Optimisation of Partitioned Temporal Joins

Thomas Zurek

University of Edinburgh

Overview

1. temporal joins

2. temporal join processing

3. optimisation process

4. summary

An Example

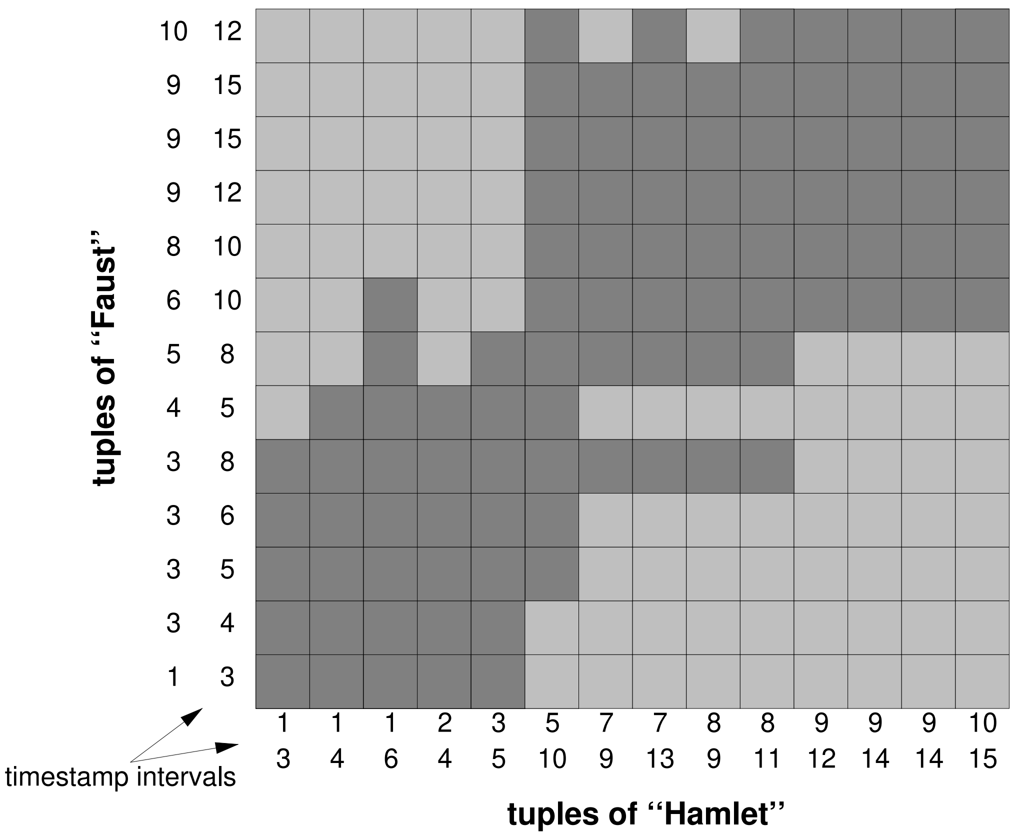
Relation **Hamlet**	
City	Timestamp
Paris	[3,4]
Berlin	[3,5]
Rome	[4,6]
Vienna	[8,8]
Oslo	[8,9]
Berlin	[8,9]
...	

Relation **Faust**	
City	Timestamp
Berlin	[1,3]
Oslo	[1,3]
Paris	[3,3]
Vienna	[6,7]
Madrid	[6,9]
Paris	[7,9]
...	

Hamlet $\bowtie_T$ **Faust**

with $T \equiv$ Hamlet.Timestamp *intersects* Faust.Timestamp

Nested-Loops Temporal Join **Hamlet** $\bowtie_T$ **Faust**



Symmetric Partitioning

$$\textbf{Hamlet} \bowtie_T \textbf{Faust} =$$

$$\textbf{Hamlet}_1 \bowtie_T \textbf{Faust}_1 \cup \textbf{Hamlet}_2 \bowtie_T \textbf{Faust}_2 \cup \textbf{Hamlet}_3 \bowtie_T \textbf{Faust}_3$$

$$\textbf{Hamlet} = \bigcup_{k=1}^{3} \textbf{Hamlet}_k \qquad \textbf{Faust} = \bigcup_{k=1}^{3} \textbf{Faust}_k$$

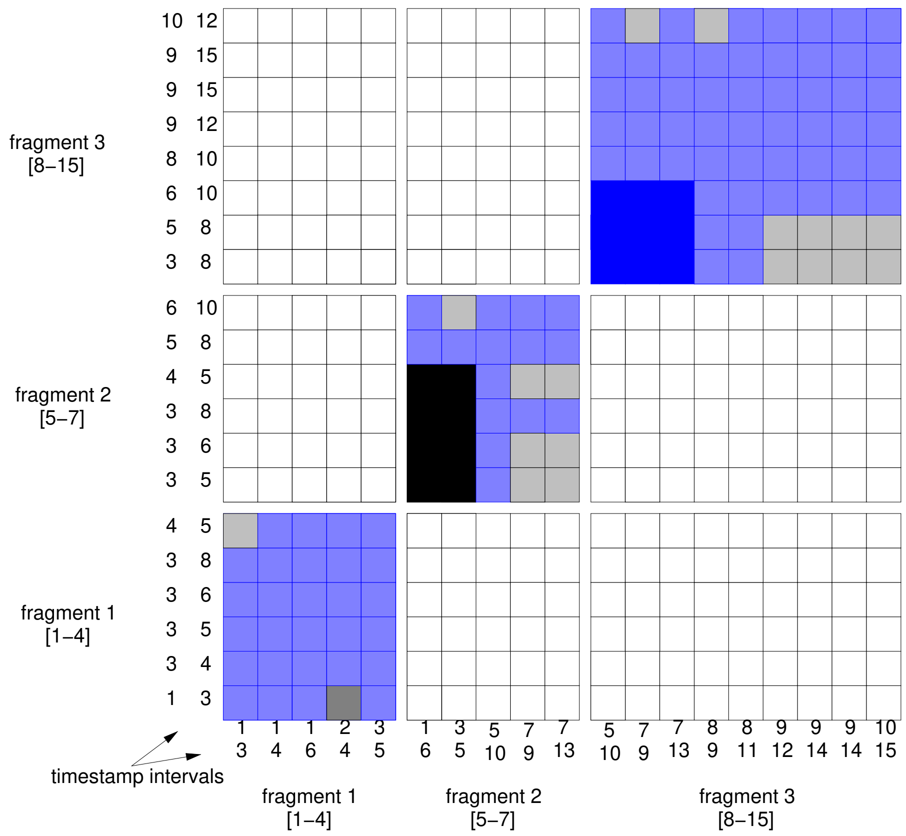
Partitioned Equi-Join Processing

$$\bigcap_{k=1}^{3} \textsf{\textbf{Hamlet}}_k = \emptyset \qquad \bigcap_{k=1}^{3} \textsf{\textbf{Faust}}_k = \emptyset$$

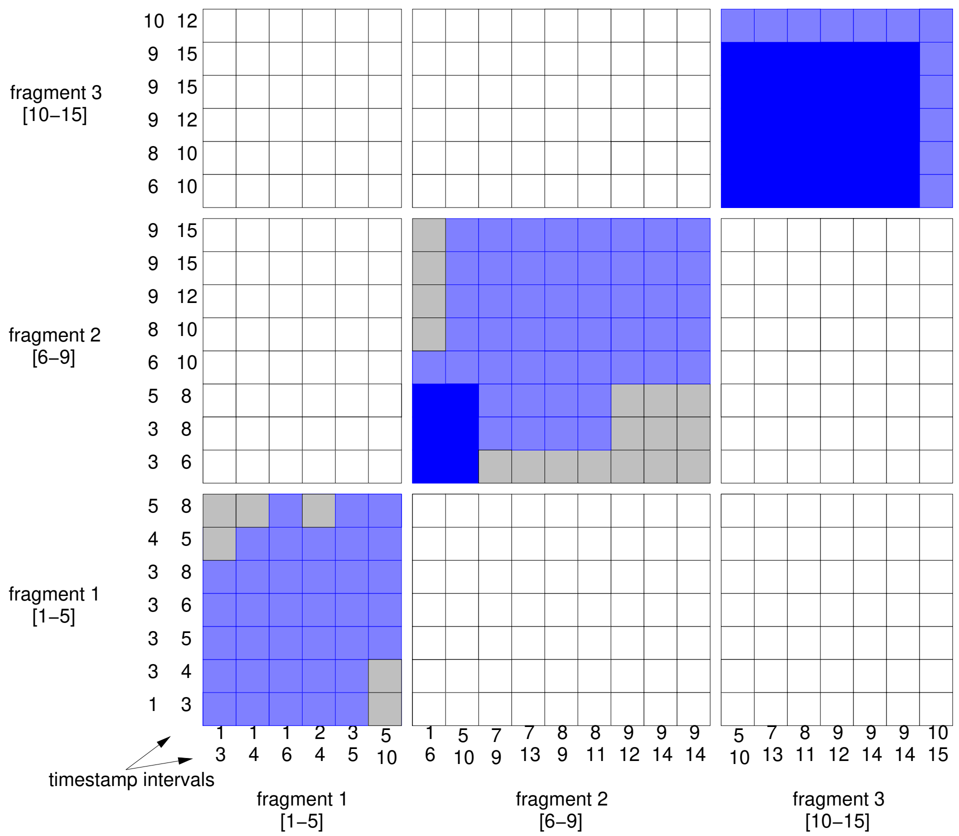
Partitioned Temporal Join Processing

$$\bigcap_{k=1}^{3} \textsf{\textbf{Hamlet}}_k \neq \emptyset \qquad \bigcap_{k=1}^{3} \textsf{\textbf{Faust}}_k \neq \emptyset$$

Partitioned Nested-Loops Temporal Join **Hamlet** $\bowtie_T$ **Faust**



and using a different partition . . .



Problems

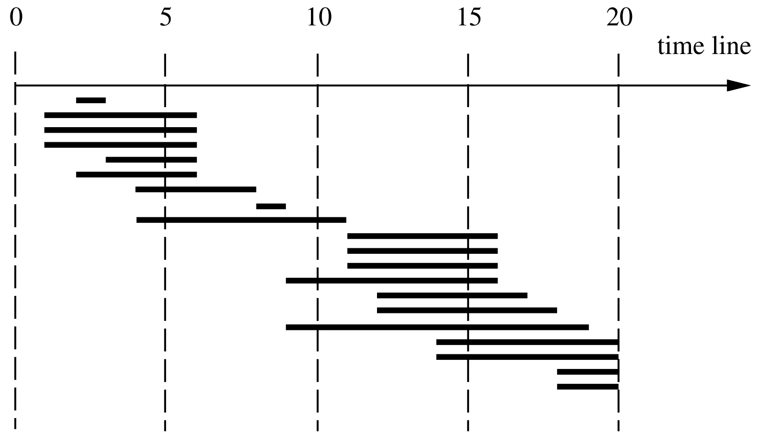
- replication overhead
- processing overhead
- duplicates overhead

$\longrightarrow$ choice of partition is critical

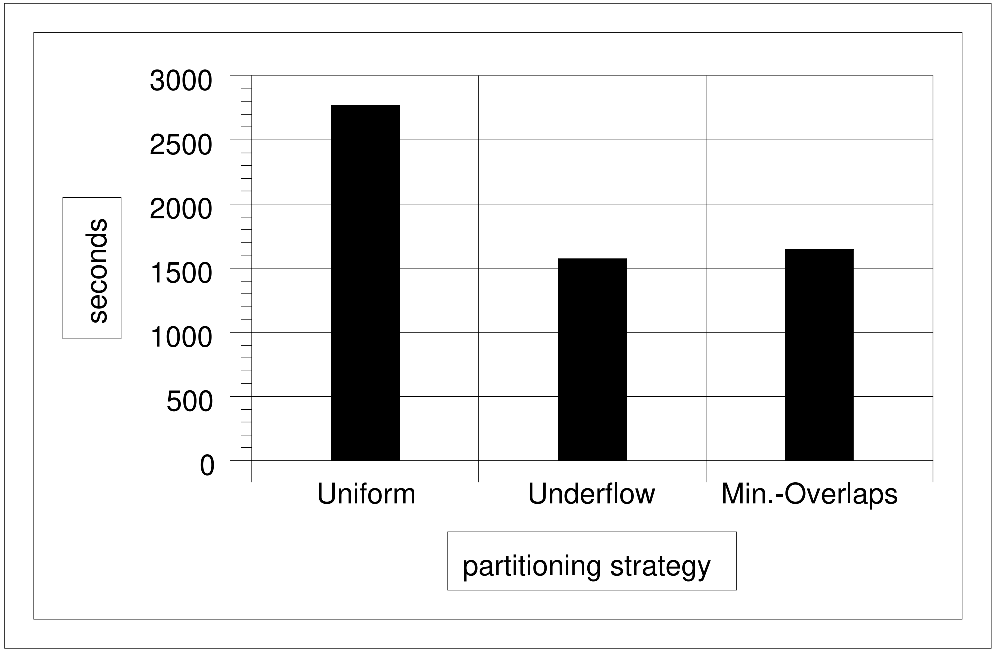
IP-Table for a Relation R

j	t_j	$s_R(t_j)$	$o_R(t_j)$
1	1	3	3
2	2	2	5
3	3	1	5
4	4	2	7
5	6	0	2
6	8	1	2
7	9	2	3
8	11	3	5
9	12	2	7
10	14	2	9
11	16	0	5
12	17	0	4
13	18	2	5
14	19	0	4
15	20	0	0

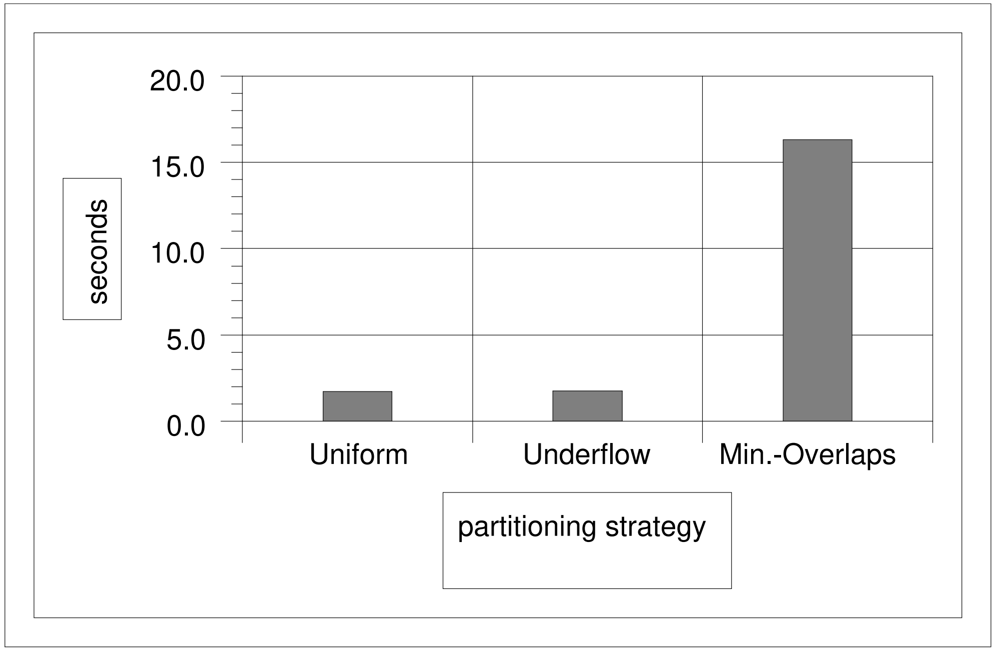
Example Scenario: Intervals of R



Performance Simulation



Elapsed Times for Optimisation



Summary

- temporal joins
- partitioned temporal join processing
- optimiser chooses partition
- IP-table