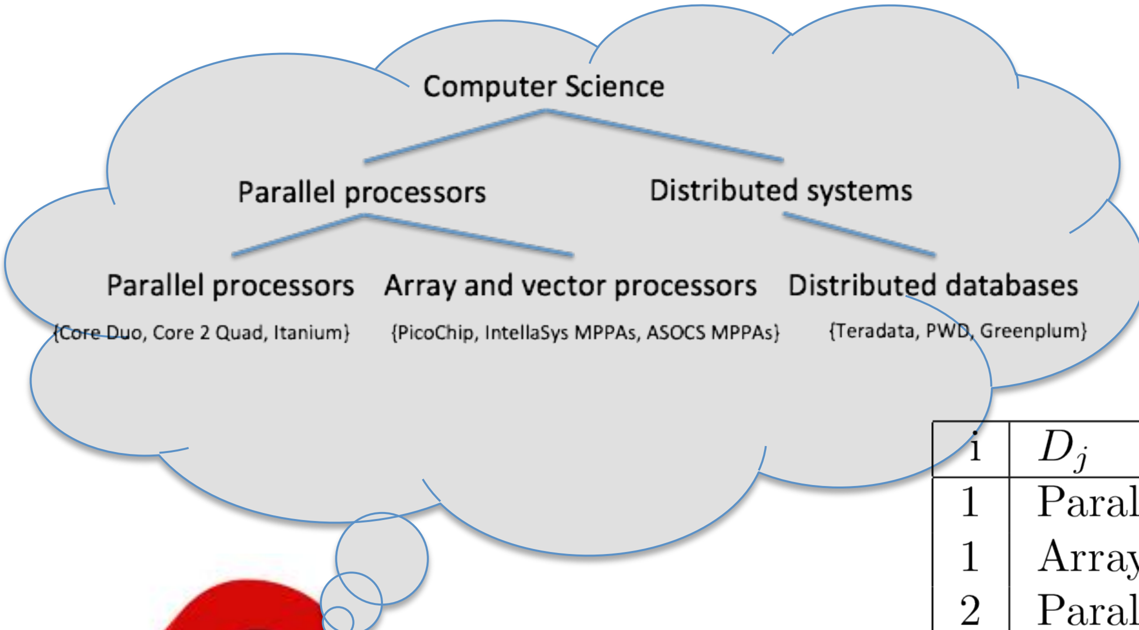
UNIVERSITY of HOUSTON

Query Processing on Cubes Mapped from Ontologies to Dimension Hierarchies

Carlos Garcia-Alvarado
Greenplum EMC
USA

Carlos Ordonez
University of Houston
USA

Scenario



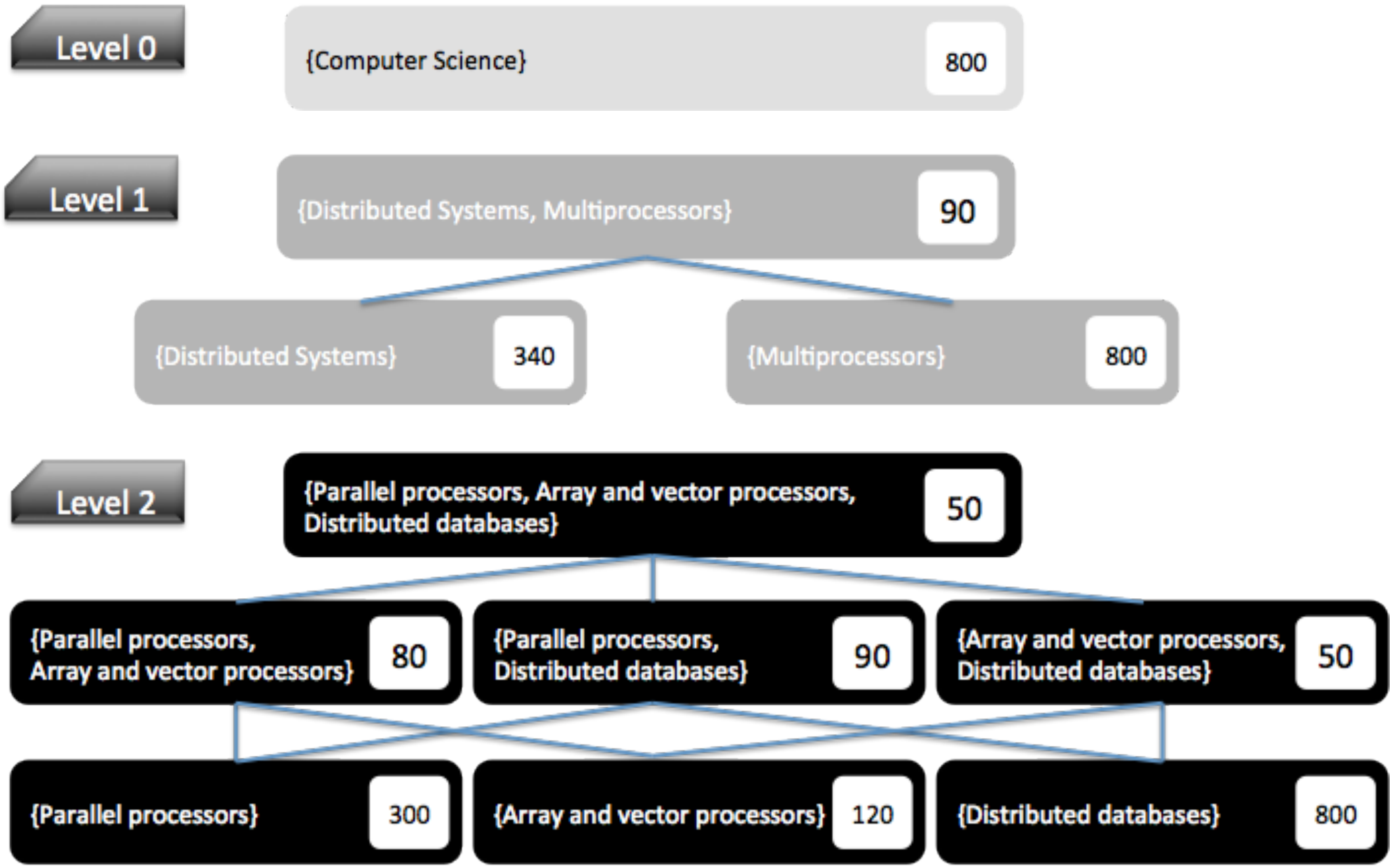
Explore
Digital Library

Dimension — Measurements

Document

i	D_j	A_1
1	Parallel Processors	30
1	Array and Vector Processors	30
2	Parallel Processors	40
2	Distributed databases	40
3	Array and Vector Processors	50
3	Distributed databases	50
3	Parallel Processors	50
4	Parallel Processors	180
5	Array and Vector Processors	40
6	Distributed databases	310
7	Distributed databases	400

Dimension Hierarchies



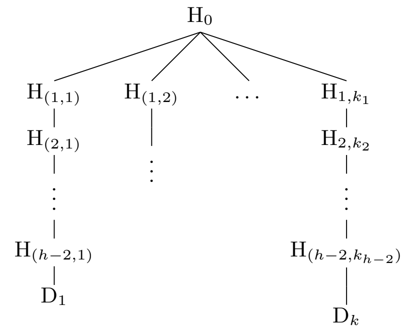
Problem

Efficient summarization of text corpora mapped from ontologies to dimension hierarchies.

OLAP Cube is an excellent candidate to represent concept hierarchies and perform efficient aggregations on multiple combinations.

Previous work required star schema or cubes on demand with all concepts [1][2][3].

Definitions



- Let a collection C, or corpus, of n documents.
- Each document has $\{D_1, D_2, \ldots, D_k\}$ dimensions.
- Each document has a measurements $\{A_1, A_2, \ldots, A_e\}$
- Fact table is in vertical format: $F(i, D_j, A_1, \ldots, A_e)$
- An ontology O is mapped to a dimension hierarchy as a tree-like structure.
- A query Q is a subset of dimensions from F which builds an OLAP cube.

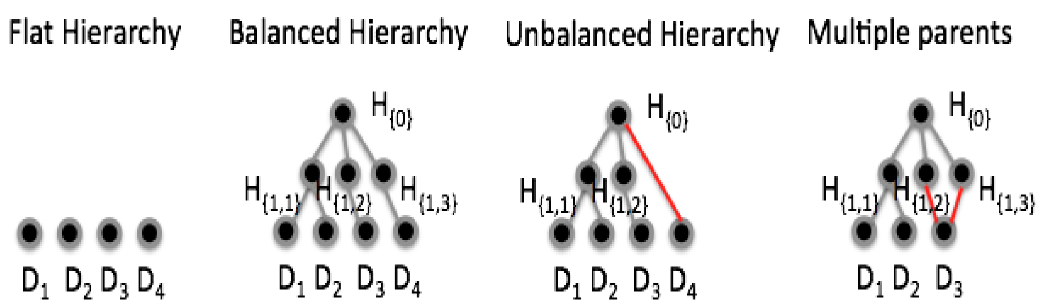


Figure 4: Hierarchies.

CUBO: CUBed Ontologies

- Take advantage of sparse frequency matrix.
- Perform single pass through the data.
- Store the result in a Hash-table.
- Load the Ontology in main memory for summarization by level.

Fact Table Computation

Algorithm 1: CUBO

Input: $\mathcal{O}$,F,Q,T,$\{A_1, \ldots\}$
Output: R

/* Init CUBO struct */
R $\leftarrow \emptyset$;
/* Load ontology in main memory. */
$\mathcal{O} \leftarrow$ LoadOntologyFromOWL();
/* Filter F to consider only those D_j in Q */
$\hat{F} \leftarrow \{t_i | t_i \in F \wedge \exists D_j \ s.t. D_j \in \ t_i \wedge D_j \in Q\}$;
/* Single data set scan. */
t $\leftarrow \emptyset$;
while *row in* $\hat{F}$ **do**
    if *document changed* **then**
        /* Add document to CUBO */
        R $\leftarrow$ R $\cup$ BuildCube(t,$\mathcal{O}$,T,R,$\{A_1, \ldots\}$);
    end
    /* $D_j \in row$. */
    t $\leftarrow$ t $\cup$ $\{D_j\}$;
end
/* Add last document to CUBO */
BuildCube(t,$\mathcal{O}$,T,R,$\{A_1, \ldots\}$);

Build Cube per Document

Algorithm 2: BuildCube

```
Input: t,𝒪,T,R,{A_1,...}
Output: R
s_h ← Combos();
/* Aggregate all the existing combos of the h-1
   level.                                        */
foreach combo do
  | R ← R ∪ {1, combo, {A_1,...}};
end
/* Recursive function to extract all unique
   concepts by level h-2 to 0.                   */
s_{0,...,h-2} ← CombosForOntologyLevel(s,𝒪,T);
/* Increments found combos by level.             */
foreach l in s_{0,...,h-2} do
  | foreach combo in s_l do
  |   | R ← R ∪ {l, combo, {A_1,...}};
  | end
end
return R;
```

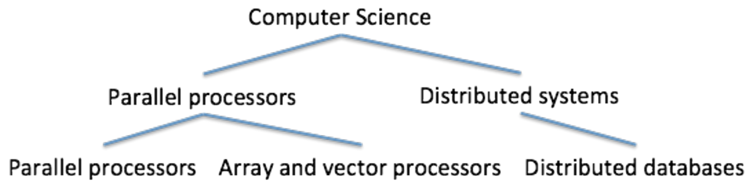
Example

Fact Table

Q={Parallel processors, Array and vector processors, Distributed databases}

…
3, Array and Vector Processors, 50
3, Distributed databases, 50
3, Parallel Processors, 50

$A_1 = 50$



i	D_j	A_1
1	Parallel Processors	30
1	Array and Vector Processors	30
2	Parallel Processors	40
2	Distributed databases	40
3	Array and Vector Processors	50
3	Distributed databases	50
3	Parallel Processors	50
4	Parallel Processors	180
5	Array and Vector Processors	40
6	Distributed databases	310
7	Distributed databases	400

Cube per Document

H_{h-1}
{Array and Vector Processors, Distributed databases} 50 → Hash Table H_{h-1} — Level 2
{Distributed databases, Parallel Processors} 50 →
{Array and Vector Processors, Parallel Processors} 50 →

H_{h-2}
{Parallel Processors, Distributed Systems} 50 → Hash Table H_{h-2} — Level 1

H_0
{Computer Science} 50 → Hash Table H_0 — Level 0

Time Complexity

- Traditional data cube computation $O(nh2^k)$
- The average number of k and h is small.
- Our algorithm has a worst time complexity of $O(n2^{kh})$, but on average performs less computations.

Experiments in a DBMS

- CUBO is a User-Defined Function in C#.
- Our experiments were run on:
  - Intel Xeon Dual Core @3.00 GHz
  - 1 TB Hard drive
  - 4 GB RAM
  - SQL SERVER 2005

Data Sets

Table 1: TPCH Corpora.

n	Max k_j	Min k_j	Total k
1K	3	1	1038
10K	3	1	6589
100K	5	1	9702
1M	5	1	9702
10M	5	1	9702

Table 2: dbpedia Corpora.

n	Max k_j	Min k_j	Total k
1K	9	1	156
10K	14	1	231
100K	16	1	263
1M	26	1	302
10M	46	1	308

Experiments

Table 3: Performance of Traditional Cube and CUBO (* unable to compute)

d	Traditional Single Level	CUBO
2	36	5
4	36	8
8	37	9
16	*	15
32	*	44
64	*	96

Experiments

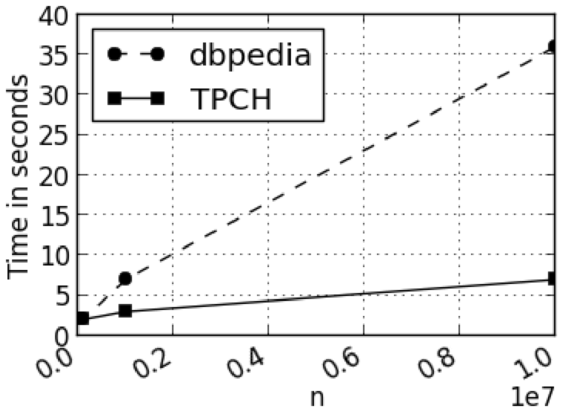


Figure 6: Varying Corpus Size.

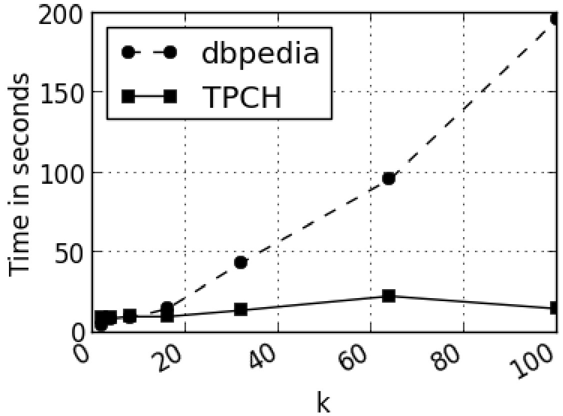


Figure 7: Varying Number of Dimensions.

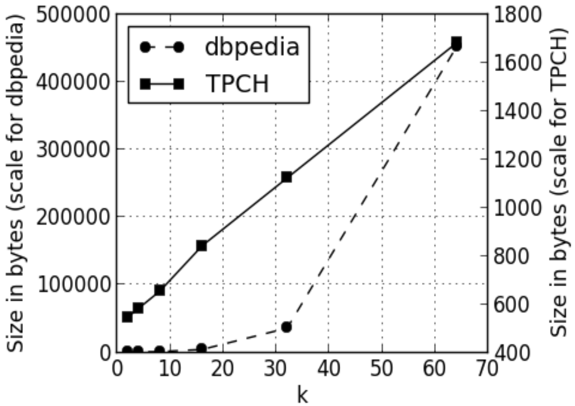


Figure 8: CUBO Size when Varying Number of Dimensions.

Table 5: Varying Ontology Levels in TPCH (time in seconds).

n	ALL	MAX 2	MAX 1
1K	2	2	2
10K	2	2	2
100K	2	2	2
1M	3	2	2
10M	7	7	7

Conclusions

- CUBO is an efficient and single pass algorithm for summarizing hierarchical data.
- CUBO is faster than using a traditional OLAP algorithm.
- CUBO performs faster than the theoretical upper bound.
- CUBO not sensitive to the branching factor.

Future Work

- Support ontologies that do not fit in main memory.
- Improve scalability on h (more than 5 levels deep).
- Support unbalanced trees (ontologies) and ontologies with multiple parents.
- Support incremental computation of new dimensions.
- CUBO needs to be explored in MPP databases.

References

[1] J. Lee, D. Grossman, O. Frieder, and M.C. McCabe. Integrating structured data and text: A multi-dimensional approach. In Proc. of IEEE International Conference on Information Technology: Coding and Computing, pages 264-269, 2000.

[2] C.X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. Text cube: Computing IR measures for multidimensional text database analysis. In Proc. of IEEE ICDM, pages 905-910, 2008.

[3] D. Zhang, C. Zhai, and J. Han. Topic cube: Topic modeling for OLAP on multidimensional text databases. In Proc. of SIAM SDM Conference, 2009.