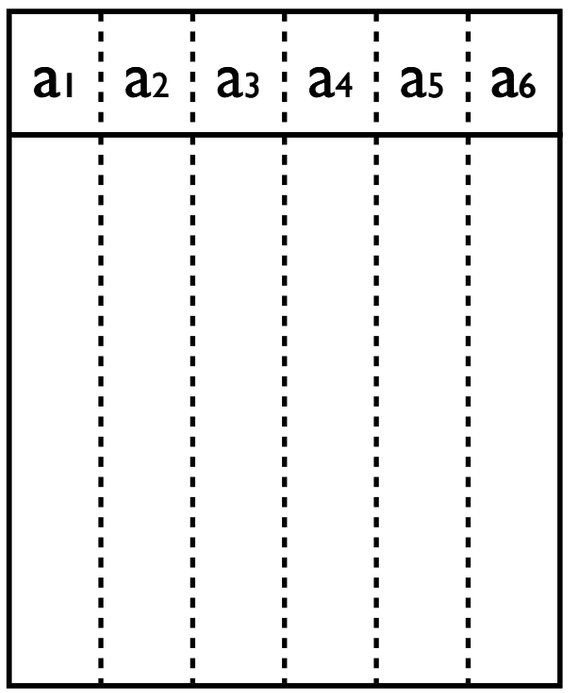
- Partitioning unit P_u e.g. a1, a2, a3, a4, a5, a6
- P_u ordering $\preceq$ e.g. a3≤a2≤a1≤a5≤a4≤a6
- Split line, Split vector S e.g. [01001]
- Partition $p_{m,r}(S,\preceq)$ e.g. (a1, a2)
- Partitioning scheme $P(S,\preceq)$ e.g. (a1, a2), (a3, a4, a5), (a6) {(a1 , a2) (a3 , a4, a5)(a6) }

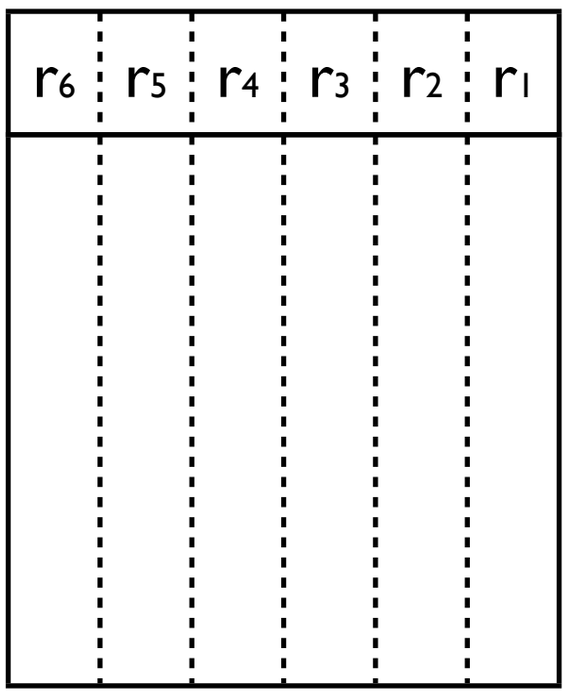




What about Horizontal Partitioning?

- Just rotate the table by 90 degrees
- P_u abstraction allows us to solve both problems
- P_u can be attributes, row-ranges, or any other table slice





Partitioning Problem: What to Analyze?

- Partitioning unit P_u e.g. $a_1, a_2, a_3, a_4, a_5, a_6$
- P_u ordering $\preceq$ e.g. $a_3 \preceq a_2 \preceq a_1 \preceq a_5 \preceq a_4 \preceq a_6$
- Split line, Split vector S e.g. [01001]
- Partition $p_{m,r}(S, \preceq)$ e.g. (a_1, a_2)
- Partitioning scheme $P(S, \preceq)$ e.g. $(a_1, a_2), (a_3, a_4, a_5), (a_6)$
- Workload W_{t_k}
- Problem statement
  Find $\preceq$, S' such that: $$S' = \underset{S}{\operatorname{argmin}}\, C_{\text{est.}}\left(W_{t_k}, P(S, \preceq)\right)$$

How to Analyze the Workload?

Step 1: Finding Partitioning Unit Ordering

- offline approach: create affinity matrix and cluster it once, as proposed by Navathe et. al.

- online approach: leverage the affinity idea, but dynamically update and cluster the affinity matrix

Offline Partitioning Unit Ordering

- Create affinity matrix having attributes co-occurrences

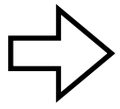
	PartKey	SuppKey	Quantity
PartKey	8	5	6
SuppKey	5	8	4
Quantity	6	4	9

- Cluster affinity matrix to maximize the affinity measure

$$M(\preceq) = \sum_{i=1}^{x} \sum_{j=1}^{x} A(a_i, a_j)[A(a_i, a_{j-1}) + A(a_i, a_{j+1})]$$

$M(\preceq) = 404$

	PartKey	SuppKey	Quantity
PartKey	8	5	6
SuppKey	5	8	4
Quantity	6	4	9



$M(\preceq) = 440$

	PartKey	Quantity	SuppKey
PartKey	8	6	5
Quantity	6	9	4
SuppKey	5	4	8

Online Partitioning Unit Ordering

- Update *only* the referenced P_u in affinity matrix

	PartKey	Quantity	SuppKey
PartKey	8	6	5
Quantity	6	9	4
SuppKey	5	4	8

(PartKey, SuppKey)

	PartKey	Quantity	SuppKey
PartKey	9	6	6
Quantity	6	9	4
SuppKey	6	4	9

- Re-cluster only the referenced P_u in affinity matrix

+48 0

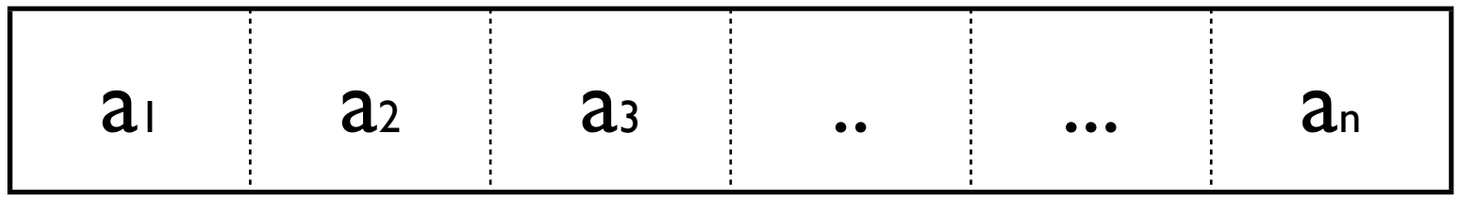
	PartKey	Quantity	SuppKey
PartKey	9	6	6
Quantity	6	9	4
SuppKey	6	4	9

	SuppKey	PartKey	Quantity
SuppKey	6	9	6
PartKey	4	6	9
Quantity	9	6	4

How to Analyze the Workload?

Step 2: Enumerating Split Vectors

- offline approach: consider all possible split vectors (brute force)



Complexity: 2^{n-1}

How to Analyze the Workload?

Step 2: Enumerating Split Vectors

- offline approach: consider all possible split vectors (brute force)
- online approach: One-dimensional Online Partitioning (O^2P) Algorithm

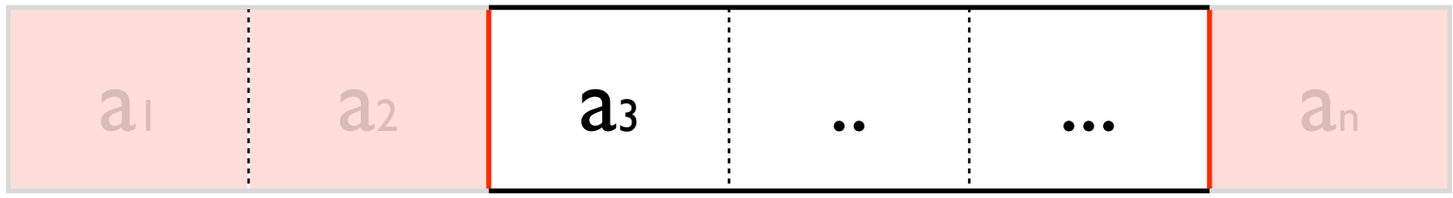
  Technique 1: prune non-referenced partitioning units

  Technique 2: consider split vectors greedily

  Technique 3: save previous best split vectors using dynamic programming

Partitioning Unit Pruning

Idea: Prune the unused (non-referenced) P_u in at most two separate partitions

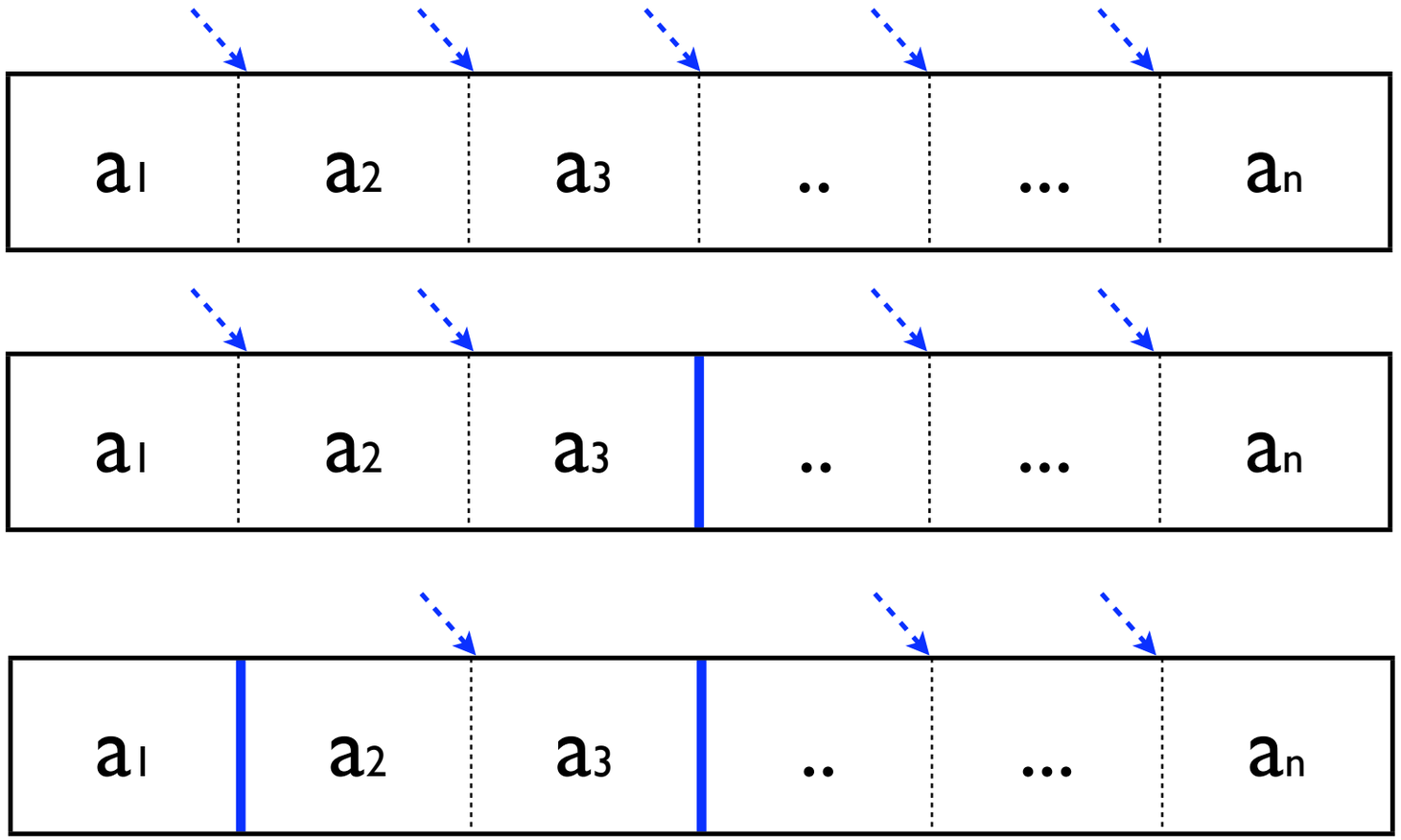


Complexity: For p leading and q trailing unused P_u

$$2^{n-p-q-1}$$

Greedy Split Vector Enumeration

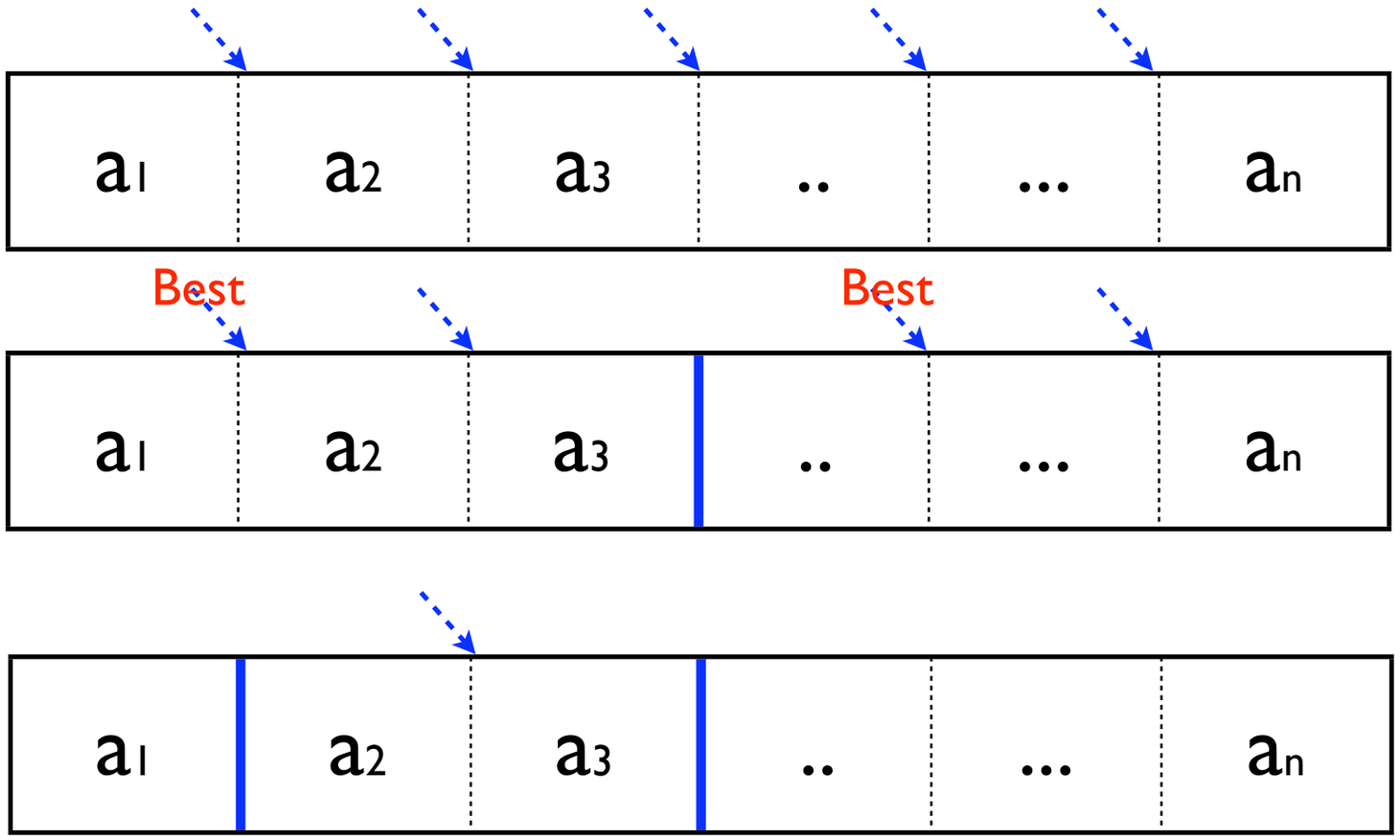
Idea: Mark only one (best) split vector at a time



Complexity: worst case n^2

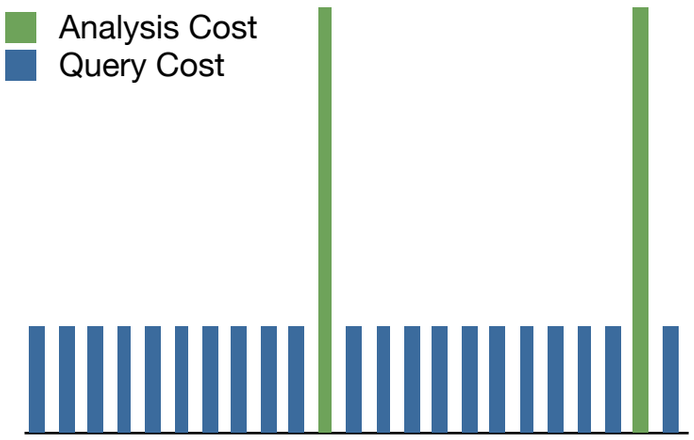
Dynamic Programming

Idea: save best split vectors in un-split partitions

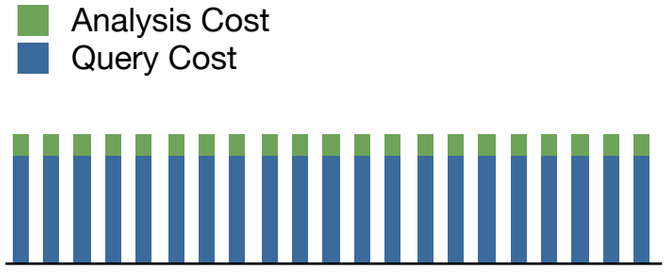


How to Amortize Partitioning Analysis?

- offline approach: querying after computing and creating partitions



- online approach:
  option1: interleave queries with partitioning analysis
  option2: queries in a separate thread



Goals of the Experiments

- Does greedy partitioning hurt Quality?
- How much is O2P faster?
- Can such a system adapt to changing workload ?
- Will our approach work on real systems?

Dynamic Workload

- Mix of OLTP and OLAP style queries
- OLTP: 1% selectivity and 75-100% attributes
- OLAP: 10% selectivity and 1-25% attributes
- Vary the fraction of OLTP-OLAP over time

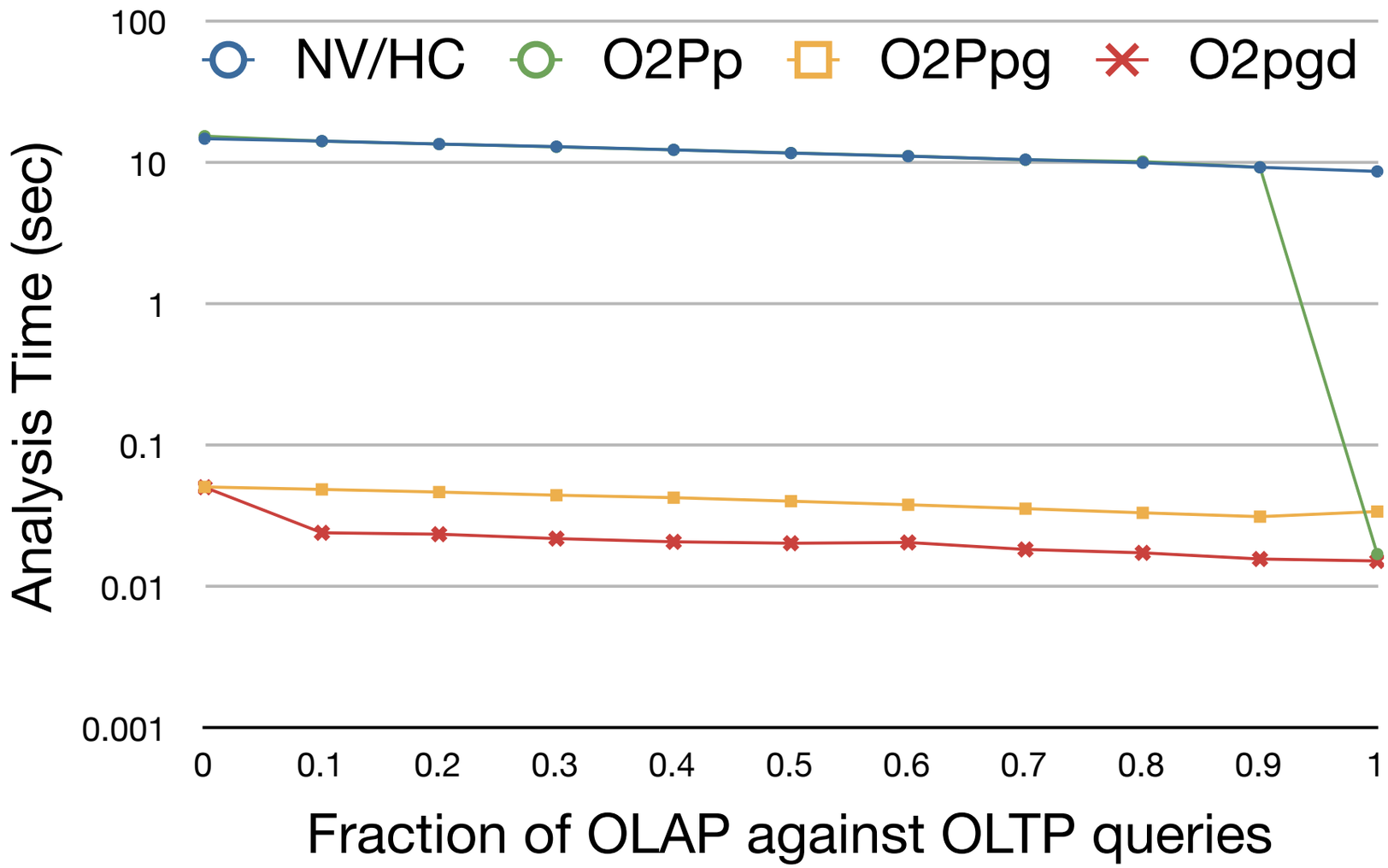
Does Greedy Partitioning Hurt Quality?

Quality: Ratio of expected query costs of optimal partitioning and the partitioning produced by the algorithm

	Customer			Lineitem		
	Optimal	Navathe	O2P	Optimal	Navathe	O2P
Quality	100%	99.29%	92.76%	100%	97.45%	95.80%
Iterations	100%	14.60%	2.28%	100%	2.42%	0.14%

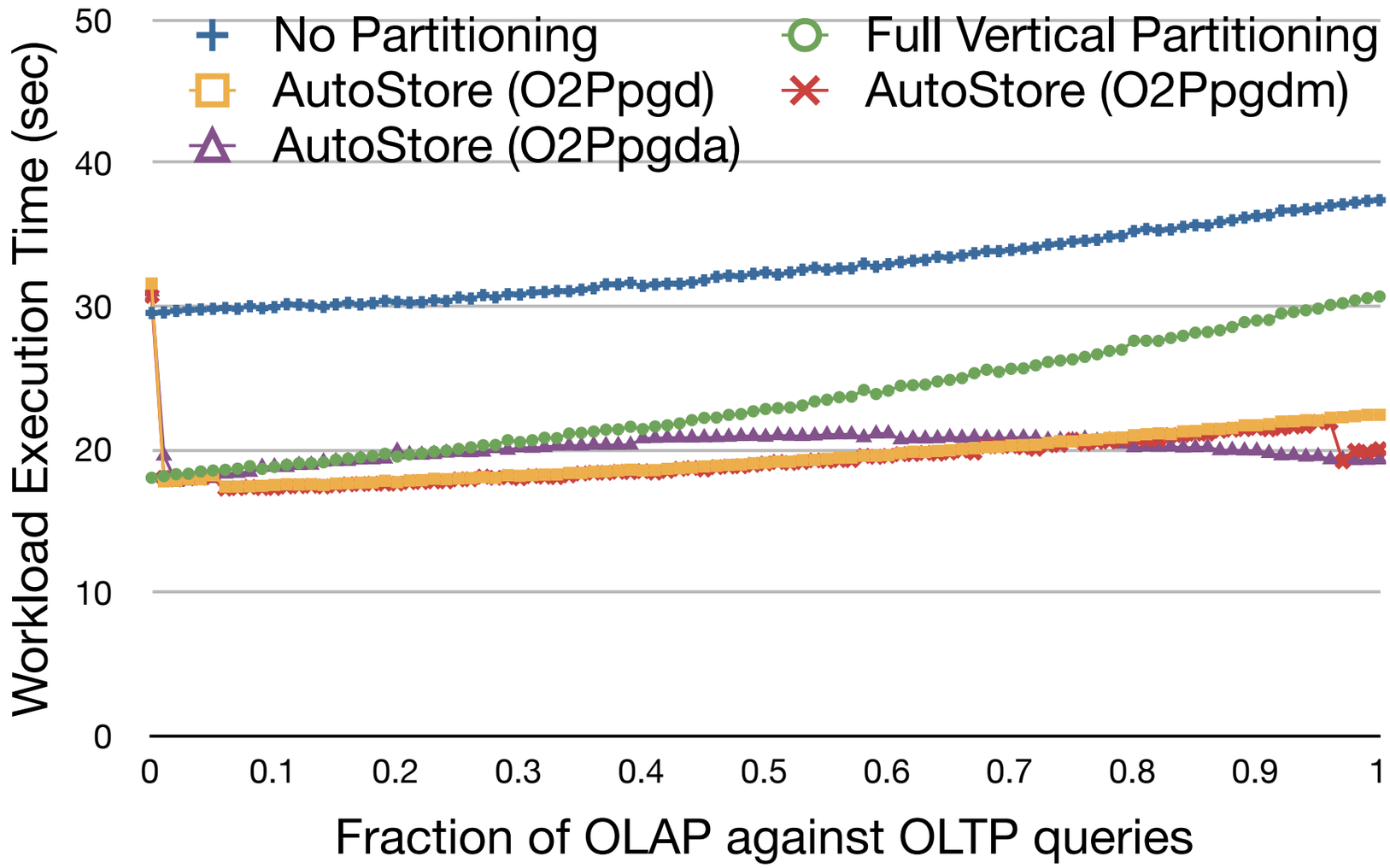
How much is O2P Faster?

Setup: TPC-H Lineitem table, 10,000 queries in total



Can such a System Adapt to Changing Workload ?

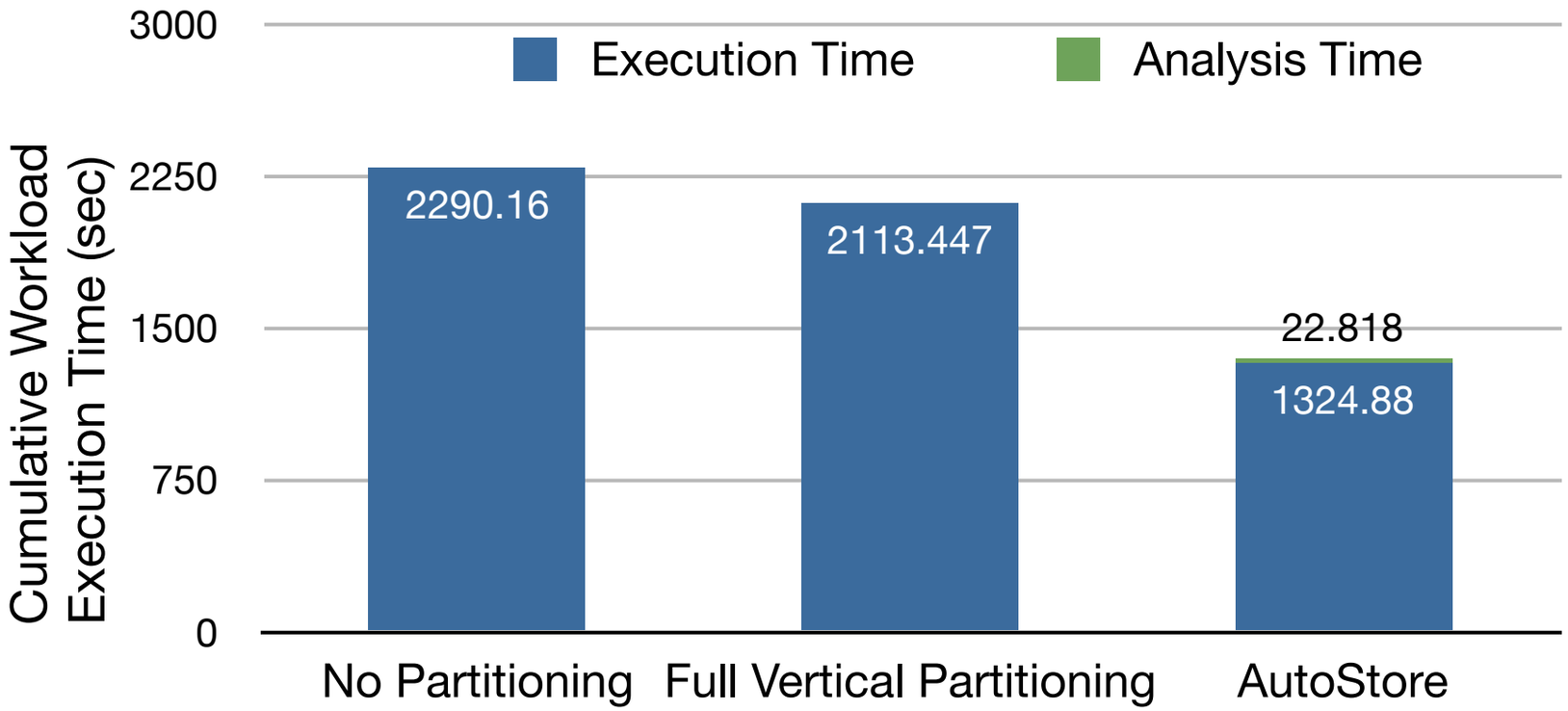
Setup: Universal relation de-normalized from TPC-H schema *, SF **1**



* Constant-Time Query Processing, V. Raman et.al., ICDE 2008

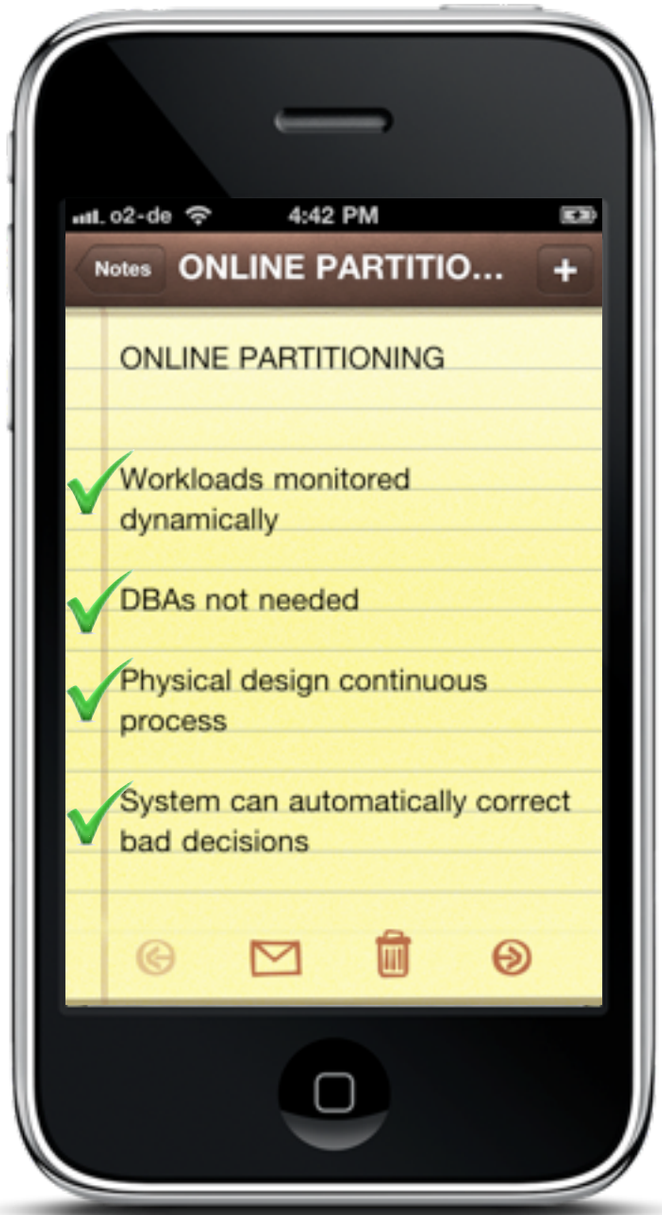
Will our Approach Work on Real System?

Setup: TPC-H Customer table, SF 1, BerkeleyDB



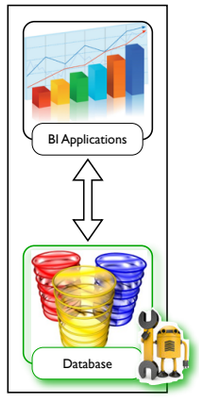
So Whats the Point Again?

- ~~Workloads infrequently change over time~~
- ~~DBAs always available~~
- ~~Physical design once in a while process~~
- ~~DBAs make perfect decisions~~



Summary

Motivation: Online Physical Database Design



Sub-Problem	Proposed Solution
Indexing	Online Indexing Database Cracking Adaptive Indexing
Materialized Views	Dynamic Materialized Views
Partitioning	WE!

5

Partitioning Problem: What to Analyze?

- Partitioning unit P_u e.g. $a_1, a_2, a_3, a_4, a_5, a_6$
- P_u ordering $\preceq$ e.g. $a_3 \preceq a_2 \preceq a_1 \preceq a_5 \preceq a_4 \preceq a_6$
- Split line, Split vector S e.g. [01001]
- Partition $p_{m,r}(S, \preceq)$ e.g. (a_1, a_2)
- Partitioning scheme $P(S, \preceq)$ e.g. $(a_1, a_2), (a_3, a_4), (a_5, a_6)$
- Workload W_{t_k}
- Problem statement
  Find $\preceq, S'$ such that: $S' = \underset{S}{\operatorname{argmin}}\, C_{\text{est.}}\left(W_{t_k}, P(S, \preceq)\right)$

10

Online Partitioning Unit Ordering

- Update *only* the referenced P_u in affinity matrix

	PartKey	Quantity	SuppKey
PartKey	8	6	5
Quantity	6	9	4
SuppKey	5	4	8

(PartKey, SuppKey) ⇨

	PartKey	Quantity	SuppKey
PartKey	9	6	6
Quantity	6	9	4
SuppKey	6	4	9

- Re-cluster only the referenced P_u in affinity matrix

+48, 0

	PartKey	Quantity	SuppKey
PartKey	9	6	6
Quantity	6	9	4
SuppKey	6	4	9

⇨

	SuppKey	PartKey	Quantity
SuppKey	6	9	6
PartKey	4	6	9
Quantity	9	6	4

13

How to Analyze the Workload?

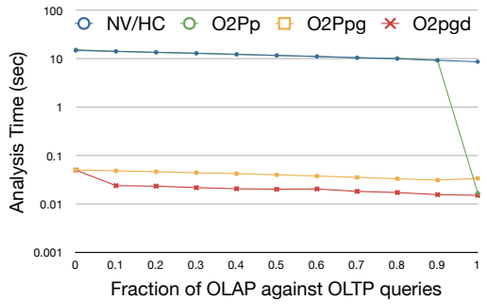
Step 2: Enumerating Split Vectors

- offline approach: consider all possible split vectors (brute force)
- online approach: One-dimensional Online Partitioning (O_2P) Algorithm
  - prune non-referenced partitioning units
  - consider split vectors greedily
  - save previous best split vectors using dynamic programming

15

How much is O_2P Faster?

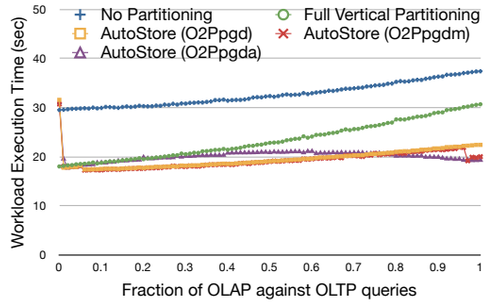
Setup: TPC-H Lineitem table, 10,000 queries in total



23

Can such a System Adapt to Changing Workload ?

Setup: Universal relation de-normalized from TPC-H schema, 11 attributes, SF 1



24