Empowered by Innovation NEC

BPOE-5

Record Placement Based on Data Skew Using Solid State Drives

Jun Suzuki[1,2], Shivaram Venkataraman[2], Sameer Agarwal[2], Michael Franklin[2], and Ion Stoica[2]

[1]Green Platform Research Laboratories, NEC, Japan

[2]University of California, Berkeley

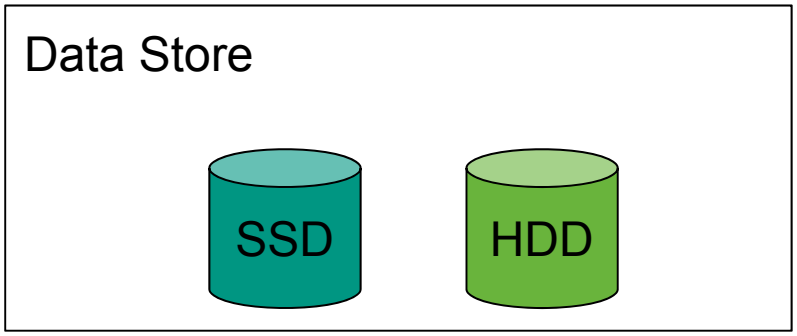
Background

- Integrating SSDs into data stores has been subject of much attention
- Performance in random access outperforms HDDs by >100 times while that in sequential access does not
- Price per byte between SSDs and HDDs differs by orders of magnitude

Device	Random 4K Read IOPS	Sequential Read BW	Price
SSD	41,000-89,000	500-550MB/s	$1/GB
HDD	>91-118	125-156MB/s	$0.05-0.07/GB

Research Motivation

- Configure hybrid data store of SSDs and HDDs
- Introduce data placement method between SSDs and HDDs to maximize I/O performance under constrained SSD usage
- Especially, suppress sharp performance decline that happens when HDDs are randomly accessed by introducing SSDs

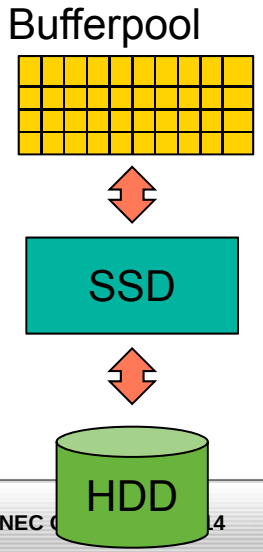


Related Work

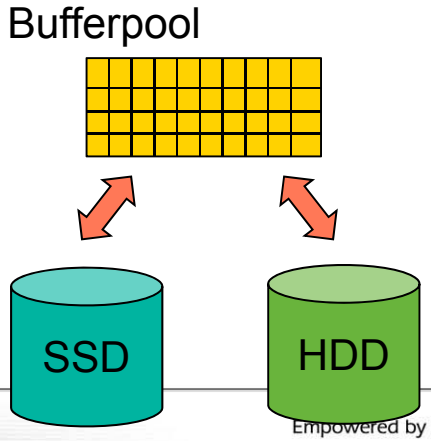
- Most of work are based on reference locality
  - One uses SSDs as cache between memory and HDDs [1]
  - Other places SSDs and HDDs in the same storage tier and optimizes placement of database objects [2]

- Performance gain depends on data access pattern of application
  - Gain cannot be obtained when data are accessed for the first time

Bufferpool extension
[1] M. Canim *et al.*, VLDB 2010



Optimize object placement
[2] M. Canim *et al.*, VLDB 2009

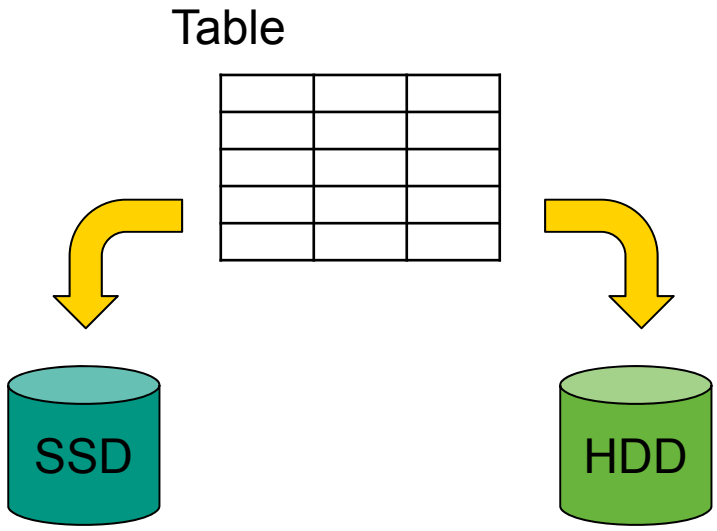


Proposal

SDP: Skew-based Data Placement

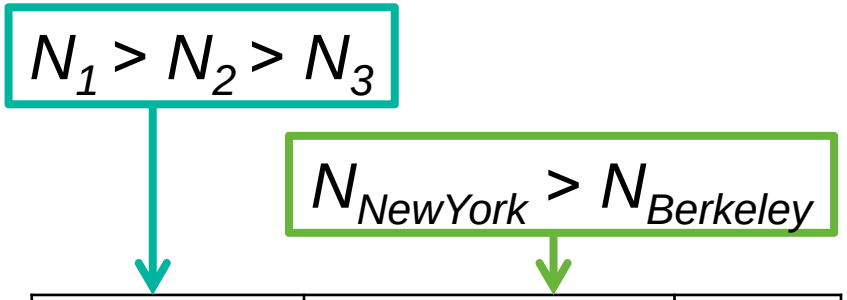
- Data partition method between SSDs and HDDs that is based on data skew
- Orthogonal to previous methods that use reference locality

Performance gain is stable and does not depend on application's access pattern



SDP partitions records in table based on data skew

Data Skew



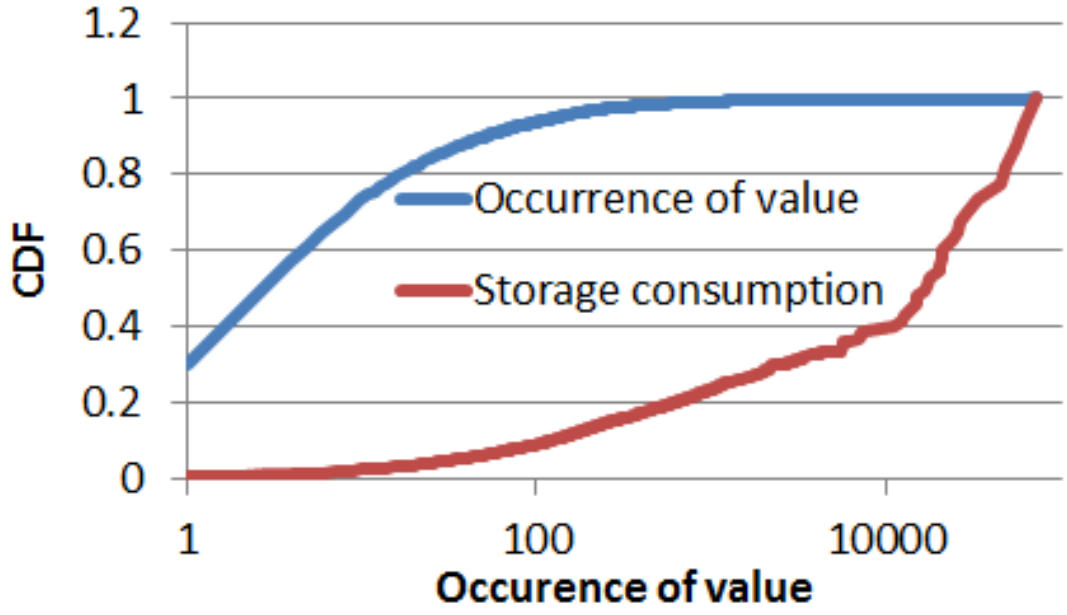
Service Provider ID	City	....
1	New York	....
1	New York	....
1	New York	....
1	New York	....
1	Berkeley	....
1	Berkeley	....
2	New York	....
2	Berkeley	....
3	New York	....

- #entries of values appear in one column generally has skew
- e.g., suppose column is on residence of customers, big cities appear more often than small ones
  - Big city: New York
  - Small city: Berkeley
- On the other hand, cardinality of small cities is large

Data Skew in Real Case

Log of Internet Company, Conviva

- Table has 1,000,000 rows and 104 attributes
- Figure: CDF of occurrence of each data value in composite key and its storage consumption
  - Composite key: (endedFlag, customerId, country, city)
- 90% of values just appear in 6% of records
  - 5% of major values occupy 90% of records



SDP: Skew-based Data Placement 1/3

- SDP enhances performance of fetching records using indices in multi-dimensional way

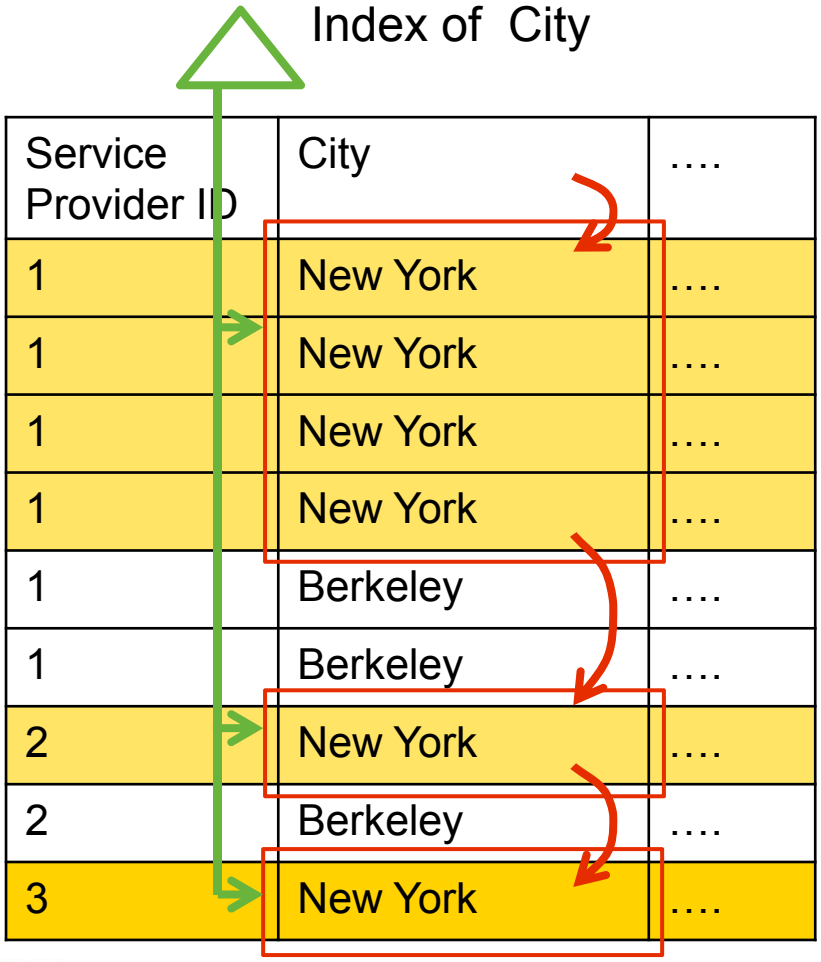
Composite Key

Service Provider ID	City	….
1	New York	….
1	New York	….
1	New York	….
1	New York	….
1	Berkeley	….
1	Berkeley	….
2	New York	….
2	Berkeley	….
3	New York	….

- Records in table are sorted by composite key consisted of columns used to fetch records
  - Suppose Service Provider ID, City
  - Application fetches records by using either of indices of these columns

SDP: Skew-based Data Placement 1/3

- SDP enhances performance of fetching records using indices in multi-dimensional way



- Records in table are sorted by composite key consisted of columns used to fetch records
  - Suppose Service Provider ID, City
  - Application fetches records by using either of indices of these columns
- If whole table were stored in HDD, fetching records with “New York” using index of City requires three HDD seeks
  - Secondary index

SDP: Skew-based Data Placement 2/3

- If up to three records are allowed to move to SSD, selecting records with Service Provider ID of “2” and “3” reduces two HDD seeks
  - Reduction of 66% seeks by moving 33% of records

Service Provider ID	City	….
1	New York	….
1	New York	….
1	New York	….
1	New York	….
1	Berkeley	….
1	Berkeley	….
2	New York	….
2	Berkeley	….
3	New York	….

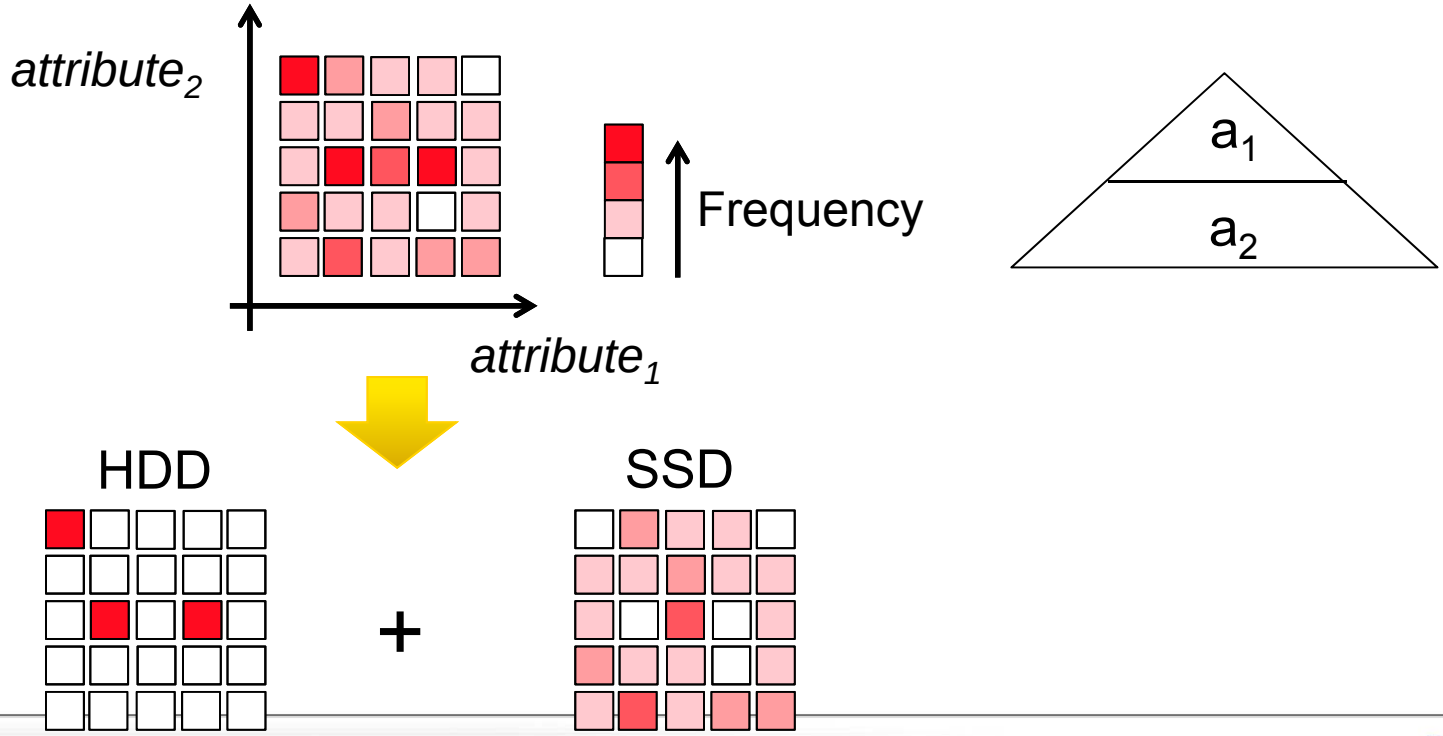
- Speed-up nonlinear to SSD consumptions owes to data skew in Service Provider ID

Moved to SSD

SDP: Skew-based Data Placement 3/3

Intuitive Explanation

- Each axis in figure correspond to one of columns in composite key
- Squares represent possible value
- Colored squares represent existing value
- By storing less frequent values in SSD, large number of colored squares are moved to SSD and random accesses in HDD are reduced



ILP (Integer Linear Programing) in Data Placement 1/3

- Suppose attributes a_1, a_2, a_3 are multi-dimensionally optimized to fetch records by SDP
- Selecting which records to be stored in SSD is ILP (NP-hard)
  - Records that share specific combination of values (k_1, k_2, k_3) in those attributes are stored in the same kind of device

a_1	a_2	a_3	….
			….
			….
			….
k_1	k_2	k_3	….
			….
			….
			….
			….

SSD?,
HDD?

ILP in Data Placement 2/3

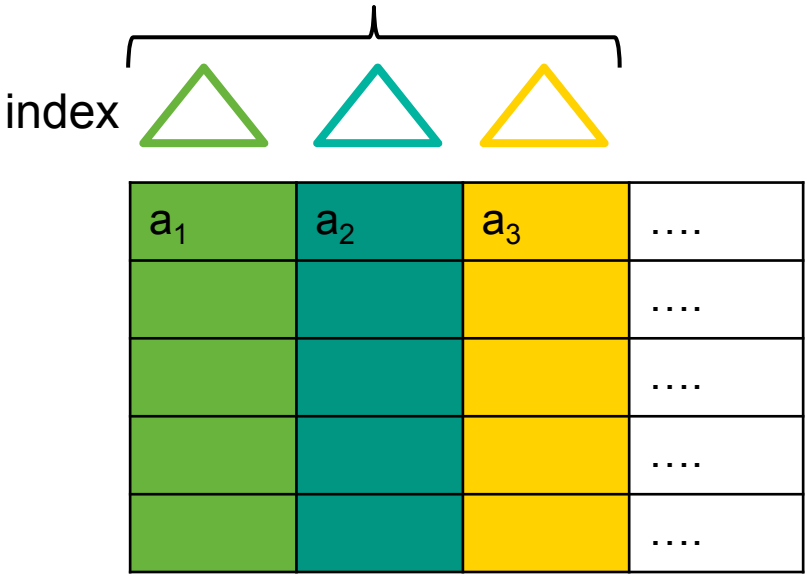
- If, for specific value k_2 in attribute a_2 (e.g. 3), all possible value in a_3 (1, 5) are stored in SSD,
  - a_2 : single seek of HDD in fetching records with k_2 =3 is reduced
  - a_3 : two seeks in fetching records with k_3 =1 or 5 is reduced
- In this way, hierarchical reduction of seeks is modeled

a_1	a_2	a_3	….
			….
1	2	4	….
1	3	1	….
1	3	1	….
1	3	5	….
1	4	1	….
			….
			….

ILP Formulation 3/3

- Select records stored in SSD under constraint of SSD consumption
- Optimize linear combination of average cost to fetch records by using index of each column
- Cost of seeks of HDD is modeled using hierarchical reduction model

Optimize average cost to fetch records by using index of each column



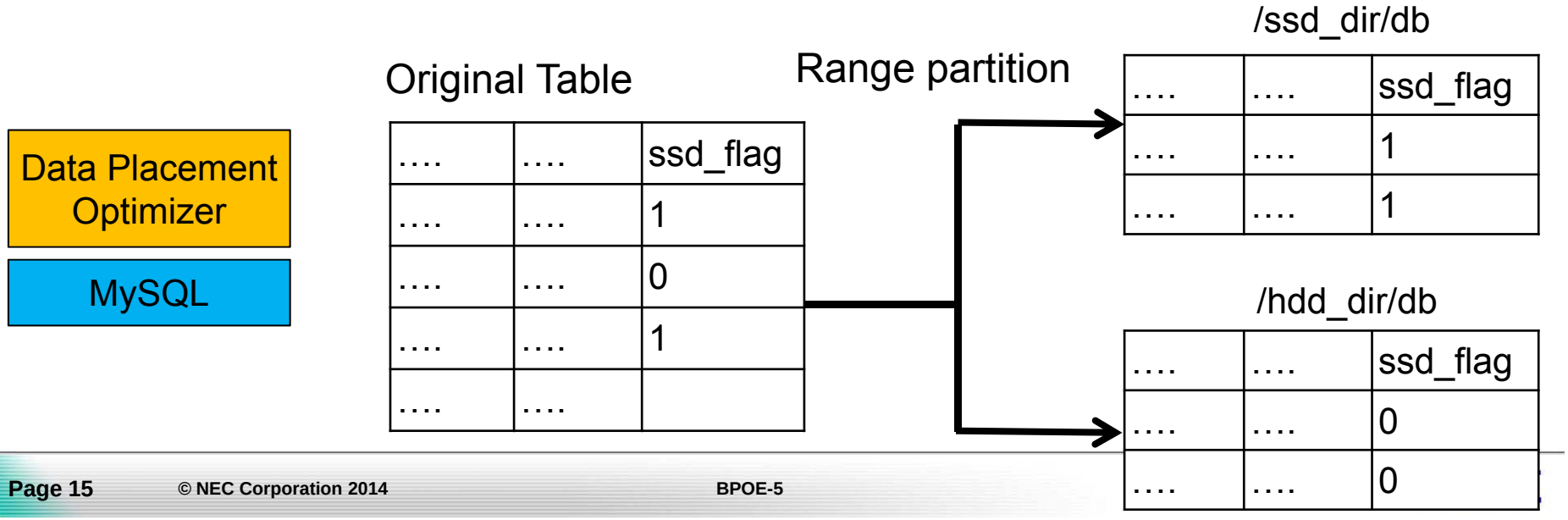
Implementation

Data Placement Optimizer

- Takes statistics of values appear in target columns
- Decides which record is stored in SSD using SDP
  - Greedy method is used to calculate ILP
  - Make new column, ssd_flag, and set it to store its record in SSD

MySQL

- Range partition function on ssd_flag is used to divide storing records between SSD and HDD



Evaluation Setup

Data: Conviva log shown in previous slide

- 104 columns, 1,062,701 records
- Record length: 2KB

Host

- 8-core Xeon
- 128GB Mem
- SATA SSD (Intel 520) and SATA HDD

Evaluation Scenarios

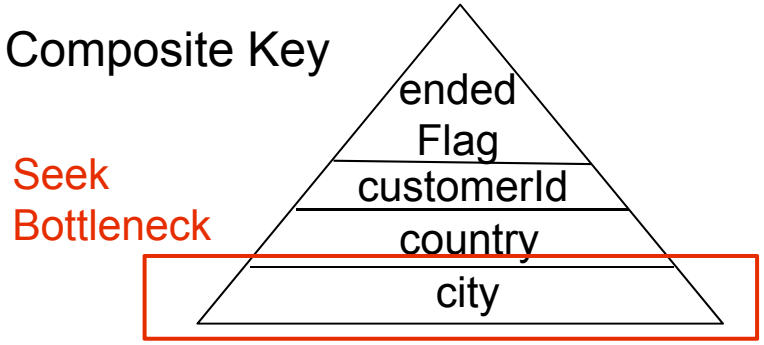
- Cost to fetch records (selection) using index of columns targeted for optimization was evaluated

- Experiment 1
  - Optimized columns were combination of ones frequently used in Conviva analysis
  - Records were not so much fragmented because of correlation

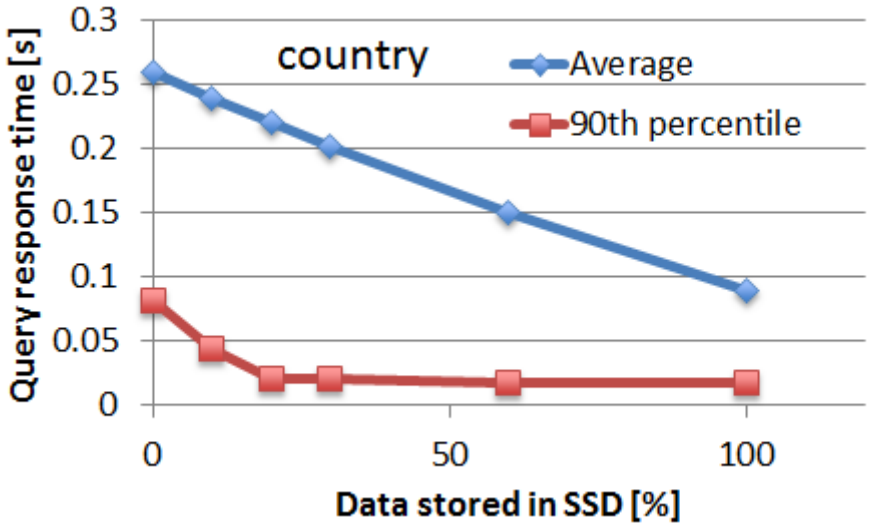
- Experiment 2
  - Different columns were selected so that records were fragmented more

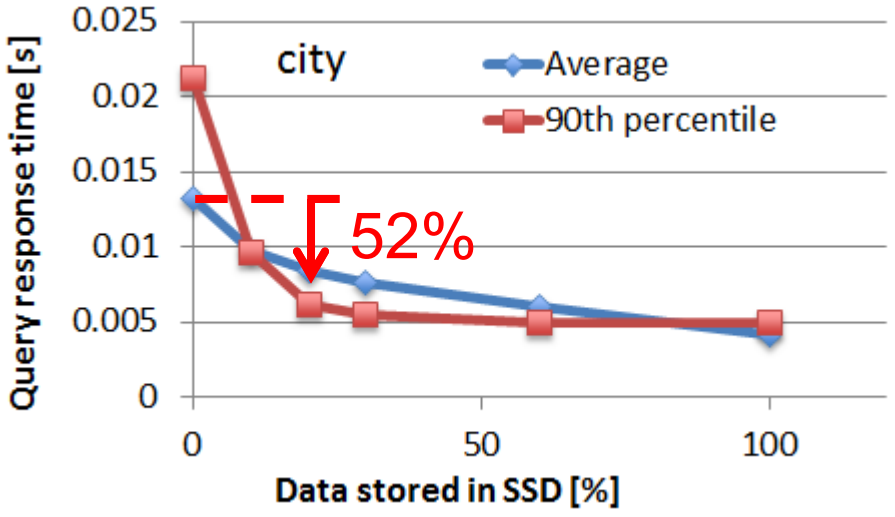
Experiment 1: Frequently used columns for selection

- Only city's bottleneck was seeks of HDD
  - 52% of speed-up was obtained when 20% of records were stored in SSD
  - 90th percentile was improved more, which depends on data distribution
  - Linear performance gain for country because its bottleneck was sequential read bandwidth of HDD



Key	Cardinality
endedFlag	2
(endedFlag, customerId)	12
(endedFlag, customerId, country)	462
(endedFlag, customerId, country, city)	10813





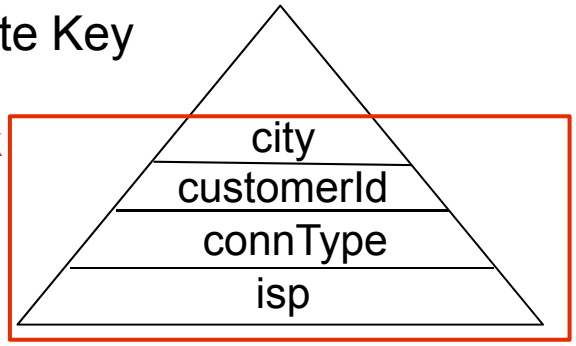
Experiment 2: More Fragmentation

Different columns were selected

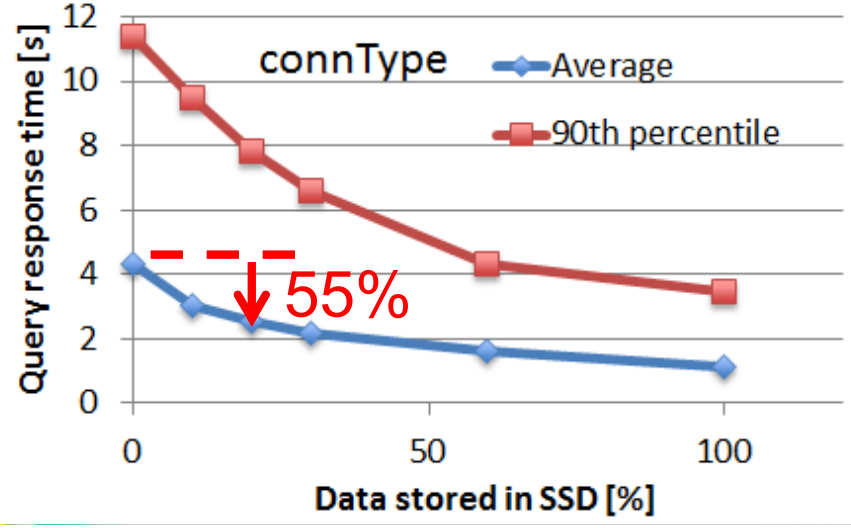
- Length of records was halved to 1KB to increase affects of HDD seek
- Non-linear speed-up were obtained for all of columns
- 55% and 68% of speed-up were obtained for connType and isp, respectively, when 20% of records were stored in SSD

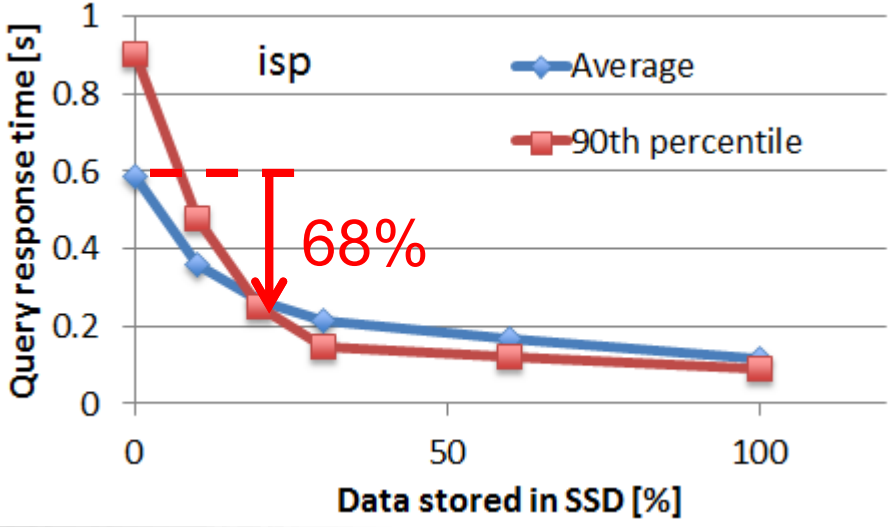
Composite Key

Seek Bottleneck



Key	Cardinality
city	6086
(city, customerId)	9657
(city, customerId, connType)	15462
(city, customerId, connType, isp)	19587





Conclusion

- Proposed SDP (Skew-based Data Placement) to enhance performance of fetching records in hybrid data stores using indices in different columns
  - Unlike caching, its performance is stable
- Evaluation showed speed-up that is nonlinear to SSD consumption
- Future work
  - Compare performance of SDP and other methods based on reference locality
  - Use of real queries for evaluation

Empowered by Innovation