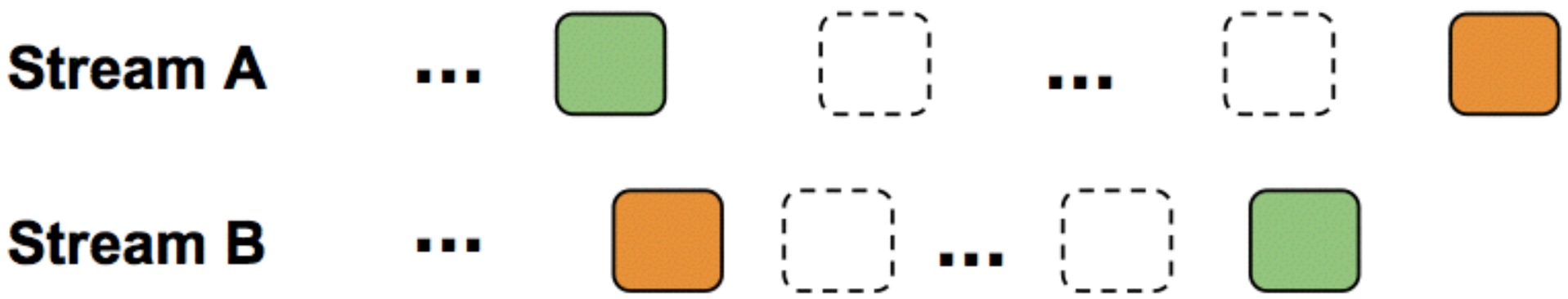
Joining Punctuated Streams.

Luping Ding, Nishant Mehta, Elke A. Rundensteiner, and George T.

Motivation.

- Traditional Join: Symmetric Hash, etc, but...
- Join on Streams



Steam is potentially unbounded : Memory overflow

Existing Approach.

- Memory & Disk: XJoin
- Sliding Window

Paper's Approach.

- Punctuation: **PJoin**

• What is Punctuation?

"signal end of transmitting certain attribute values"

"ordered set of patterns"

Open Stream

item_id	**seller_id**	**open_price**	**timestamp**	◄ *Schema*
1080	jsmith	130.00	Nov-10-03 9:03:00	◄ *Tuple*
<1080, *, *, *>				◄ *Punctuation*
1082	melissa	20.00	Nov-10-03 9:10:00	
<1082, *, *, *>				

Bid Stream

item_id	**bidder_id**	**bid_price**	**timestamp**
1080	pclover	175.00	Nov-14-03 8:27:00
1082	smartguy	30.00	Nov-14-03 8:30:00
1080	richman	177.00	Nov-14-03 8:52:00
<1080, *, *, *>			

(a) Streams

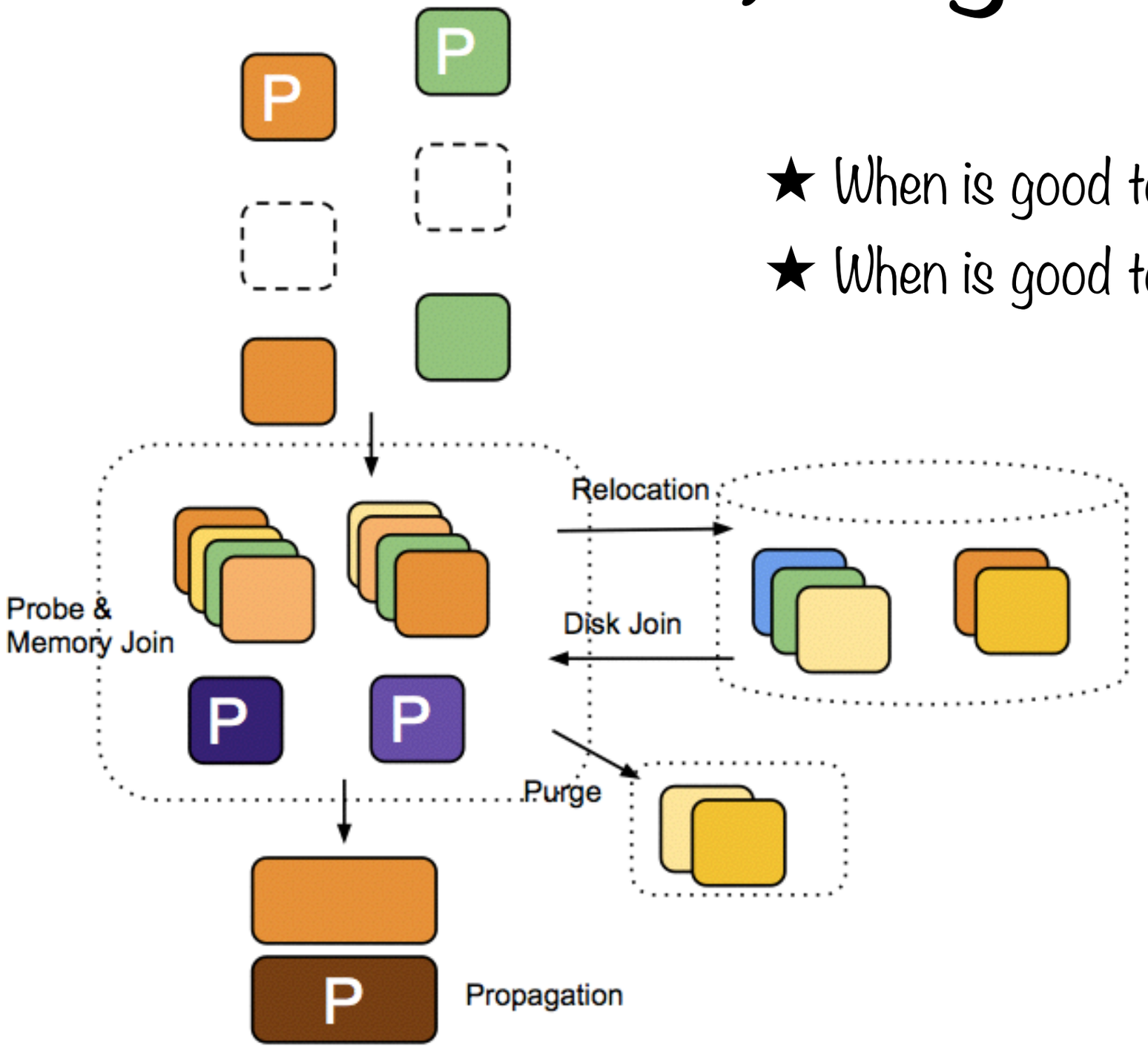
- Why use punctuation?

  Short answer: Purge & Propagation

- How PJoin use punctuation?

- How does PJoin work?

PJoin, in general.



★ When is good to purge to tuple?

★ When is good to propagate a punctuation?

★ When is good to purge to tuple?

First, let's introduce some denotation ...

Let $TS_A(T)$ be set of all tuples arrived before time T from stream A
Let $PS_A(T)$ be set of all punctuations arrived before time T from stream A

Say $match(t, p)$ if tuple t has a join value that mathes the pattern declared by punctuation p.

Say $setMatch(t, PS_A(T))$ if $\exists p \in PS_A(T)$ such that $match(t, p)$

When is it safe to purge a tuple from a stream?

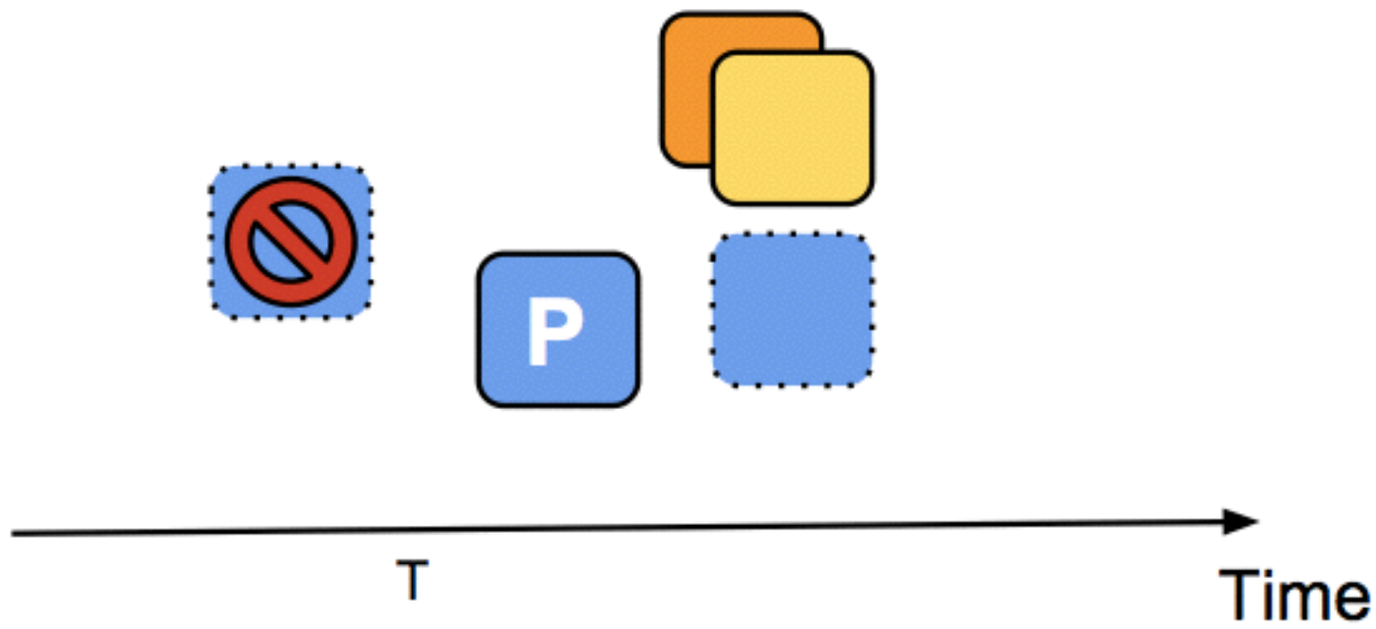
Given stream A and B,
$\forall t \in TS_A(T)$, if $setMatch(t, PS_B(T))$ then safe to purge t
$\forall t \in TS_B(T)$, if $setMatch(t, PS_A(T))$ then safe to purge t

Purge Mode: Eager Purge

Lazy Purge (with a purge threshold)

★ When is good to propagate a punctuation?

Theorem 1. *Given $TS_A(T)$ and $PS_A(T)$, for any punctuation p_A in $PS_A(T)$, if at time T, no tuple t_A exists in $TS_A(T)$ such that match(t_A,p_A), then no tuple t_R such that match(t_R, p_A) will be generated as a join result at or after time T.*



When is it safe to propagate a punctuation?

$$\forall p_A \in PS_A(T),\ propagate(p_A)\ \ if\ \ \forall t_A \in TS_A(T), \neg match(t_A, p_A)$$
$$\forall p_B \in PS_B(T),\ propagate(p_B)\ \ if\ \ \forall t_B \in TS_B(T), \neg match(t_B, p_B)$$

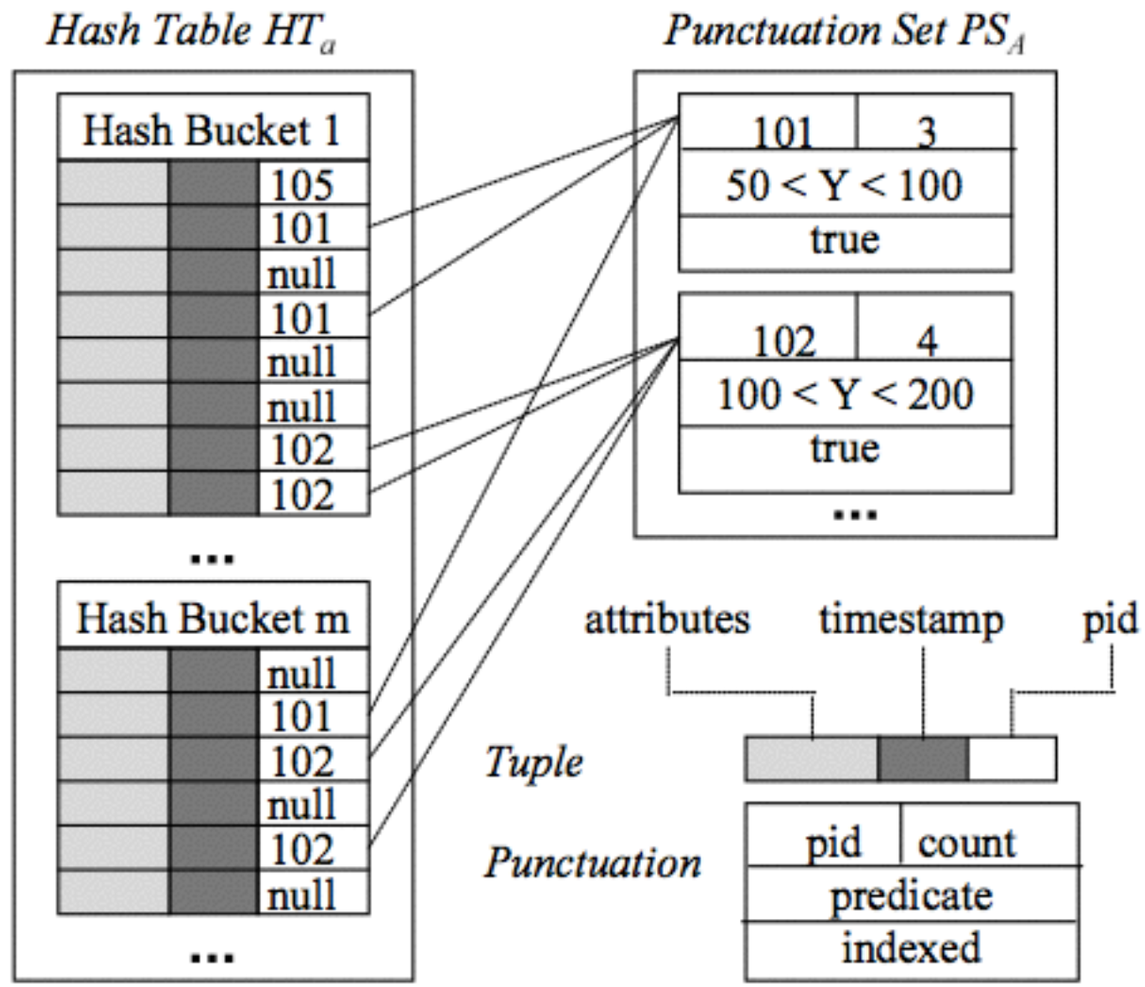
A propagation optimizer: Punctuation Index

```
Class Punctuation {
    long pid;
    int count;
    Expression predicate;
    boolean indexed;
}
```

(a) Definition of Punctuation

```
Class Tuple {
    Object[] attributes;
    long timestamp;
    long pid;
}
```

(b) Definition of Tuple

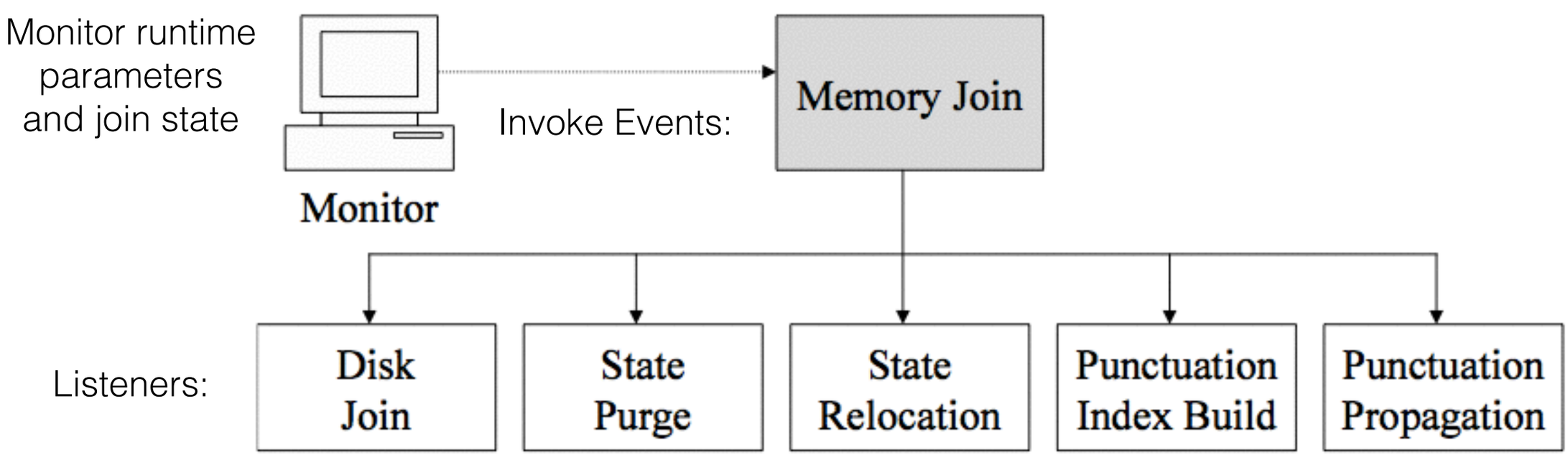


(c) Punctuation Index

- Eager & Lazy build
- Pull & Pull propagation

For Experimenting PJoin:
An Event Driven Framework

1. "keep track of a variety of runtime parameters"
2. "model the different coupling alternatives among components"



Experiments Setup.

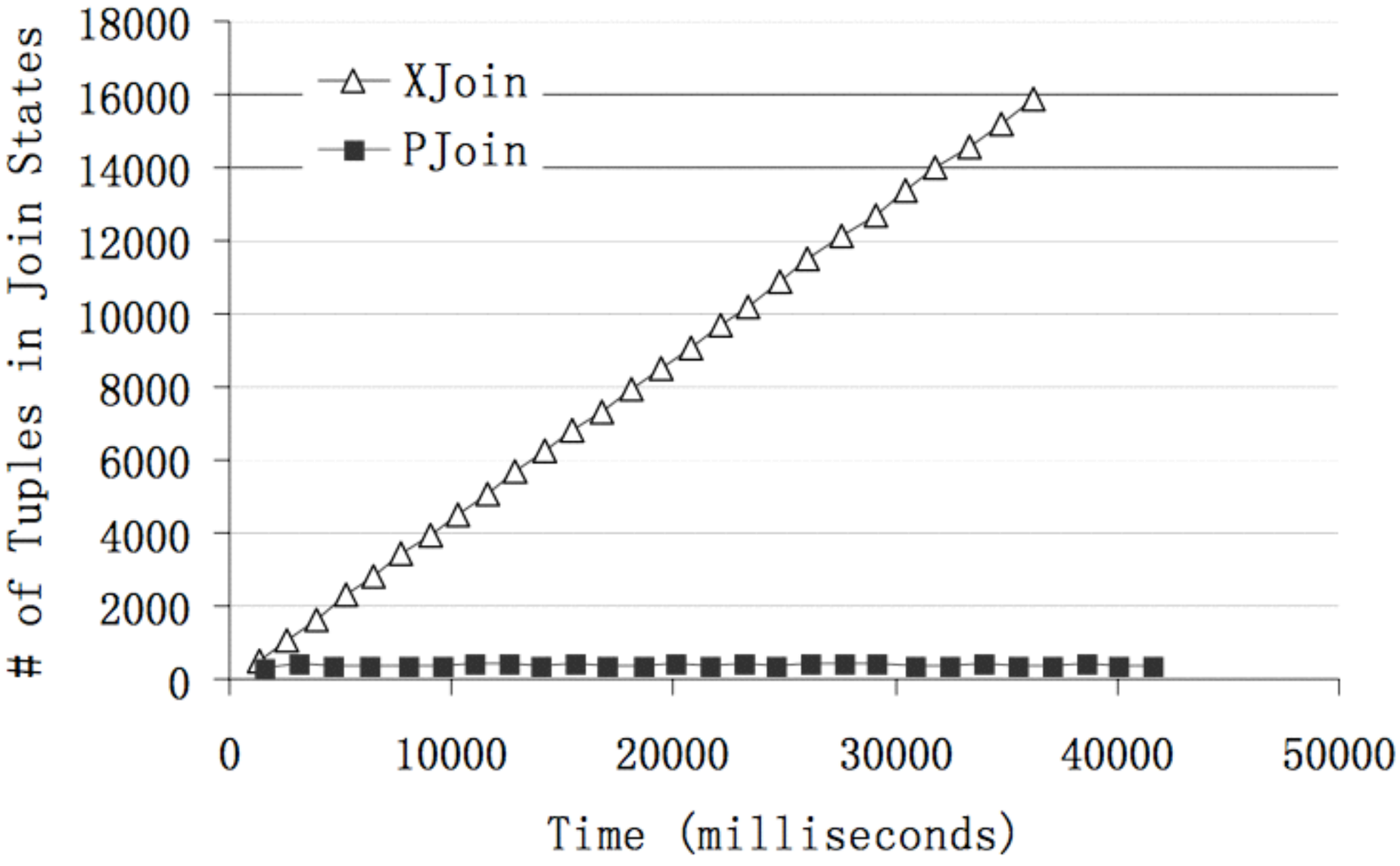
- Raindrop
- Synthetic stream
- Data with poisson inter-arrival time
- Many-to-many relationship

Are we able to reduce memory overhead?
(if so, how much?)

Are we able to increase data throughput?
(if so, how much?)

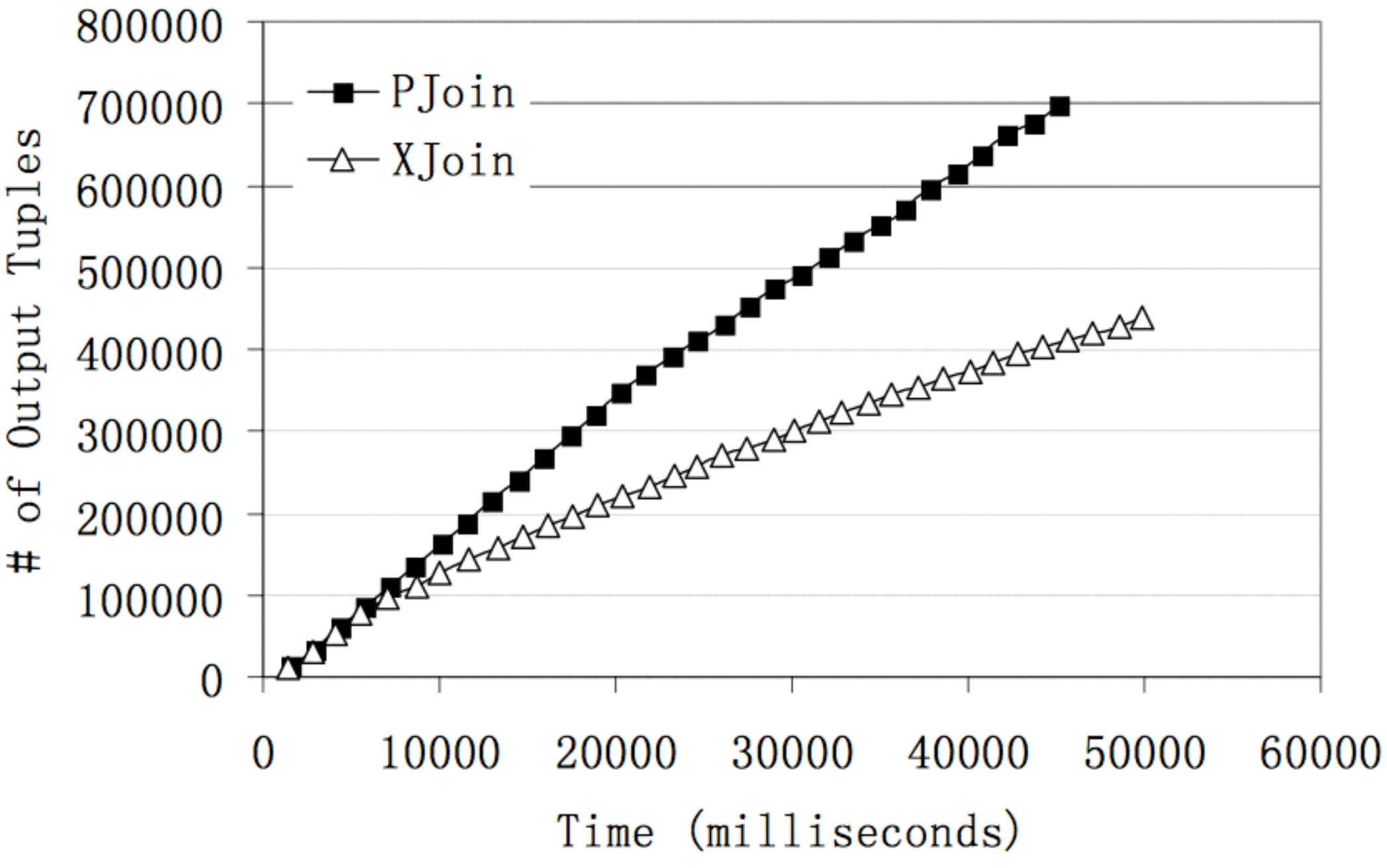
Round-1: PJoin VS. XJoin

Memory Overhead:



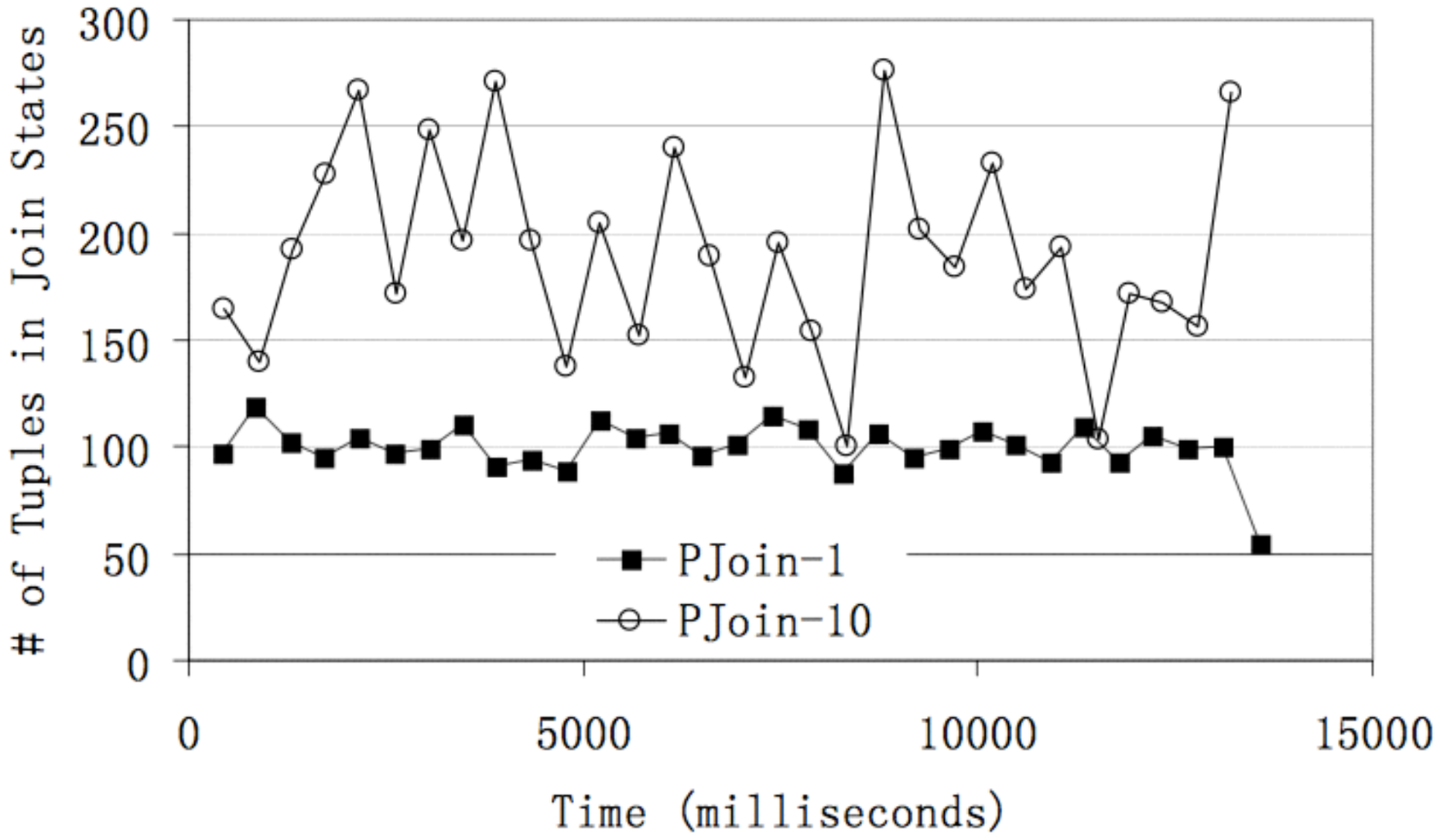
Round-1: PJoin VS. XJoin

Data Throughput:



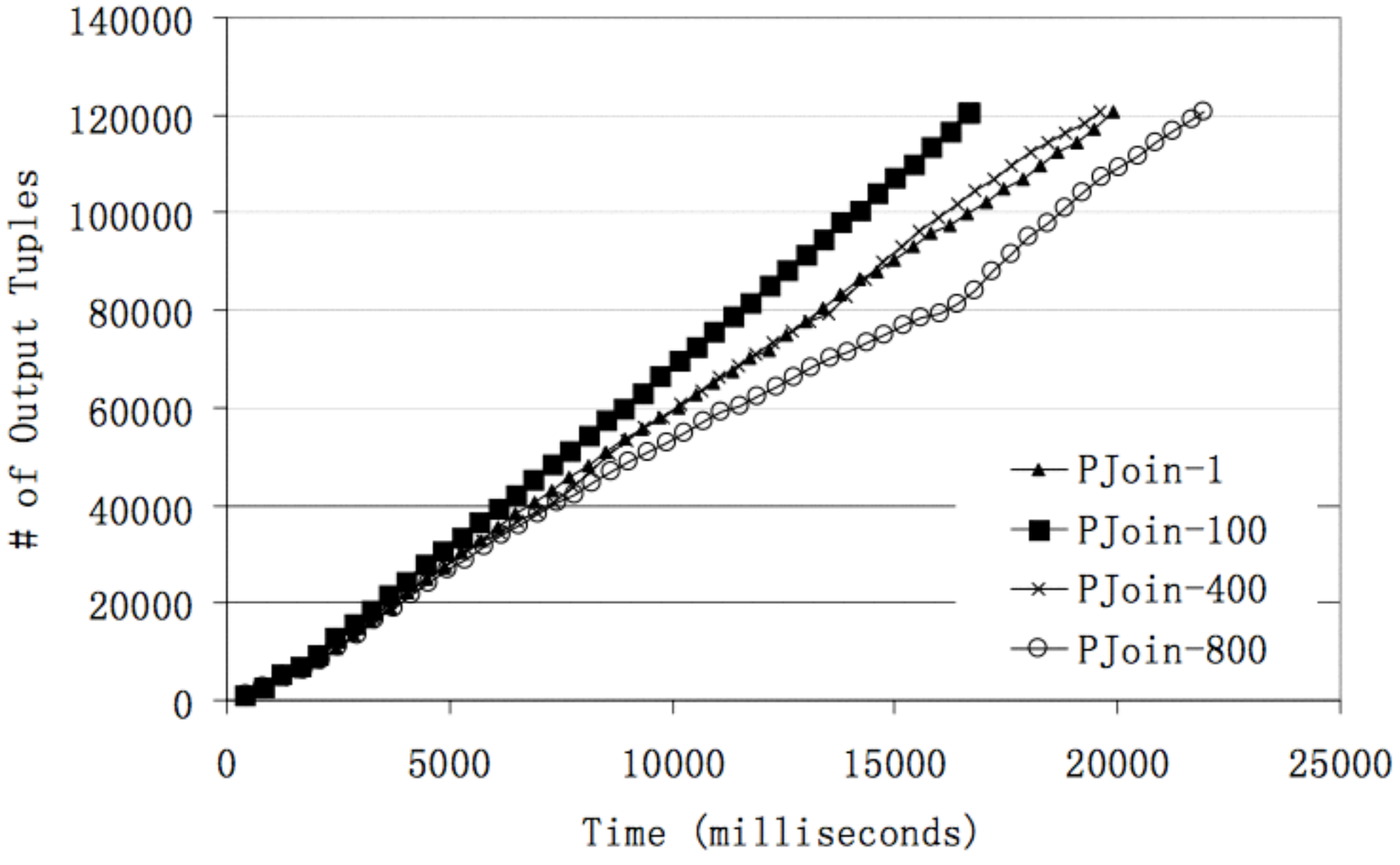
Round-2: Eager purge VS. Lazy purge

Memory Overhead:



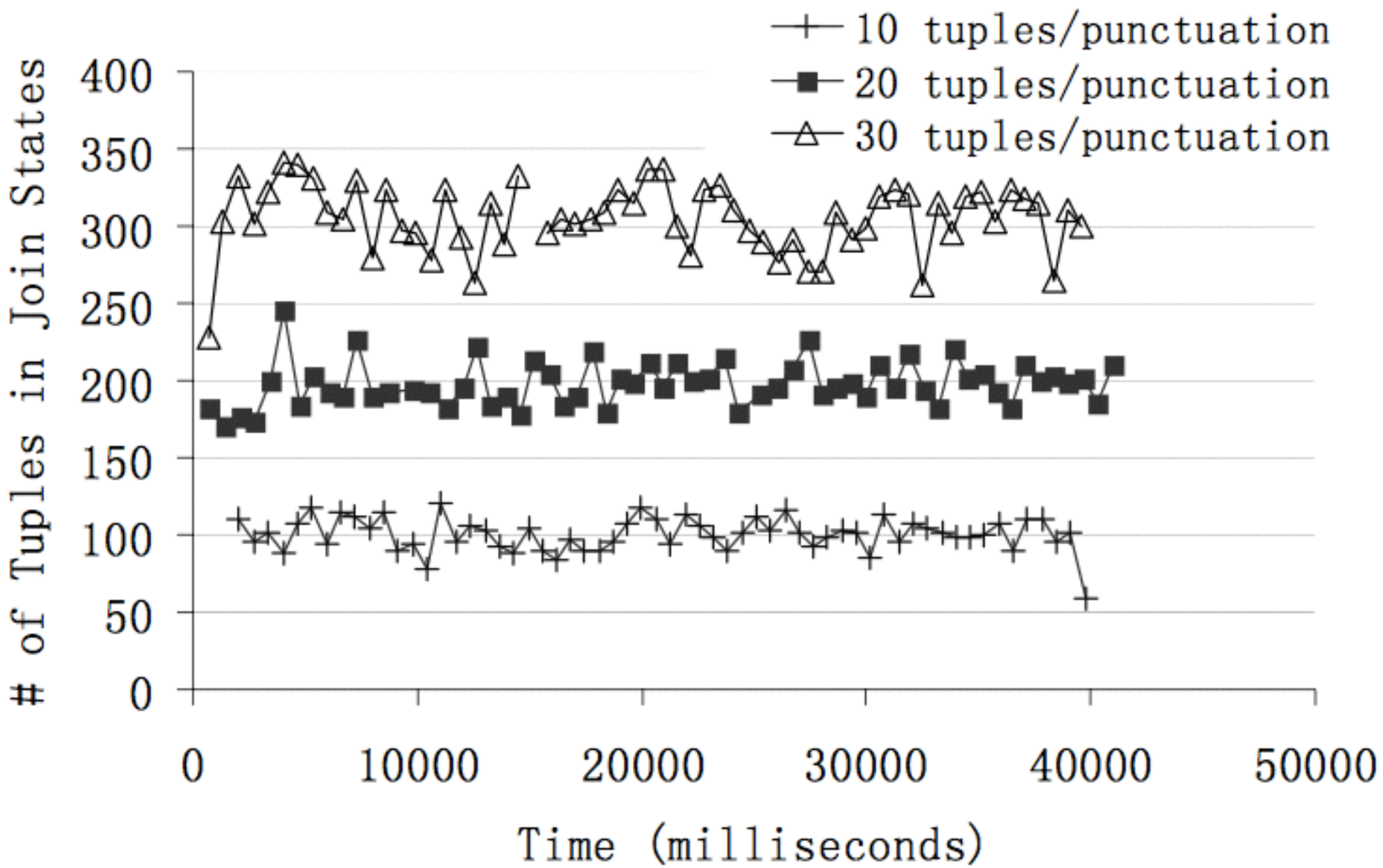
Round-2: Eager purge VS. Lazy purge

Data Throughput:



A closer look at PJoin - 1:

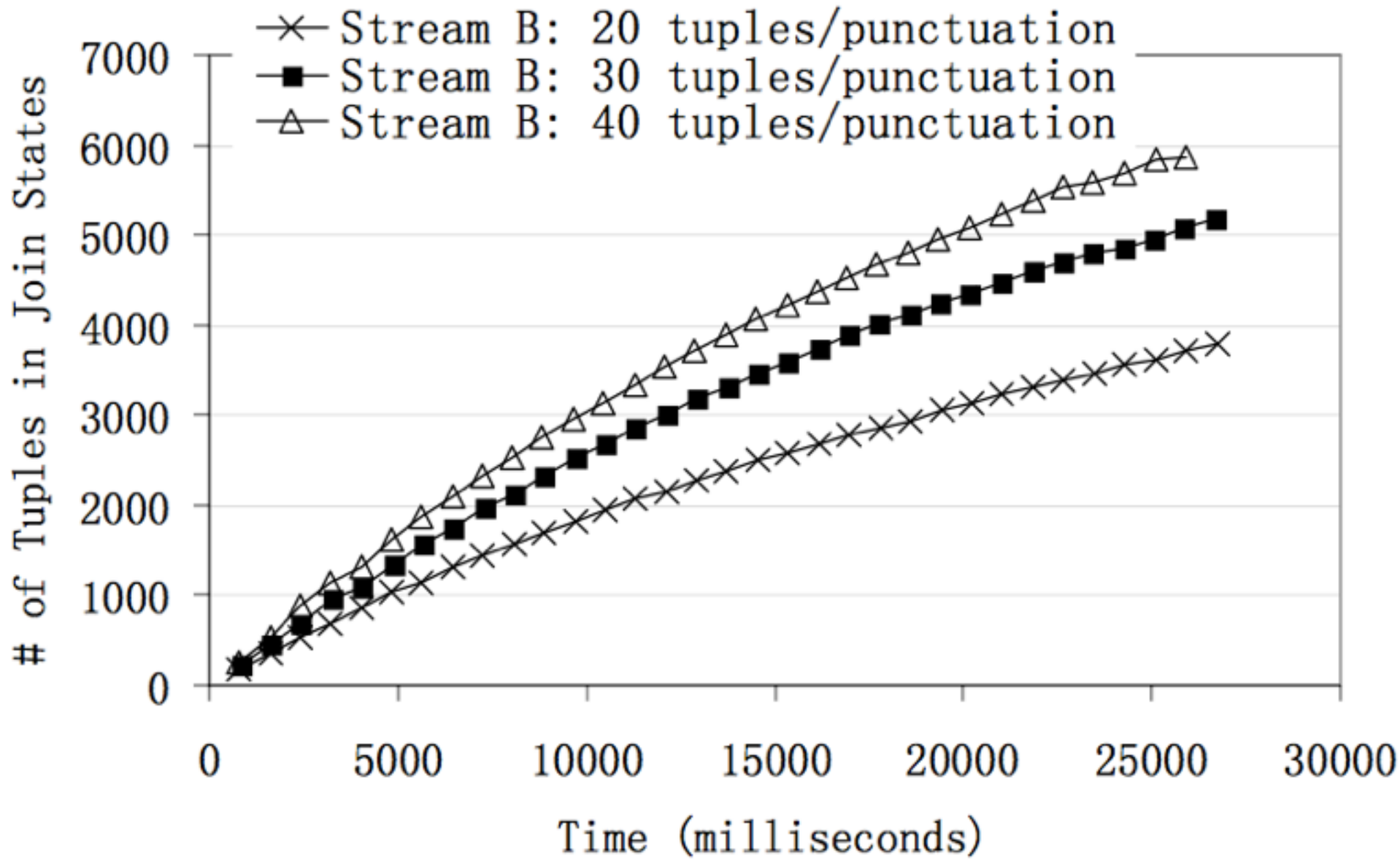
Different punctuation inter-arrival time



A closer look at PJoin - 2:

Asymmetric punctuation inter-arrival time

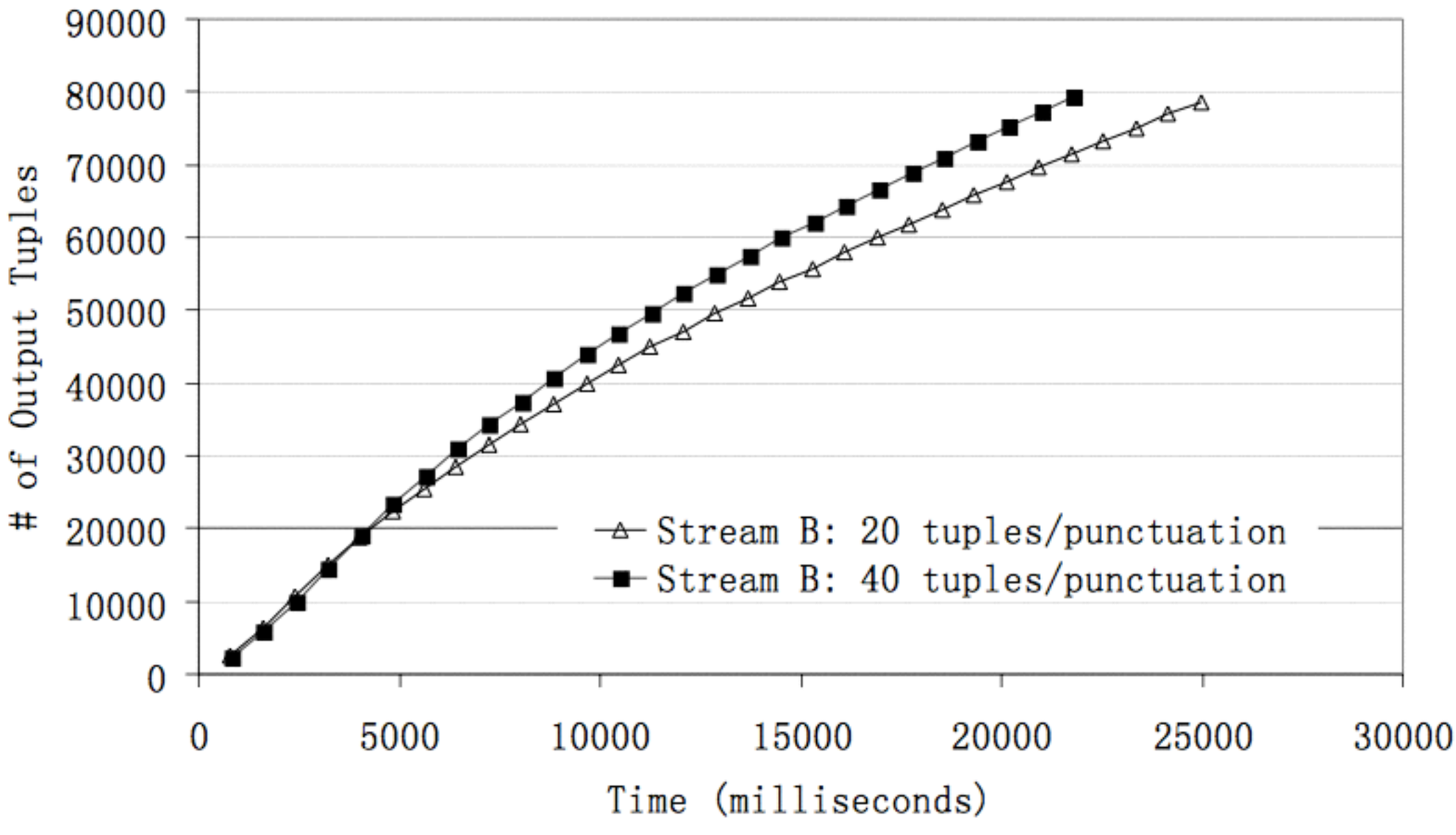
Memory Overhead:



A closer look at PJoin - 2:

Asymmetric punctuation inter-arrival time

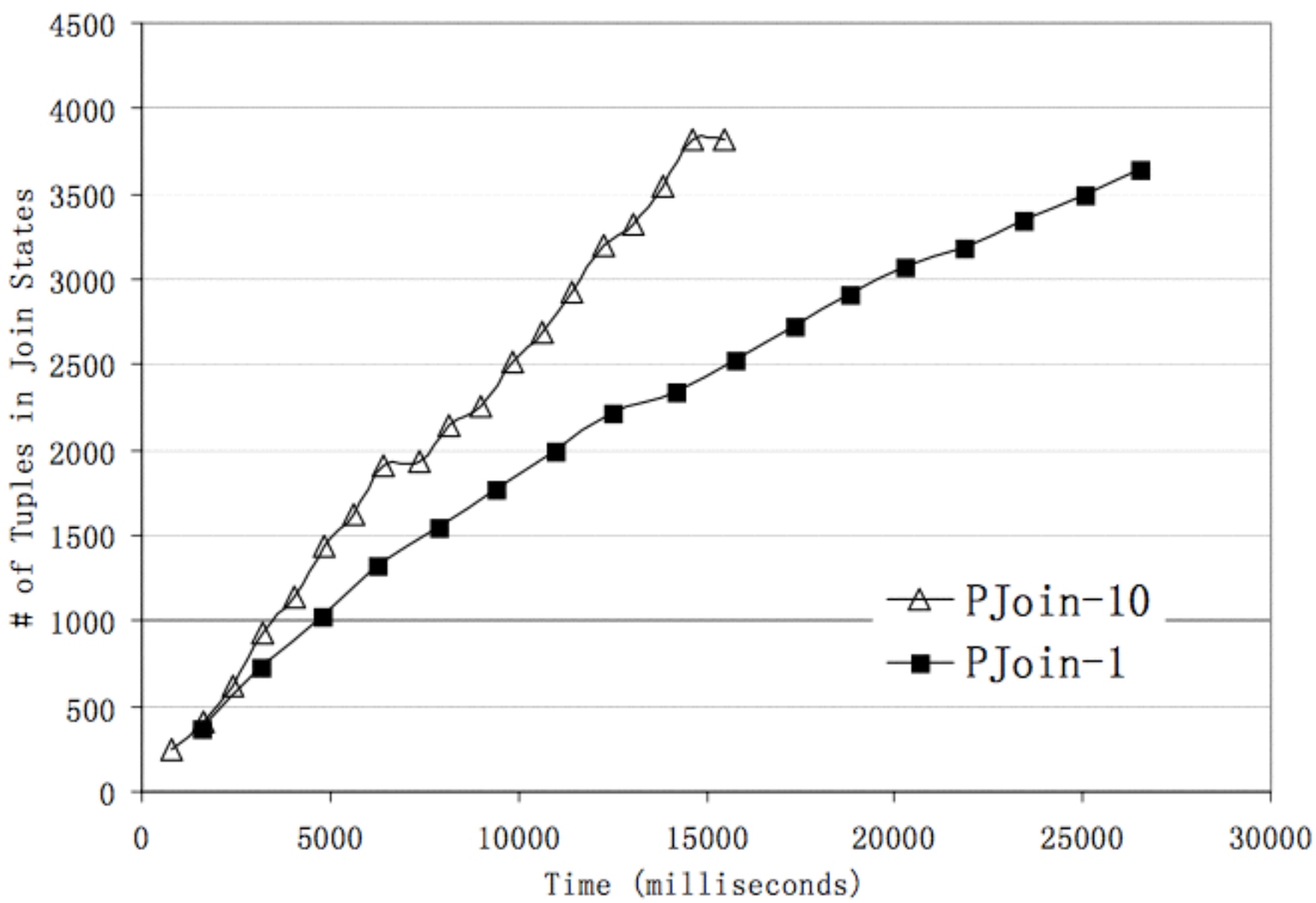
Data Throughput:



A closer look at PJoin - 2.5:

Asymmetric punctuation inter-arrival time Combined with Different Purge Modes

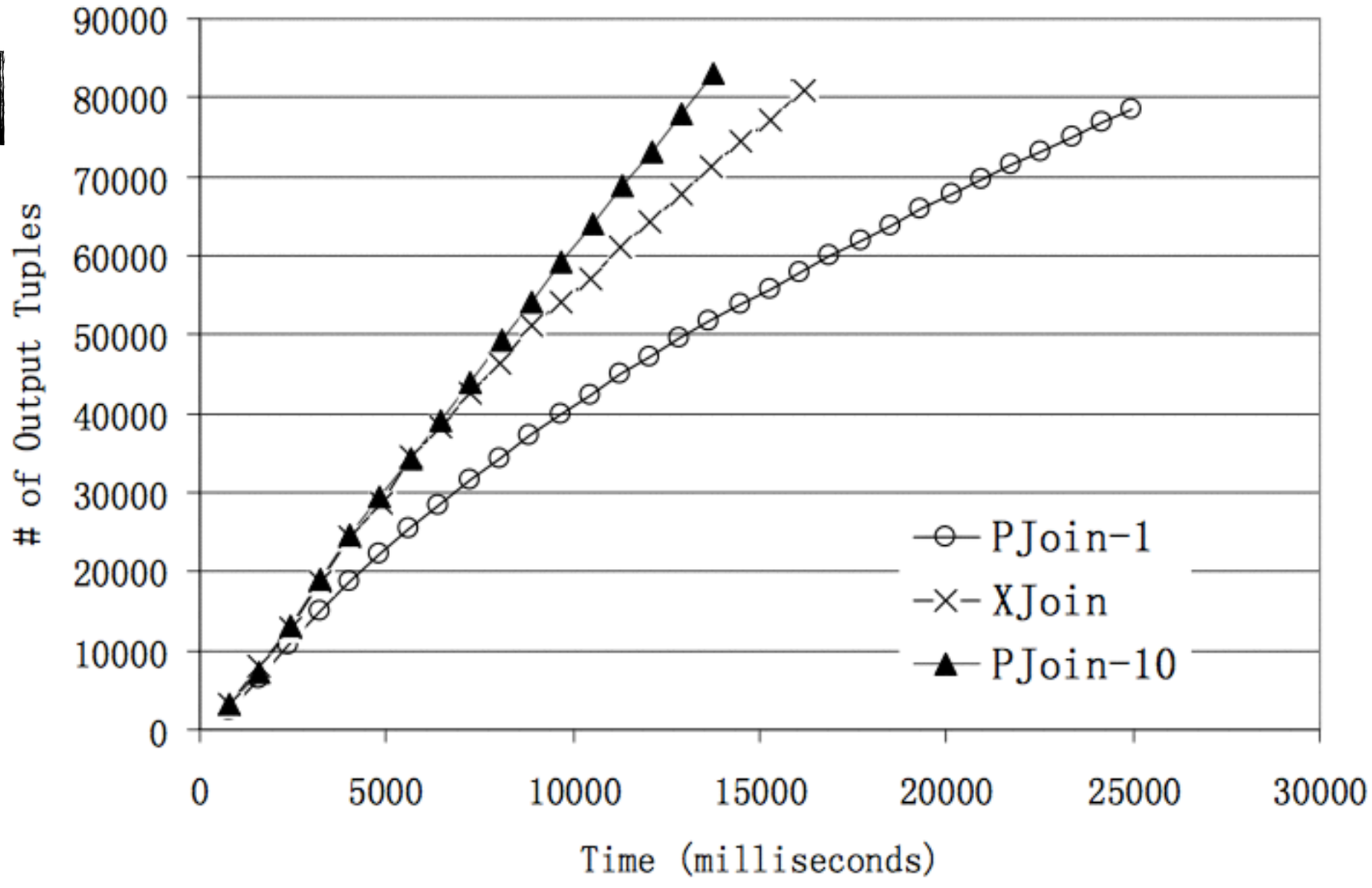
Memory Overhead:



A closer look at PJoin - 2.5:

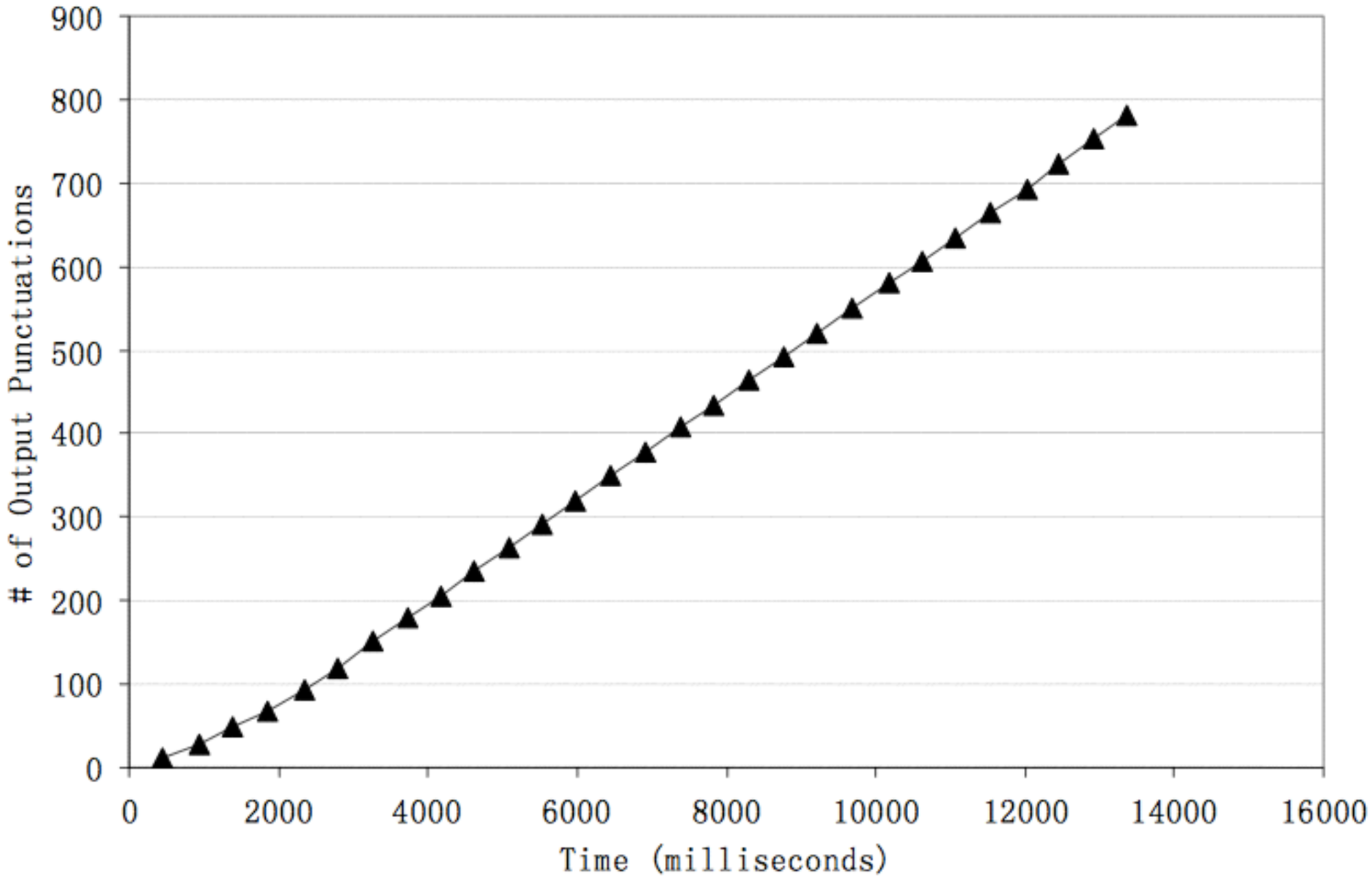
Asymmetric punctuation inter-arrival time Combined with Different Purge Modes

Data Throughput



A closer look at PJoin - 3:

Punctuation Propagation Capability



Future Work

- To support sliding window
- To support n-ary join

Reference

Joining Punctuated Streams, Luping Ding, Nishant Mehta, Elke A. Rundensteiner, and George T.