INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

ICPP/2021/CHICAGO/USA

acm In-Cooperation

sighpc

AUGUST 9-12, 2021

Efficient Complete Event Trend Detection over High-Velocity Streams

Huiyao Mei[1], **Hanhua Chen**[1], **Hai Jin**[1], **Qiang-shen Hua**[1], **Bing Bing Zhou**[2]

[1]Huazhong University of Science and Technology

[2]The University of Sydney

BDTS SCTS CGCL

Complete Event Trend Detection

E-commerce

Anti-terrorism Monitoring

Financial Transaction

Campus loan

Stock Trading

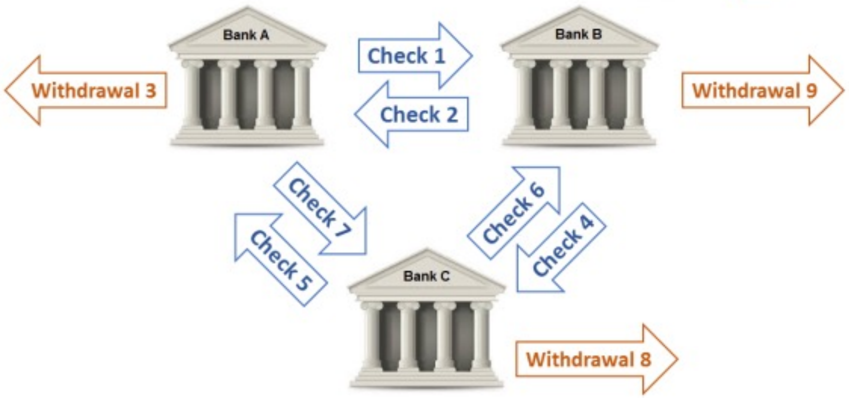
Price rise or fall

Cluster Monitoring

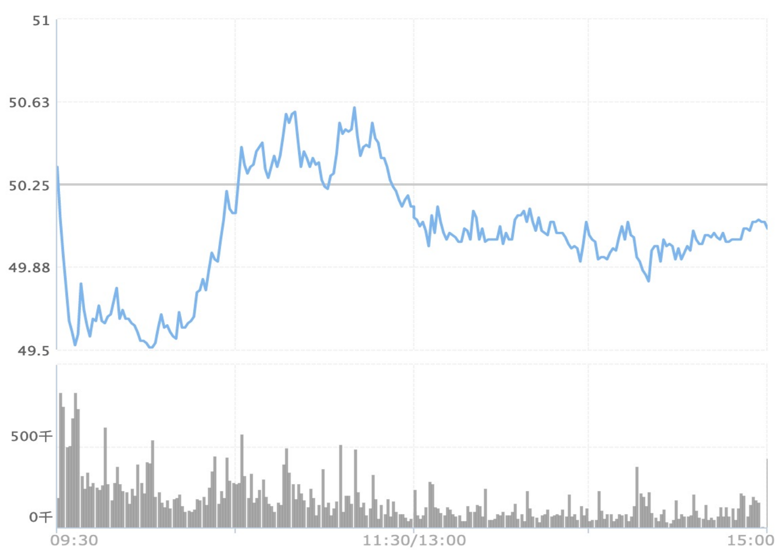
Load imbalance

- According to the user-defined event pattern, detecting all matching event sequences from high-speed event streams,

Complete Event Trend Detection



$$
\begin{array}{lll}
Q1: & PATTERN & check+\ c[\,] \\
 & WHERE & c.type='not\ covered'\ AND \\
 & & c[i].src=c[i-1].dest \\
 & WITHIN & 1\ week\ SLIDE\ 1\ day
\end{array}
$$



$$
\begin{array}{lll}
Q2: & PATTERN & stock+\ s[\,] \\
 & WHERE & [id]\ AND\ s[i].price > ratio * s[i-1].price \\
 & WITHIN & 60\ min\ \ SLIDE\ 10\ min
\end{array}
$$

Challenges

- Massive Partial Results
  - Partial results must be stored in memory to match with ongoing events
  - One partial result can match any number of events
- High Performances Requirement
- Difficult to be parallized

State-of-the-art Works

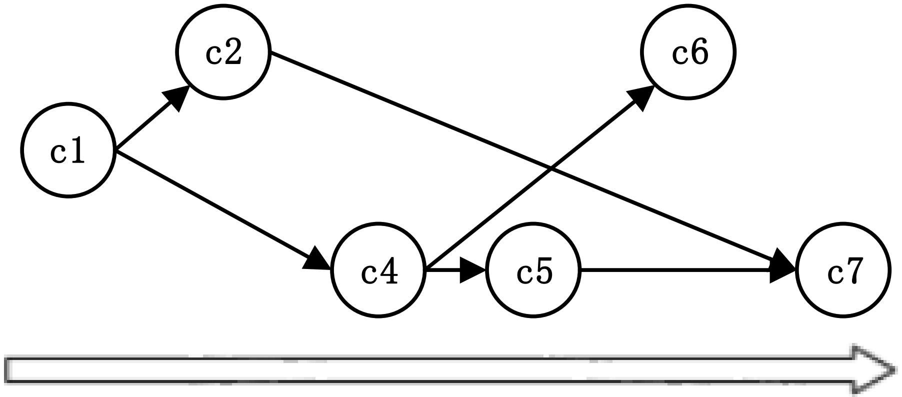
- Detection - CET Graph[1]
  - High graph construction latency
  - Impractical graph partition method
- Parallelization Solution - Coarse-grained
  - pipeline-based [2]
  - Key-based[3]

[1]: Olga Poppe, et al. Complete Event Trend Detection in High-Rate Event Streams. SIGMOD'2017

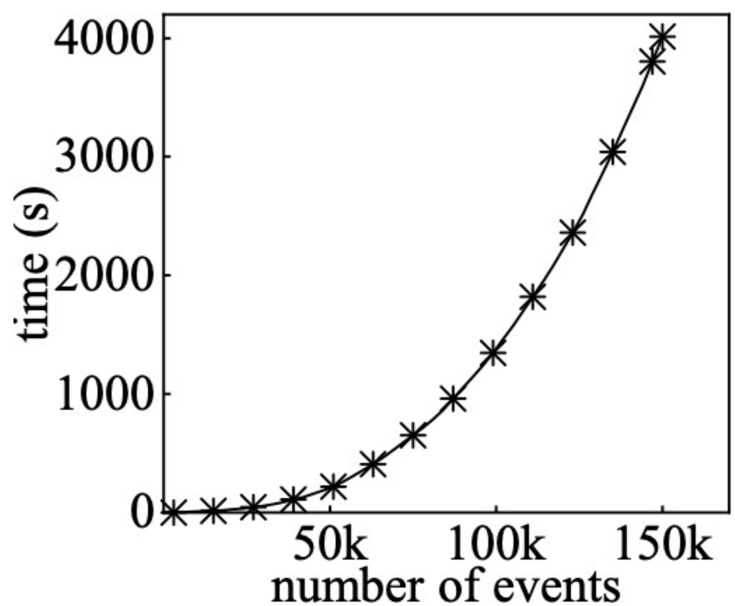
[2]: Cagri Balkesen, et al. RIP: run-based intra-query parallelism for scalable complex event. DEBS'2013

[3]: flink CEP. https://flink.apache.org/

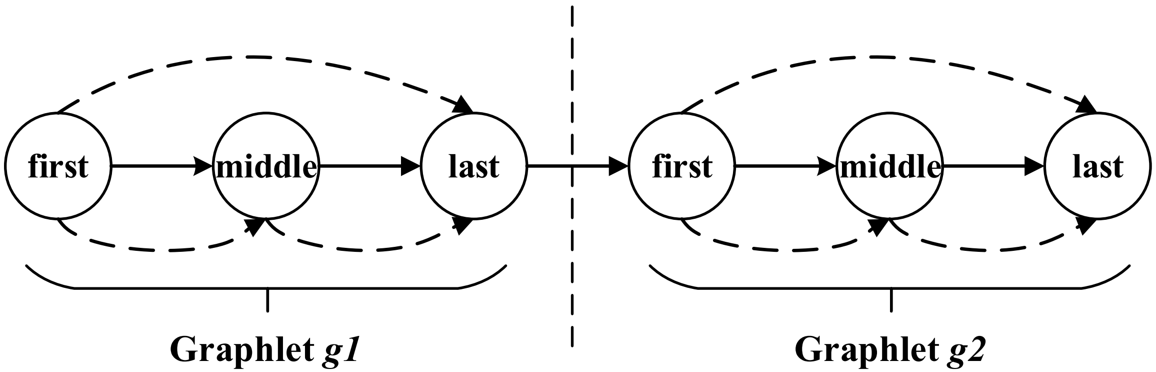
CET Graph — Construction



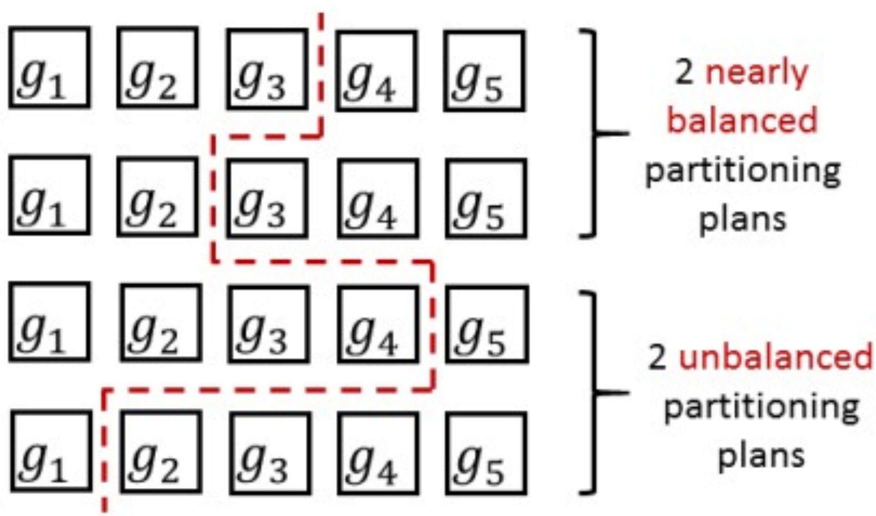
- CET Graph： share common event sequences by paths
- Traverse to find matches
  - $O(|V|^2)$ time and space costs



CET Graph - Extraction



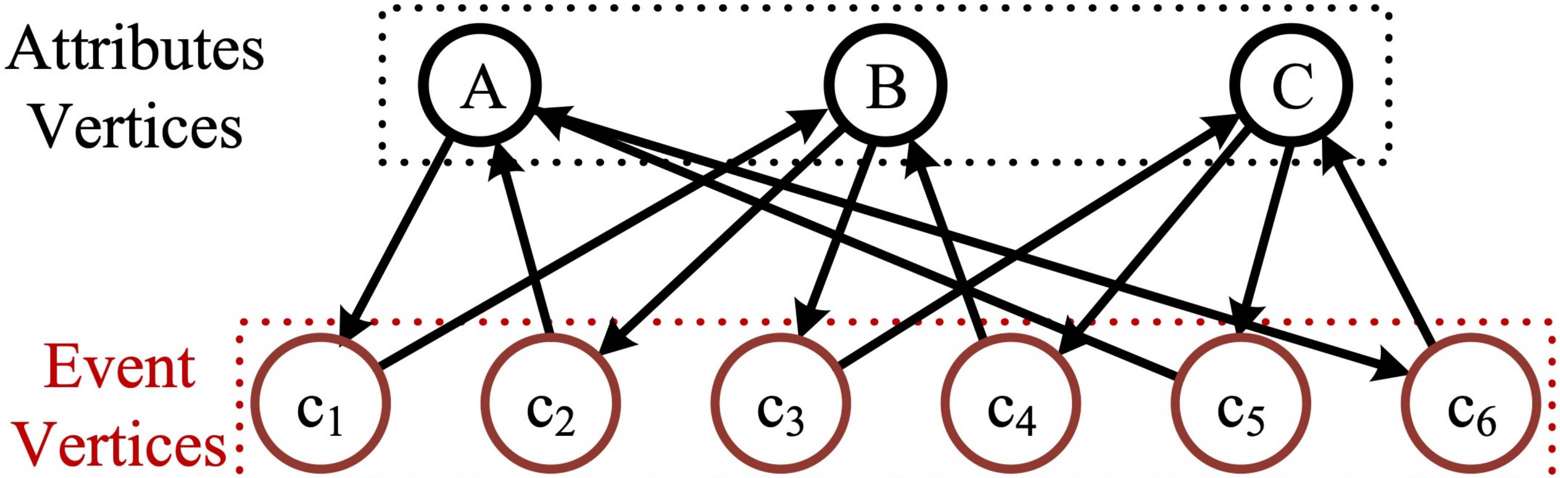
- Only connect by last and first vertices
- Imbalanced graphlets



- |partition| < avg + |g1|
- exponential time complexity

ABI Graph Model

- Event Vertex → Attribute Vertex (to-edge)：the event is relevant to the attribute value
- Attribute Vertex →Event Vertex (from-edge)：the attribute value of the event

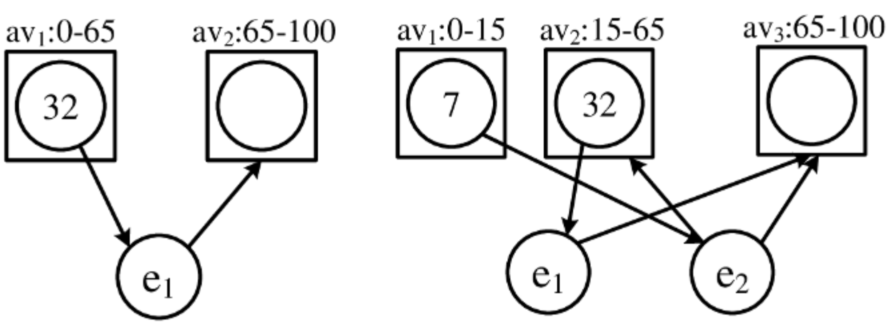


Dynamic Graph Construction

- Composite Attribute Vertex
  - respond to many attribute values
  - most of attribute values are unnecessary
- **A Stipulation: values of a composite attribute vertex must only:**
  - **be all relevant to an event**
  - **or all irrelevant to an event**
- Violate (Composite) Attribute Vertex is split

$$e[i].value > 2 * e[i-1].value$$

event	e_1	e_2	e_3	e_4	e_5	e_6
value	32	7	15	35	40	17

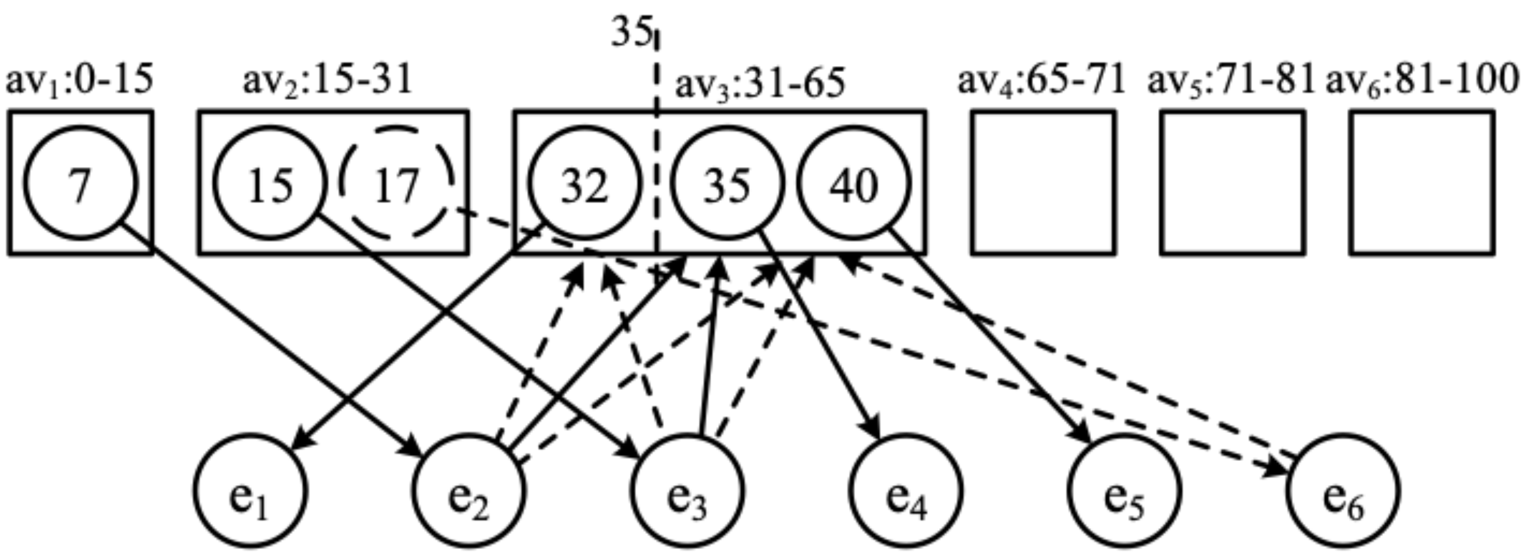


Dynamic Graph Construction

- from-edge： assigned by the value of its source attribute vertex
- to-edge： copied and pointed to all child attribute vertex

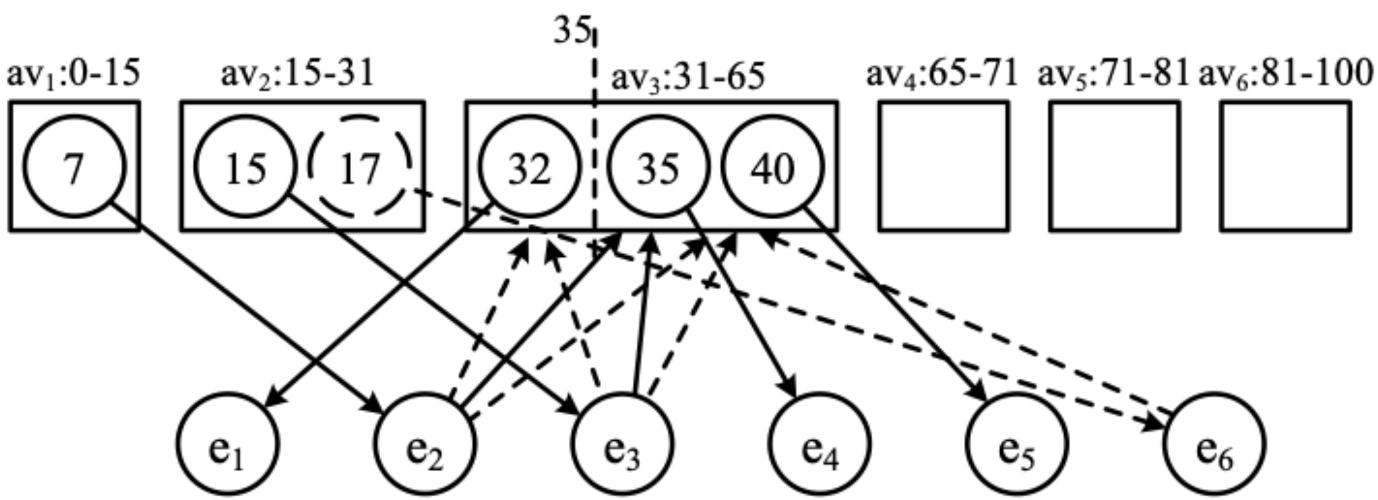
$$e[i].value > 2 * e[i-1].value$$

event	e_1	e_2	e_3	e_4	e_5	e_6
value	32	7	15	35	40	17



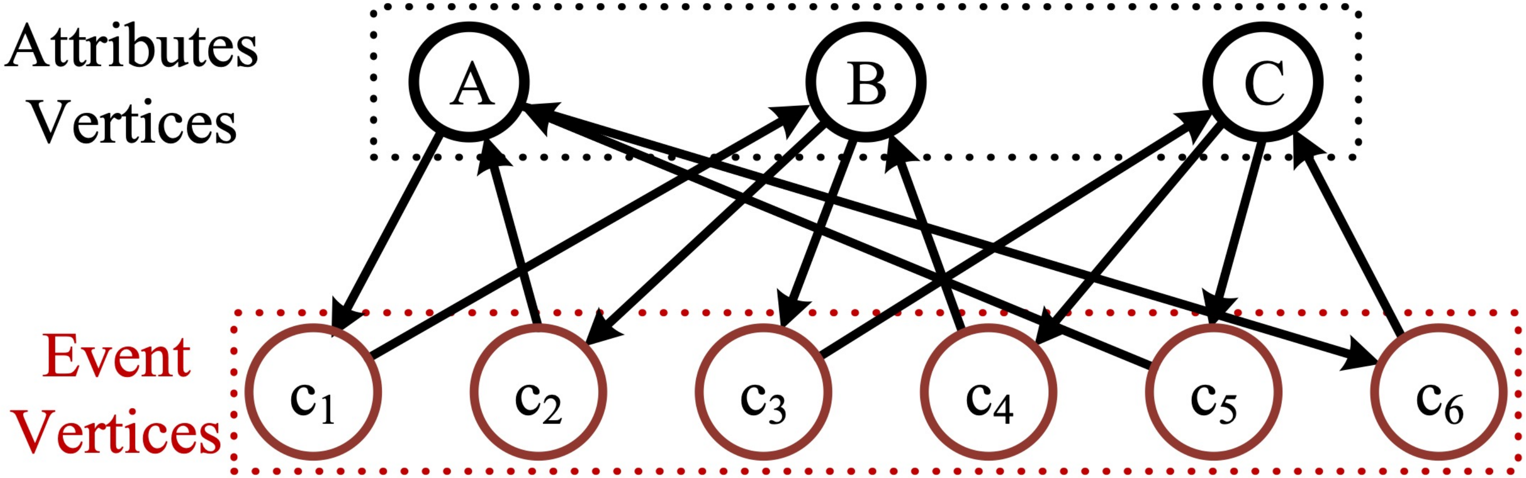
Parallel Graph Construction

- Read-Write Conflict
  - Write: split violated attribute vertices
  - Read: search attribute vertices
- Parallel Solution – postponed construction
  - the split point only relate to event and predicate
  - construct the graph after all split points are calculated



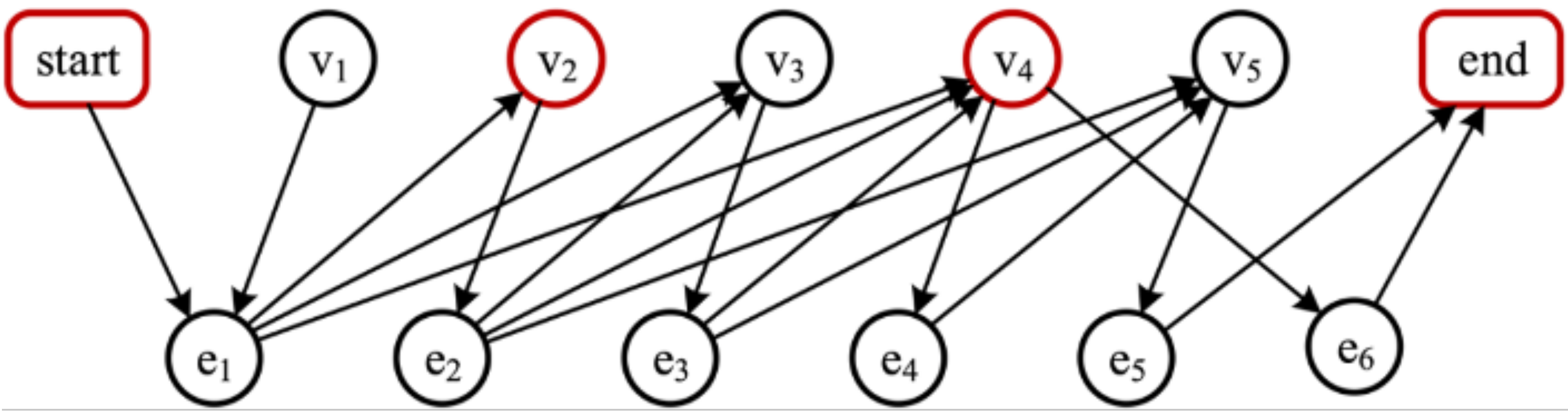
CET Path In ABI Graph

- two condition for relevance of two events：
  - their corresponding event vertices are connected by an attribute vertex
  - following the direction of edges, the timestamp of the later event is greater than that of the previous
- start（end）event vertex：the start or end event vertex of CET paths

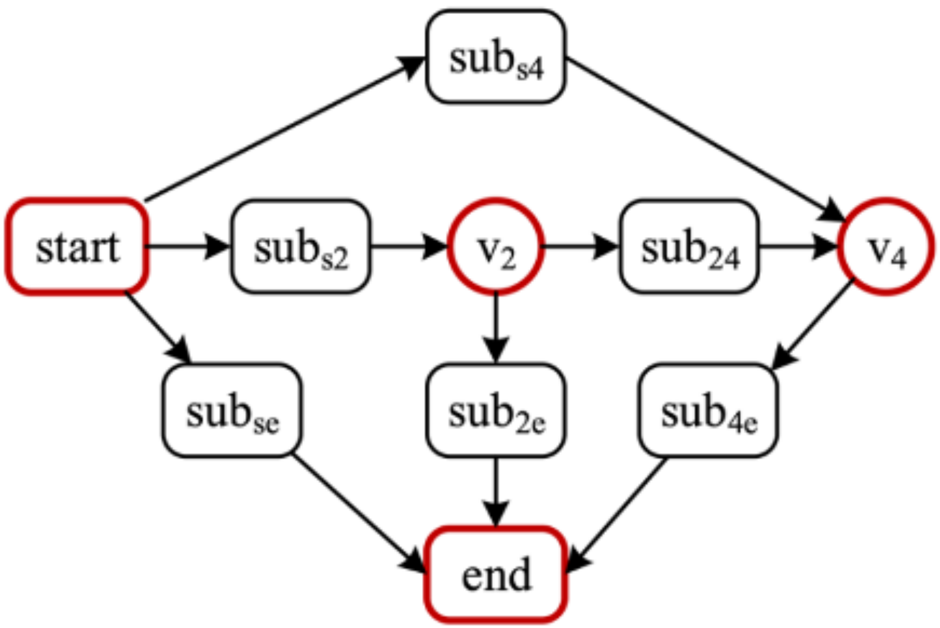


Anchor-based Extraction – BFS stage

- anchors： evenly selected attribute vertices
- start anchor： points to all start event vertices
- end anchor： pointed by all end event vertices
- BFS stage: between two anchors



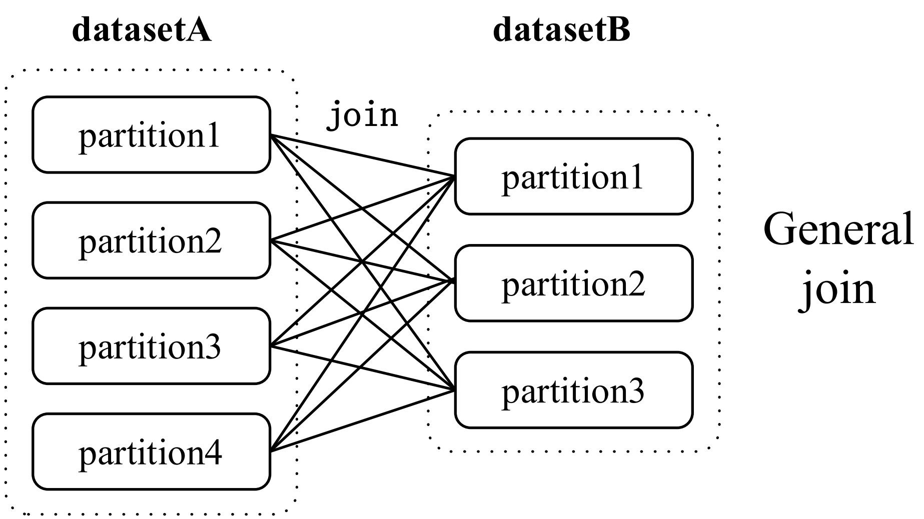
Anchor-based Extraction – DFS stage

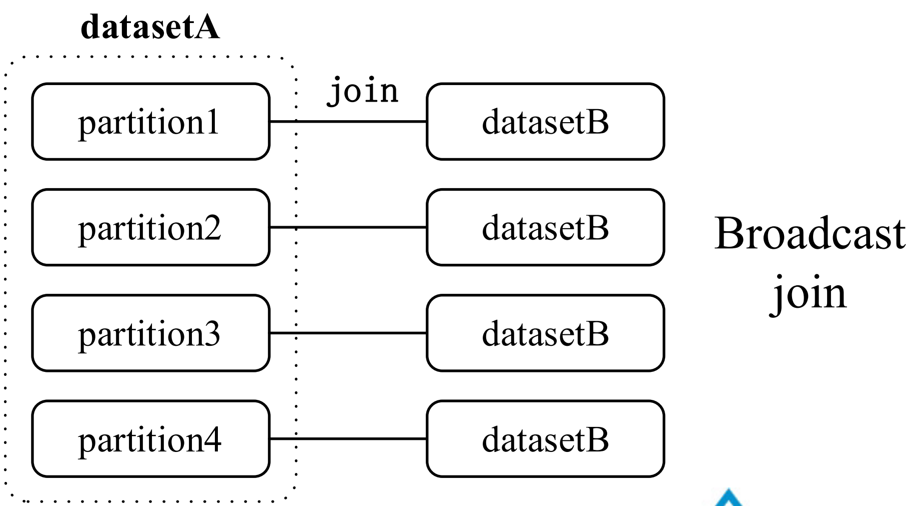


sub_{s2}	(e_1)
sub_{s4}	$(e_1),(e_1,e_3)$
sub_{se}	(e_1,e_3,e_5)
sub_{24}	$(e_2),(e_2,e_3)$
sub_{2e}	(e_2,e_5)
sub_{4e}	$(e_4,e_5),(e_6)$

Broadcast Join-based Extraction

- General Graph Processing
  - large scale graph, not large scale message
  - transfer by general join
- CET Extraction
  - small scale graph， large scale message
  - transfer by broadcast join



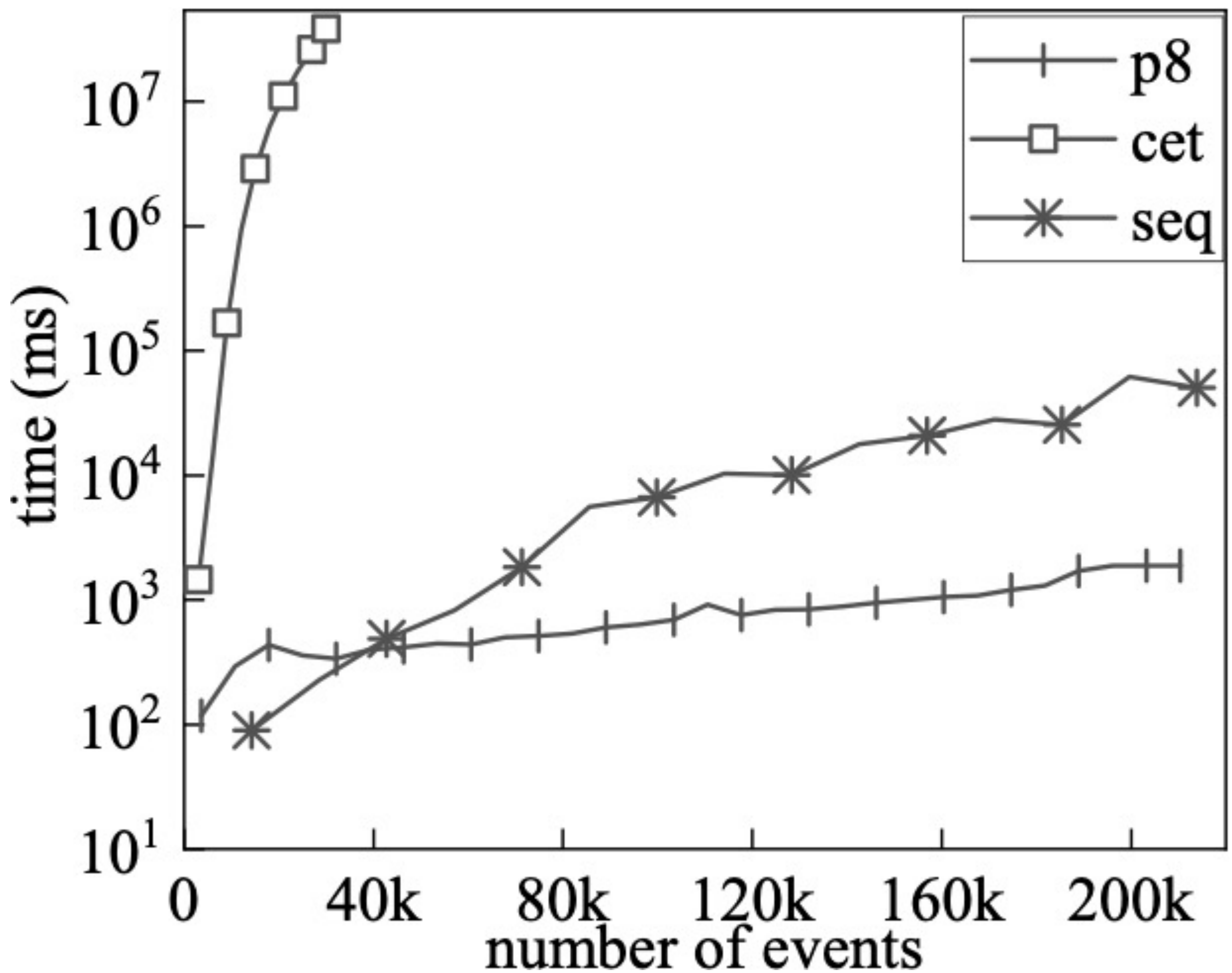


Experiment Setup

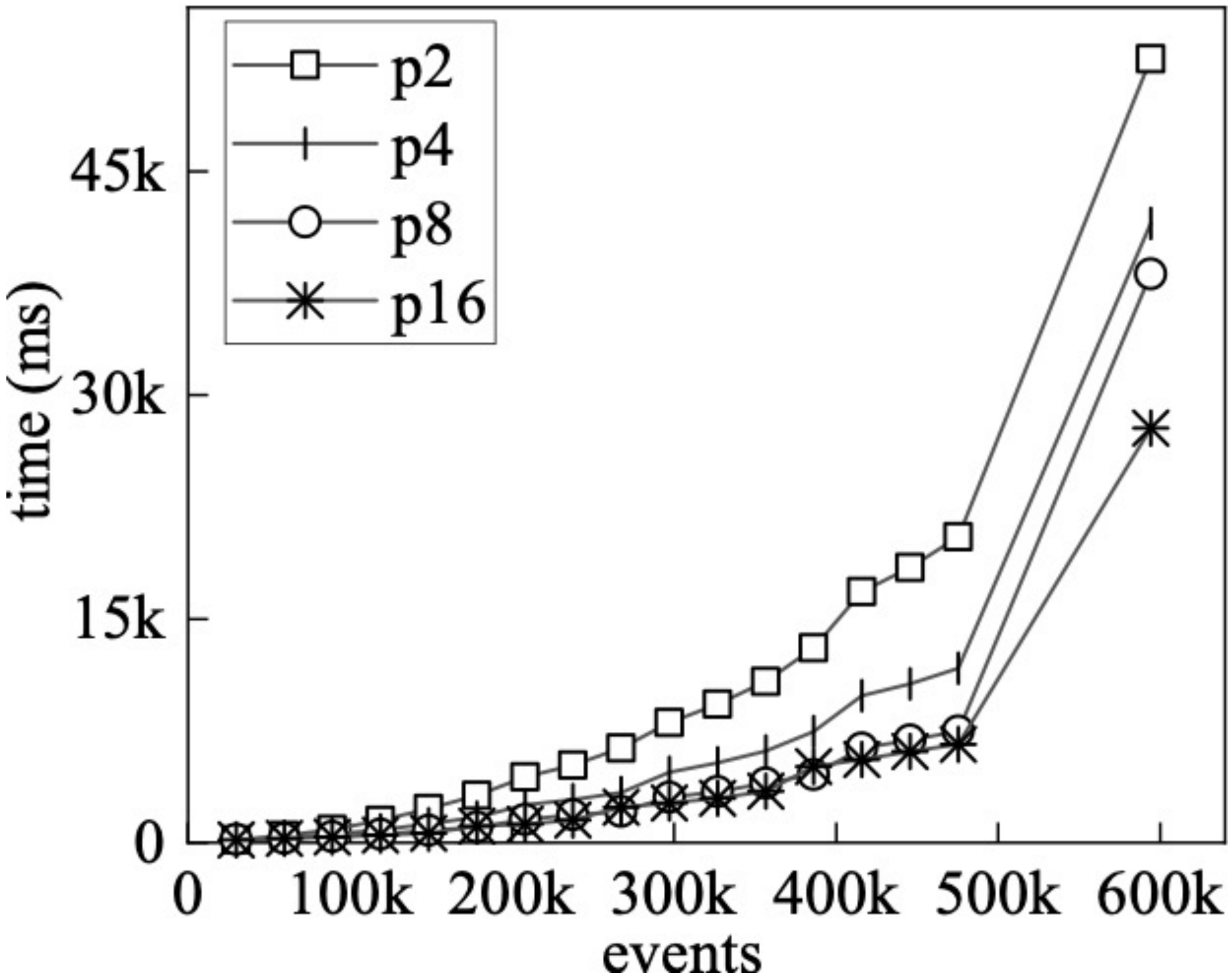
CPU	Intel Xeon(Cascade Lake) Platinum 8269CY 16cores
Memory	64GB
Disk	SATA 40GB
Operating System	CentOS 8.0
Machines	6
Spark Version	3.0.1

- Dataset 1: crawled stock data and copy for 5000 times
  - 17M events
  - timestamp, price
- Dataset 2: generated check dataset
  - timestamp, source bank account, destination bank account

Graph Construction



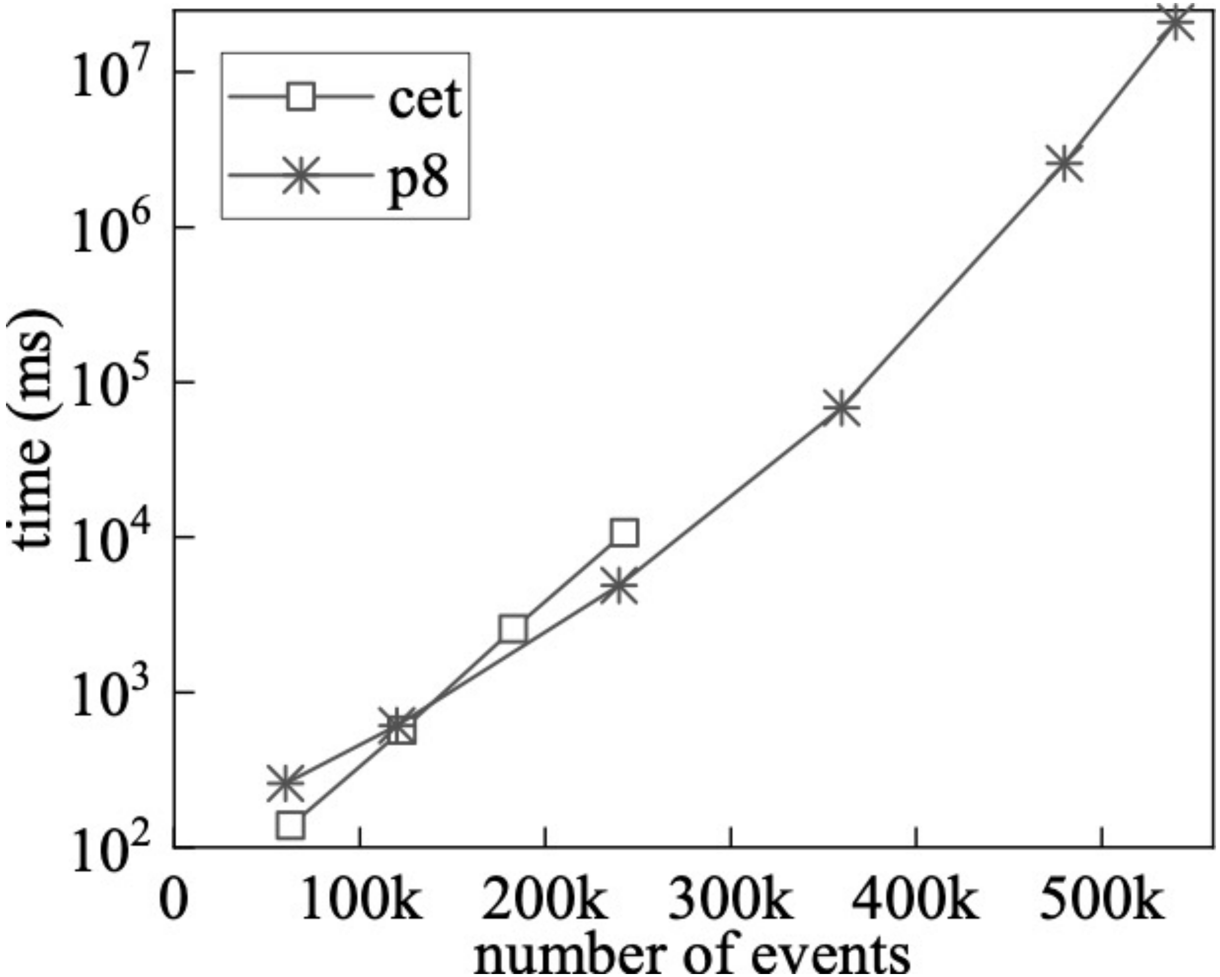
Graph Construction



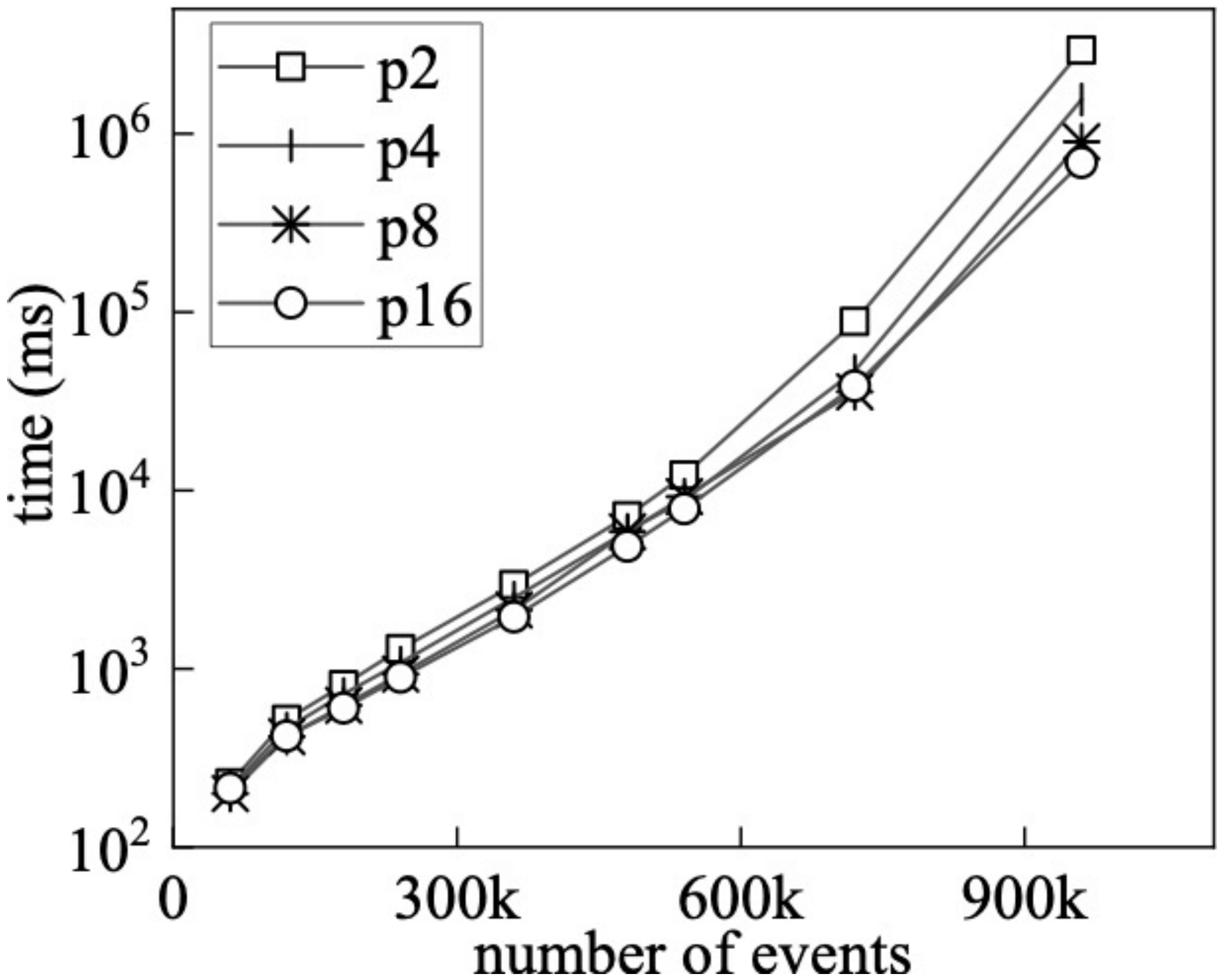
INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation sighpc

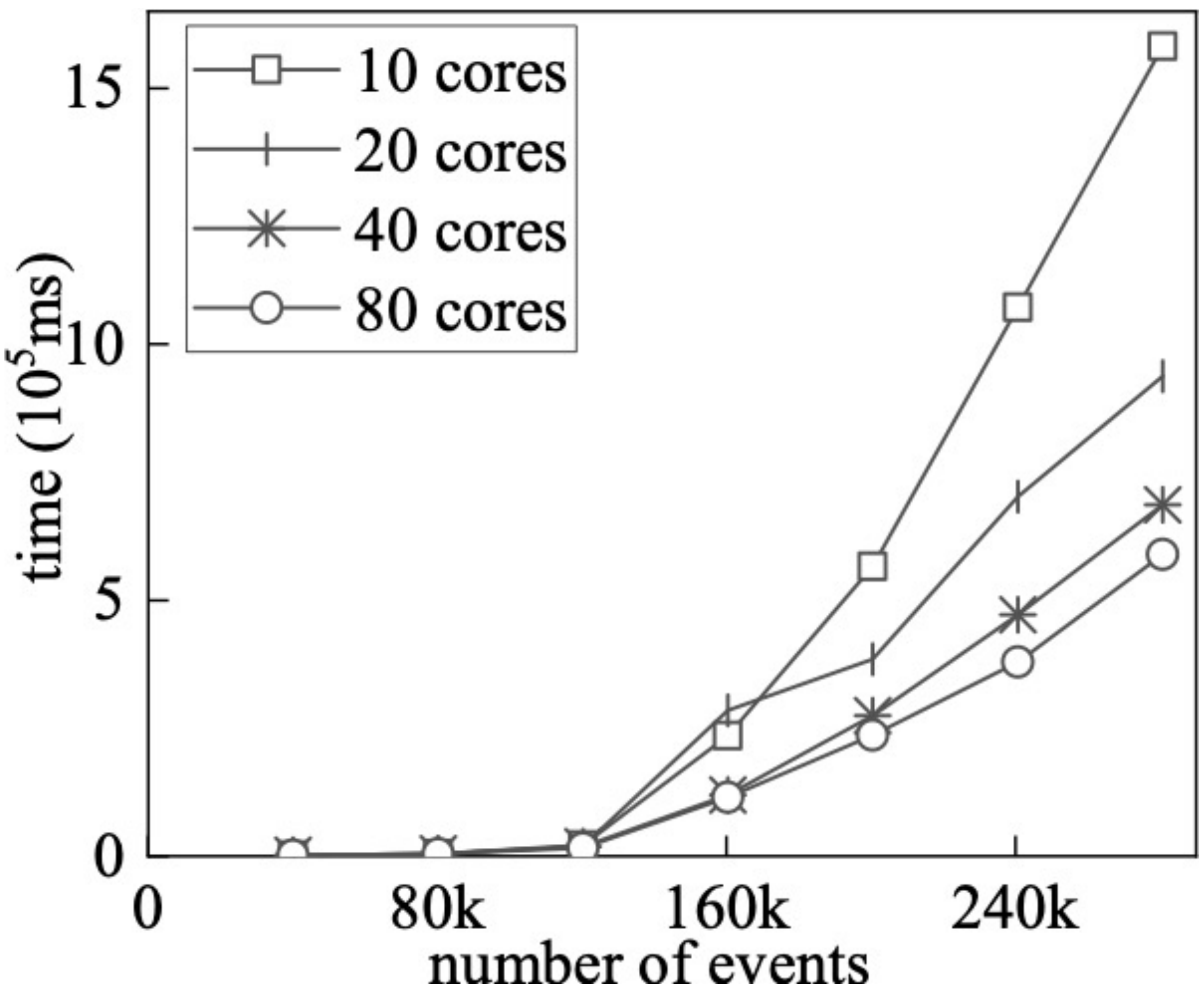
CET Extraction



CET Extraction



CET Extraction



Thank you for your attention!

Source code available at:
https://github.com/CGCL-codes/DynamicTG