

Data Warehousing and Data Mining

- The Generalized Multi-dimensional Join -

1. Definition of the GMD-join
2. An Algorithm for the GMD-join
3. Evaluating Subqueries in an OLAP Context
4. Distributed Data Warehousing
5. Conclusion

Acknowledgements: I am indebted to M. Böhlen for providing me the lecture notes.

Goals

- We need an algebraic operator for OLAP queries:
 - The operator must allow us to **easily express** OLAP queries.
 - The operator must admit **efficient evaluation** algorithms.
 - The operator must **interact seamlessly** with the other operators of the relational algebra (query optimization).

Motivating Application

- Analysis of network data
 - Collect, correlate, and analyze huge amounts of data across the network
 - Complex OLAP operations
- Examples:
 - **Network usage:** For each IP address, what fraction of the total traffic is due to web flows?
 - **Principal components:** On an hourly basis, what fraction of the total traffic is from IP subnets whose hourly traffic is within 10% of the max?
 - **Pattern identification:** Break down all flows recorded on US election day by all combinations of source router, destination router, and protocol.

Data Warehouse Schema

- **IP Flows star schema**

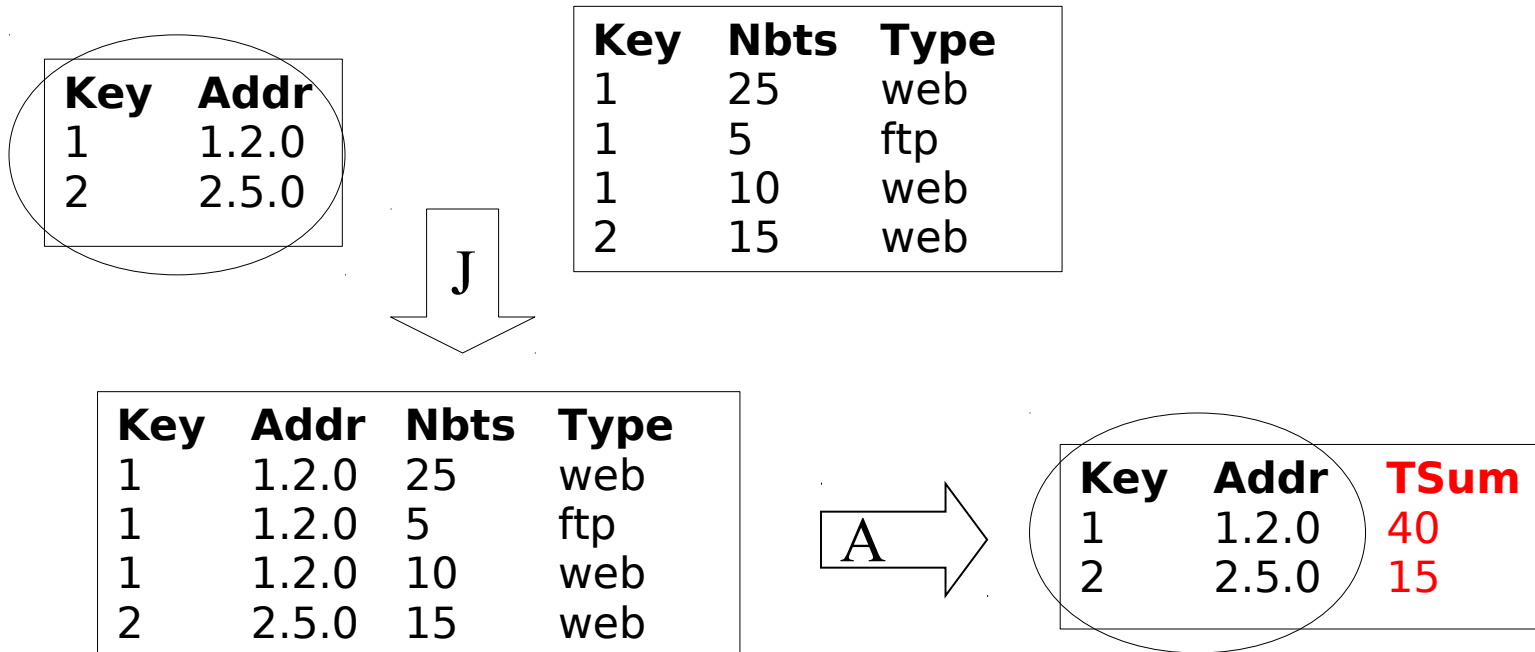
- UserDim(UserId, Name, Addr, Balance, ...)
- IPRegDim(IPKey, UserId, IP, Port, Mask, ASsystem, ...)
- RouterDim(RouterId, Addr, Type, ...)
- HourDim(HourKey, Start, End)
- FlowFact(RouterId, SourceKey, DestKey, Protocol, StartTime, EndTime, NumPackets, NumBytes, ...)

- **Denormalized IP Flow schema**

- Flow(RouterId, SourceIP, SourcePort, SourceAS, DestIP, DestPort, DestAS, Protocol, StartTime, EndTime, NumPackets, NumBytes, ...)

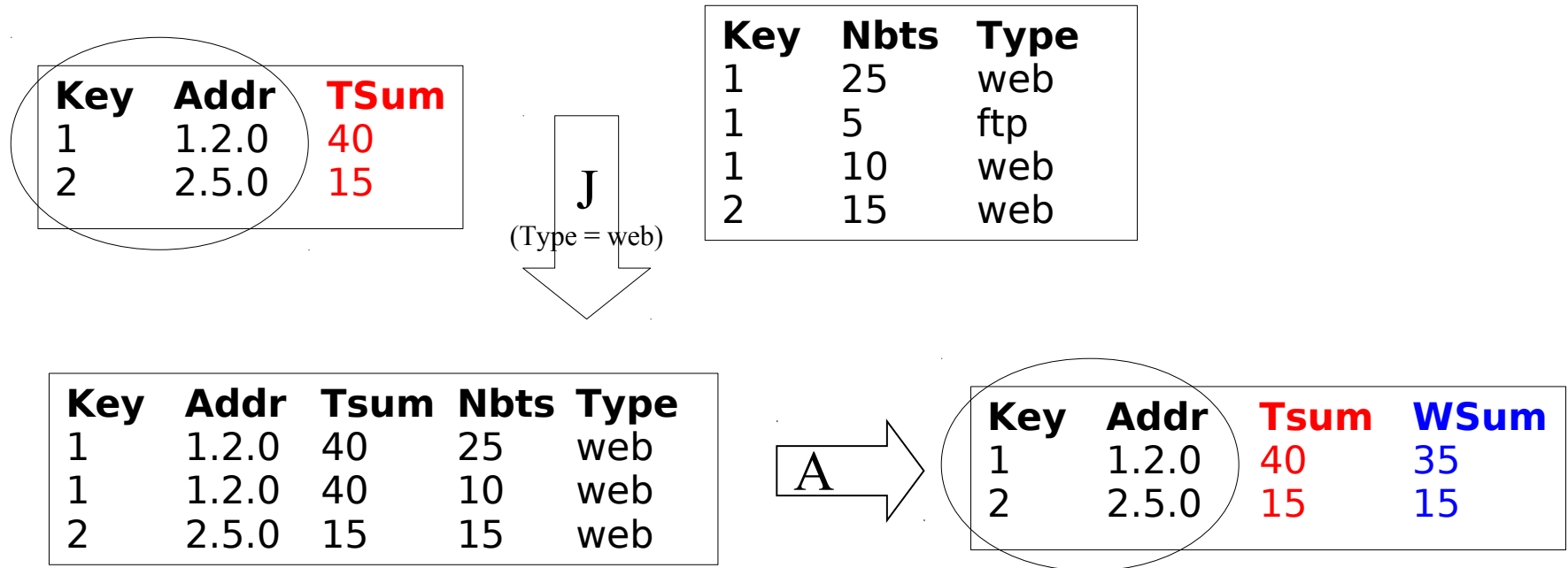
Network Usage (Relational)

- **Network usage:** For each IP address, what fraction of the total traffic is due to web flows?
- Step 1: Determine **total traffic**



Network Usage (Relational)/2

- Step 2: Determine **web traffic**

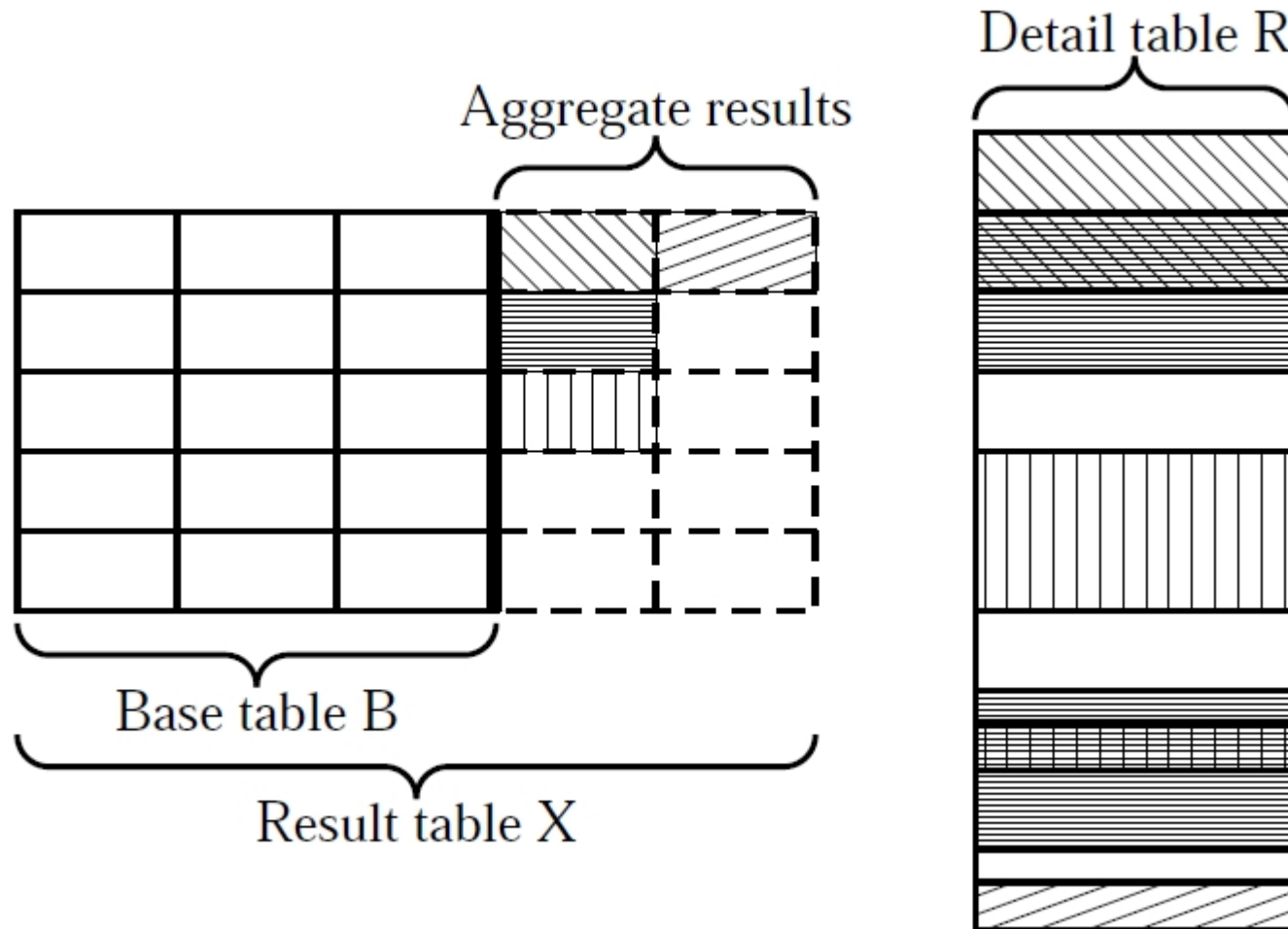


- OLAP extension provide some support, but some queries are still difficult to express

The Generalized MD-join

- $MD(B, R, (l_1, \dots, l_m), (\Phi_1, \dots, \Phi_m))$
 - B is the base table (the “groups”)
 - R is the detail table (fact data)
 - l_1 to l_m are list of aggregate functions
 - Φ_1 to Φ_m are possibly complex conditions over B and R describing what data is to be aggregated for each l_i respectively
- Result: The base table B **extended** with the aggregates in l_1 to l_m
- Salient feature: decouples grouping from aggregation (SQL does not do this).

The Generalized MD-join/2



Network Usage (GMD-join)

- **Network usage:** For each IP address, what fraction of the total number of flows is due to web traffic?
- Step 1: total traffic
 $X1 = MD(IP, Flow, (SUM(*):Cnt1), (IP.key=Flow.key))$
 - Schema of X1: (Key, Addr, Cnt1)
- Step 2: web traffic
 $X2 = MD(X1, Flow, (SUM(*):Cnt2), (X1.key=Flow.key \text{ and } Flow.Source='web'))$
 - Schema of X2: (Key, Addr, Cnt1, Cnt2)
- Sequences of GMDJs instead of multiple aggregate-join expressions

Network Usage (GMD-join)/2

- GMD-join can do it in one step

MD (IP, Flow, ((SUM(NBts)),(SUM(NBts))), (Φ_1 , Φ_2))

Φ_1 : (IP.key = Flow.key)

Φ_2 : (IP.key = Flow.key and Flow.Type = web)

- Equivalent RA expression (left outer join L0J):

π [key,IP,S1,SUM(NB)](
 (π [key,IP,SUM(NB)/S1](IP L0J[Φ_1] Flow)) L0J[Φ_2] Flow)

Φ_1 : (IP.key = Flow.key)

Φ_2 : (IP.key = Flow.key and Flow.Type = web)

Network Usage (GMD-join)/3

- Execution of GMD-join

$MD(IP, Flow, ((SUM(NBts)), (SUM(NBts))), (\Phi_1, \Phi_2))$

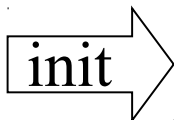
$\Phi_1 : (IP.key = Flow.key)$

$\Phi_2 : (IP.key = Flow.key \text{ and } Flow.Type = \text{web})$

- Step 1: Init base-result structure

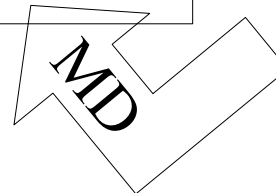
IP

Key	Addr
1	1.2.0
2	2.5.0



Base-Result Structure

Key	Addr	Tsum	WSum
1	1.2.0	0	0
2	2.5.0	0	0



Flow

Key	Nbts	Type
1	25	web
1	5	ftp
1	10	web
2	15	web

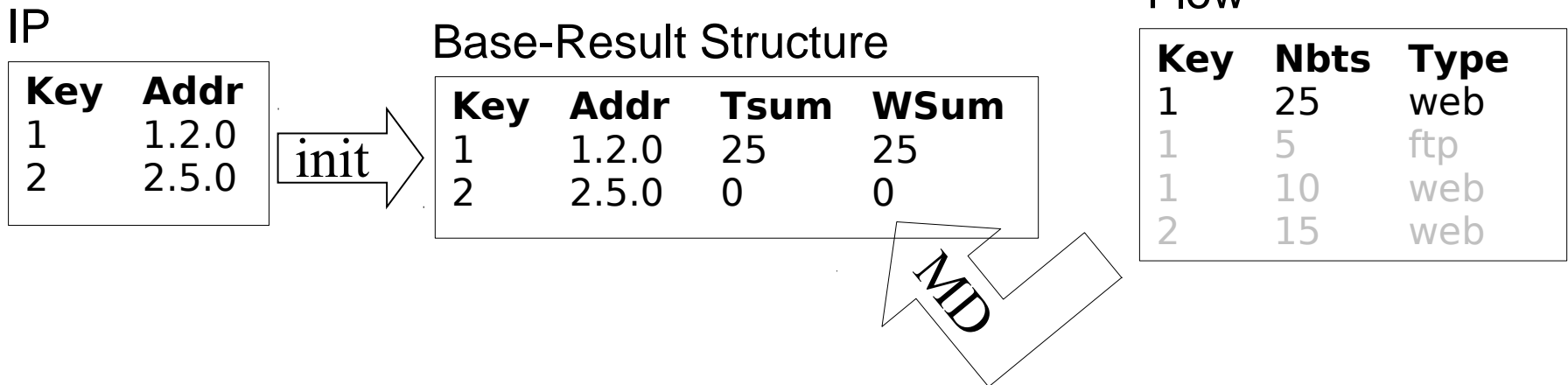
Network Usage (GMD-join)/4

MD(IP, Flow, ((SUM(NBts)),(SUM(NBts))), (Φ_1 , Φ_2))

Φ_1 : (IP.key = Flow.key)

Φ_2 : (IP.key = Flow.key and Flow.Type = web)

- Step 2: Scan result tuples and update base-result structure

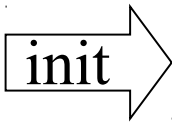


Network Usage (GMD-join)/5

- Step 2 (contd.)

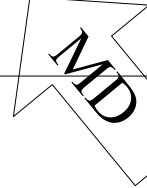
IP

Key	Addr
1	1.2.0
2	2.5.0



Base-Result Structure

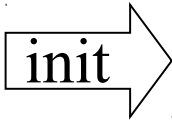
Key	Addr	Tsum	WSum
1	1.2.0	30	25
2	2.5.0	0	0



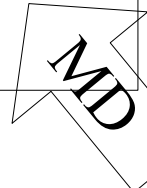
Flow

Key	Nbts	Type
1	25	web
1	5	ftp
1	10	web
2	15	web

Key	Addr
1	1.2.0
2	2.5.0

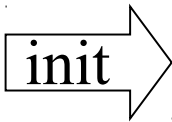


Key	Addr	Tsum	WSum
1	1.2.0	40	35
2	2.5.0	0	0



Key	Nbts	Type
1	25	web
1	5	ftp
1	10	web
2	15	web

Key	Addr
1	1.2.0
2	2.5.0



Key	Addr	Tsum	WSum
1	1.2.0	40	35
2	2.5.0	15	15



Key	Nbts	Type
1	25	web
1	5	ftp
1	10	web
2	15	web

Advantages of the GMD-join

- Flexible Operator - allows the “simple” expression of very complex queries
 - Allows the expression and evaluation of multiple complex aggregations using just a single operator and a **single pass** through the detail data
- Very efficient to evaluate
 - Indexing on the base-result structure constructed at query execution time performs very effectively
- Can be optimized effectively
 - GMD-joins allow efficient optimization schemes to be developed for complex aggregate queries

Complex Agg. Example/1

- **Example:** Assume the denormalized IP Flow schema:

```
Flow(RouterId, SourceIP, SourcePort, SourceAS, DestIP,  
     DestPort, DestAS, Protocol, StartTime, EndTime,  
     NumPackets, NumBytes, ...)
```

- Query: Determine the cumulative total, two hour moving average, and hourly total of network flow broken down by hour of the day.

Complex Agg. Example/2

- Base and detail table

Hours

HId	HStart	HEnd
1	0	59
2	60	119
3	120	179
...

Flows

SIP	DIP	NB	T
5	29	3	30
5	7	8	45
5	29	6	110
7	29	6	161
6	29	10	170

- Result

Result

HId	HStart	HEnd	CSum	MAvg	HSum
1	0	59	11	11/2	11
2	60	119	17	17/3	6
3	120	179	33	22/3	16

Complex Agg. Example/3

- GMD-join formulation
 - MD(Hours→B, Flows→R, (I_1, I_2, I_3), (Φ_1, Φ_2, Φ_3))
 - ◆ L_1 : (SUM(NB) → CSum)
 - ◆ Φ_1 : (R.T ≤ B.HEnd)
 - ◆ L_2 : (SUM(NB) → MSum, COUNT(*) → Mcnt)
 - ◆ Φ_2 : (R.T ≥ B.HStart-60 and R.T ≤ B.HEnd)
 - ◆ L_3 : (SUM(NB) → Hsum)
 - ◆ Φ_3 : (R.T ≥ B.HStart and R.T ≤ B.HEnd)

2D Cumulative Aggregates/1

- **Example:** Consider the following relation

Lineitem

	<i>Ordkey</i>	<i>Partkey</i>	<i>Suppkey</i>	<i>Quant</i>	<i>Price</i>	<i>Disc</i>	<i>Shipdate</i>
<i>r</i> ₁	O1	P1	S1	2	220	0.00	2008.01.23
<i>r</i> ₂	O2	P1	S1	4	440	0.05	2008.01.23
<i>r</i> ₃	O3	P2	S1	6	300	0.10	2008.01.23
<i>r</i> ₄	O4	P2	S2	7	420	0.10	2008.01.23
<i>r</i> ₅	O5	P2	S1	2	100	0.00	2008.01.24
<i>r</i> ₆	O6	P1	S2	3	240	0.05	2008.01.24
<i>r</i> ₇	O7	P2	S1	9	450	0.05	2008.01.24
<i>r</i> ₈	O8	P1	S2	8	640	0.10	2008.01.24

- Query: *Compute the number of orders per day and disc rate, the cumulative number of orders per day, and the cumulative number of orders per day and disc rate.*

2D Cumulative Aggregates/2

- Result

X

	<i>Shipdate</i>	<i>Disc</i>	<i>CntDD</i>	<i>CumCntD</i>	<i>CumCntDD</i>
x_1	2008.01.23	0.00	1	4	1
x_2	2008.01.23	0.05	1	4	2
x_3	2008.01.23	0.10	2	4	4
x_4	2008.01.24	0.00	1	8	2
x_5	2008.01.24	0.05	2	8	5
x_6	2008.01.24	0.10	1	8	8

2D Cumulative Aggregates/3

- Different types of aggregates
 - *CntDD*: groups are defined by identical values on the grouping attr. (GROUP BY)
 - *CumCntD*: 1D cumulative aggregate
 - ◆ Groups are contiguous rows (SQL window functions)
 - *CumCntDD*: 2D cumulative aggregate
 - ◆ Groups are specified along 2 dimensions
 - ◆ Ordering with contiguous rows might not exist
 - ◆ Difficult and inefficient in SQL

Aggregate value	Aggregation group
$x_1.CntDD$	$\{r_1\}$
$x_3.CntDD$	$\{r_3, r_4\}$
$x_1.CumCntD$	$\{r_1, r_2, r_3, r_4\}$
$x_4.CumCntD$	$\{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$
$x_3.CumCntDD$	$\{r_1, r_2, r_3, r_4\}$
$x_5.CumCntDD$	$\{r_1, r_2, r_5, r_6, r_7\}$

2D Cumulative Aggregates/4

- GMD-join formulation
 - $MD(\pi[\text{Shipdate}, \text{Disc}] \text{Lineitem} \rightarrow B, \text{Lineitem} \rightarrow L, (L_1, L_2, L_3), (\Phi_1, \Phi_2, \Phi_3))$
 - ♦ $L_1: (\text{COUNT}(\text{Quantity}) \rightarrow \text{CntDD})$
 - ♦ $\Phi_1: (\text{L.Shipdate} = \text{B.Shipdate} \text{ and } \text{L.Disc} = \text{B.Disc})$
 - ♦ $L_2: (\text{COUNT}(\text{Quantity}) \rightarrow \text{CumCntD})$
 - ♦ $\Phi_2: (\text{L.Shipdate} \leq \text{B.Shipdate})$
 - ♦ $L_3: (\text{COUNT}(\text{Quantity}) \rightarrow \text{CumCntDD})$
 - ♦ $\Phi_3: (\text{L.Shipdate} \leq \text{B.Shipdate} \text{ and } \text{L.Disc} \leq \text{B.Disc})$

Reduction to SQL/1

- The GMD-join $MD(B, R, (l_1, \dots, l_m), (\Phi_1, \dots, \Phi_m))$ can be expressed in SQL as

```
SELECT
  B1, ..., Bh
  f1,1(CASE WHEN  $\Phi_1$  THEN R.A1,1 ELSE N1,1 END) AS C1,1
  ...
  fm,k(CASE WHEN  $\Phi_m$  THEN R.Am,k ELSE Nm,k END) AS Cm,k
FROM
  B LEFT OUTER JOIN R ON  $\Phi_c$ 
GROUP BY
  B
```

- Φ_c is the common subpart of all Φ_i
 - ♦ Cartesian Product if Φ_c is empty

Reduction to SQL/2

```
SELECT      B.Shipdate, B.Disc,
            COUNT(CASE WHEN L.Shipdate = B.Shipdate
                          AND L.Disc = B.Disc)
              THEN Quantity
              ELSE 0 END) AS CntDD,
            COUNT(CASE WHEN L.Shipdate <= B.Shipdate)
              THEN Quantity
              ELSE 0 END) AS CumCntD,
            COUNT(CASE WHEN L.Shipdate <= B.Shipdate
                          AND L.Disc <= B.Disc)
              THEN Quantity
              ELSE 0 END) AS CumCntDD
FROM        (SELECT DISTINCT Shipdate, Disc FROM Lineitem) B,
            Lineitem L
GROUP BY    B.Shipdate, B.Disc;
```

Reduction to SQL/3

- Further ad hoc optimizations are possible

```
WITH
```

```
q1 AS (
```

```
    SELECT    Shipdate, Disc,  
              COUNT(Quantity) AS CntDD
```

```
    FROM Lineitem
```

```
    GROUP BY Shipdate, Disc
```

```
),
```

```
q2 AS (
```

```
    SELECT    Shipdate,  
              SUM(COUNT(Quantity)) OVER  
                (ORDER BY Shipdate ROWS UNBOUNDED PRECEDING)  
              AS CumCntD
```

```
    FROM Lineitem
```

```
    GROUP BY Shipdate, Disc
```

```
),
```

Reduction to SQL/4

(contd.)

```
q3 AS (  
  SELECT    B.Shipdate, B.Disc,  
            COUNT(Quantity) AS CumCntDD  
  FROM (SELECT DISTINCT Shipdate, Disc FROM Lineitem) B  
  LEFT OUTER JOIN  
    (SELECT DISTINCT Shipdate, Disc FROM Lineitem) L  
  ON L.Shipdate <= B.Shipdate AND L.Disc <= B.Disc  
  GROUP BY Shipdate, Disc  
)  
SELECT * FROM q1 NATURAL JOIN q2 NATURAL JOIN q3;
```

Outline

- The Generalized Multi-dimensional Join
- **An Algorithm for the GMD-join**
- Evaluating Subqueries in an OLAP Context
- Distributed Data Warehousing
- Conclusion

Evaluating the GMD-join

ComputeGMDJ_Basic: MD($B, R, (l_1, \dots, l_m), (\Phi_1, \dots, \Phi_m)$)

```
X := B extended with a column per aggregate
      initialize to NULL for MAX and MIN
      initialize to 0 for SUM and COUNT
for all r in R do
  for all x in X do
    for all  $\Phi_i$  in {  $\Phi_1, \dots, \Phi_m$  } do
      if  $\Phi_i(x, r)$  then update aggregates  $l_i$  of x
    end
  end
end
end
```

Hash Index for the GMD-join

ComputeGMDJ_Index: MD(B , R , (l_1, \dots, l_m) , (Φ_1, \dots, Φ_m))

$X := B$ extended with a column per aggregate
initialize to NULL for MAX and MIN
initialize to 0 for SUM and COUNT

construct hash index for X

for all r in R do

for all Φ_i in $\{ \Phi_1, \dots, \Phi_m \}$ do

use index to find $X_i = \{x \mid x \text{ in } X, \Phi_i(x, r)\}$

for all x in X_i do

update aggregates l_i of x

end

end

end

Incremental Aggregation

- The GMDJ **incrementally** computes the aggregate values in the result table.
- Different types of aggregate functions (AF) need to be considered
- **Distributive AF**
 - The result is the aggregation of partial results
 - $F(A) = G\{F(A_1), \dots, F(A_k)\}$
 - Example: MIN, MAX, SUM, COUNT
 - ♦ e.g., $\text{COUNT}(A) = \text{SUM}\{\text{COUNT}(A_1), \dots, \text{COUNT}(A_k)\}$

Incremental Aggregation/2

- **Algebraic AF**

- The computation can be reduced to a fixed number of distributive sub-aggregates
- $F(A) = H\{F_1(A), \dots, F_m(A)\}$
- Example: $AVG(A) = DIV(SUM(A), COUNT(A))$

- **Holistic AF**

- Cannot be reduced to a fixed number of sub-aggregates
- $F(A) = F\{A_1 \cup \dots \cup A_k\}$
- Example: MEDIAN, MODE, RANK

Evaluation of Algebraic AF

26

ComputeGMDJ_Algebraic: MD(B, R, l, Φ)

$l' = l;$

for each algebraic F_i in l do

 Replace F_i with sub-aggregates F_{i1}, \dots, F_{im}

end

$X = \text{ComputeGMDJ_Basic}(B, R, l', \Phi)$

for each row x in X do

 for each algebraic F_i in l do

 Replace in x columns F_{i1}, \dots, F_{im} by

 a single column $H(x.F_{i1}, \dots, x.F_{im})$

 end

end

Example

- For each combination of source (S) and destination (D) systems, compute the total number of flows and the number of flows whose NumBytes (NB) value exceeds the average value of NumBytes
- Data relation F

S	D	NB
5	29	3
5	7	8
5	29	6
7	29	4
7	29	6
6	29	10

Example/2

- $B1 = MD(\pi[S,D]F \rightarrow B0, F \rightarrow R, I_1, \Phi_1)$
 - $I_1 = (\text{avg}(NB) \rightarrow \text{avg})$
 - $\Phi_1 = (R.S = B0.S \text{ and } R.D = B0.D)$
- I_1 contains the algebraic aggregate AVG and is therefore translated into
 - $I_1 = (\text{count}(\ast) \rightarrow \text{cnt1}, \text{sum}(NB) \rightarrow \text{sum1})$

S	D	cnt1	sum1
5	29	2	9
5	7	1	8
7	29	2	10
6	29	1	10

Example/3

- $MD(B1, F \rightarrow R, I_2, \Phi_2)$

$$I_2 = (\text{count}(\ast) \rightarrow \text{cnt2})$$

$$\Phi_2 = (R.S = B1.S \text{ and } R.D = B1.D \text{ and } R.NB > B1.\text{sum1}/B1.\text{cnt1})$$

S	D	cnt1	sum1	cnt2
5	29	2	9	1
5	7	1	8	0
7	29	2	10	1
6	29	1	10	0

Advanced Memory Management

- Efficiency decreases if X does not fit in memory
- Solutions
 - **Partitioning** the result table X
 - ◆ Compute the GMDJ for each partition $\rightarrow X_1, X_2, \dots$
 - ◆ Take the union of the results X_1, X_2, \dots
 - **Completed result tuples**
 - ◆ During the evaluation, some parts of X might become obsolete, i.e., they will not be affected by not yet processed detail tuples
 - ◆ Such tuples in X can be written to the output and removed from memory

Advanced Memory Management/2

- **Example:** Compute the number of sales orders per day and discount rate
- $MD(\pi[\text{Shipdate}, \text{Disc}]\text{Lineitem} \rightarrow B, \text{Lineitem} \rightarrow R, I_1, \Phi_1)$
 $I_1: (\text{COUNT}(\ast) \rightarrow \text{CntDD})$
 $\Phi_1: (\text{L.Shipdate} = \text{B.Shipdate} \text{ and } \text{L.Disc} = \text{B.Disc})$
- If the tuples in the Lineitem relation are sorted first on Shipdate and then on Disc, tuple completion can be applied.

Advanced Memory Management/3

- Only tuples in boldface are kept in memory

X

	<i>Shipdate</i>	<i>Disc</i>	<i>CntDD</i>
x_1	2008.01.23	0.00	0
x_2	2008.01.23	0.05	0
x_3	2008.01.23	0.10	0
x_4	2008.01.24	0.00	0
x_5	2008.01.24	0.05	0
x_6	2008.01.24	0.10	0

	<i>Shipdate</i>	<i>Disc</i>	<i>CntDD</i>
x_1	2008.01.23	0.00	1
x_2	2008.01.23	0.05	0
x_3	2008.01.23	0.10	0
x_4	2008.01.24	0.00	0
x_5	2008.01.24	0.05	0
x_6	2008.01.24	0.10	0

	<i>Shipdate</i>	<i>Disc</i>	<i>CntDD</i>
x_1	2008.01.23	0.00	1
x_2	2008.01.23	0.05	2
x_3	2008.01.23	0.10	0
x_4	2008.01.24	0.00	0
x_5	2008.01.24	0.05	0
x_6	2008.01.24	0.10	0

Lineitem

	<i>Disc</i>	<i>Shipdate</i>
r_1	0.00	2008.01.23
r_2	0.05	2008.01.23
r_3	0.10	2008.01.23
r_4	0.10	2008.01.23
r_5	0.00	2008.01.24
r_6	0.05	2008.01.24
r_7	0.05	2008.01.24
r_8	0.10	2008.01.24

	<i>Disc</i>	<i>Shipdate</i>
r_1	0.00	2008.01.23
r_2	0.05	2008.01.23
r_3	0.10	2008.01.23
r_4	0.10	2008.01.23
r_5	0.00	2008.01.24
r_6	0.05	2008.01.24
r_7	0.05	2008.01.24
r_8	0.10	2008.01.24

	<i>Disc</i>	<i>Shipdate</i>
r_1	0.00	2008.01.23
r_2	0.05	2008.01.23
r_3	0.10	2008.01.23
r_4	0.10	2008.01.23
r_5	0.00	2008.01.24
r_6	0.05	2008.01.24
r_7	0.05	2008.01.24
r_8	0.10	2008.01.24

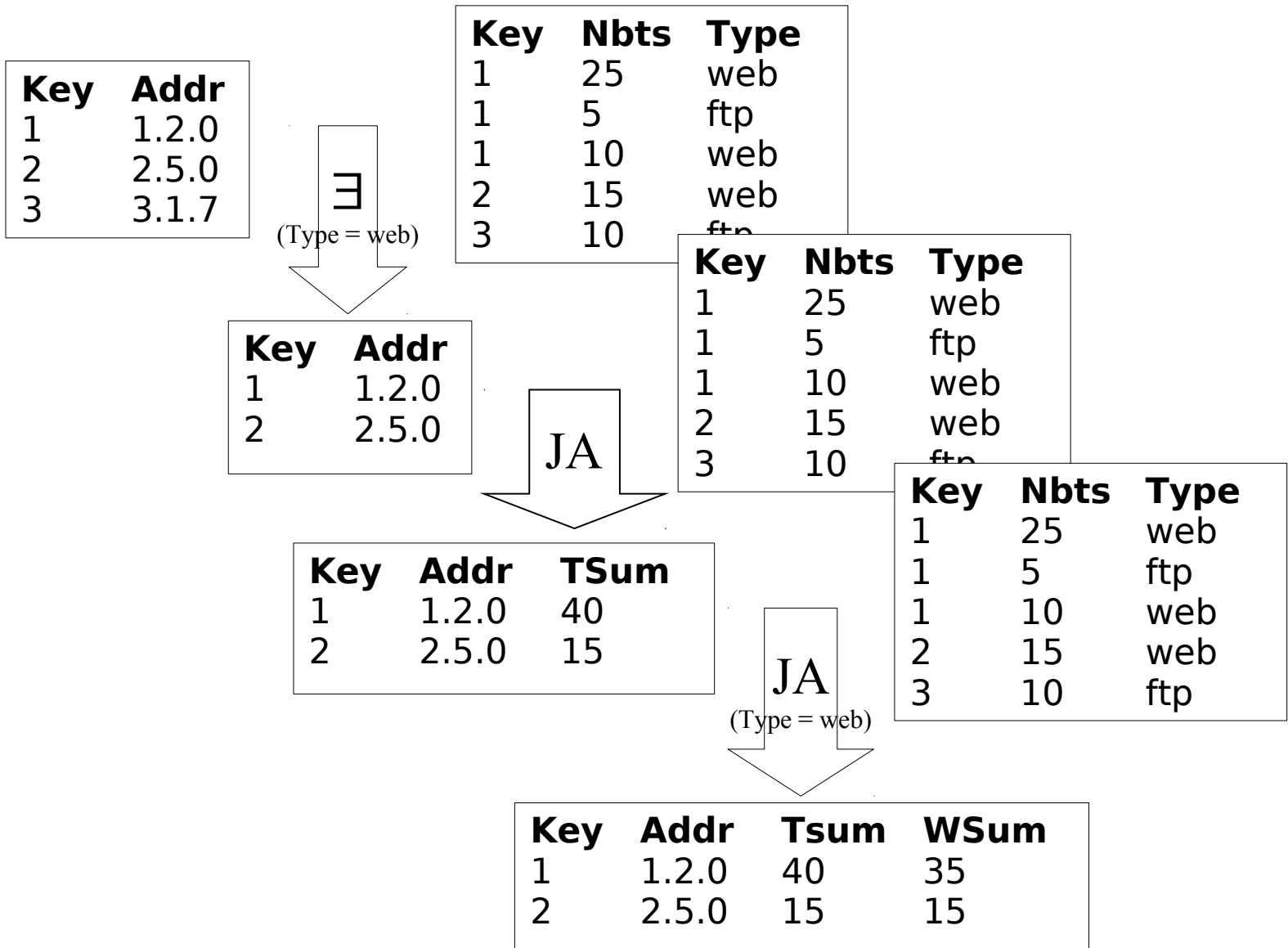
Outline

- The Generalized Multi-dimensional Join
- An Algorithm for the GMDJ
- Evaluating Subqueries in an OLAP Context
- Distributed Data Warehousing
- Conclusion

Subqueries in Complex OLAP

- Common in Complex OLAP applications
- Example: For each IP address, **for which there exists web traffic**, what fraction of the total traffic is due to web flows
- State of the Art:
 - Complex, specialized algorithms
 - Convert to join/outer-join (e.g., [Dayal'87, Seshadri et al.'96] or union/difference operations (e.g., [Bækgaard & Mark'95])).
- Not always good solution for OLAP applications (See [Pirahesh et al.'92, Rao & Ross'98])

Network Usage (Subquery)

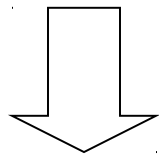


Solving Exists with GMD-joins

- For each IP address, **for which there exists web traffic**, what fraction of the total traffic is due to web flows
- Exists can be expressed as a GMD-join followed by a selection

$\pi[\text{Key}, \text{IP}] \sigma[\exists(\sigma[\Phi_3] \text{Flow})] \text{IP}$

$\Phi_3 : (\text{IP.key} = \text{Flow.key} \text{ and } \text{Flow.Type} = \text{web})$



$\sigma[\text{cnt} > 0] \text{MD}(\text{IP}, \text{Flow}, (\text{cnt}(*)), (\Phi_3))$

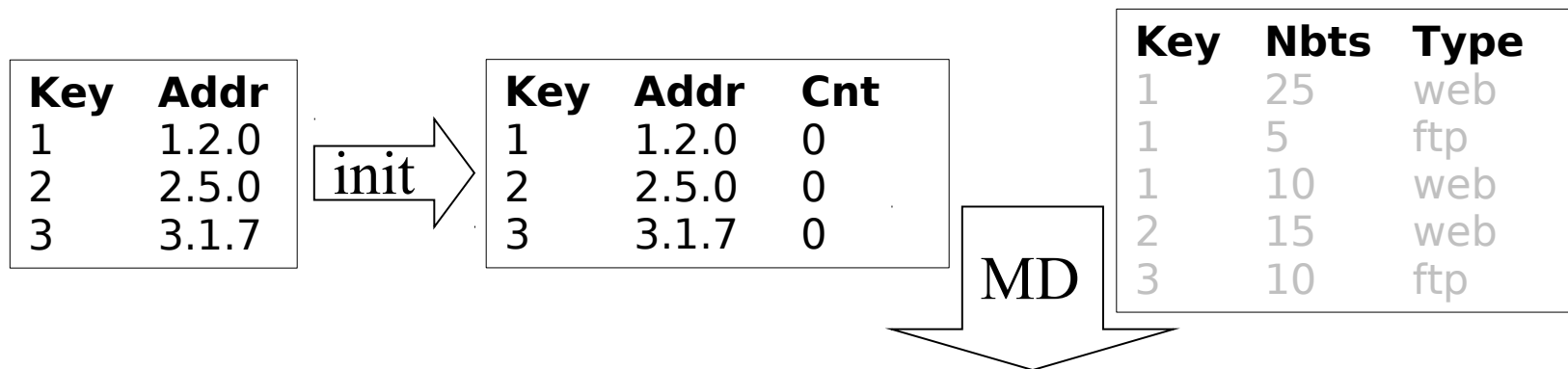
$\Phi_3 : (\text{IP.key} = \text{Flow.key} \text{ and } \text{Flow.Type} = \text{web})$

Solving Exists with GMD-joins/2

- Replace exists by a GMD-join followed by a selection

$\pi[\text{Key}, \text{IP}] \sigma[\text{cnt} > 0] \text{MD}(\text{IP}, \text{Flow}, (\text{cnt}(*)), (\Phi_3))$

$\Phi_3 : (\text{IP.key} = \text{Flow.key} \text{ and } \text{Flow.Type} = \text{web})$

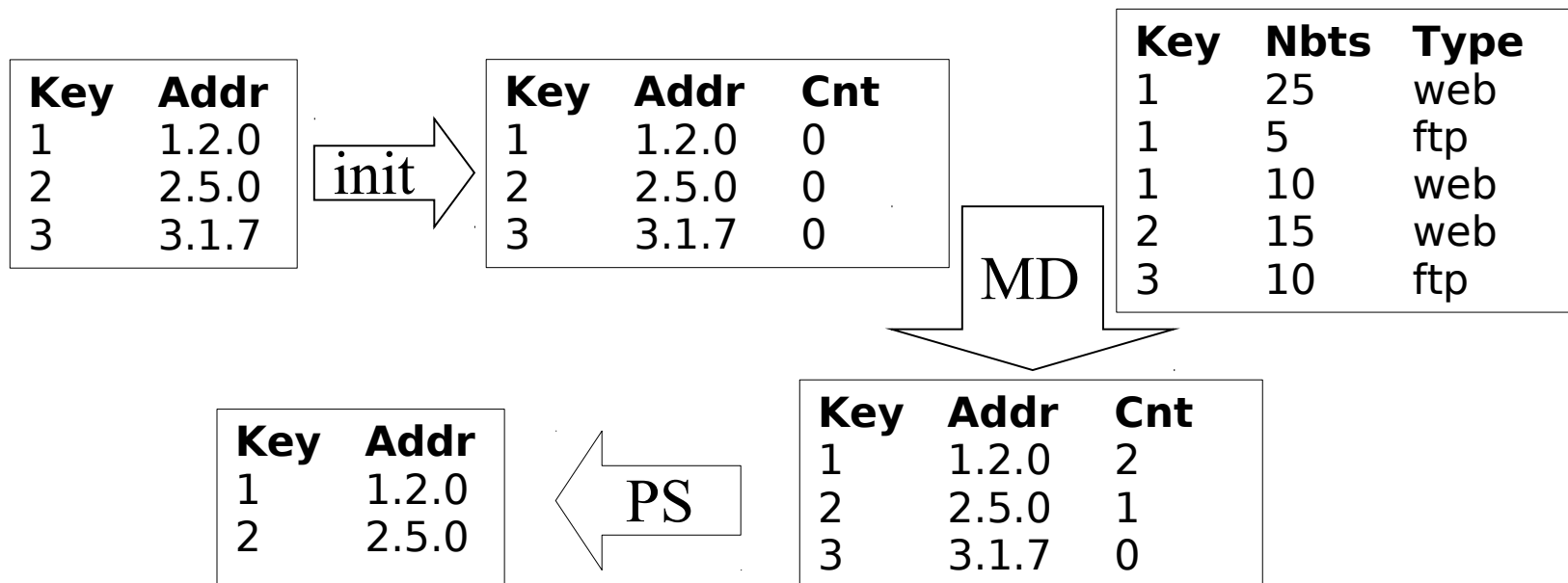


Solving Exists with GMD-joins/3

- For each IP address, **for which there exists web traffic**, what fraction of the total traffic is due to web flows?

$\pi[\text{Key}, \text{IP}] \sigma[\text{cnt} > 0] \text{MD}(\text{IP}, \text{Flow}, (\text{cnt}(*)), (\Phi_3))$

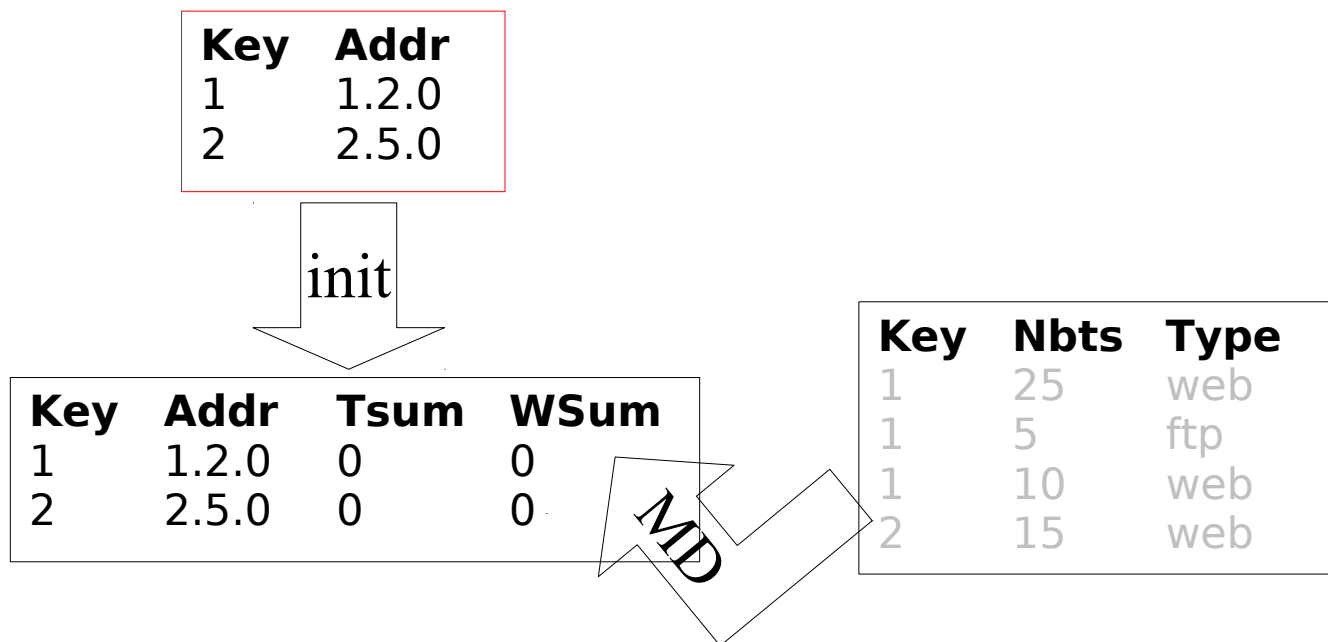
$\Phi_3 : (\text{IP}.key = \text{Flow}.key \text{ and } \text{Flow}.Type = \text{web})$



Solving Exists with GMD-joins/4

- Nested GMD-join
 - The inner GMD-join computes the exists, the outer computes the traffic.

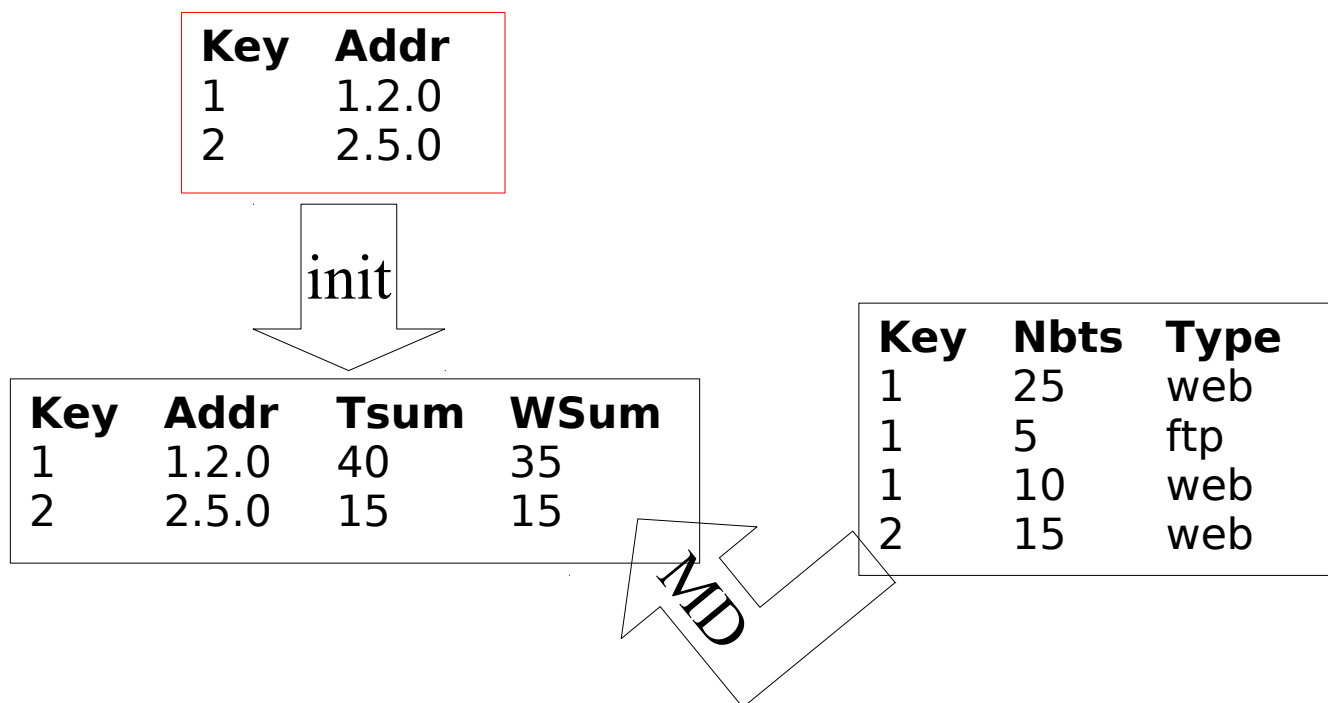
$MD(\pi[\text{Key}, \text{IP}] \sigma[\text{cnt} > 0] MD(\text{IP}, \text{Flow}, (\text{cnt}(*)), (\Phi_3)), \text{Flow}, ((\text{Sum}(\text{NBts}), (\text{Sum}(\text{NBts}))), (\Phi_1, \Phi_2)))$



Solving Exists with GMD-joins/5

- Nested GMD-join (contd.)

$MD(\pi[\text{Key}, \text{IP}] \sigma[\text{cnt} > 0] MD(\text{IP}, \text{Flow}, (\text{cnt}(*)), (\Phi_3)),$
 $\text{Flow}, ((\text{SUM}(\text{NBts}), (\text{SUM}(\text{NBts}))), (\Phi_1, \Phi_2)))$



Advantages of GMD-joins

- The GMD-join allows us to transform complex predicates into simple predicates over complex aggregations.
- The GMDJ interacts with other algebraic operators.
- Optimizations for the GMD-join can be exploited to improve the performance of algebraic expressions.

Transformation Rules

$$\pi[\mathbf{A}]\mathcal{G}^\theta(B, R, \vec{l}, \vec{\theta}) = \mathcal{G}^\theta(\pi[\mathbf{A}']B, R, \vec{l}, \vec{\theta})$$

$$\pi[\mathbf{A}]\mathcal{G}^\theta(B, R, \vec{l}, \vec{\theta}) = \pi[\mathbf{A}]\mathcal{G}^\theta(\pi[\mathbf{A}']B, R, \vec{l}, \vec{\theta})$$

$$\sigma[\theta_S]\mathcal{G}^\theta(B, R, \vec{l}, \vec{\theta}) = \mathcal{G}^\theta(\sigma[\theta_S]B, R, \vec{l}, \vec{\theta})$$

$$\mathcal{G}^\theta(B, R, \vec{l}, \vec{\theta}) = \mathcal{G}^\theta(B, \sigma[\theta_1^R \vee \dots \vee \theta_m^R]R, \vec{l}, \vec{\theta})$$

$$\sigma[>0]\mathcal{G}^\theta(B, R, \vec{l}, \vec{\theta}) = \sigma[>0]\mathcal{G}^\theta(\sigma[\theta_1^B \vee \dots \vee \theta_m^B]B, R, \vec{l}, \vec{\theta})$$

$$\mathcal{G}^\theta(\mathcal{G}^\theta(B, R_1, \vec{l}_1, \vec{\theta}_1), R_2, \vec{l}_2, \vec{\theta}_2) = \mathcal{G}^\theta(\mathcal{G}^\theta(B, R_2, \vec{l}_2, \vec{\theta}_2), R_1, \vec{l}_1, \vec{\theta}_1)$$

$$\mathcal{G}^\theta(\mathcal{G}^\theta(B, R, \vec{l}_1, \vec{\theta}_1), R, \vec{l}_2, \vec{\theta}_2) = \mathcal{G}^\theta(B, R, (\vec{l}_1, \vec{l}_2), (\vec{\theta}_1, \vec{\theta}_2))$$

$$\mathcal{G}^\theta(\mathcal{G}^\theta(B, R_1, \vec{l}_1, \vec{\theta}_1), R_2, \vec{l}_2, \vec{\theta}_2) = \mathcal{G}^\theta(B, R_1, \vec{l}_1, \vec{\theta}_1) \rightarrow U \bowtie_{\theta_B} \mathcal{G}^\theta(B, R_2, \vec{l}_2, \vec{\theta}_2) \rightarrow V$$

$$\mathcal{G}^\theta(B, R, \vec{l}, \vec{\theta}) = \mathcal{G}^\theta(B_1, R, \vec{l}, \vec{\theta}) \cup \dots \cup \mathcal{G}^\theta(B_n, R, \vec{l}, \vec{\theta})$$

$$attr(\vec{\theta}) \cap \mathbf{B} \subseteq \mathbf{A}, \mathbf{A}' = \mathbf{A} \setminus \mathbf{C} \quad (\text{E1})$$

$$\mathbf{A}' = (\mathbf{A} \cup (attr(\vec{\theta}) \cap \mathbf{B})) \setminus \mathbf{C} \quad (\text{E2})$$

$$attr(\theta_s) \subseteq \mathbf{B} \quad (\text{E3})$$

$$\theta_i = \theta'_i \wedge \theta_i^R, attr(\theta_i^R) \subseteq \mathbf{R} \quad (\text{E4})$$

$$\theta_i = \theta'_i \wedge \theta_i^B, attr(\theta_i^B) \subseteq \mathbf{B} \quad (\text{E5})$$

$$attr(\vec{\theta}_2) \subseteq (\mathbf{B} \cup \mathbf{R}_2) \quad (\text{E6})$$

$$attr(\vec{\theta}_2) \subseteq (\mathbf{B} \cup \mathbf{R}) \quad (\text{E7})$$

$$attr(\vec{\theta}_2) \subseteq (\mathbf{B} \cup \mathbf{R}_2), \quad (\text{E8})$$

$$\theta_B = (U.B = V.B)$$

$$B = B_1 \cup \dots \cup B_n, \quad (\text{E9})$$

$$B_i \cap B_j = \emptyset$$

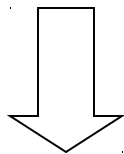
Commutation with Selections

$$\sigma[\Phi_S] \text{ MD } (B, R, (I_1, \dots, I_m), (\Phi_1, \dots, \Phi_m)) = \text{ MD } (\sigma[\Phi_S](B), R, (I_1, \dots, I_m), (\Phi_1, \dots, \Phi_m))$$

Rule

$$\text{ MD } ((\sigma[\text{cnt} > 0] \text{ MD } (\text{IP}, \text{Flow}, (\text{cnt}(*)), (\Phi_3))), \text{Flow}, ((\text{Sum}(\text{NBts}), (\text{Sum}(\text{NBts}))), (\Phi_1, \Phi_2)))$$

Example

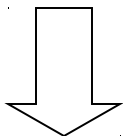


$$\sigma[\text{cnt} > 0] \text{ MD } (\text{ MD } (\text{IP}, \text{Flow}, (\text{cnt}(*)), (\Phi_3)), \text{Flow}, ((\text{Sum}(\text{NBts}), (\text{Sum}(\text{NBts}))), (\Phi_1, \Phi_2)))$$

Coalescing

$$\text{MD} (\text{MD}(\mathbf{B}, R_1, (\underline{l}_1), (\underline{\Phi}_1)), R_2, (\underline{l}_2), (\underline{\Phi}_2)) = \text{Rule}$$
$$\text{MD} (\mathbf{B}, R, (\underline{l}_1, \underline{l}_2), (\underline{\Phi}_1, \underline{\Phi}_2))$$

$$\sigma[\text{cnt}>0]\text{MD}(\text{MD} (\text{IP}, \text{Flow}, (\text{cnt}(*)), (\Phi_3)), \text{Example}$$
$$\text{Flow}, ((\text{Sum}(\text{NBts}), (\text{Sum}(\text{NBts}))), (\Phi_1, \Phi_2))$$



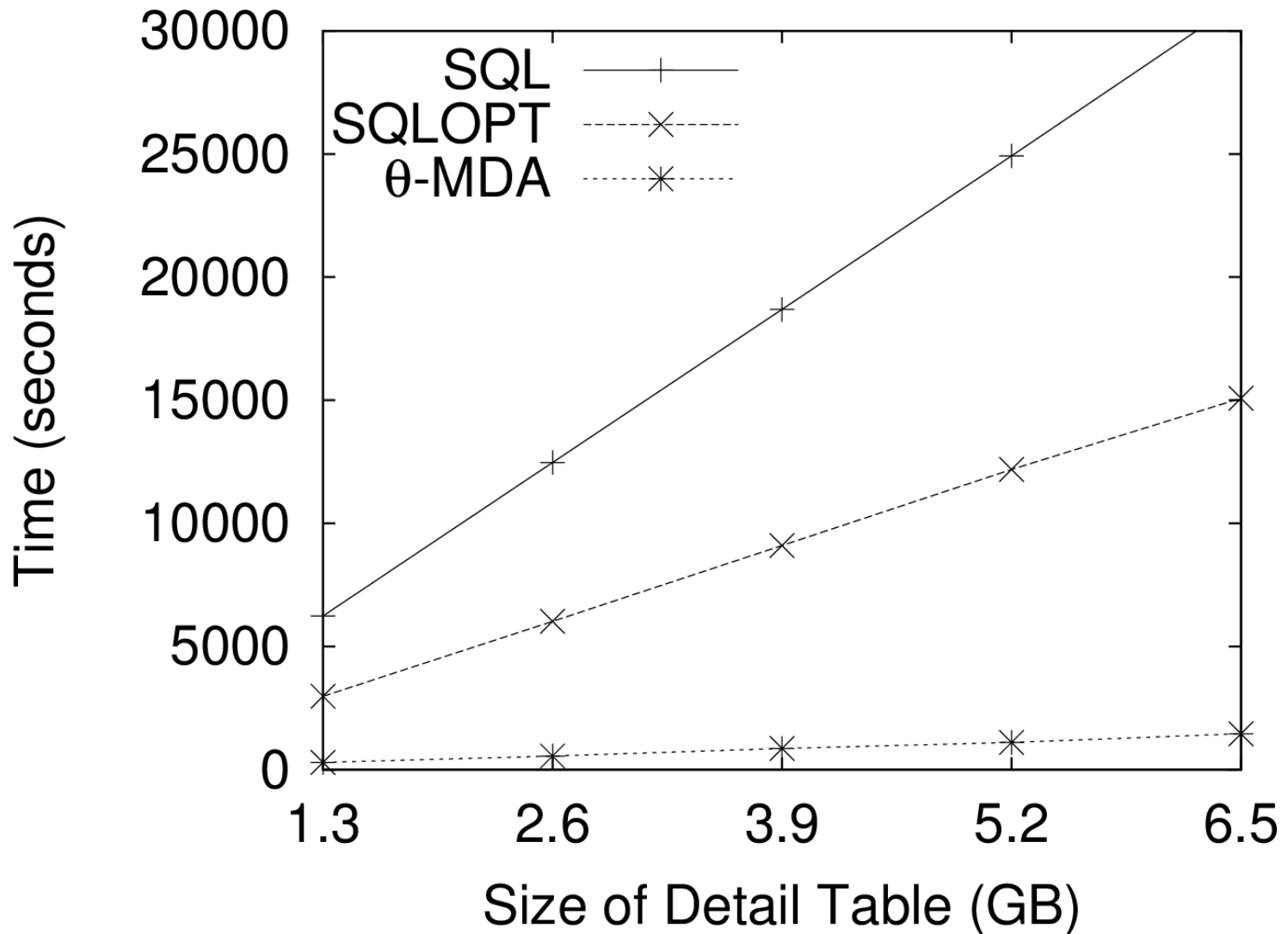
$$\sigma[\text{cnt}>0]\text{MD}(\text{IP}, \text{Flow}, ((\text{Sum}(\text{NBts}), (\text{Sum}(\text{NBts})), (\text{cnt}(*))), (\Phi_1, \Phi_2, \Phi_3))$$

$$\Phi_1 : (\text{IP.key} = \text{Flow.key})$$

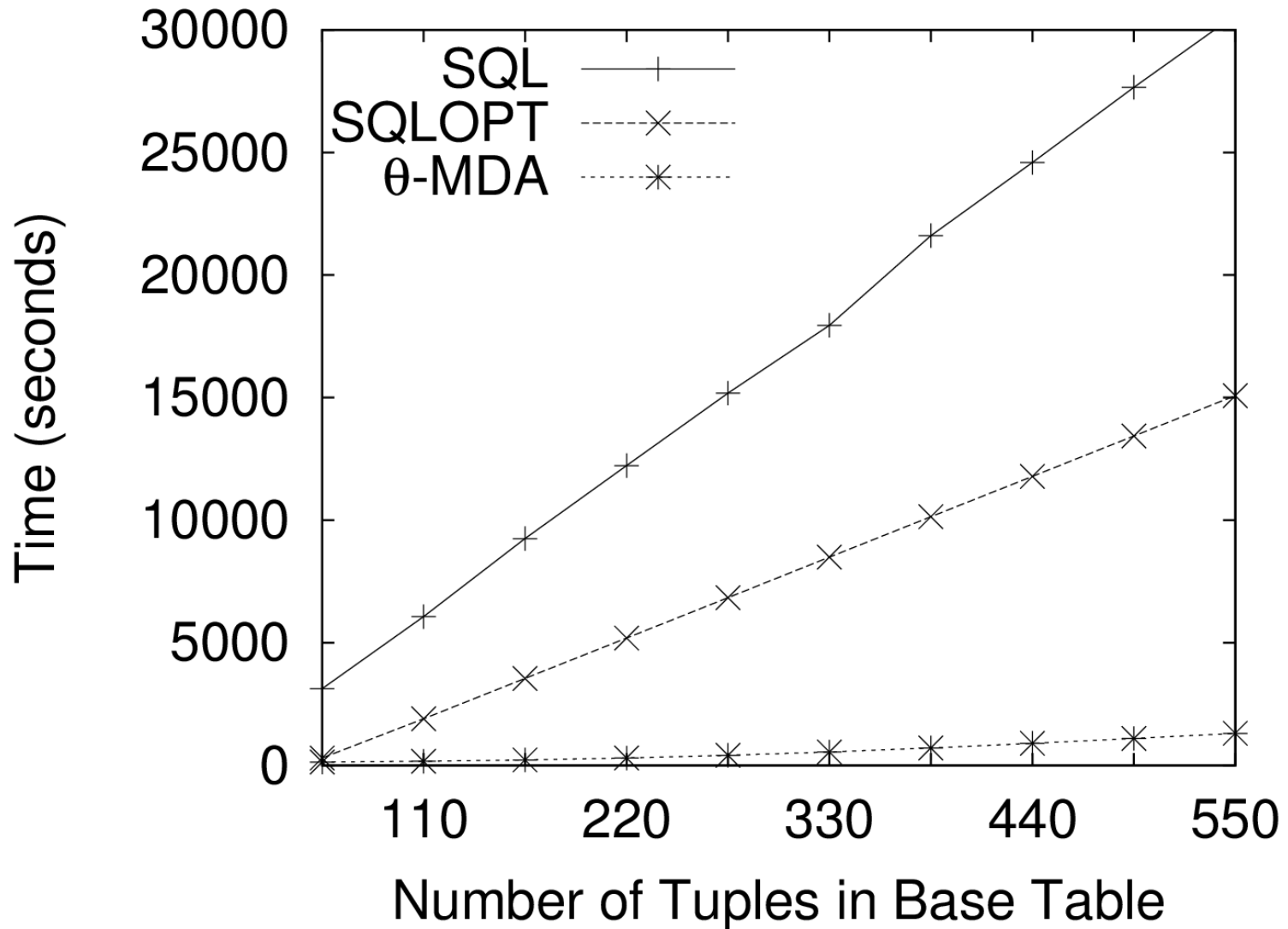
$$\Phi_2 : (\text{IP.key} = \text{Flow.key} \text{ and } \text{Flow.Type} = \text{web})$$

$$\Phi_3 : (\text{IP.key} = \text{Flow.key} \text{ and } \text{Flow.Type} = \text{web})$$

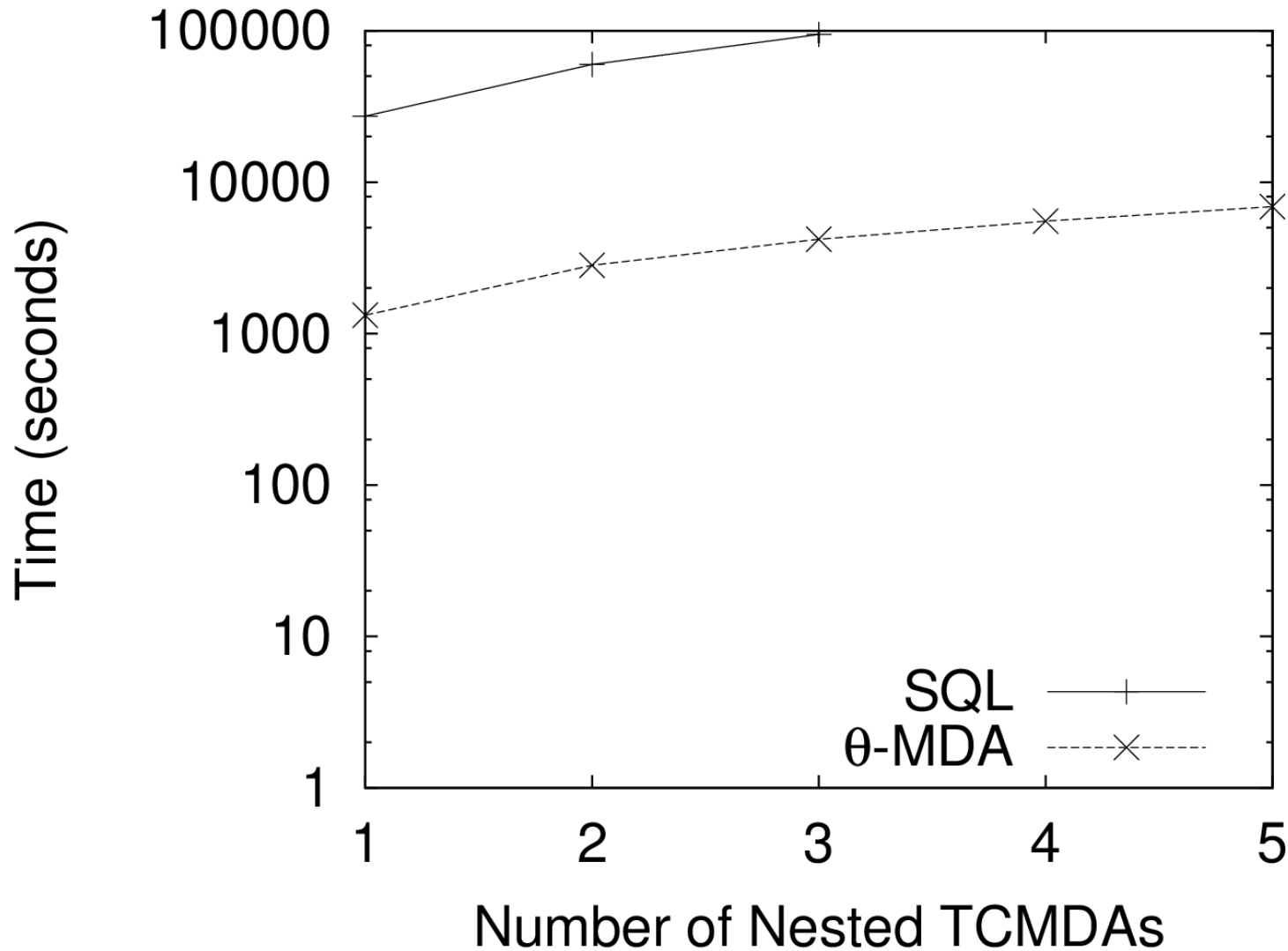
Experimental Evaluation



Experimental Evaluation/2



Experimental Evaluation/3



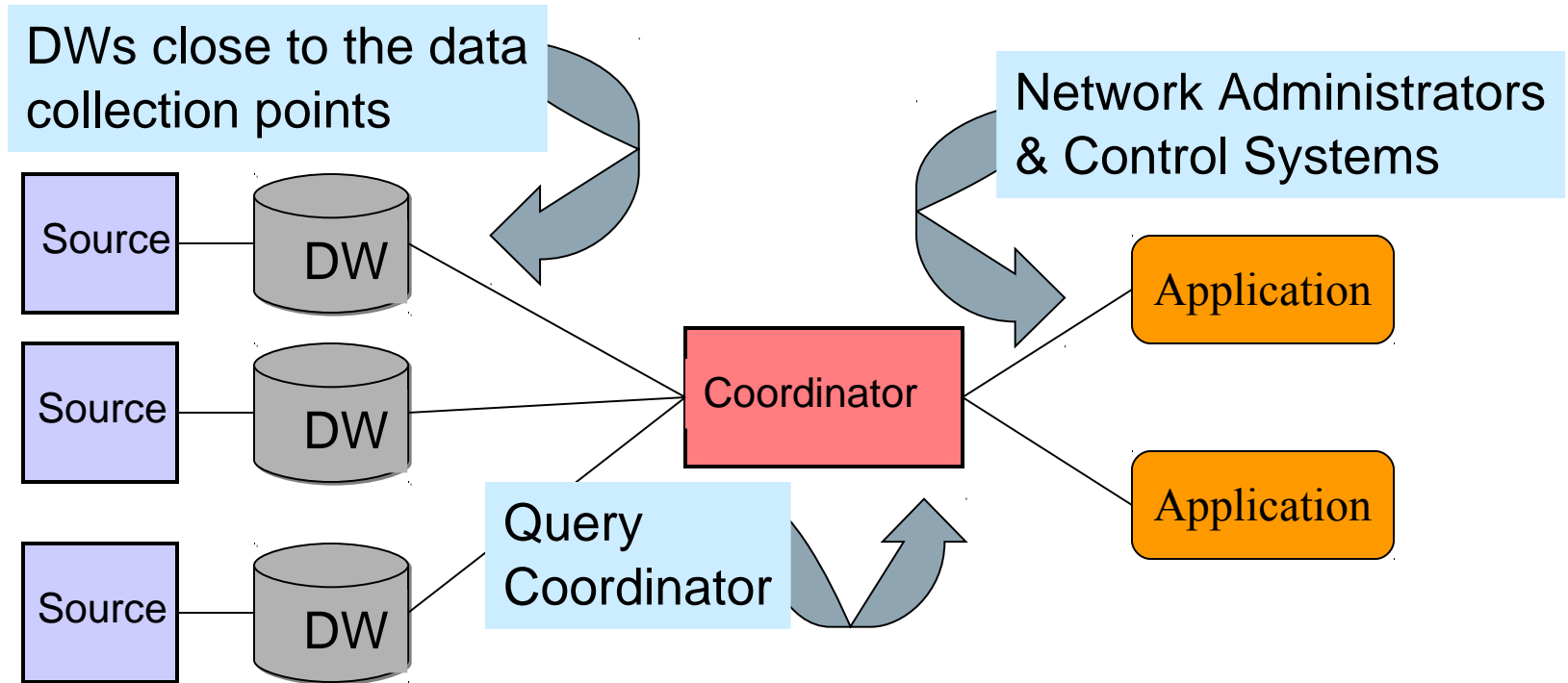
Advantages of GMD-joins

- The GMD-join allows us to transform complex predicates into simple predicates over complex aggregations
 - optimizations for the GMD-join can be exploited to improve the performance of the algebraic expression
 - GMD-join unnesting performs well, even for subquery types where “conventional” algorithms perform poorly
 - performs particularly well for ad-hoc queries
- Many specialized optimization algorithms can be generalized as algebraic transformations using GMD-joins

Outline

- The Generalized Multi-dimensional Join
- An Algorithm for the GMDJ
- Evaluating Subqueries in an OLAP Context
- **Distributed Data Warehousing**
- Conclusion

A Distributed Data Warehouse

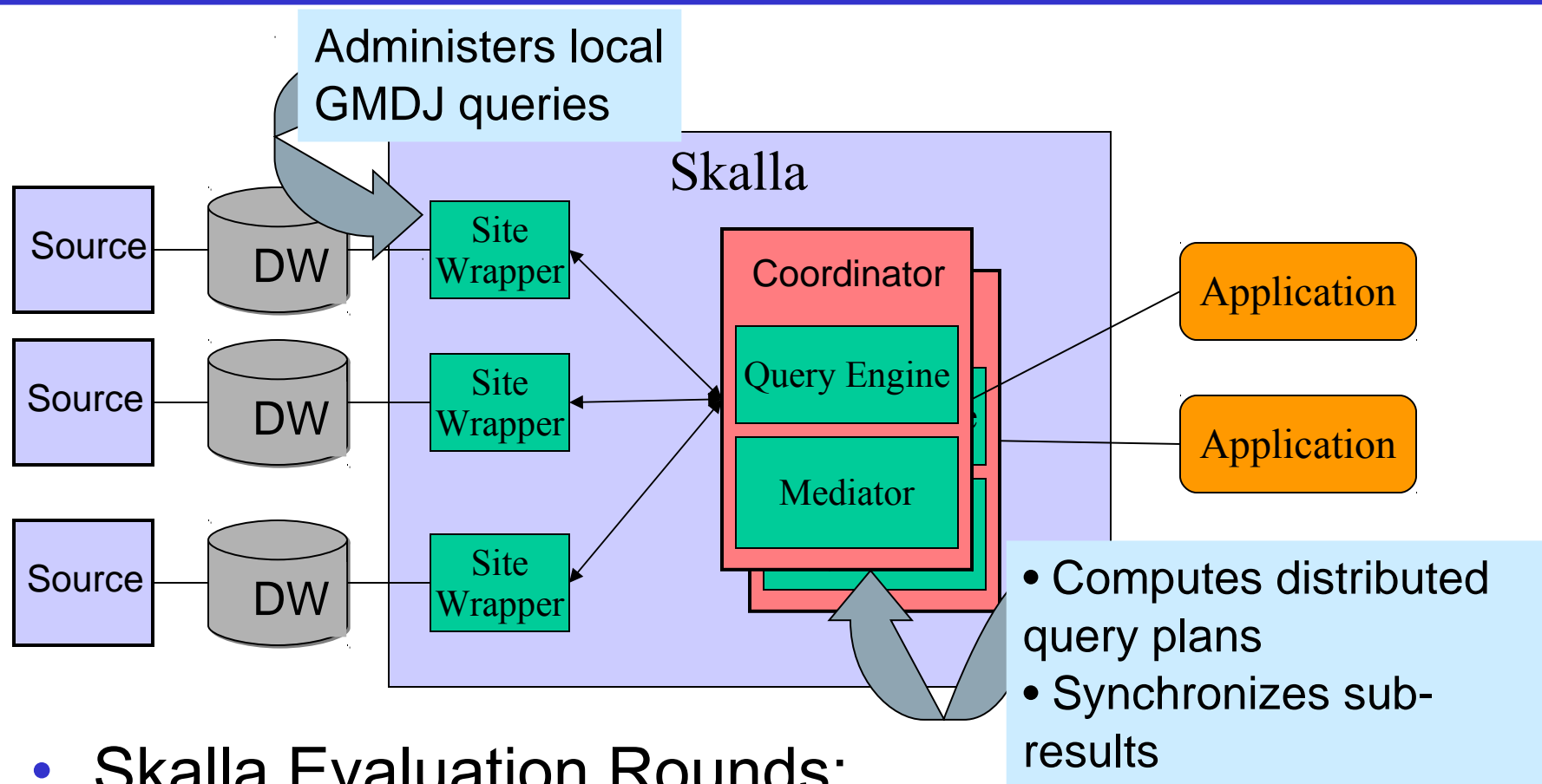


- Local DW at each collection point (e.g., router)
- Integration of data is infeasible.
- Compute queries across multiple DWs
- A technology is needed for *distributed processing* of complex OLAP queries

Skalla

- Translates OLAP queries expressed using an *algebra extended with GMD-joins*, into distributed query evaluation plans
- Salient Features:
 - Efficiently handles a significant variety of complex OLAP queries (incl. pivots, correlations, etc.)
 - Only partial results are shipped between the sites and the coordinator -- never subsets of the detail data
 - No site to site communication

Skalla Architecture & Evaluation



- **Skalla Evaluation Rounds:**

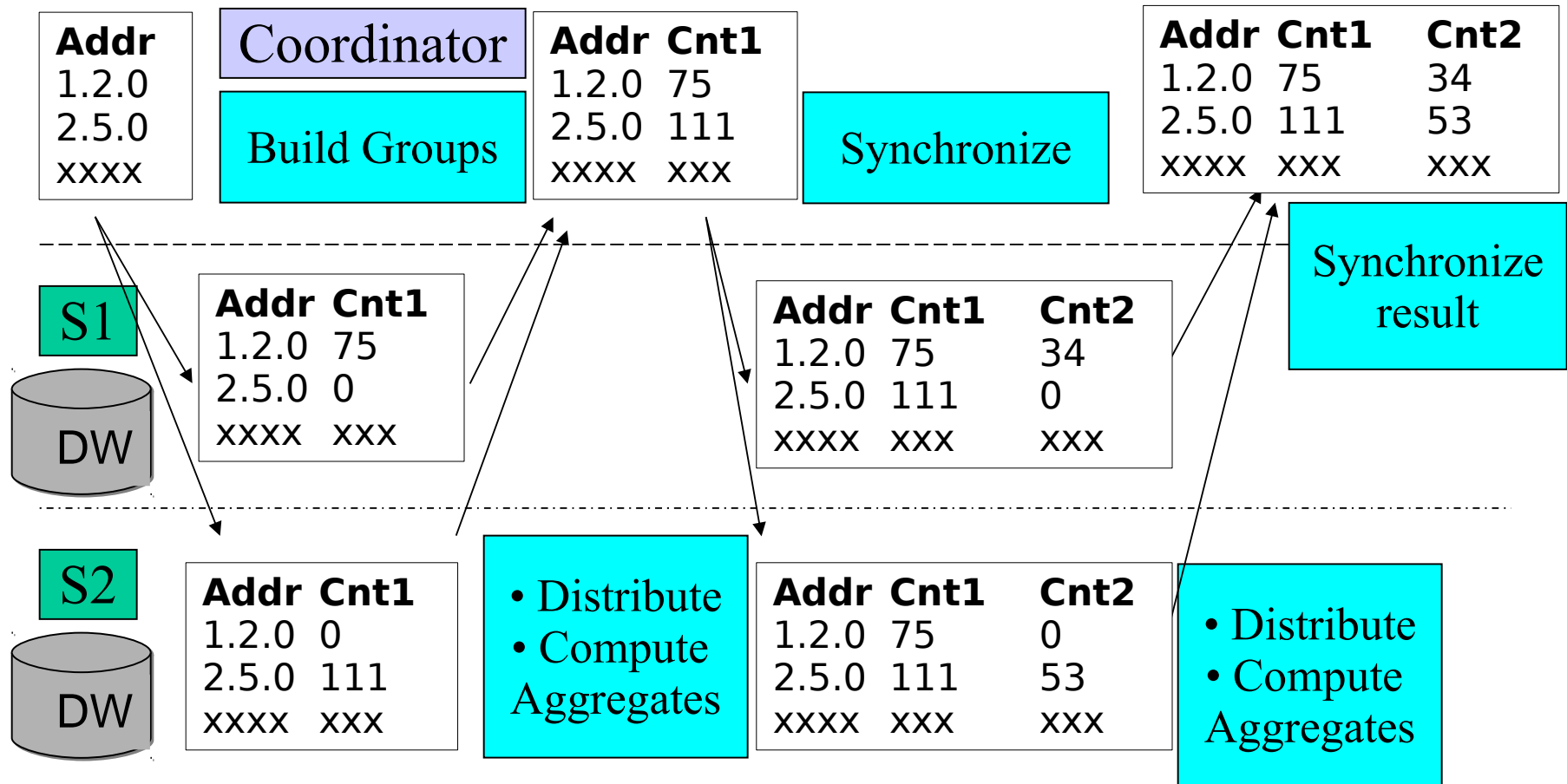
- Computation of GMDJ at the local DWs
- Synchronize sub-results at coordinator

Skalla Example

- Complex OLAP queries are sequences of GMD-joins
- For each IP address, what fraction of the total number of flows is due to web traffic?

```
MD( MD(IP, Flow, (cnt(*):cnt1), (IP.key = Flow.key)),  
    Flow,  
    (cnt(*):cnt2),  
    (IP.key = Flow.key and Flow.Source = web ))
```

Skalla Evaluation: Example



For each IP address, what fraction of the total number of flows is due to web traffic?

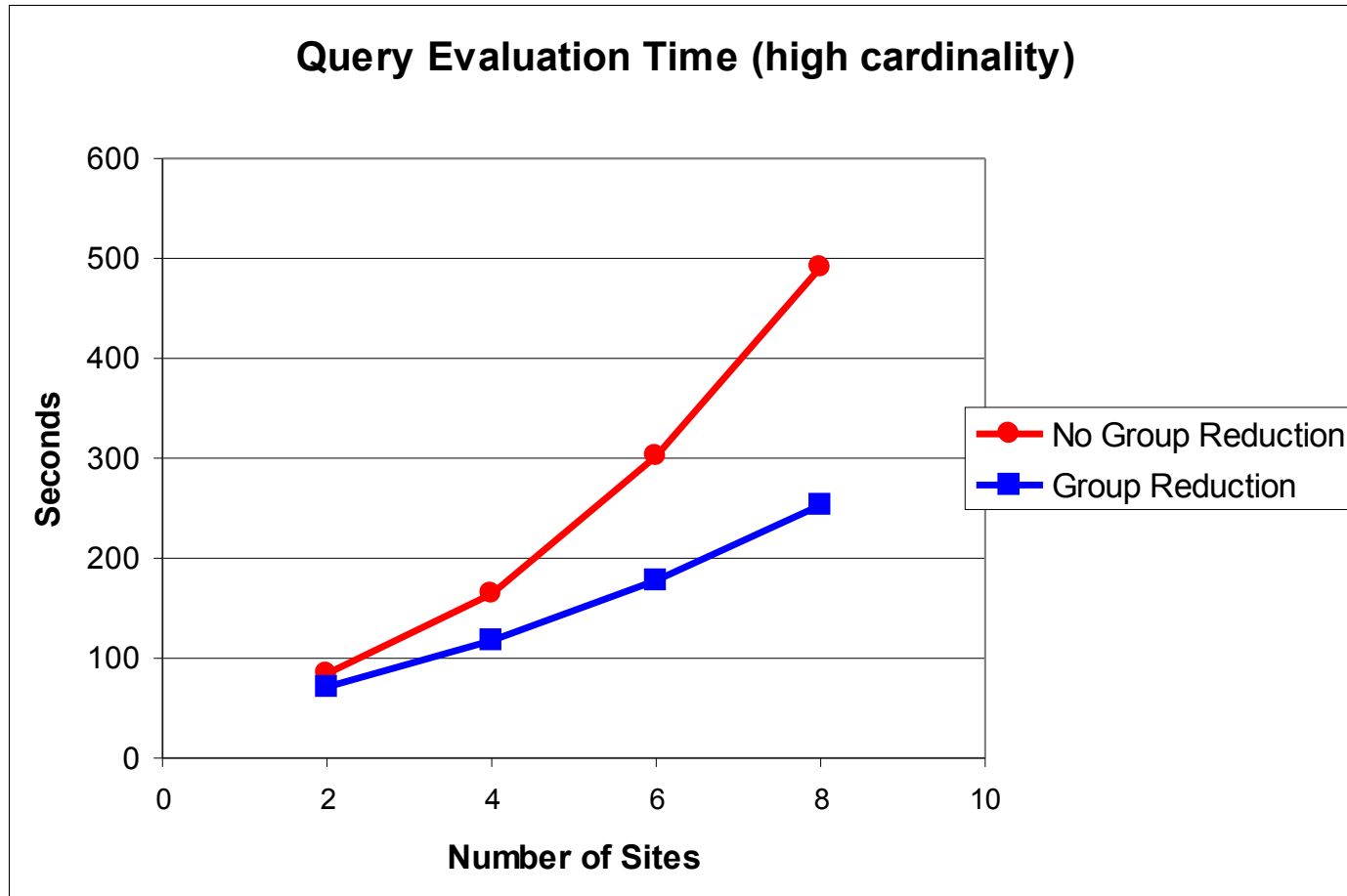
Skalla Evaluation: Features

- Each round of computation in the distributed query evaluation computes a single GMDJ.
- Features of the Evaluation:
 - Semantics of the query plans ensure that the amount of data shipped by the algorithm is dependent on the number of groups and aggregate functions and *independent of the size of the fact relation in the database!*
 - The algorithm permits for a wide variety of optimizations

Group Reduction

- During processing, we only ship data that has actually been changed
- Example:
 - Query: For each IP address, what fraction of the total number of flows is due to web traffic?
 - Each local DW receives a base-values table containing all source data
 - Coordinator has a copy of base-values table
 - Local DWs ship only those tuples back that have actually been changed

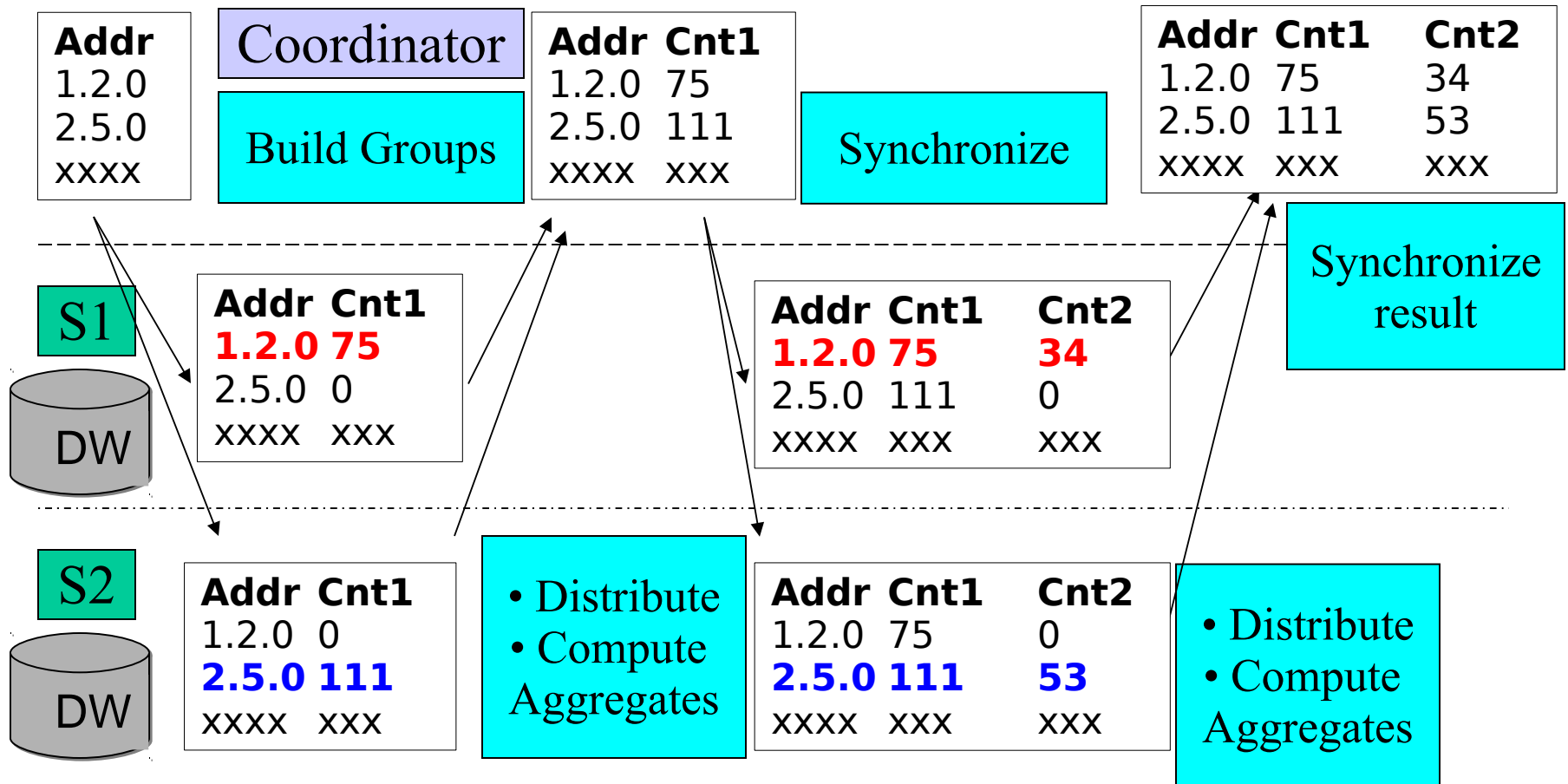
Number of Sites (GR)



Synchronization Reduction

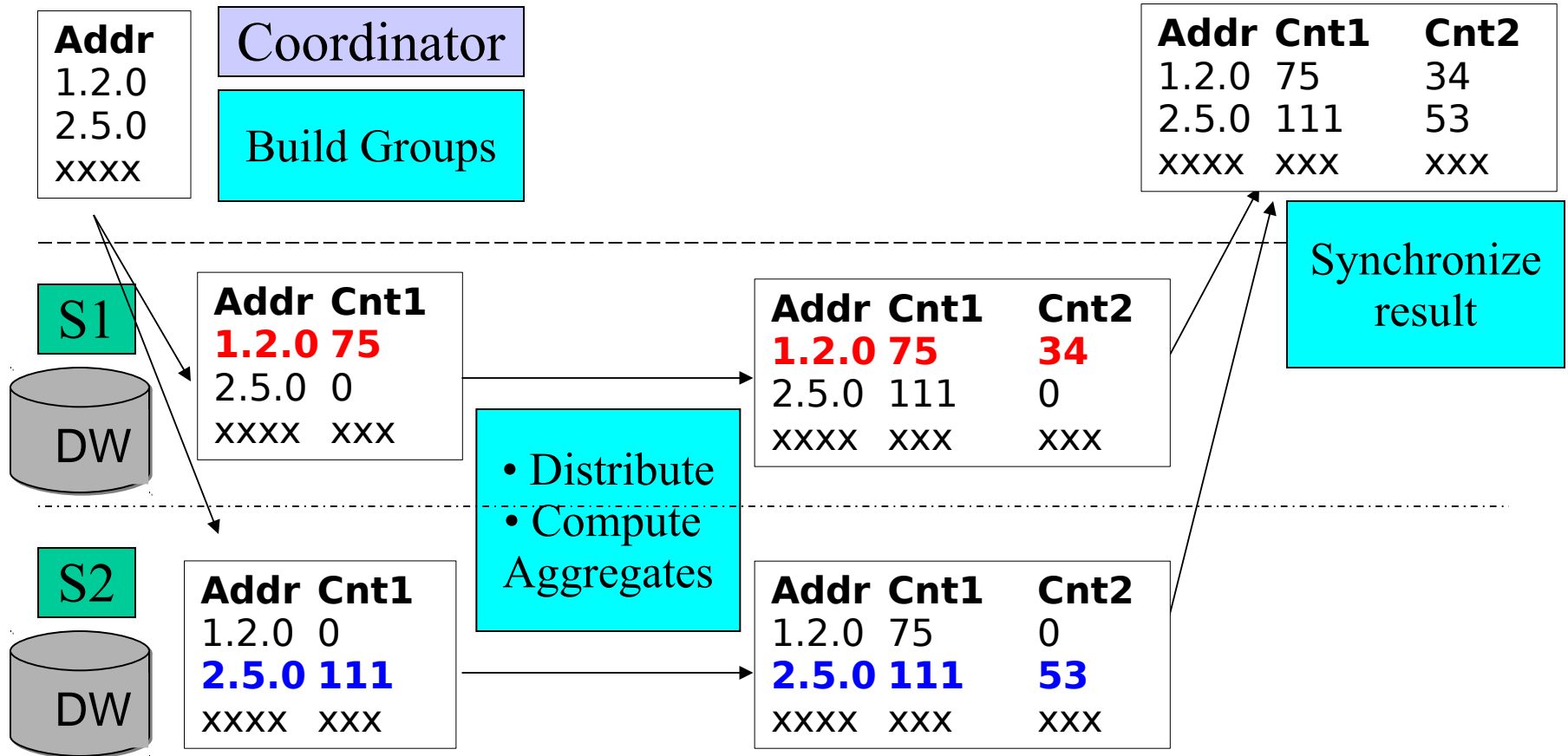
- It is possible to detect cases where no synchronization is required between passes.
- Example:
 - DW data: All the flows of a particular router are always registered (stored) at a particular local DW
 - Query: For each IP address, what fraction of the total number of flows is due to web traffic?
 - Each IP address belongs to a particular router; i.e., all data for a particular IP address is located at the system storing the flows of its router

Synch Reduction: Example



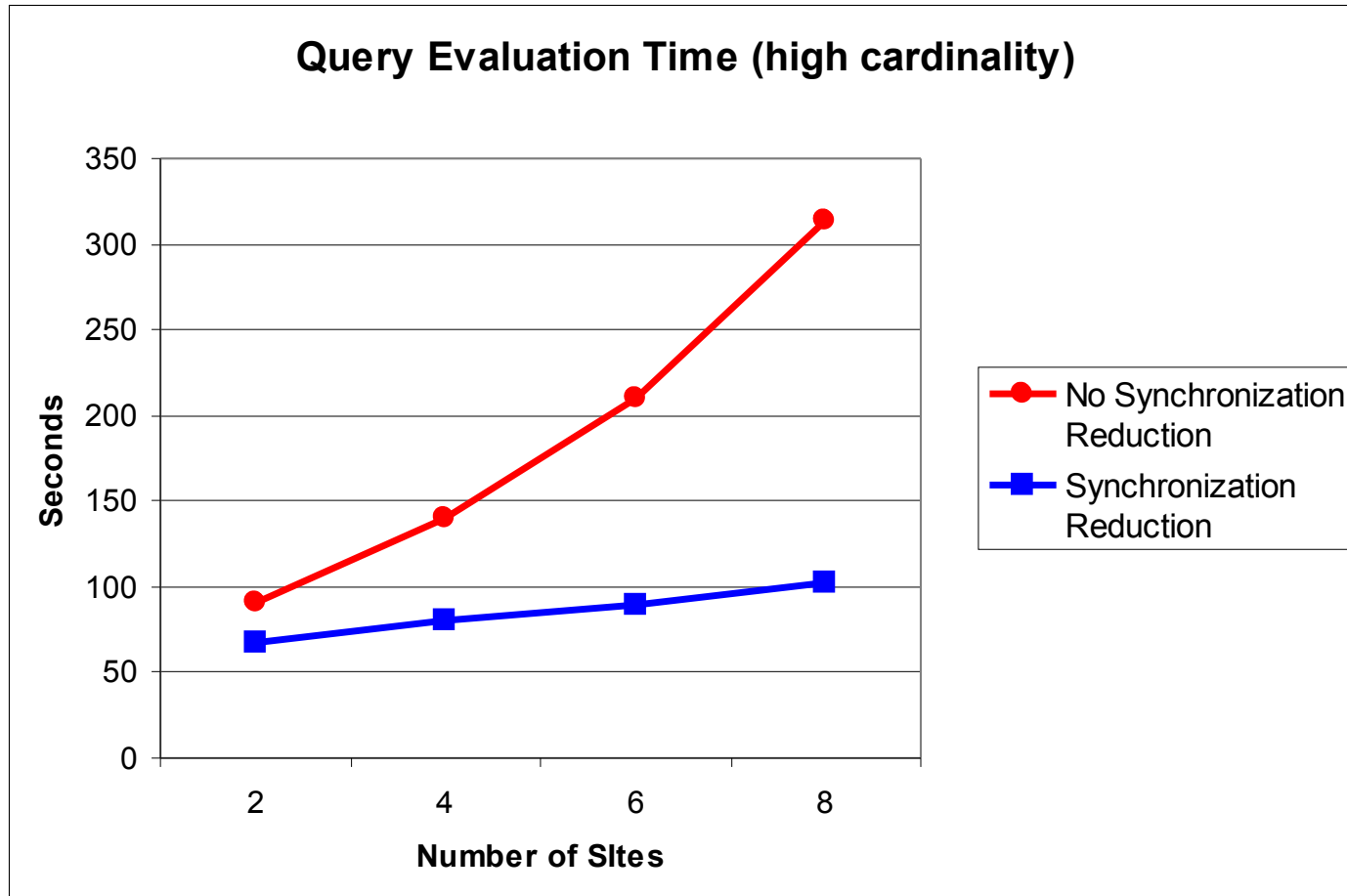
For each IP address, what fraction of the total number of flows is due to web traffic?

Synch Reduction: Example

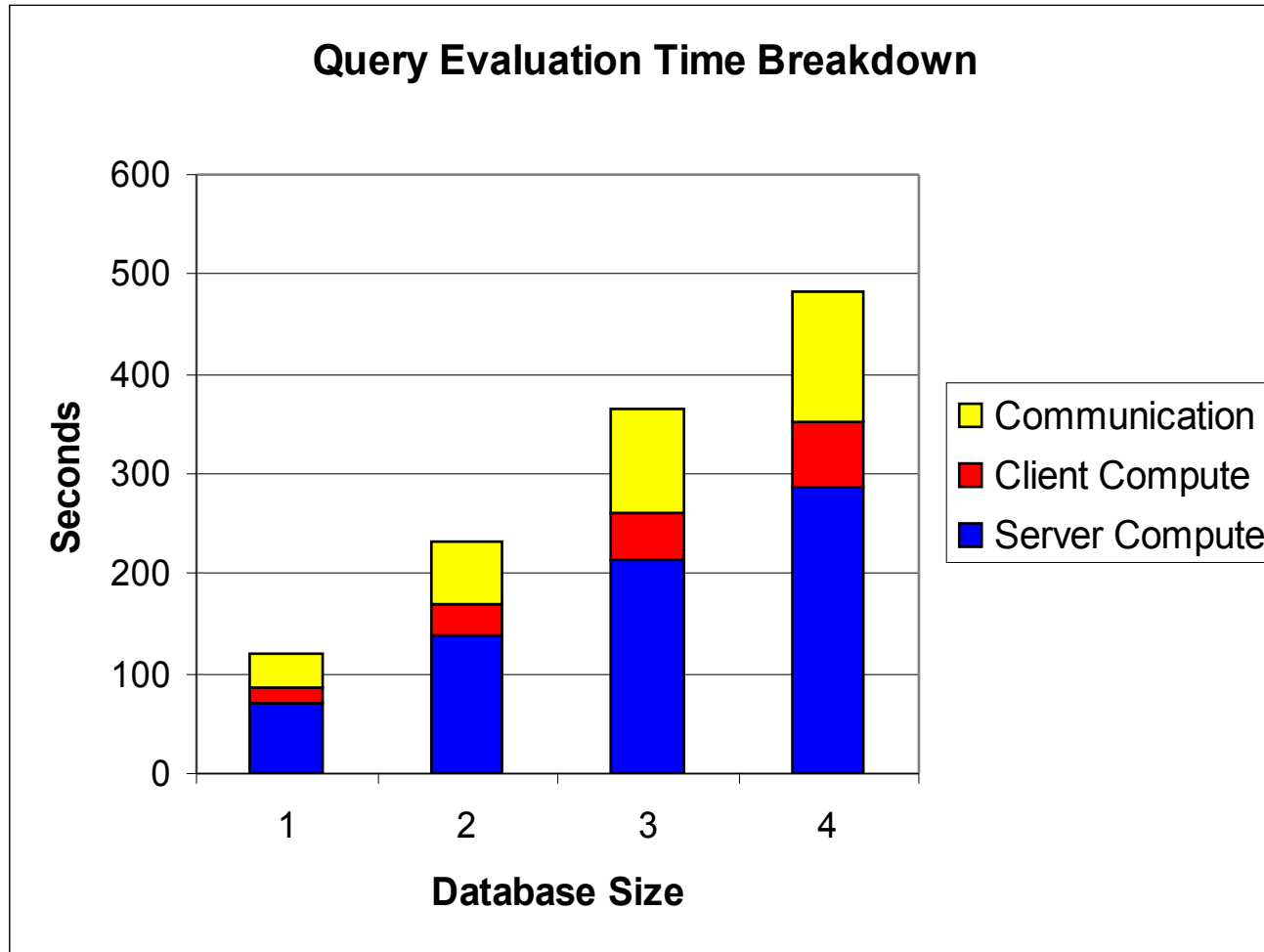


For each IP address, what fraction of the total number of flows is due to web traffic?

Number of Sites (SR)



Cost Breakdown



Outline

- The Generalized Multi-dimensional Join
- An Algorithm for the GMDJ
- Evaluating Subqueries in an OLAP Context
- Distributed Data Warehousing
- **Conclusion**

Literature

- Multi-feature syntax for SQL [VLDB96 & VLDB97]
- Chatziantoniou's PhD thesis (1997)
- Extended Multi-Feature Syntax for SQL [SSDBM99, ICDE99, CIKM99]
- The multi-dimensional join operator: an algebraic operator for complex OLAP [ICDE01]
- GMD-join [VLDB TelcoWS 2001]
- Segmented aggregation [Galindo-Legaria, SIGMOD01]
- Akinde's PhD thesis (2003)
- Complex OLAP subqueries [ICDE03]
- Distributed evaluation of complex OLAP queries [EDBT02, IS03]
- Theta-Constrained multidimensional aggregation **[IS 2011]**

Summary

- The GMDJ offers a systematic and robust strategy to evaluate complex OLAP.
- It separates grouping from aggregation.
- Complex predicates are mapped to simple aggregations.
- Typically, intermediate results remain small (unlike join-based solutions).
- The GMDJ has been verified empirically.