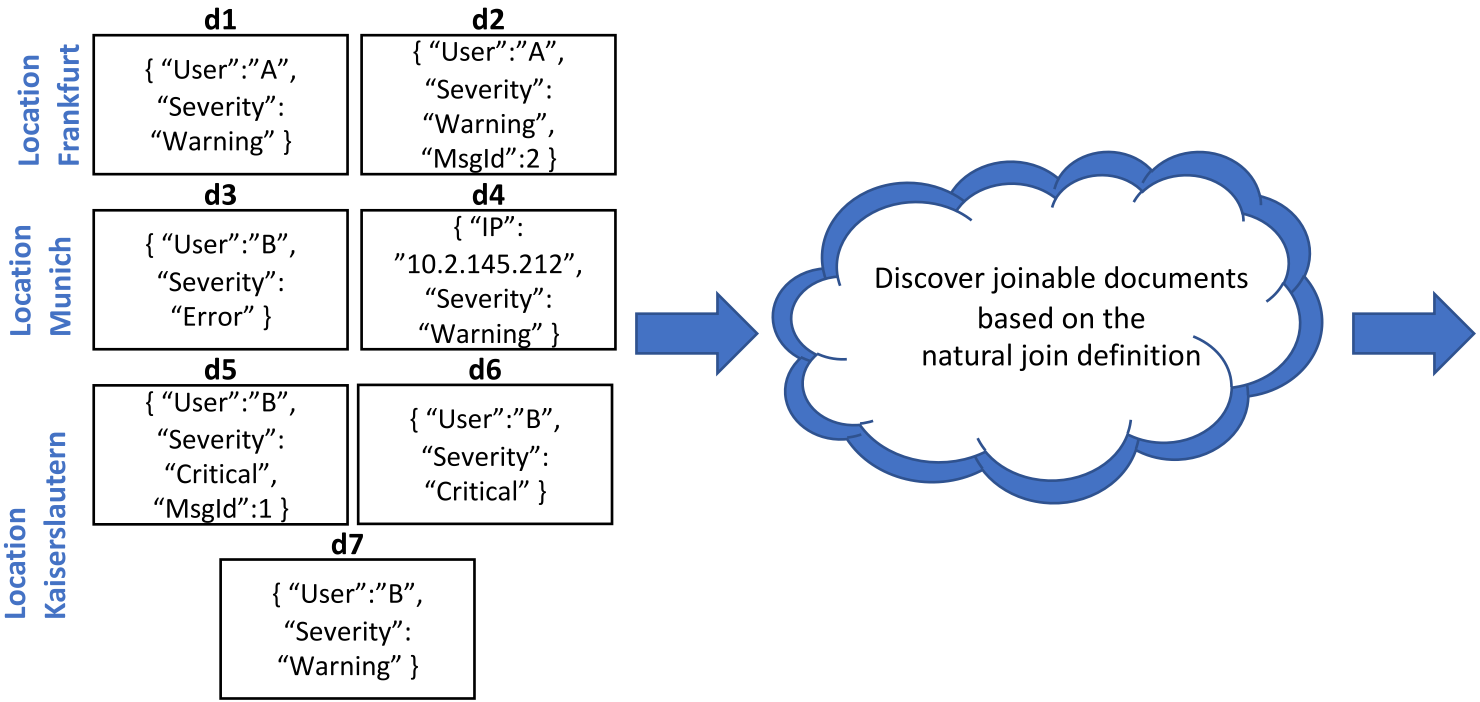
Scaling Out Schema-free Stream Joins

Damjan Gjurovski
gjurovski@cs.uni-kl.de
TU Kaiserslautern

Sebastian Michel
michel@cs.uni-kl.de
TU Kaiserslautern

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

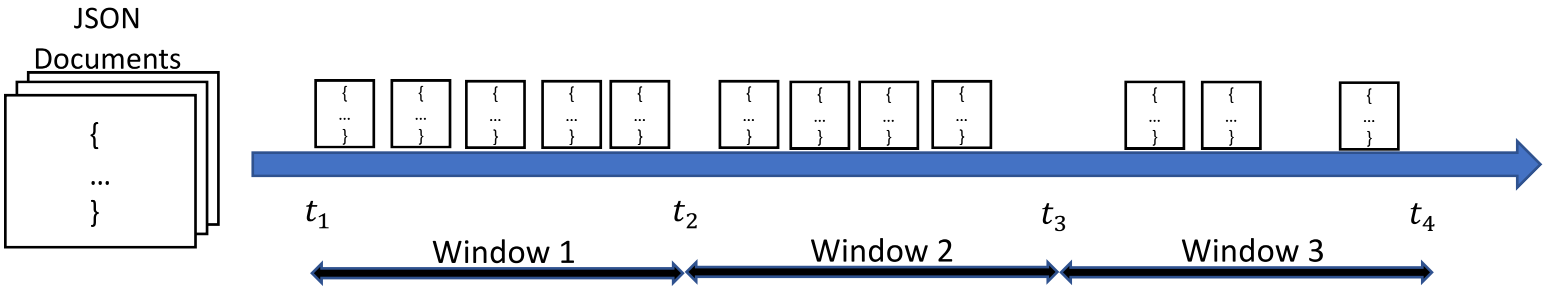
Problem Overview



Document	Joinable Documents
d1	d2, d4
d2	d1, d4
d3	/
d4	d1, d2, d7
d5	d6
d6	d5
d7	d4

Setting

- Stream of schema-free JSON documents
  - Distributed computation over a cluster of machines
- Operate over a subset of the data
  - Sliding Window
  - Tumbling Window



Naïve Approach

Single attribute-value pairs

d1
{ A:2,
B:3,
C:7 }

d2
{ A:2,
C:13,
D:7 }

d3
{ A:2,
B:3,
C:7 }

d4
{ A:2,
C:6 }

d5
{ A:2,
D:7 }

Machine 1
A:2 – d1,
A:2 – d2,
A:2 – d3,
A:2 – d4,
A:2 – d5,
D:7 – d2,
D:7 – d5

Machine 2
B:3 – d1,
B:3 – d3,
C:6 – d4

Machine 3
C:7 – d1,
C:7 – d3,
C:13 – d2

Document	Joinable Documents
d1	d3, d5
d2	d5
d3	d1,d5
d4	d5
d5	d1, d2, d3, d4

Solution Overview

1. Partition the documents
   - Number of partitions equals number of machines
   - Joinable documents are located on the same machine
   - Low replication of documents
   - Balance the load among the machines
2. Local join computation
   - Natural join algorithm performed on every machine

Association-Group-based Partitioning

- Group documents based on patterns of occurrence of the attribute-value pairs
  - Equivalence relationship
  - Implies relationship
- **Phase 1** - compute association groups based on the equivalence groups and the implications between them
- **Phase 2** - assign the computed association groups to partitions

Association-Group-based Partitioning

Phase 1

Input documents

d1 – {A:2,B:3,C:7}
d2 – {A:7,B:3,C:4}
d3 – {D:13}
d4 – {A:7,C:4}

Equivalence groups

eg1 – {A:2,C:7}
eg2 – {B:3}
eg3 – {A:7,C:4}
eg4 – {D:13}

Implications

{A:2,C:7}
implies
{B:3}

Association groups

ag1 - {A:2,C:7,B:3}
ag2 – {A:7,C:4}
ag3 – {D:13}

Association Groups	Load
{A:2,C:7,B:3}	2
{A:7,C:4}	2
{D:13}	1

Association-Group-based Partitioning

Phase 2

Number of partitions: 2

Association Groups	Load
{A:2,C:7,B:3}	2
{A:7,C:4}	2
{D:13}	1

select accosiation group with maximal load

Partitions	Load
{A:2,C:7,B:3}	2

Association-Group-based Partitioning

Phase 2

Number of partitions: 2

Association Groups	Load
{A:2,C:7,B:3}	2
{A:7,C:4}	2
{D:13}	1

select accosiation group with maximal load

Partitions	Load
{A:2,C:7,B:3}	2

select accosiation group with maximal load

Partitions	Load
{A:2,C:7,B:3}	2
{A:7,C:4}	2

Association-Group-based Partitioning

Phase 2

Number of partitions: 2

Association Groups	Load
{A:2,C:7,B:3}	2
{A:7,C:4}	2
{D:13}	1

select accosiation group with maximal load

Partitions	Load
{A:2,C:7,B:3}	2

select accosiation group with maximal load

Partitions	Load
{A:2,C:7,B:3}	2
{A:7,C:4}	2

assign to partition with minimal load

Partitions	Load
{A:2,C:7,B:3}	2
{A:7,C:4,D:13}	3

FP-tree Join: Background

Frequent Pattern Tree (FP-tree) [1]:

- Compact storing of the documents
- Reduced number of accesses over the documents
- Faster navigation through the elements of the documents
- Possibility for developing a more appropriate join algorithm

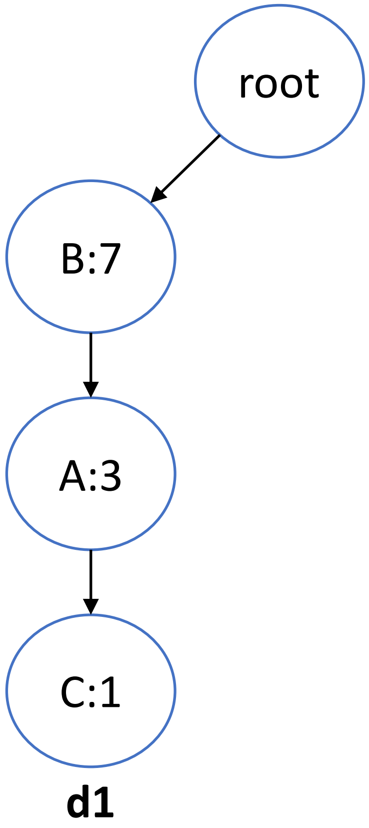
[1] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation”, SIGMOD, 2000.

FP-tree Join: Creation

ID	AV-Pairs	Ordered AV-Pairs
d1	{A:3,B:7,C:1}	{B:7,A:3,C:1}
d2	{A:3,B:8}	{B:8,A:3}
d3	{A:3,B:7}	{B:7,A:3}
d4	{B:8,C:2}	{B:8,C:2}

Attributes Order

B → A → C

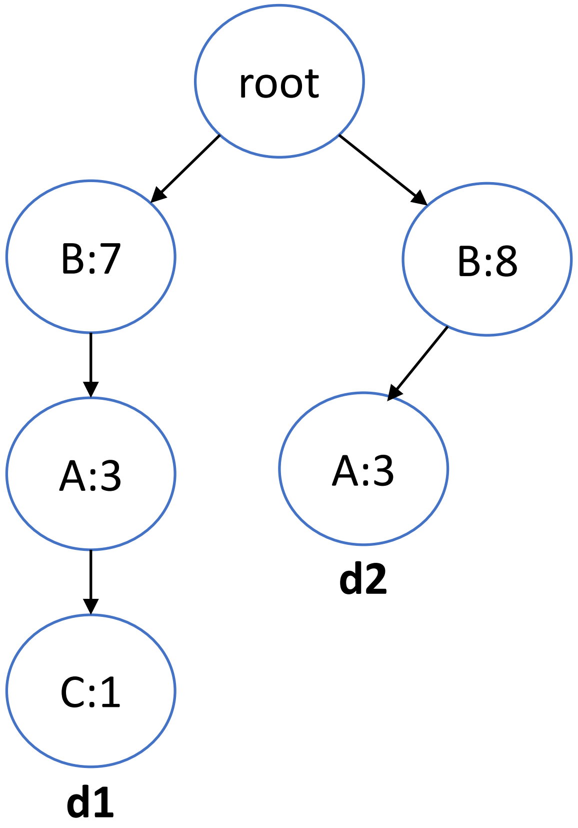


FP-tree Join: Creation

ID	AV-Pairs	Ordered AV-Pairs
d1	{A:3,B:7,C:1}	{B:7,A:3,C:1}
d2	{A:3,B:8}	{B:8,A:3}
d3	{A:3,B:7}	{B:7,A:3}
d4	{B:8,C:2}	{B:8,C:2}

Attributes Order

B → A → C



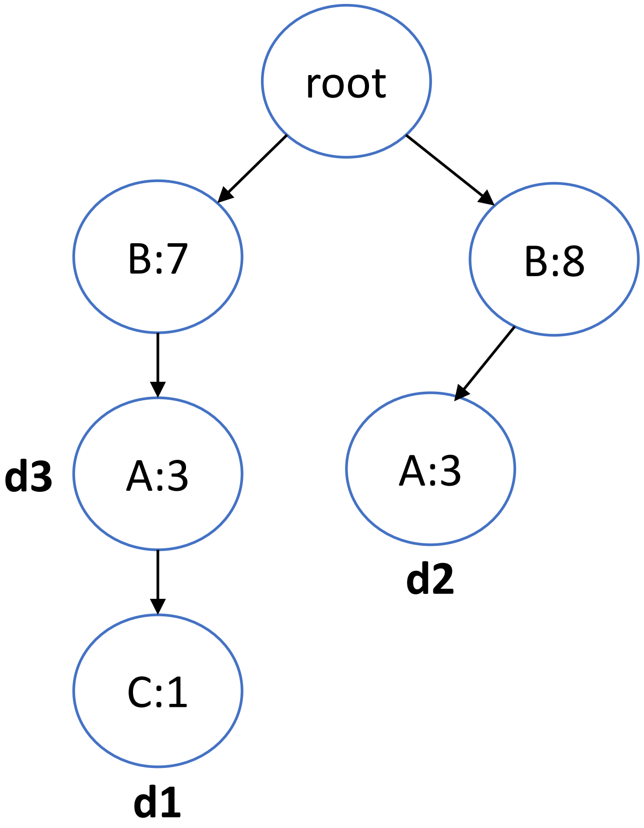
TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

FP-tree Join: Creation

ID	AV-Pairs	Ordered AV-Pairs
d1	{A:3,B:7,C:1}	{B:7,A:3,C:1}
d2	{A:3,B:8}	{B:8,A:3}
d3	{A:3,B:7}	{B:7,A:3}
d4	{B:8,C:2}	{B:8,C:2}

Attributes Order

B → A → C

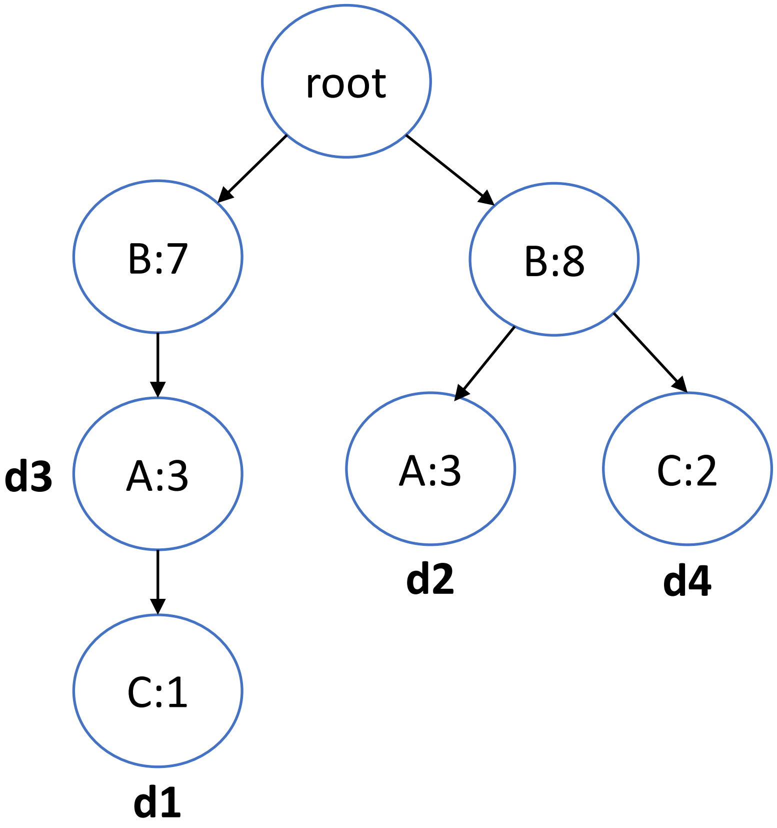


FP-tree Join: Creation

ID	AV-Pairs	Ordered AV-Pairs
d1	{A:3,B:7,C:1}	{B:7,A:3,C:1}
d2	{A:3,B:8}	{B:8,A:3}
d3	{A:3,B:7}	{B:7,A:3}
d4	{B:8,C:2}	{B:8,C:2}

Attributes Order

B → A → C

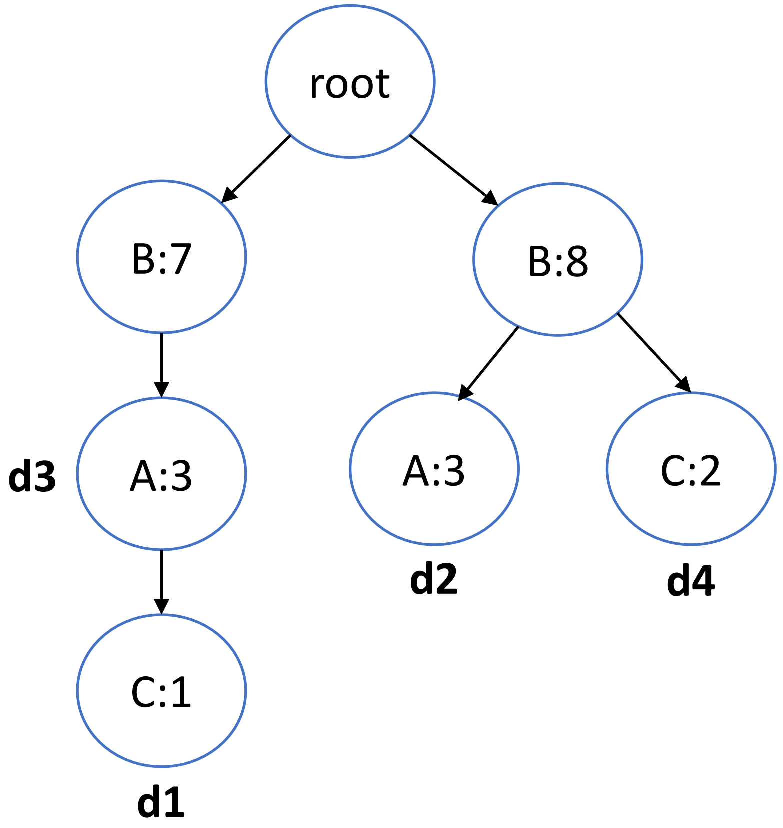


FP-tree Join: Processing

Number of ubiquitous attributes: **1**

Document	Ordered AV-Pairs
d1	{B:7,A:3,C:1}
d2	{B:8,A:3}
d3	{B:7,A:3}
d4	{B:8,C:2}

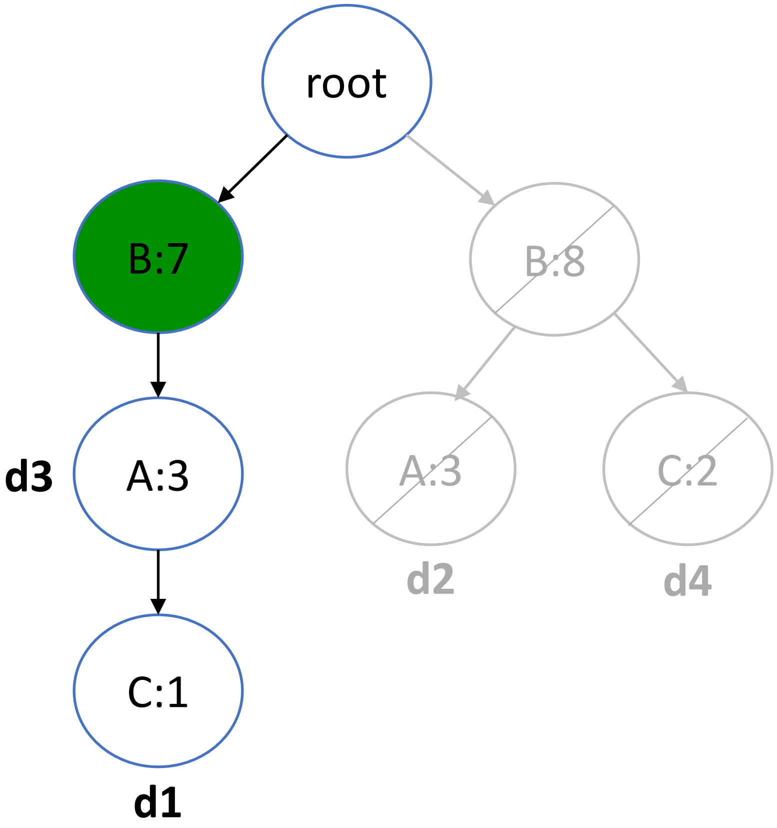
Joinable documents:



FP-tree Join: Processing

Number of ubiquitous attributes: **1**

Document	Ordered AV-Pairs
d1	{B:7,A:3,C:1}
d2	{B:8,A:3}
d3	{B:7,A:3}
d4	{B:8,C:2}



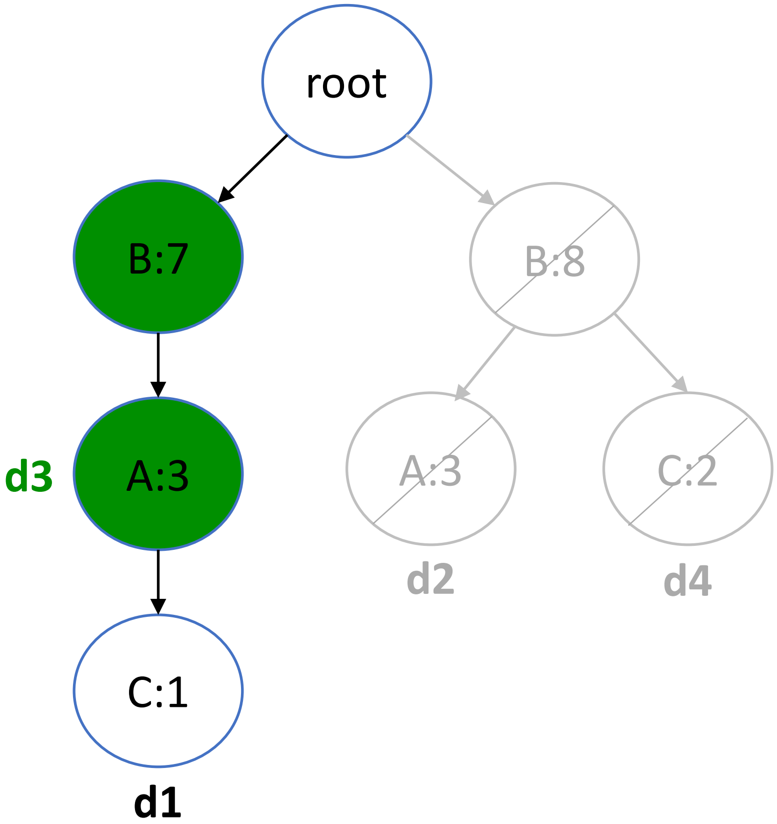
Joinable documents:

FP-tree Join: Processing

Number of ubiquitous attributes: **1**

Document	Ordered AV-Pairs
d1	{B:7,A:3,C:1}
d2	{B:8,A:3}
d3	{B:7,A:3}
d4	{B:8,C:2}

Joinable documents: **d3**

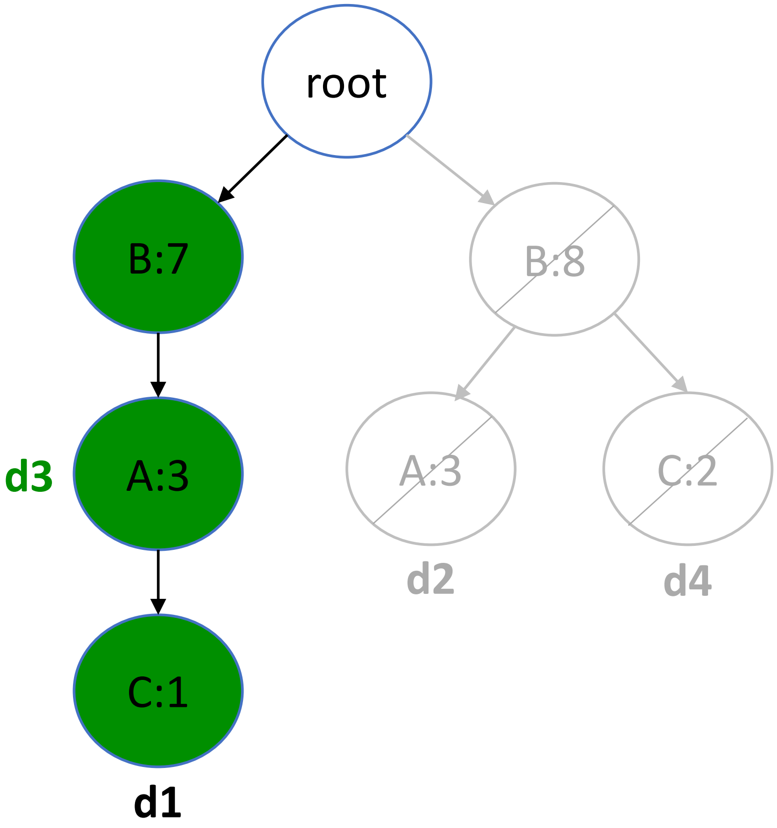


FP-tree Join: Processing

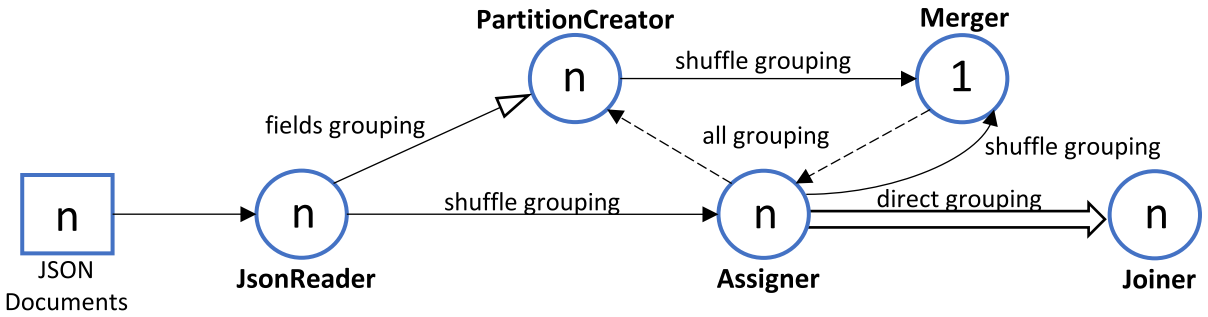
Number of ubiquitous attributes: **1**

Document	Ordered AV-Pairs
d1	{B:7,A:3,C:1}
d2	{B:8,A:3}
d3	{B:7,A:3}
d4	{B:8,C:2}

Joinable documents: **d3**



Apache Storm Topology



- **JsonReader Spout** - Parses JSON documents
- **PartitionCreator Bolt** - Creates local association groups
- **Merger Bolt**
  - Computes final partitions
  - Assigns previously unseen documents to partitions
- **Assigner Bolt**
  - Delivers documents to appropriate Joiners
  - Detects unseen documents and sends them to Merger
  - Monitors partition quality
- **Joiner Bolt** - Computes the joinable documents

Experimental Setup

- Java 8 and Apache Storm 1.2.2
- Cluster of 8 machines with 6-core Intel Xeon E5-2603 v4 CPUs @1.7 GHz and 128GB RAM
- Datasets
  - NoBench JSON data generator [2]
  - Real-world data – server logs gathered from 5 servers of a mid-size company
- Configuration parameters
  - Number of partitions $\boldsymbol{m}$
  - Window size $\boldsymbol{w}$
  - Repartitioning threshold $\boldsymbol{\theta}$

[2] C. Chasseur, Y. Li, and J. M. Patel, "Enabling JSON document stores in relational systems", WebDB, 2013.

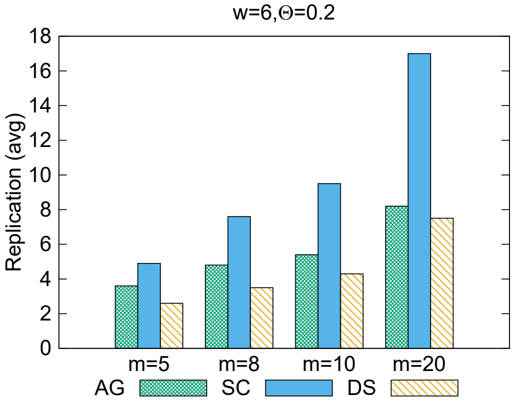
Competitors

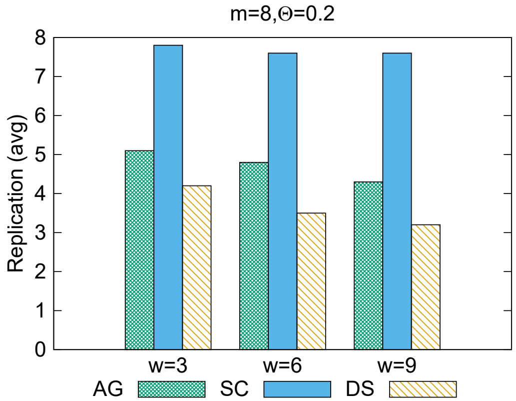
- Partitioning Algorithms
  - Disjoint Sets (DS) partitioning approach [3]
  - Set-Covers-based (SC) partitioning algorithm [3]
- Natural Join Algorithms
  - Hash-based join
  - Nested loop Join

[3] F. Alvanaki and S. Michel, “Tracking set correlations at large scale”, SIGMOD, 2014.

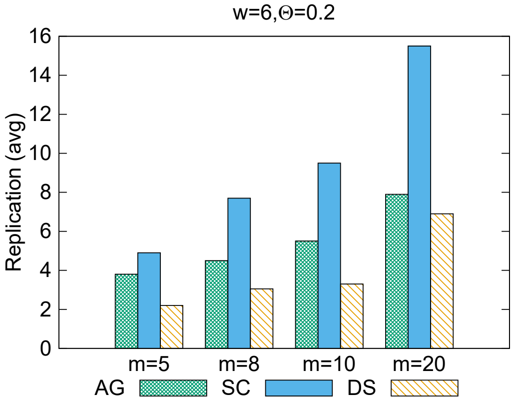
Replication

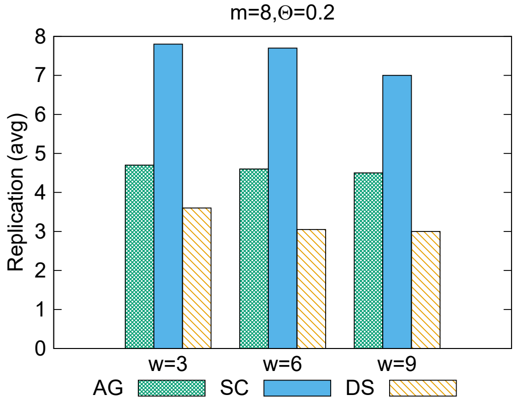
Real-world Data





NoBench Data



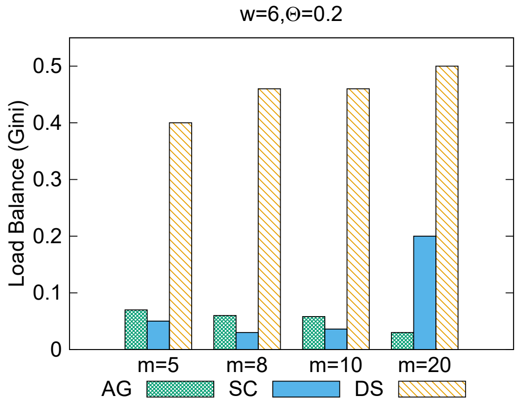


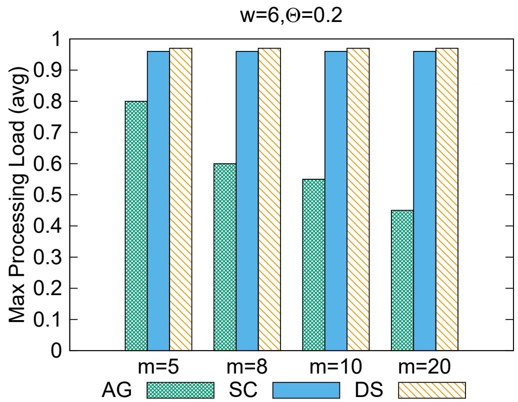
Varying Partitions

Varying Window

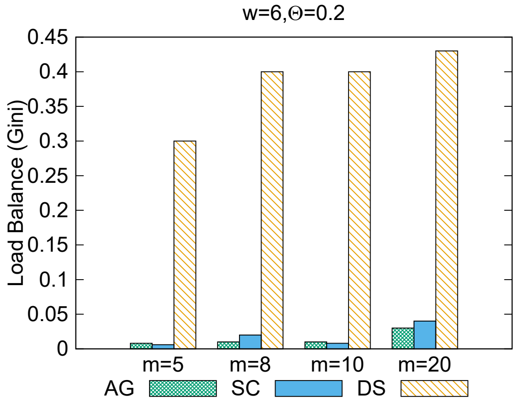
Load Balance & Maximal Processing Load

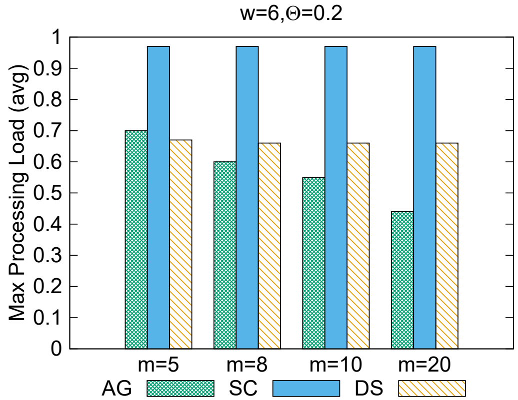
Real-world Data





NoBench Data



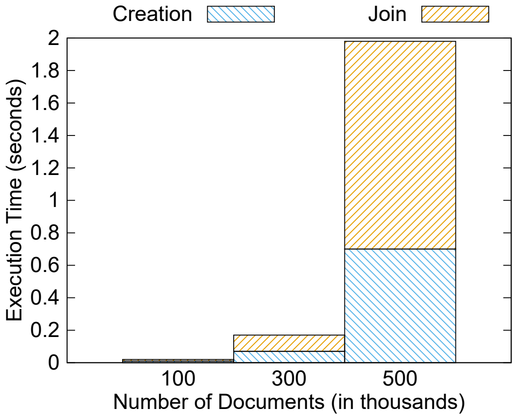


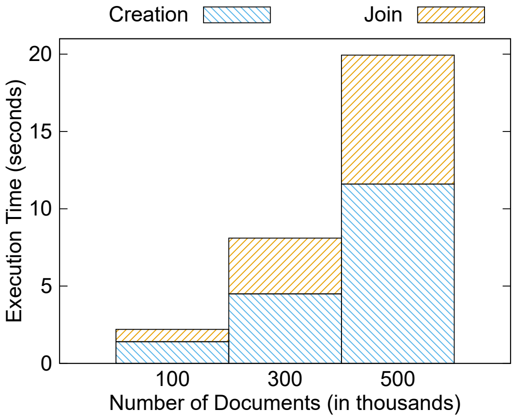
Varying Partitions

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

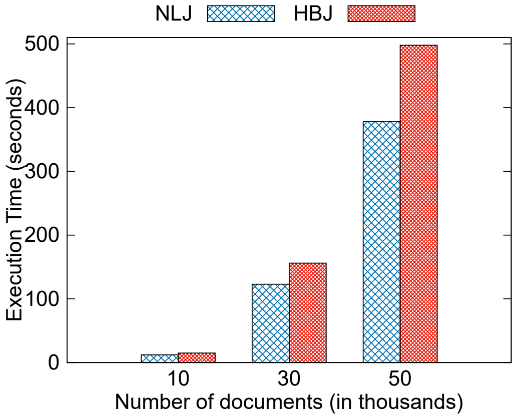
Join Execution Time

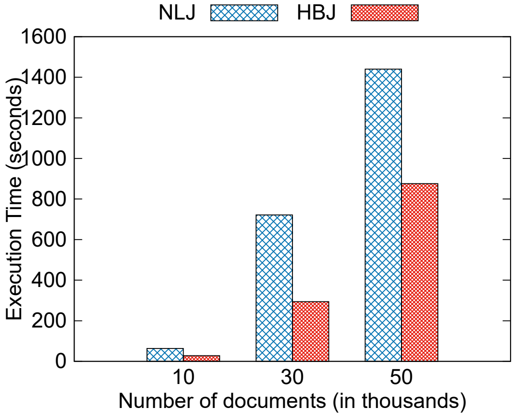
FPTreeJoin





Competitors





Real-world Data

NoBench Data

Conclusion

- Computing natural joins over schema-free JSON documents
- Partitioning based on Association Groups
  - Handling new documents with unseen attribute-values pairs
- FP-tree-based join algorithm
- Detailed experimental evaluation

Thank you for your attention !