Distributed Cube Materialization on Holistic Measures

Arnab Nandi[1], **Cong Yu[2]**, Philip Bohannon[3], Raghu Ramakrishnan[3]

[1]University of Michigan, Ann Arbor

[2]Google Research, New York

[3]Yahoo! Research, California

Motivation

- Search engines accumulate a lot of query logs
- Each query event has a lot of information
  - Location (inferred from IP)
  - Demographics (inferred from past behavior)
  - The query itself (mapped to semantic categories)
- Goal: identify interesting facts across all combinations of dimensions
  - e.g. “Women in their 40s from Michigan search a lot about knitting.”
- Typical cube analysis, except …

Challenge of Scale

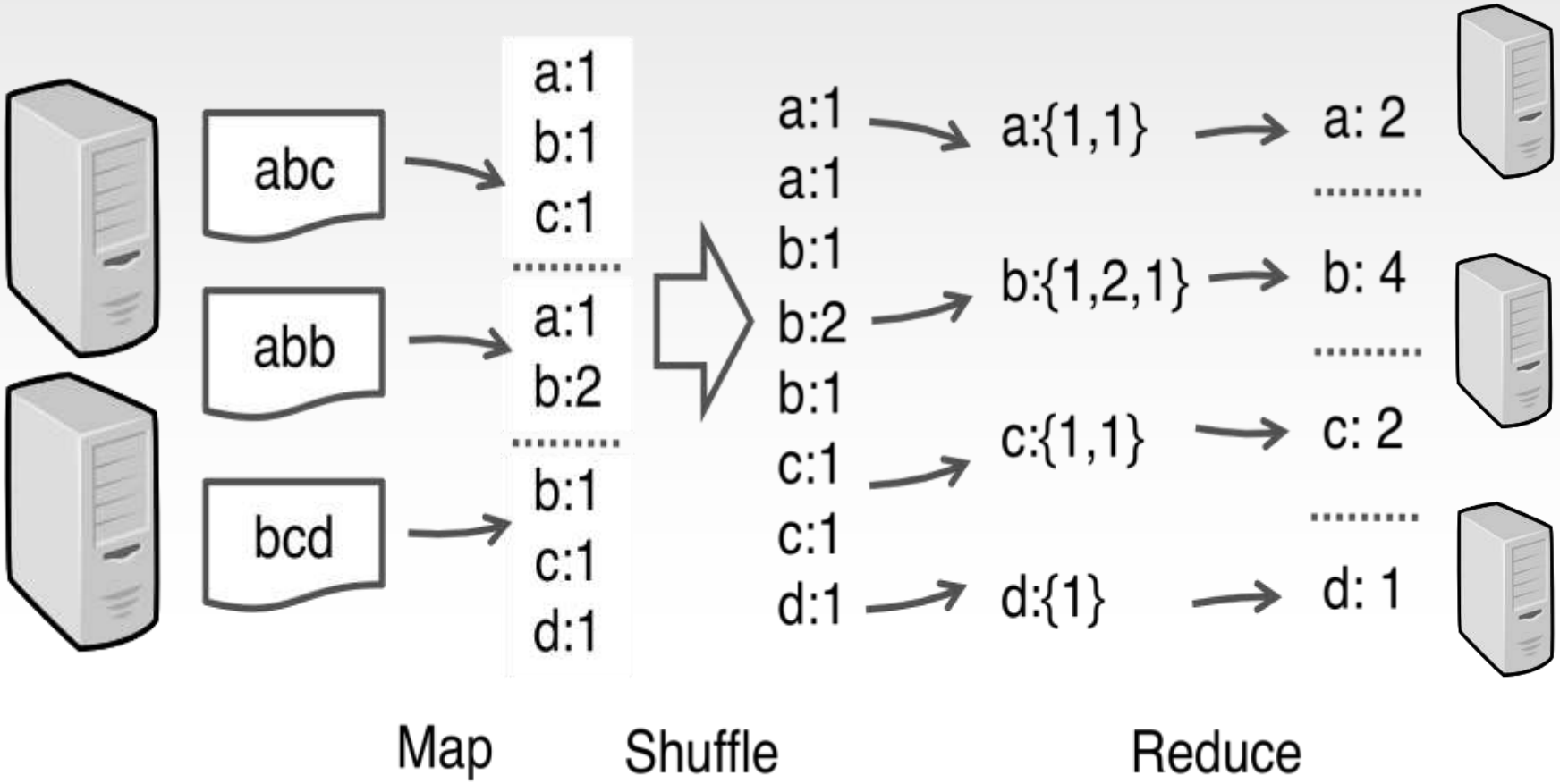
- Over any extended period of time, there can be a lot of such events (i.e., *tuples*)
  - Traditional cube analysis techniques simply can't handle this scale.
- Don't we have MapReduce and problem solved?

- Yes and no:
  - Yes: distributed computation, such as MapReduce, is necessary to handle data at this scale.
  - No: applying MapReduce techniques to cube analysis is not straight-forward.

Outline

- Introduction
- Preliminaries
- Naïve Approach and Challenges
- MR-Cube Approach
- Experimental Evaluation
- Conclusion

Preliminaries: MapReduce



Preliminaries: Data Cube

SELECT COUNT(uid)

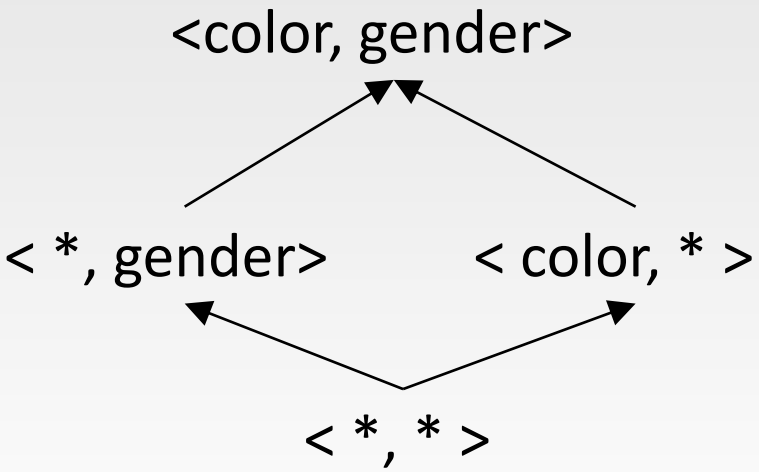
FROM ‘user.table’ as

(uid, color, gender)

CUBE ON color, gender

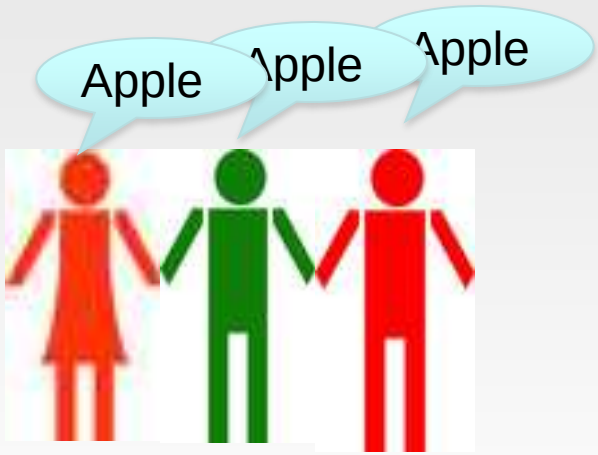
	Men	**Women**	**Total**
Red	3	1	4
Green	1	2	3
Total	4	3	7

Preliminaries: Data Cube



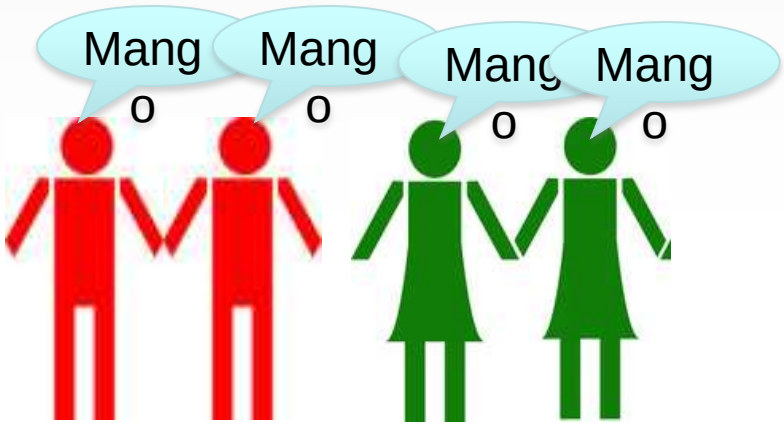
	Men	Women	Total
Red	3	1	4
Green	1	2	3
Total	4	3	7

Preliminaries: Data Cube



(mango)	Men	Women	Total
Red	2	0	2
Green	0	2	2
			4

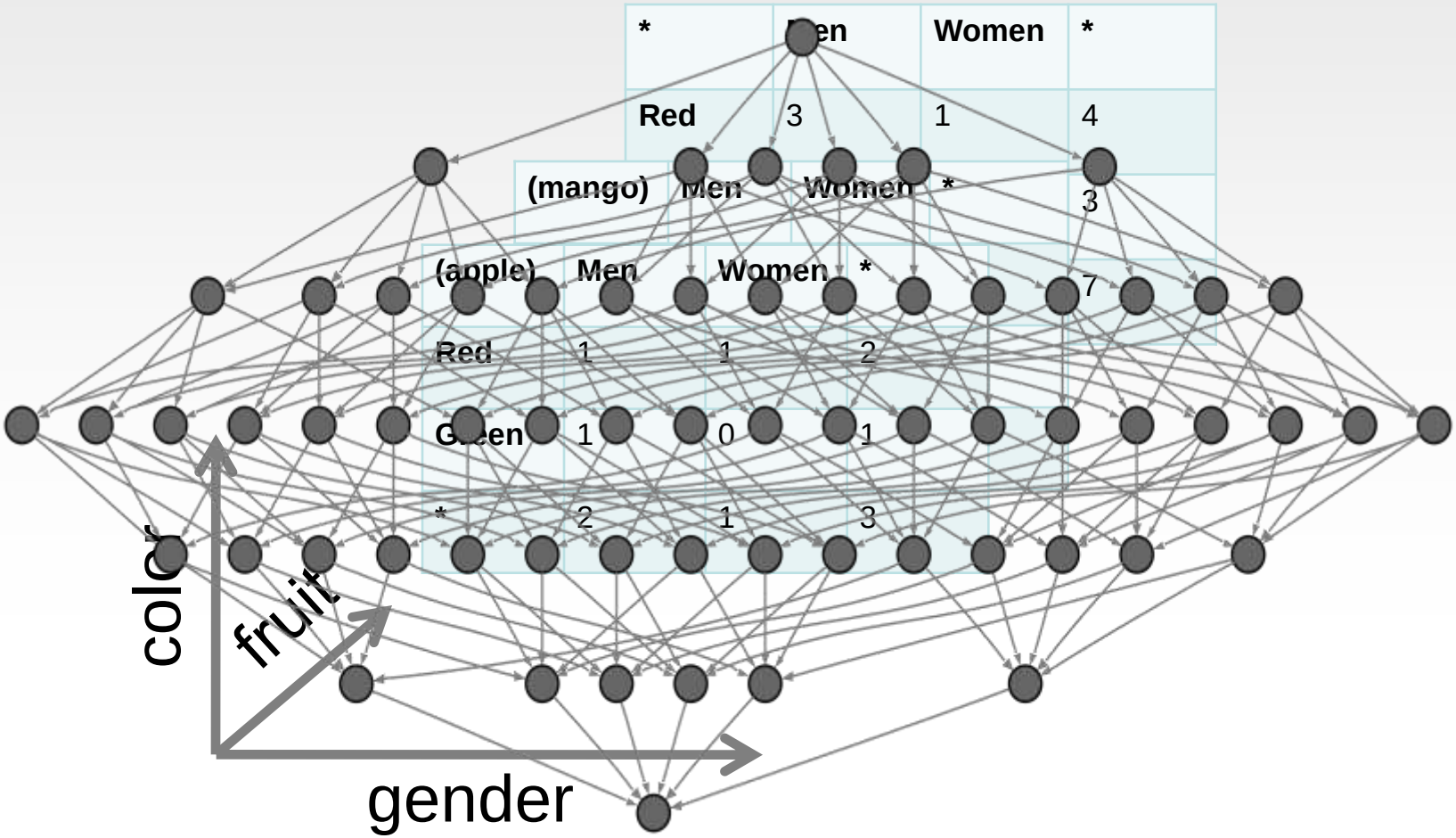
(apple)	Men	Women	Total
Red	1	1	2
Green	1	0	1
Total	2	1	3



new dimension = *fruit*

	Men	Women	Total
Red	3	1	4
Green	1	2	3
Total	4	3	7

Preliminaries: Data Cube



Key to Cube Materialization: Measure

- Algebraic measures
  - Value for parent can be computed easily from values of children.
  - COUNT(A U B) = SUM(COUNT(A), COUNT(B))
- Non-Algebraic (i.e, holistic)
  - Value for parent can not be computed from children.
  - COUNT(DISTINCT(A U B))
- Formal definitions in paper.
- Holistic measures pose significant challenges for distribution!

Outline

- Introduction
- Preliminaries
- Naïve Approach and Challenges
- MR-Cube Approach
- Experimental Evaluation
- Conclusion

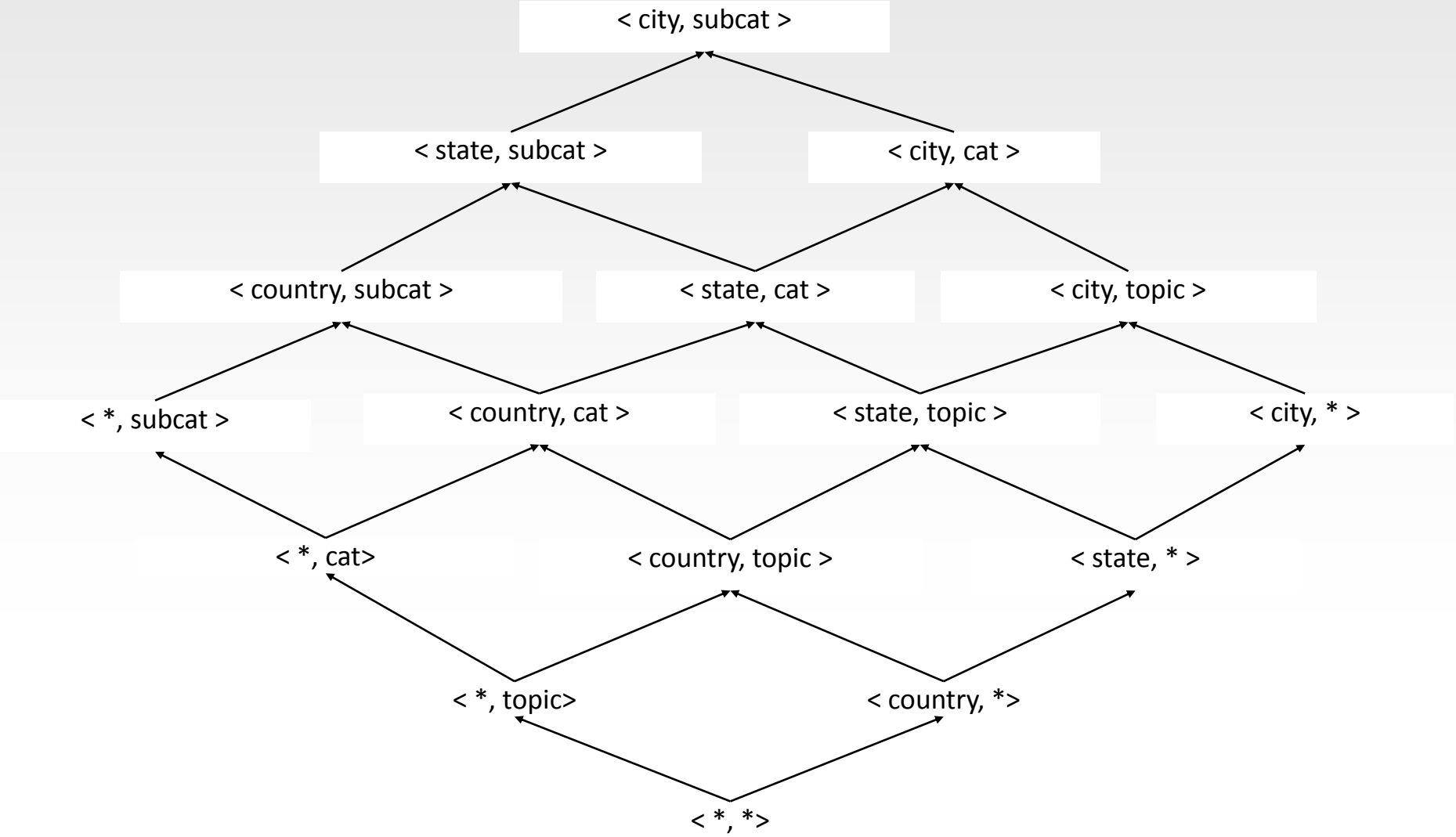
Example Task

- Given the location and topic hierarchies, compute *volume* and *reach* (number of distinct users) of all cube groups whose reach is at least 5.

User	Query (Event)	Time	**Topic …**	**Location …**
u1	Nikon d40	20090601	SLR	Ann Arbor
u2	Canon sd100	20090601	Point & Shoot	Detroit
u1	Apple iPhone	20090602	Smartphone	Ann Arbor
u3	Nikon d40	20090604	SLR	New York
u4	Nokia 1100	20090605	Basic Phone	Sunnyvale

3-levels each

Cube Lattice (dimensions = location + topic)

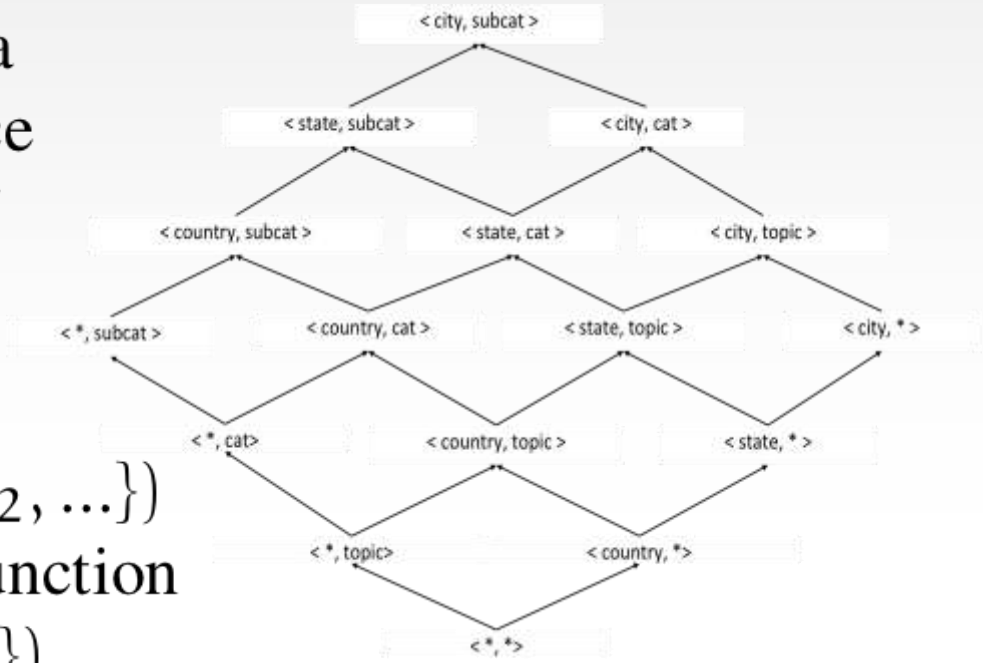


Naïve Algorithm (measure = COUNT-DISTINCT)

Algorithm 1 Naive Algorithm

```
MAP(e)
    # e is a tuple in the data
    let C be the Cube Lattice
    for each Region R in C
    do k = R(e);
        EMIT k ⇒ e

REDUCE/COMBINE(k, {e1, e2, ...})
    let M be the measure function
    EMIT k ⇒ M({e1, e2, ...})
```



Naïve Computation Example

User	Query (Event)	Time	Topic …	Location …
u1	Nikon d40	20090601	SLR	Ann Arbor
u2	Canon sd100	20090601	Point & Shoot	Detroit
u1	Apple iPhone	20090602	Smartphone	Ann Arbor
u3	Nikon d40	20090604	SLR	New York
u4	Nokia 1100	20090605	Basic Phone	Sunnyvale

<Ann Arbor, SLR> ➔ u1

…

<Michigan, Camera> ➔ u1

…

<*, *> ➔ u1

Map phase
(16 records!)

<Ann Arbor, SLR> ➔ {u1}

…

<Michigan, Camera> ➔ {u1, u2}

…

<*, *> ➔ {u1, u2, u3, u4, u5}

Reduce phase

Challenges

- **Map phase**: too many intermediate keys
  - Number of keys = |C| x |D|, where |C| is the number of regions in the lattice, and |D| is the input size.
  - As the number of dimension increases, this can cause problems for the map/shuffle phase.
- **Reduce phase**: extremely large reduce groups
  - Some reduce shards can be very large (due to the a few keys with too many values) and cause the reducer dealing with those groups to run out of memory.
  - In fact, one group is guaranteed to have this problem:

    <*, *>

Outline

- Introduction
- Preliminaries
- Naïve Approach and Challenges
- MR-Cube Approach
- Experimental Evaluation
- Conclusion

Handling Reducer Failure

- Inspired by a common task
  - Compute the unique number of users in a query log
  - The usual practice involves two steps: 1) MapReduce to get the set of unique user IDs; 2) calculate the size of result to produce the final answer
- Holistic measure computation usually can not be distributed, but
  - What if the input is split into disjoint sets with regard to the attribute of the measure (in the above example, user ID)?

Partially Algebraic Measure

- For an aggregate measure M, if there exists an attribute a, such that, given a cube group of tuples G, we can divide G into mutually exclusive partitions, $\{G_i \mid i = 1...k\}$ with the additional conditions that

  $G_i.a \wedge G_j.a = empty$, *for any (i, j) pair*, and

  and we can find functions F and H, such that

  $M(G) = F(H(G_1), \ldots, H(G_k))$

  where H returns an n-tuple and n is constant regardless of the size of the individual partition.

- We call ***a*** the algebraic attribute of M, and M a partially algebraic measure.

Example

User	Query (Event)	Time	Topic …	Location …
u1	Nikon d40	20090601	SLR	Ann Arbor
u2	Canon sd100	20090601	Point & Shoot	Detroit
u1	Apple iPhone	20090602	Smartphone	Ann Arbor
u3	Nikon d40	20090604	SLR	New York
u4	Nokia 1100	20090605	Basic Phone	Sunnyvale

<*, *, uid> ➔ {u1}
<*, *, uid> ➔ {u2}
…
<*, *, uid> ➔ {u5}

no more large groups!

Reduce phase

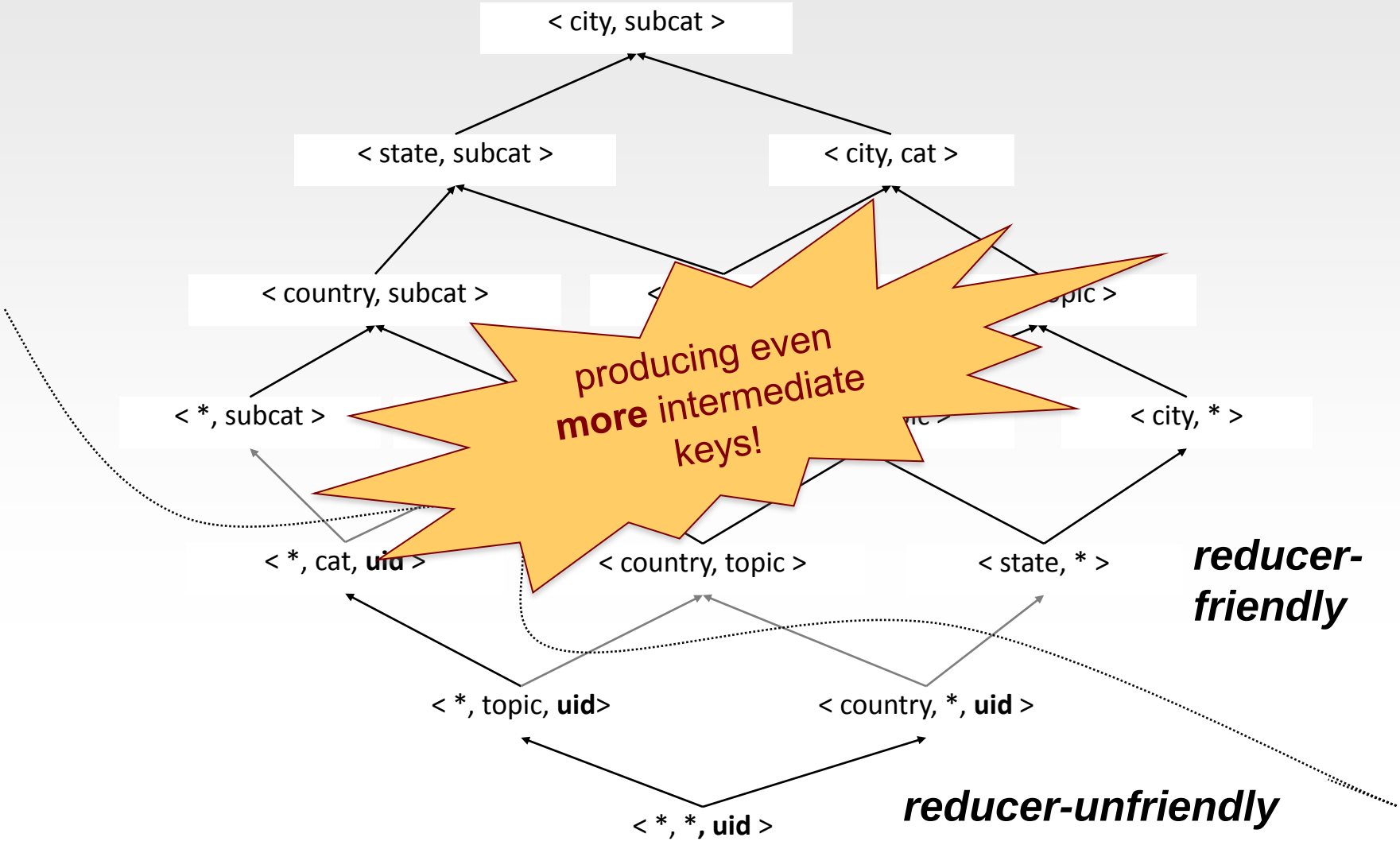
Additional Notes

- Intuitively, the notion of partially algebraicness is the same as the old notion, with the additional constraint that the partition of the data must be done in a certain way (i.e., along the algebraic attribute).
- Some intuitive guidelines for detecting whether a holistic measure is partially algebraic:
  - An aggregation A on the algebraic attribute
  - A fully algebraic measure, M, such that
  - The original measure can be computed as M(A(D))
  - E.g., reach = count(grouby-uid(D))

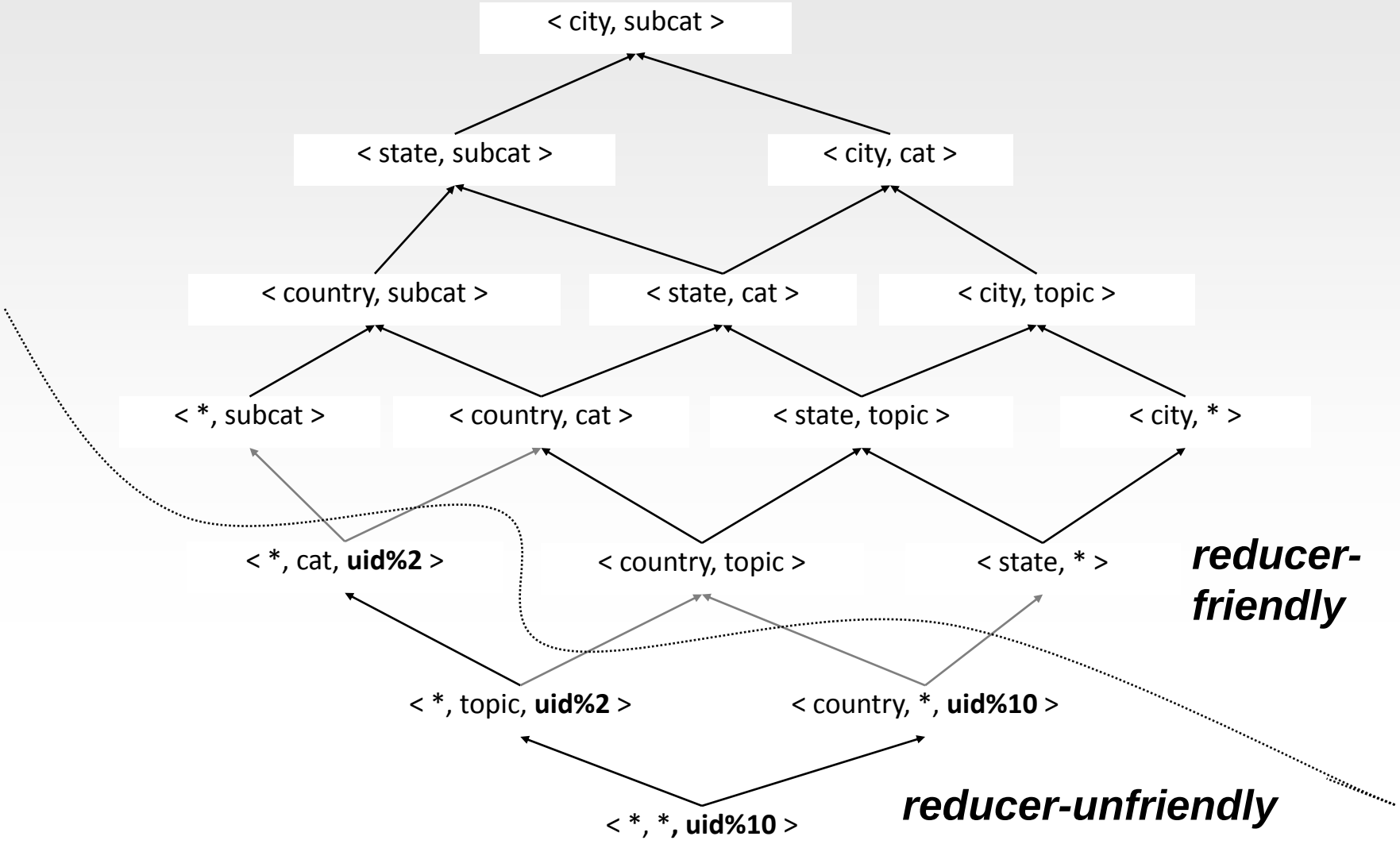
Detecting Large Cube Groups based on Sampling

- We run a small MapReduce job over a sample of the data using the naïve algorithm
  - Goal is to detect any cube group that might be reducer-unfriendly
- With a sample size of 2M tuples, we can obtain accurate enough estimation for 20B tuples
  - Proof based on Chernoff Bounds, see paper for details

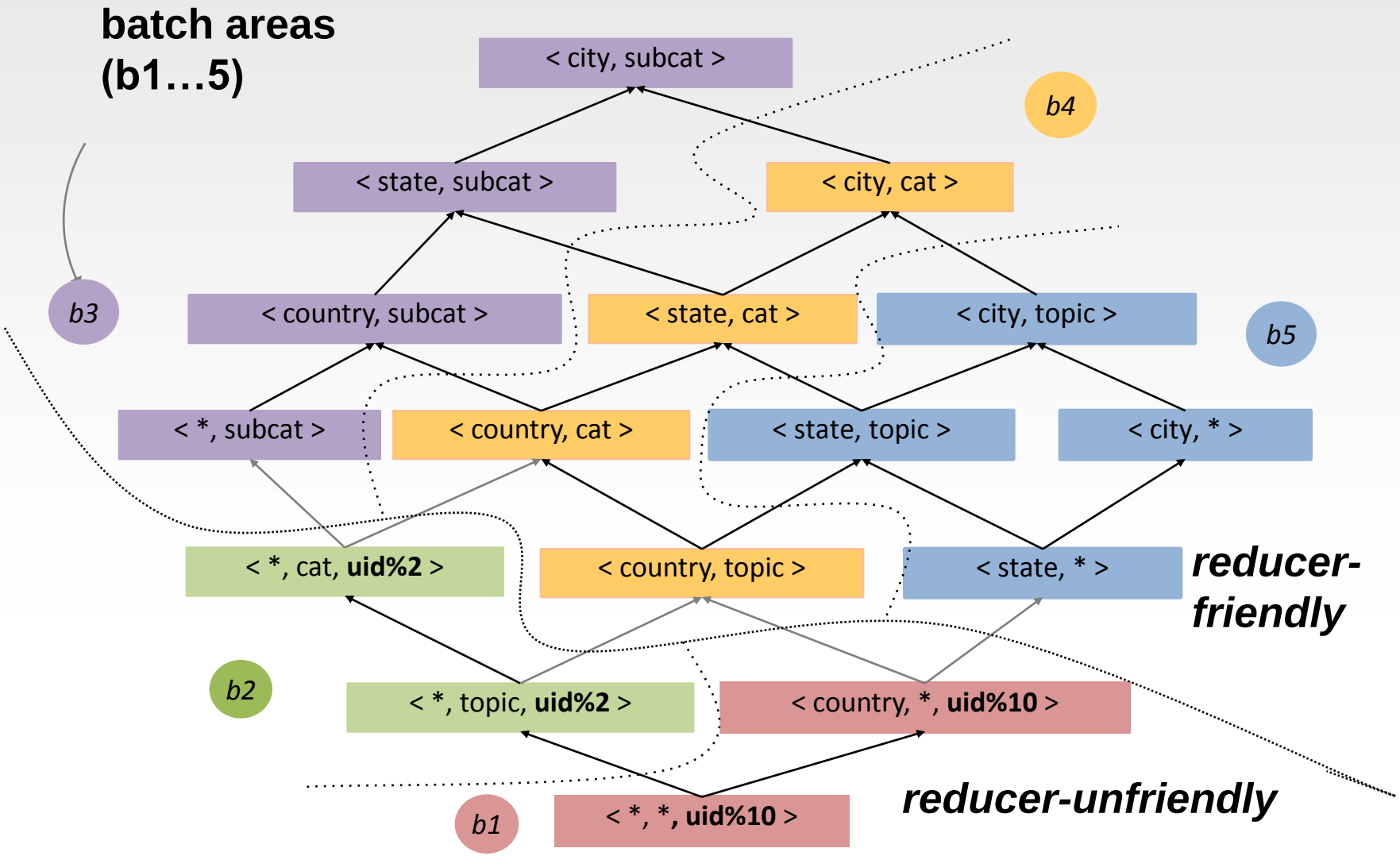
Leverage Partially Algebraic Measures, Naïvely



Value Partition (with partition factors)



Handling Too Many Intermediate Keys



Batch Area: A Constraint Satisfaction Problem

- We want to put parent/child groups together as much as possible to facilitate pruning based on monotonic measures
- Group cube regions into areas such that
  - A region that is reducer-friendly must belong to a batch area that contains at least one of its parents
  - No two regions whose parents are reducer-unfriendly can belong to the same area
  - The maximum difference in the number of regions between any pair of areas is 2, a heuristic to balance the load
- If there are more than one batch area configurations, we have a cost model to select the one with the lowest cost (see paper for details).

MR-Cube Algorithm

```
MR-CUBE(Cube Lattice C, Dataset D, Measure M)
D_sample = SAMPLE(D)
RegionSizes R = ESTIMATE-MAPREDUCE(D_sample, C)
C_a = ANNOTATE(R, C) # value part. & batching
while (D)
do R ← R ∪ MR-CUBE-MAPREDUCE(C_a, M, D)
    D ← D' # retry failed groups D' from MR-Cube-Reduce
    C_a ← INCREASE-PARTITIONING(C_a)
Result ← MERGE(R) # post-aggregate value partitions
return Result
```

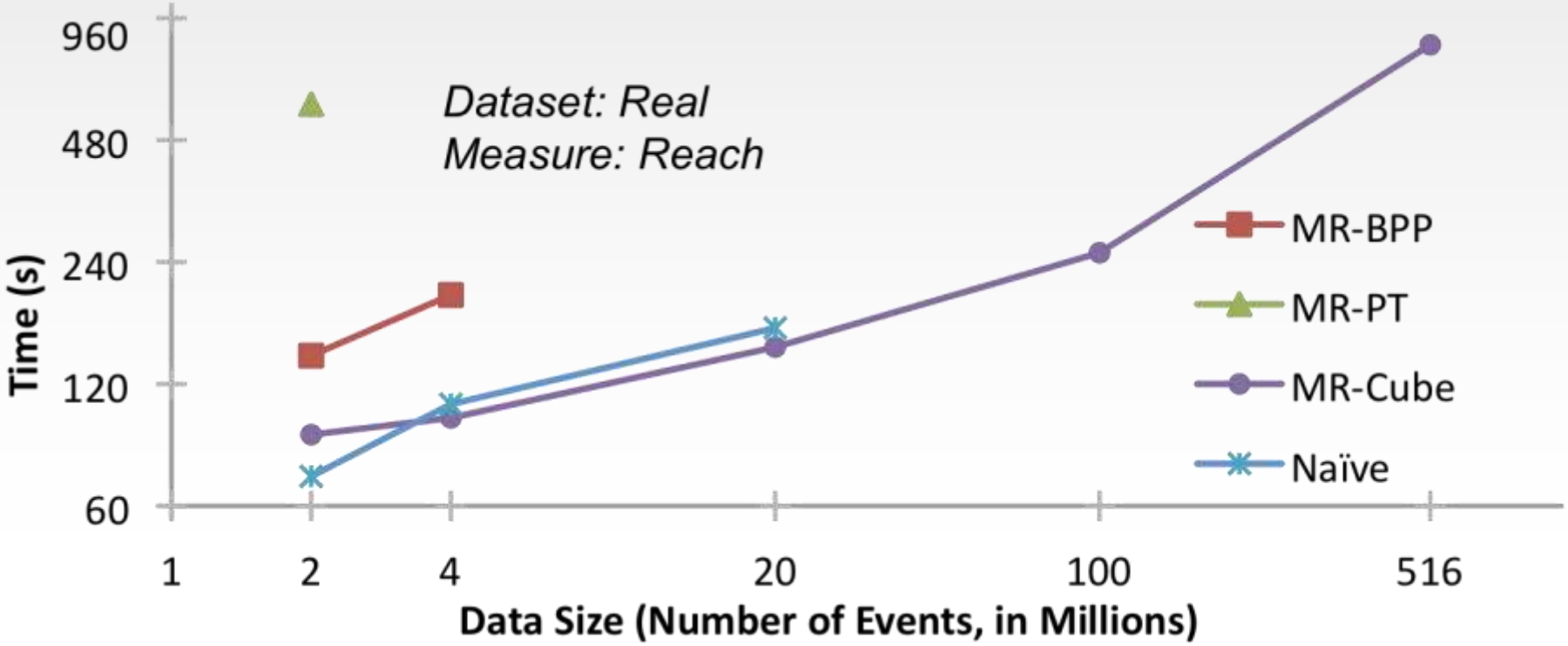
Outline

- Introduction
- Preliminaries
- Naïve Approach and Challenges
- MR-Cube Approach
- Experimental Evaluation
- Conclusion

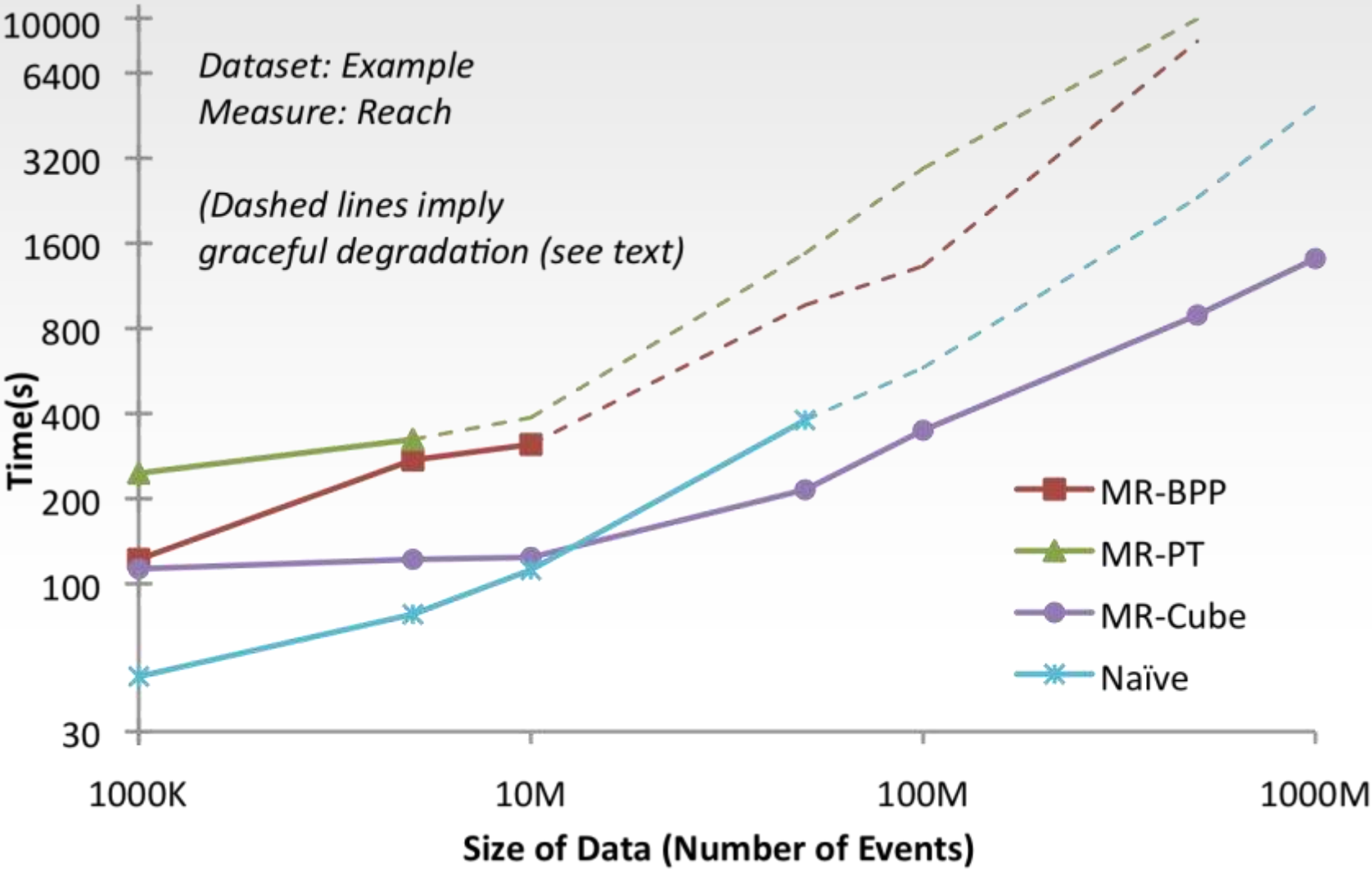
Experimental Setup

- Yahoo! Hadoop Cluster
  - Hadoop 0.20
  - Each node has 2x4core 2.5GHz Xeon/8GB RAM/4x1TB
  - 4GB per mapper and reducer
  - Python + Hadoop Streaming
- Datasets:
  - A sample of real clicklog: 516M tuples, 3 dims, 6 levels
  - Synthetic clicklog: 1B tuples, 2 dims, 6 levels
- Compare against
  - Naïve
  - MR-BPP and MR-PT, adapted from Ng et. al., You et. al., Sergey et. al.

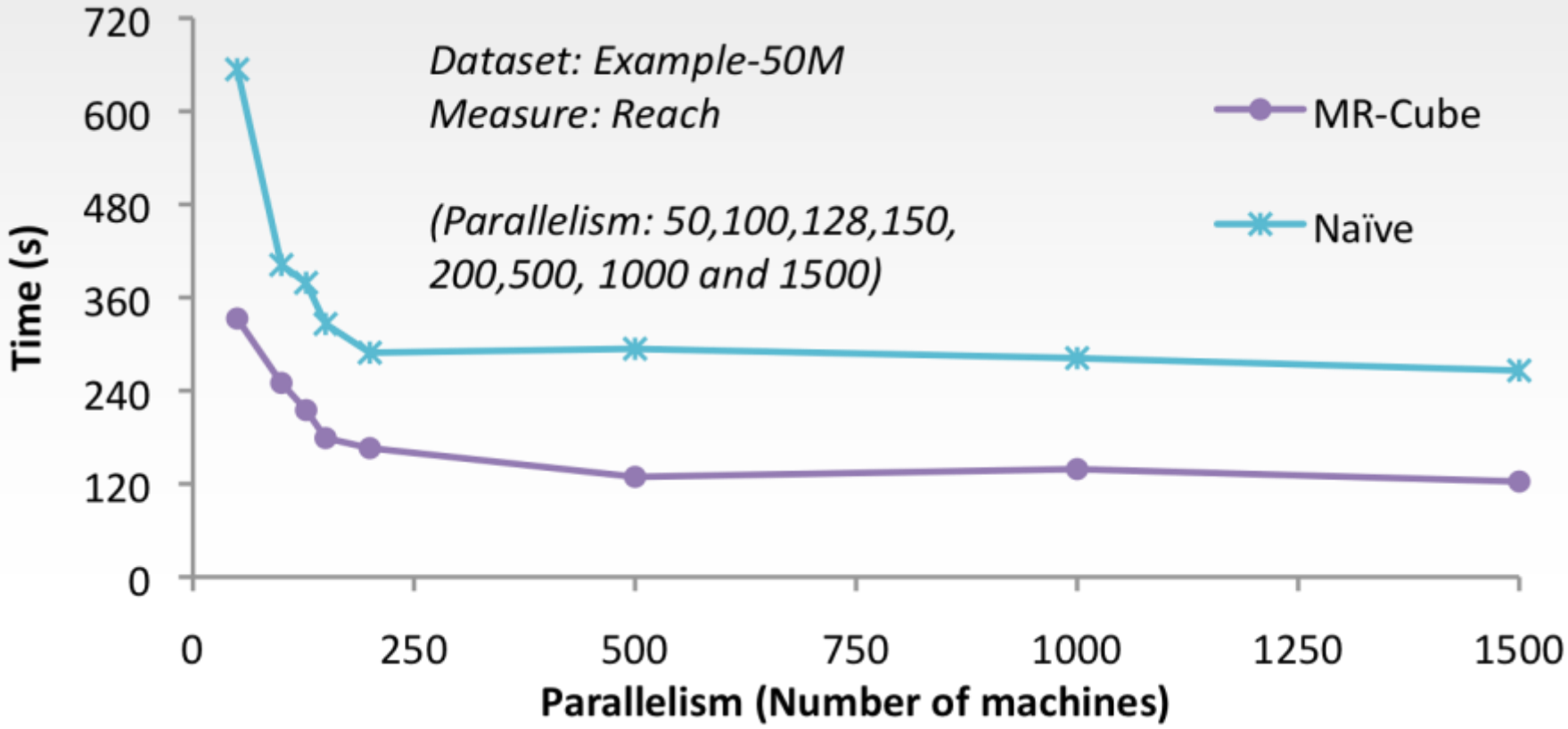
Increasing Data Size (real, reach)



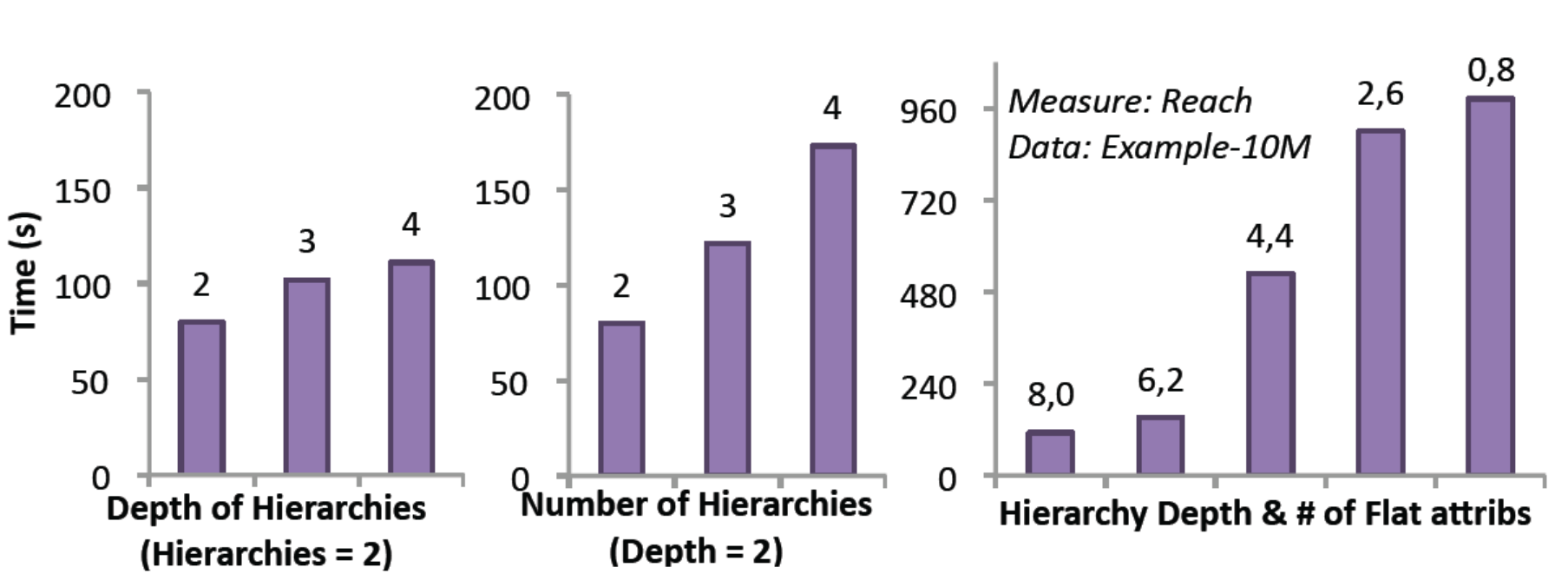
Increasing Data Size (synthetic, reach)



Increasing Parallelism



Effects of Hierarchies



Anecdotal Results

NY	RO	ME	WI
Monday	*Sunday*	*Sunday*	*Monday*
Tuesday	*Friday*	*Monday*	*Sunday*
Friday	*Monday*	*Tuesday*	*Tuesday*
Thursday	*Thursday*	*Friday*	*Friday*

(a) Top 4 frequent days for female users clicking on IMDB URLs, by state

Jan	Feb
Joanna Pacitti	*Joanna Pacitti*
Lisa Bonet	*Tonya Harding Today*
Martin Luther King Jr	*Eliza Dushku*
Kim Kardashian	*Rihanna*
Malia Obama	*Chris Brown*
March	**April**
Crystal Mckellar	*Swine Flu*
Natasha Richardson	*Twitter*
Watchmen	*Keshia Night Pullam*
XBox 360 Ring	*Sabrina Lebeauf*
Marcus Jordan	*Lady Gaga*
May	**June**
Montauk Monster	*Hyalinobatrachium pellucidum*
Kris Allen	*David Carradine*
Adam Lambert	*Dream Interpretations*
Kate Gosselin	*Frank Lloyd Wright Houses*
Derecho Storms	*Air France Flight 447*

(b) Queries with highest reach on Wikipedia URLs, by Month

Related Works

- Lots of works on single machine cube materialization
  - BUC, etc.
- Parallel cube materialization over small clusters
  - Ng et al, *SIGMOD* 2001
  - Chen et al 2008, You et al 2008, Sergey & Yury 2009
  - None of the above handle holistic measures

Conclusion

- Large scale cubing is important
- MR-Cube brings cubing to MapReduce
  - Solve scale challenges by introducing
    - Value Partitioning for Partially Algebraic Measures
    - Batch Areas Identification
  - For very large datasets, MR-Cube is the only option
- Future works
  - Approximate computation using sketches
  - Mining “interesting” cube groups at large scale