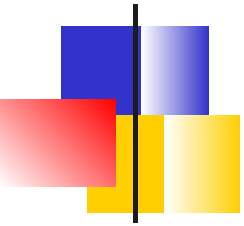


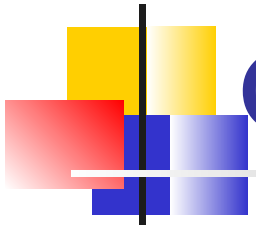
Bi-Level Online Aggregation On Raw Data

Yu Cheng[+], Weijie Zhao[*], Florin Rusu[*]

+: Amobee. Inc.

*: University of California, Merced

UCMERCED



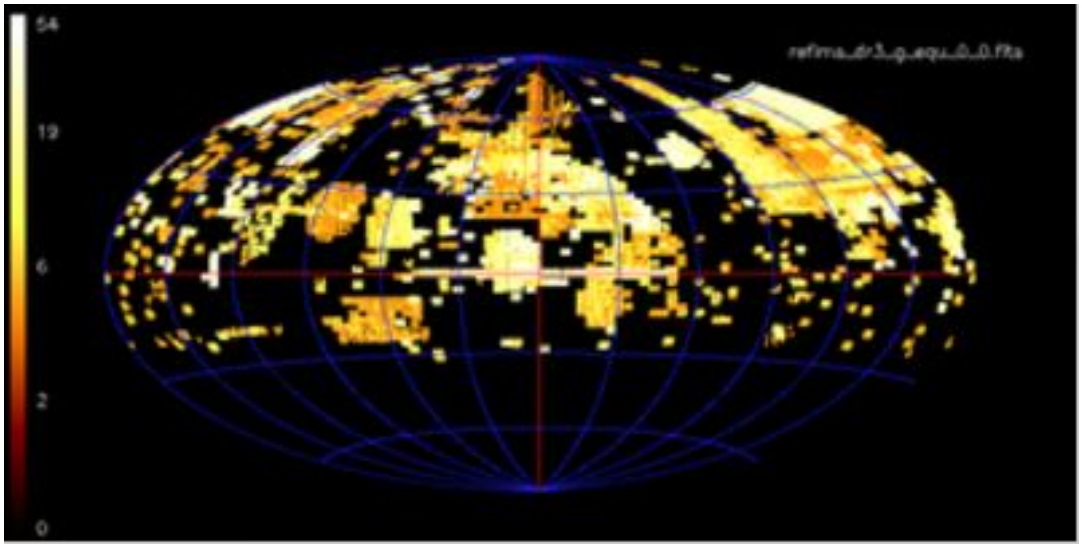
Outline

- Background
- Problem
- OLA-RAW
- Evaluation

UCMERCED

Palomar Transient Factory (PTF)

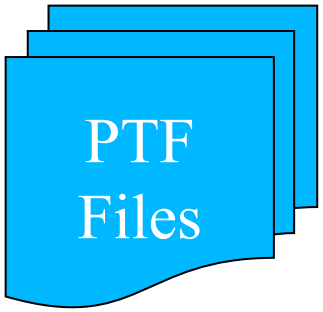
The Palomar Transient Factory (PTF) project aims to identify and automatically classify transient astrophysical objects such as variable stars and supernovae in realtime



UCMERCED

Illustrative Example

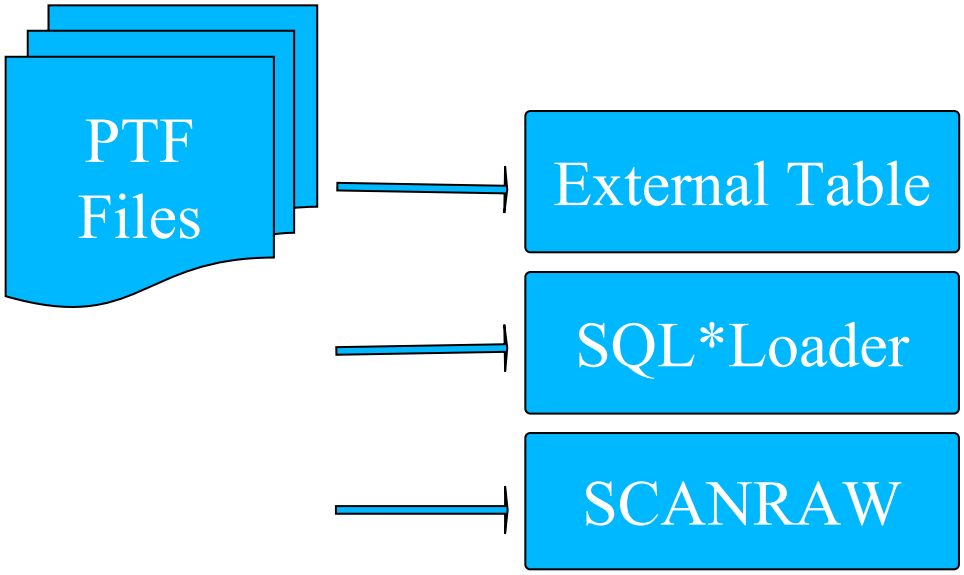
- Supernova identification




```
SELECT AGGREGATE(expression) AS agg
FROM candidate
WHERE predicate
HAVING agg < threshold
```

UCMERCED

Existing Solutions

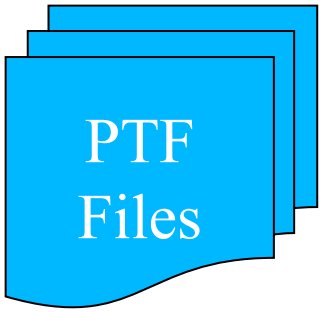


Time to query	Execution	Storage
instant	slow	zero
loading	fast	full replication
instant	fast	adaptive

UCMERCED

Illustrative Example

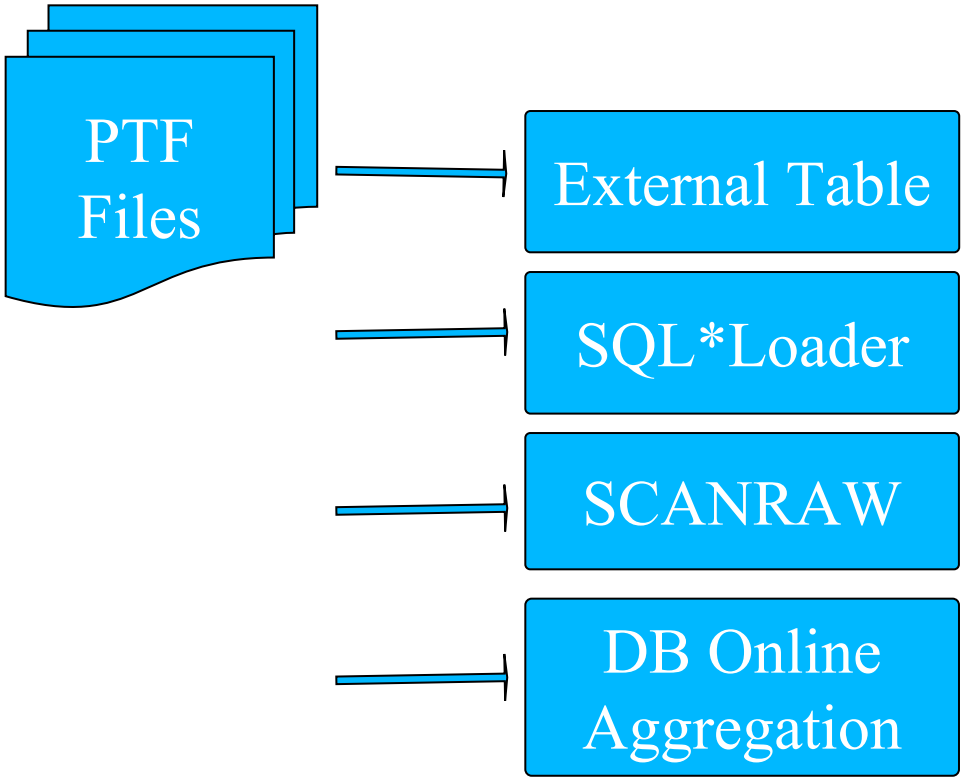
- Supernova identification



SELECT AGGREGATE(expression) AS agg
FROM candidate
WHERE predicate
HAVING agg **> threshold**
WITH ACCURACY α

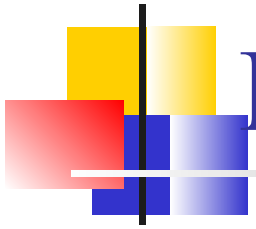
UCMERCED

Existing Solutions



Time to query	Execution	Storage
instant	slow	zero
loading	fast	full replication
instant	fast	adaptive
loading + shuffling	faster	full replication

UCMERCED



Research Problem

- Can we find a better solution to execute approximate queries in-situ over raw files?
  - ➢ Instant access to data

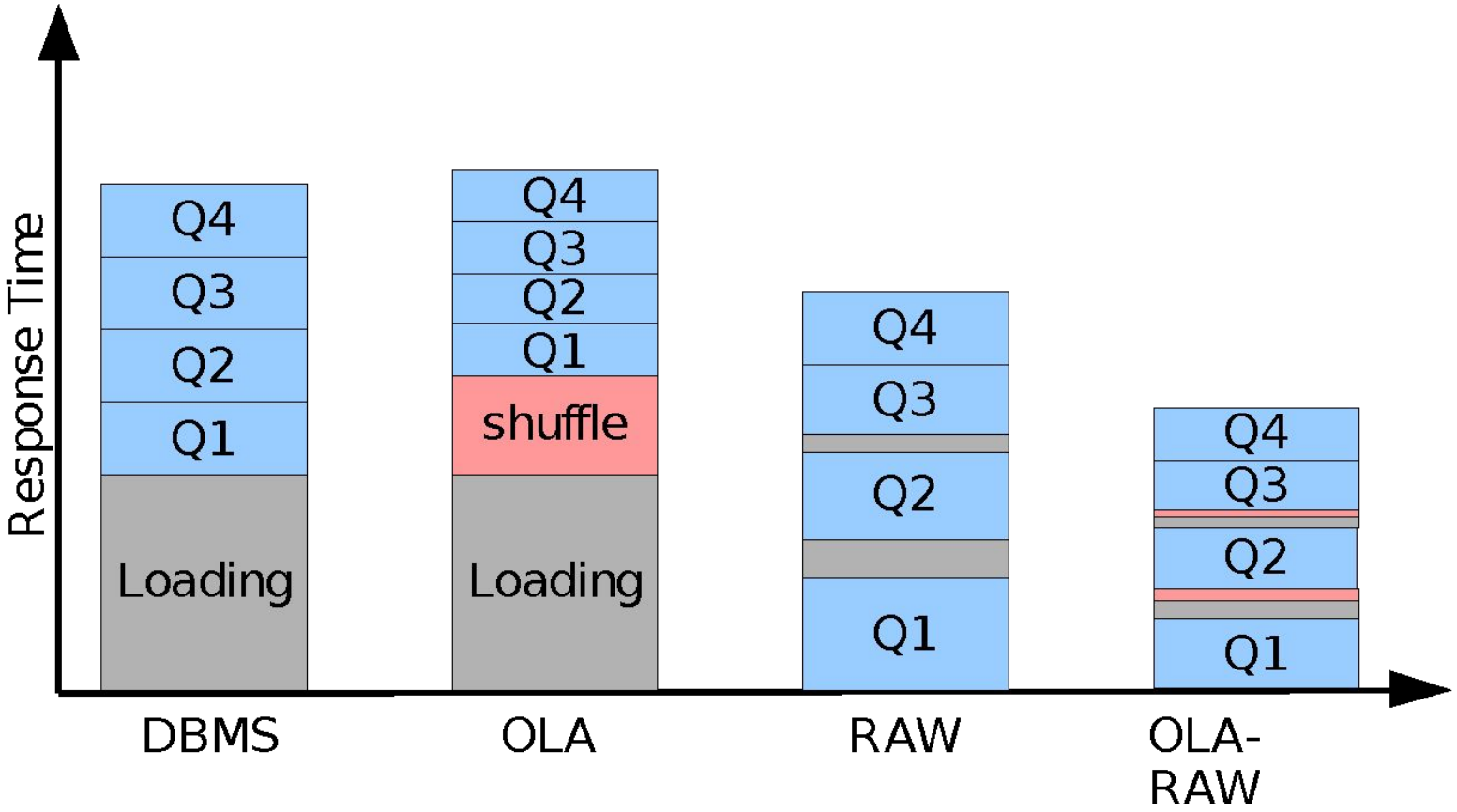
    In-situ data processing
  - ➢ Generate results faster

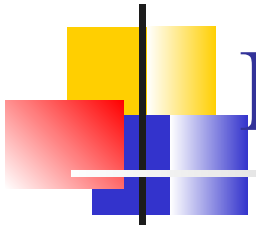
    Online aggregation (OLA)
  - ➢ Minimize used storage

    In-memory synopsis

UCMERCED

High Level Approach

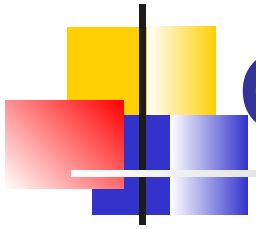




Related Work

- Adaptive partial loading [Idreos et al., CIDR 2011]
  Only load necessary attributes before query starts
- NoDB [Alagianis et al., SIGMOD 2012]
  Instead of loading, build index and cache necessary attributes in memory
- Invisible loading [Abouzied et al., EDBT/ICDT 2013]
  Portion of necessary data is loaded into database for every query
- Data vaults [Ivanova et al., SSDBM 2012]
  Memory cache for complex data in scientific repositories
- SCANRAW [Cheng and Rusu, SIGMOD 2014]
  Load data using spare system resources without affecting query processing

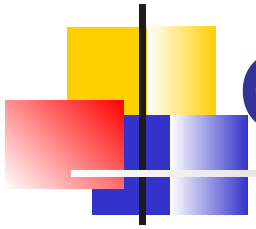
UCMERCED



OLA-RAW

- **OnLine Aggregation for RAW data processing**
  - How to generate random samples from raw files?
  - Design a feasible architecture to combine online aggregation with in-situ data processing
  - Find an efficient method to maintain extracted samples

UCMERCED

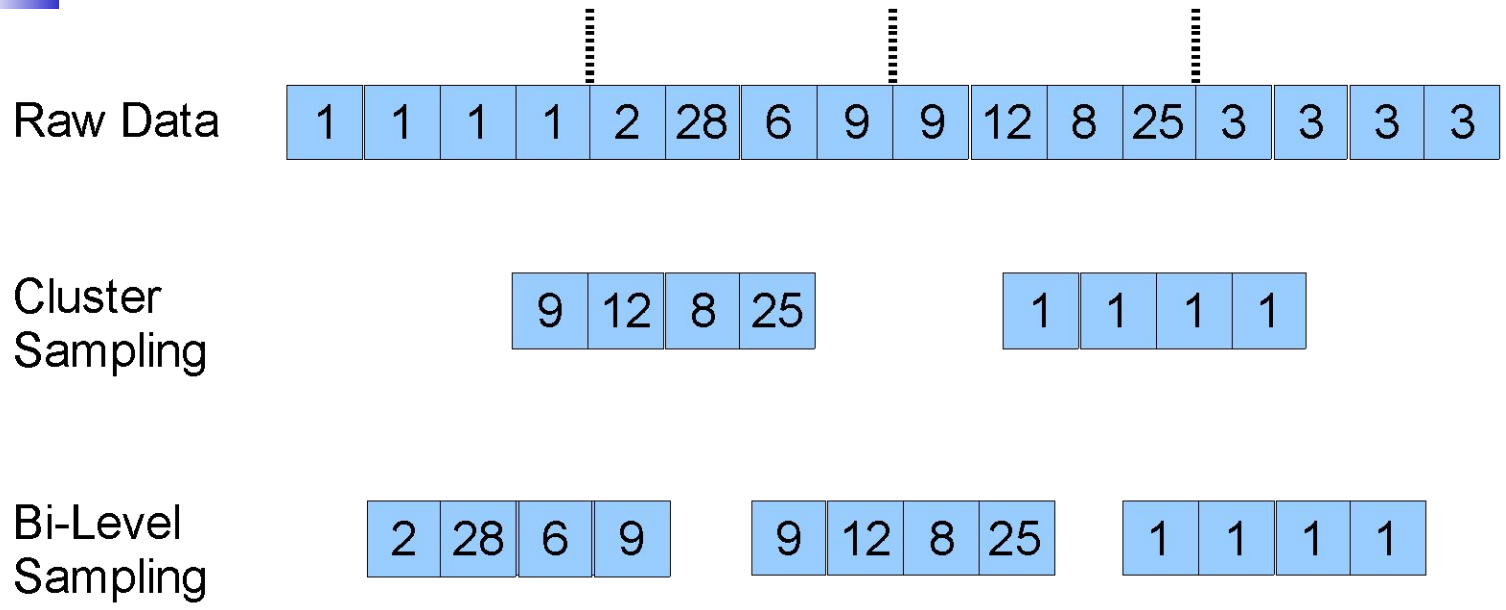


OLA-RAW

- **OnLine Aggregation for RAW data processing**
  - How to generate random samples from raw files?
    Bi-Level Sampling
  - Design a feasible architecture to combine online aggregation with in-situ data processing
  - Find an efficient method to maintain extracted samples

UCMERCED

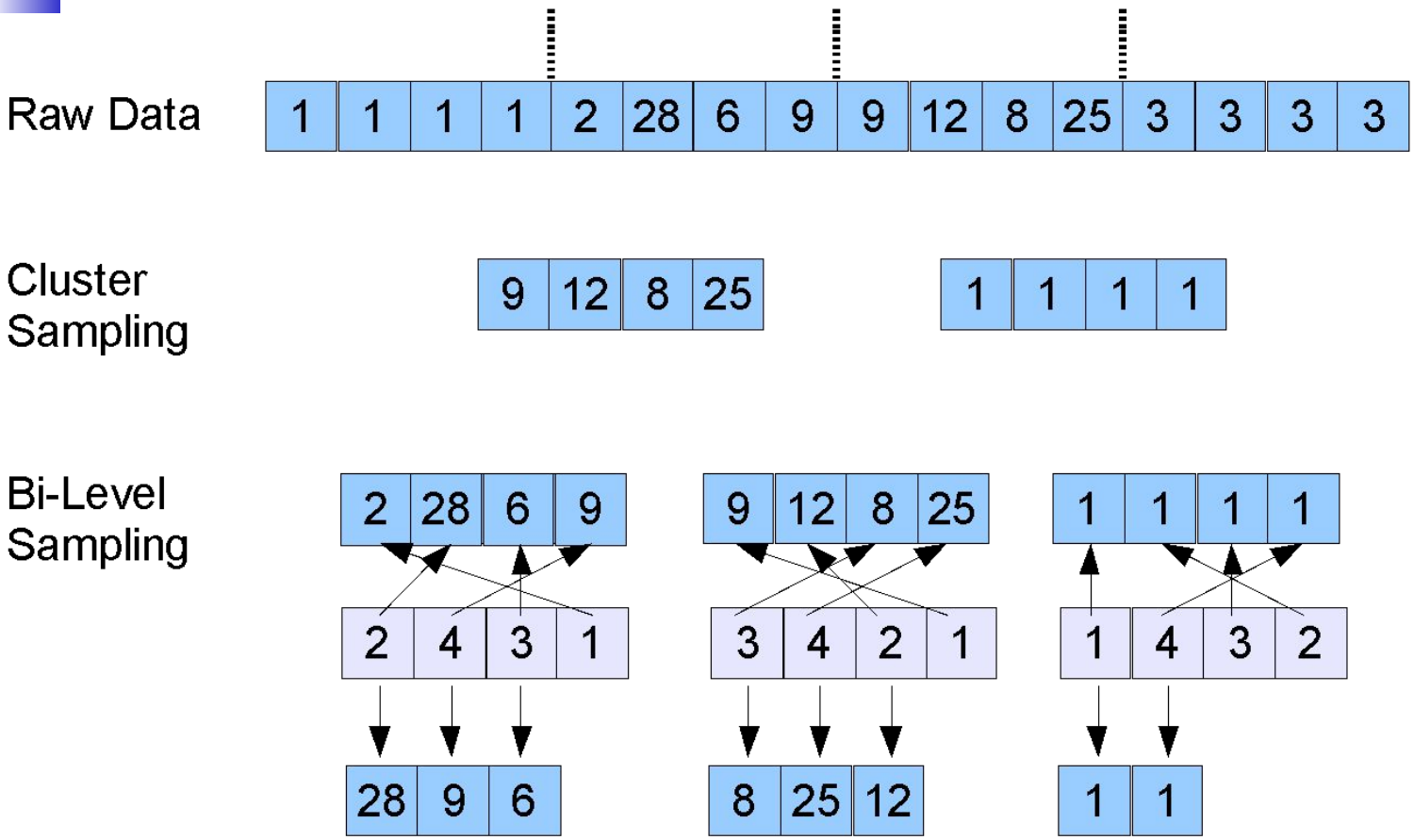
Sampling and Estimator



$$\hat{\tau} = \frac{N}{n} \sum_{i=1}^{n} \hat{y}_i \qquad \hat{y}_i = \frac{M_i}{m_i} \sum_{j=1}^{m_i} y_{ij}$$

UCMERCED

Sampling and Estimator



UCMERCED

Sampling and Estimator

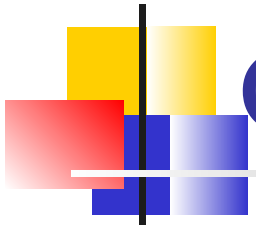
$$Var(\hat{\tau}) = N(N-n)\frac{s_u^2}{n} + \frac{N}{n}\sum_{i=1}^{n} M_i(M_i - m_i)\frac{s_i^2}{m_i}$$

$$s_u^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\hat{y}_i - \hat{\mu}_1)^2 \ , \ s_i^2 = (\frac{1}{m_i - 1})\sum_{j=1}^{m_i}(y_{ij} - \bar{y}_i)^2$$

$$\text{where } i = 1, ..., n, \ and \ \hat{\mu}_1 = (1/n)\sum_{i=1}^{n}\hat{y}_i.$$

- n : number of chunks
- m : number of processed tuples

UCMERCED

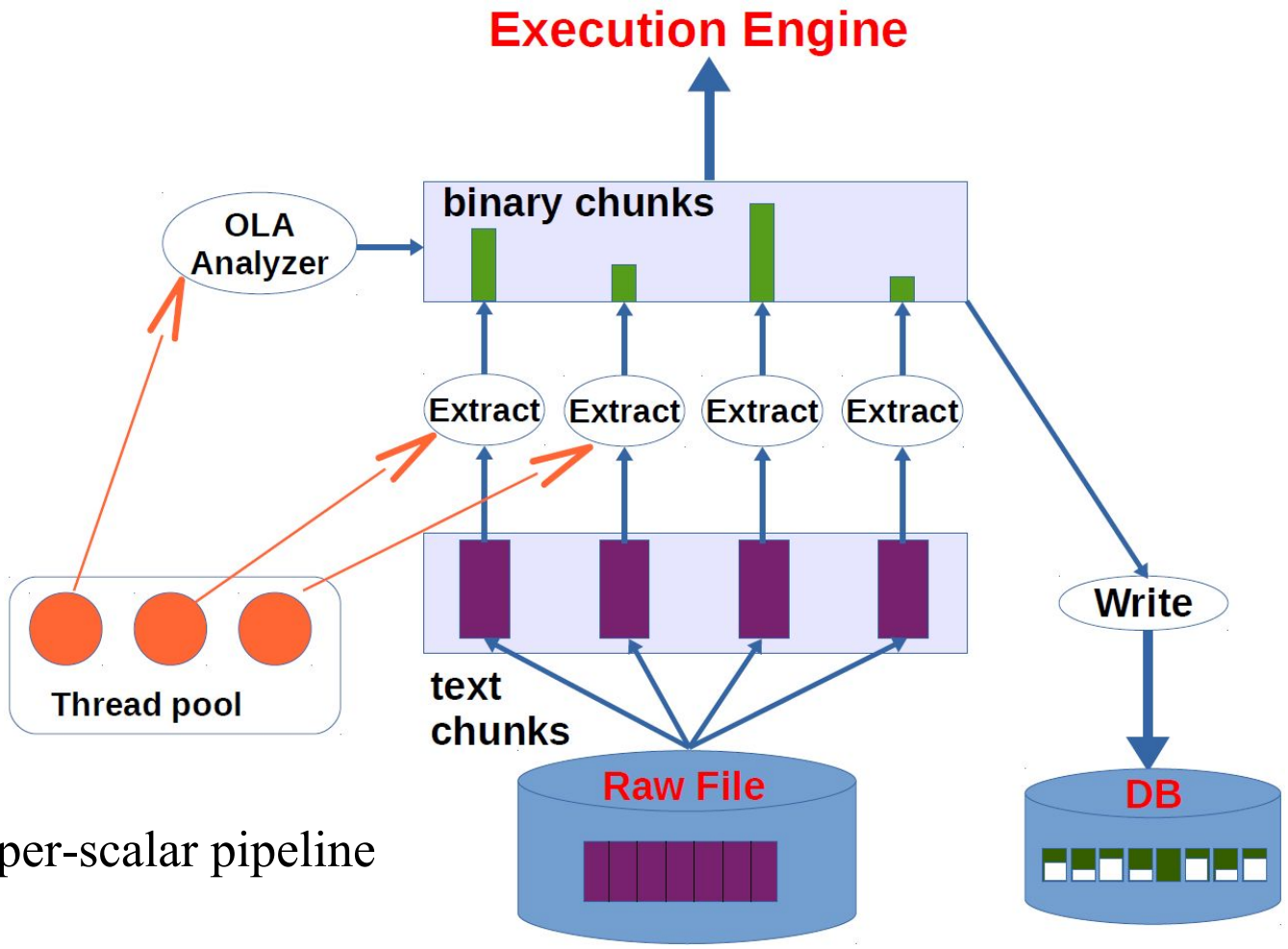


OLA-RAW

- **O**n**L**ine **A**ggregation for **RAW** data processing
  - How to generate random samples from raw files?
    Bi-Level Sampling
  - Design a feasible architecture to combine online aggregation with in-situ data processing
    OLA-RAW
  - Find an efficient method to maintain processed samples

UCMERCED

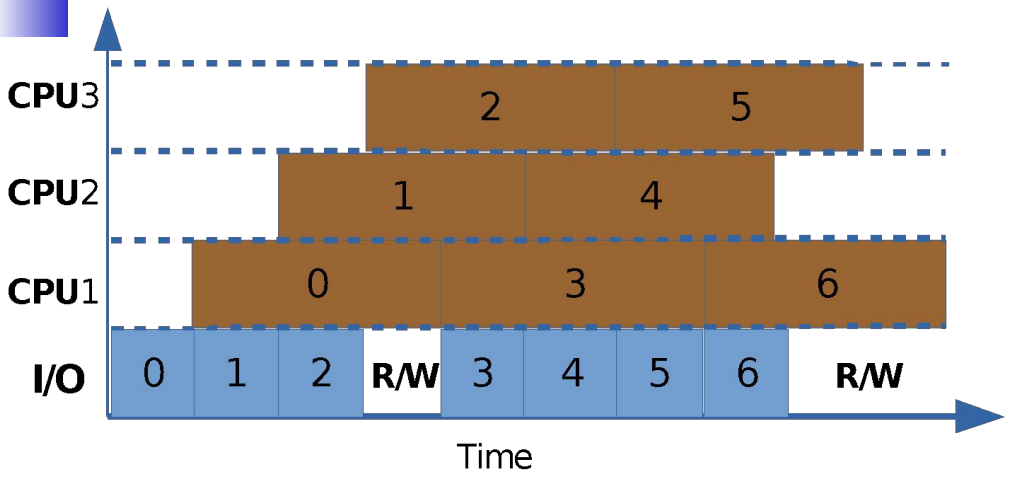
Architecture



- Parallel super-scalar pipeline

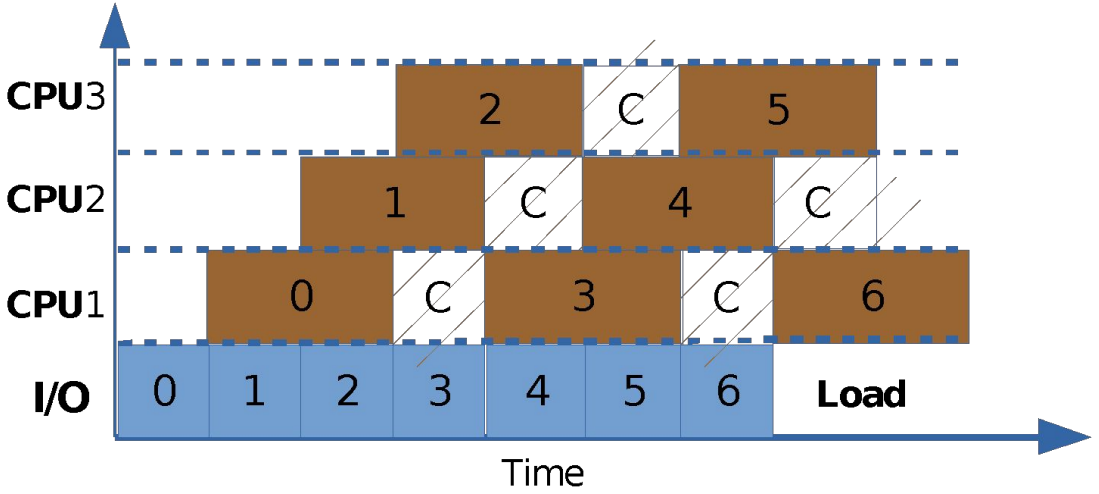
UCMERCED

Where Does the Time Go?



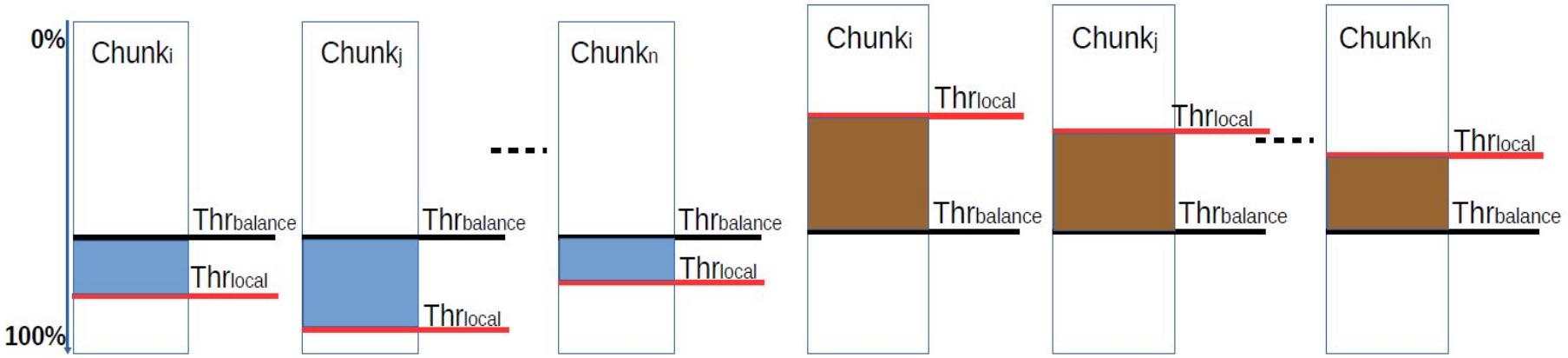
- CPU-bound
- permutation generation
- flush samples

- I/O-bound
- permutation generation
- process more tuples



UCMERCED

How many samples are enough?

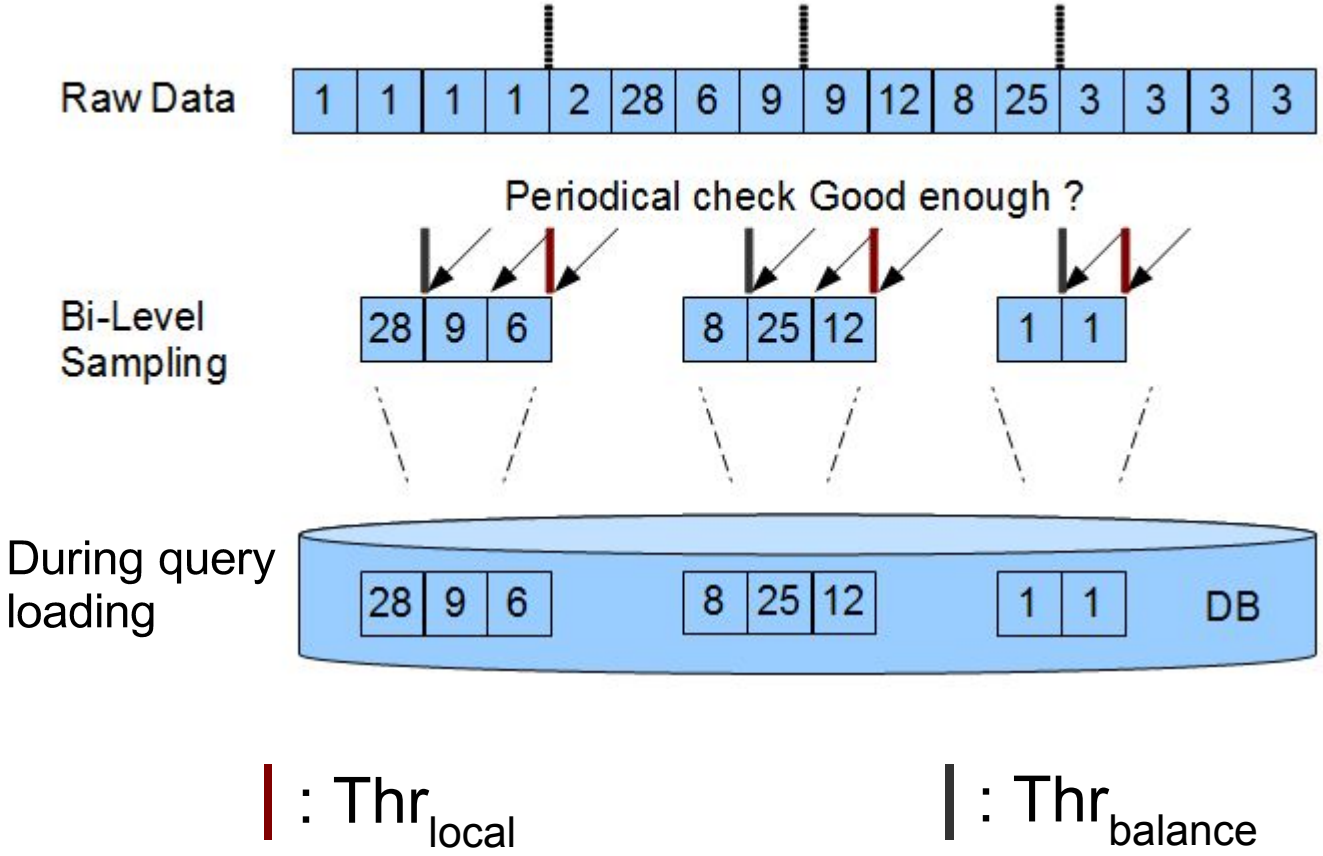


- Make sure to generate good enough estimation by accessing raw data only once
- Generate accurate estimate for each chunk

UCMERCED

Query Processing

CPU-bound process

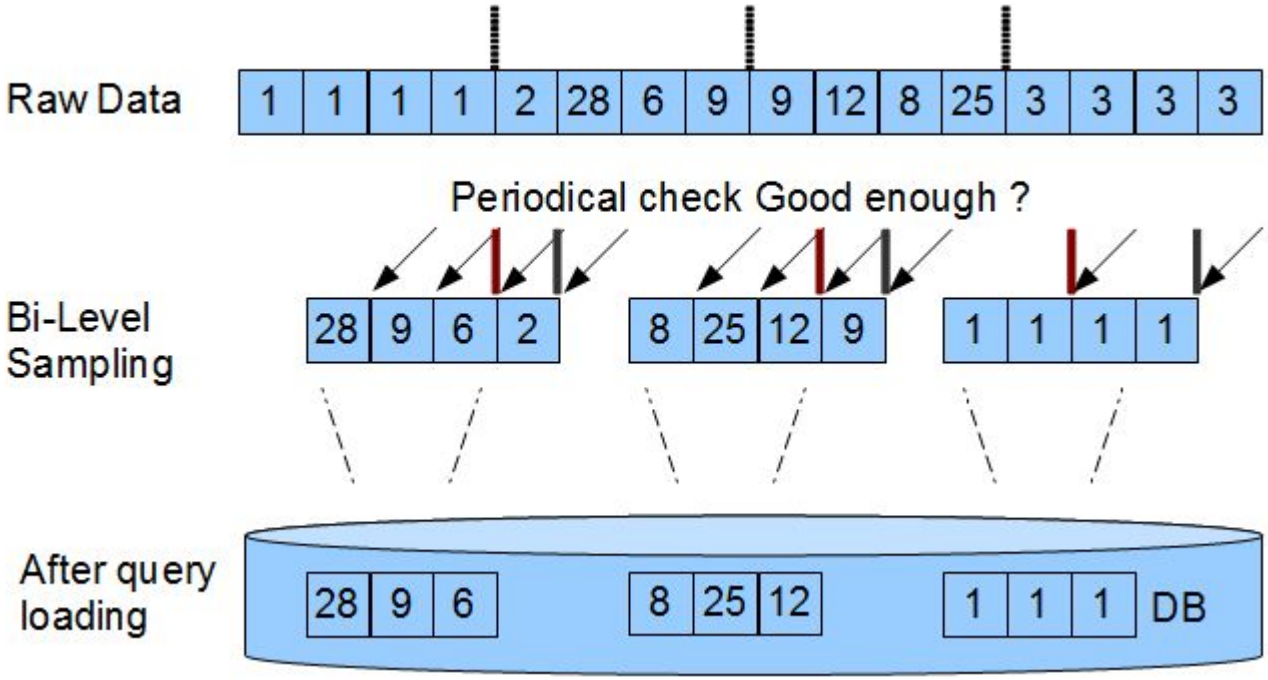


: Thr_{local}

: $Thr_{balance}$

Query Processing

IO-bound process

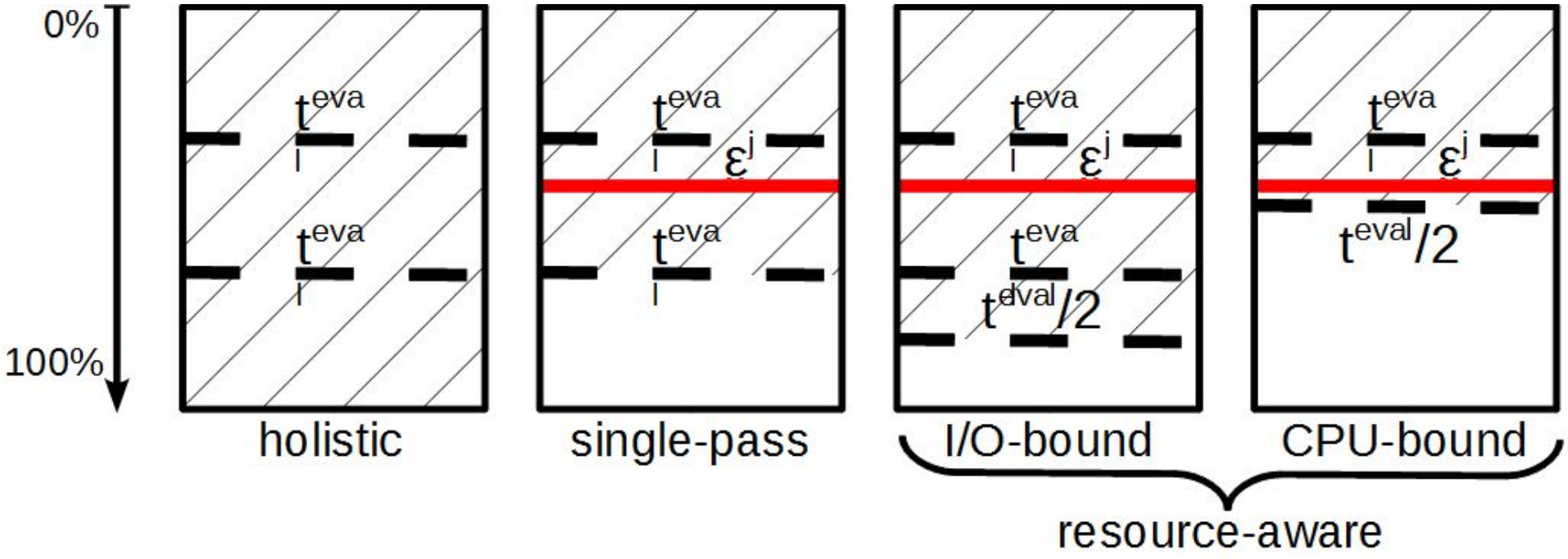


: Thr_{local}

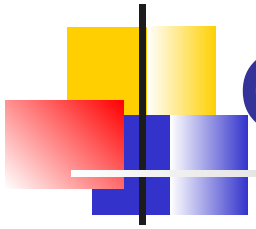
: $Thr_{balance}$

Sampling Strategy

- Parallel sampling procedure
  Result order ≠ Random chunk order → Inspection paradox



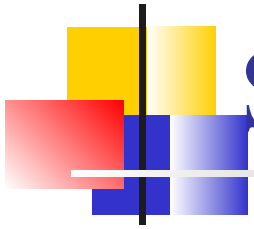
UCMERCED



OLA-RAW

- **OnLine Aggregation for RAW data processing**
  - How to generate random samples from raw files?
    Bi-Level Sampling
  - Design a feasible architecture to combine online aggregation with in-situ data processing
    OLA-RAW
  - Find an efficient method to maintain processed samples
    In-memory sample synopsis

UCMERCED



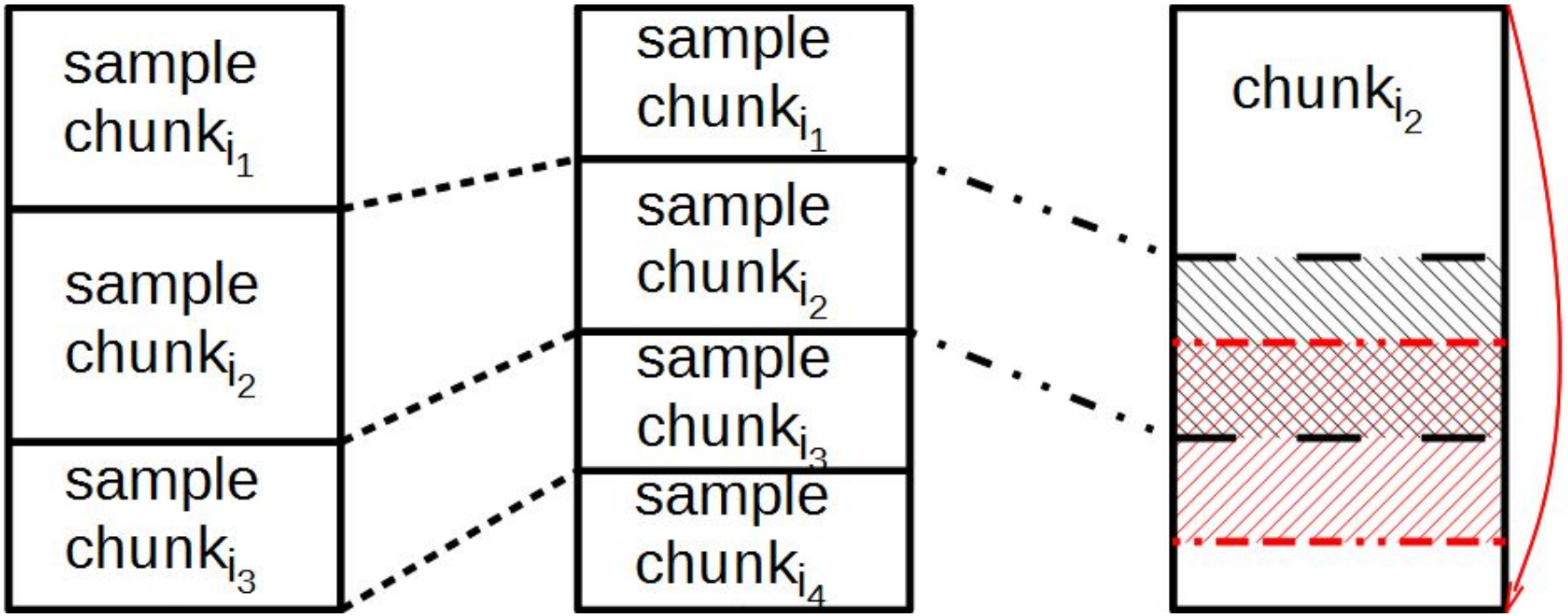
Sample Maintenance

- What kind of samples should be preserved?
  - Variance-driven
- When to load the samples?
  - During query or loading after query processing
- How to make sure the additional samples have not been selected before?
  - Permutation seeds + offset

UCMERCED

Sample Maintenance

- Variance-driven sample swap policy



UCMERCED

Evaluation

Data : The PTF dataset with 1 billion transient detection tuples. Each tuple has 8 attributes, 6 of which are real numbers with 10 decimal digits

Query :

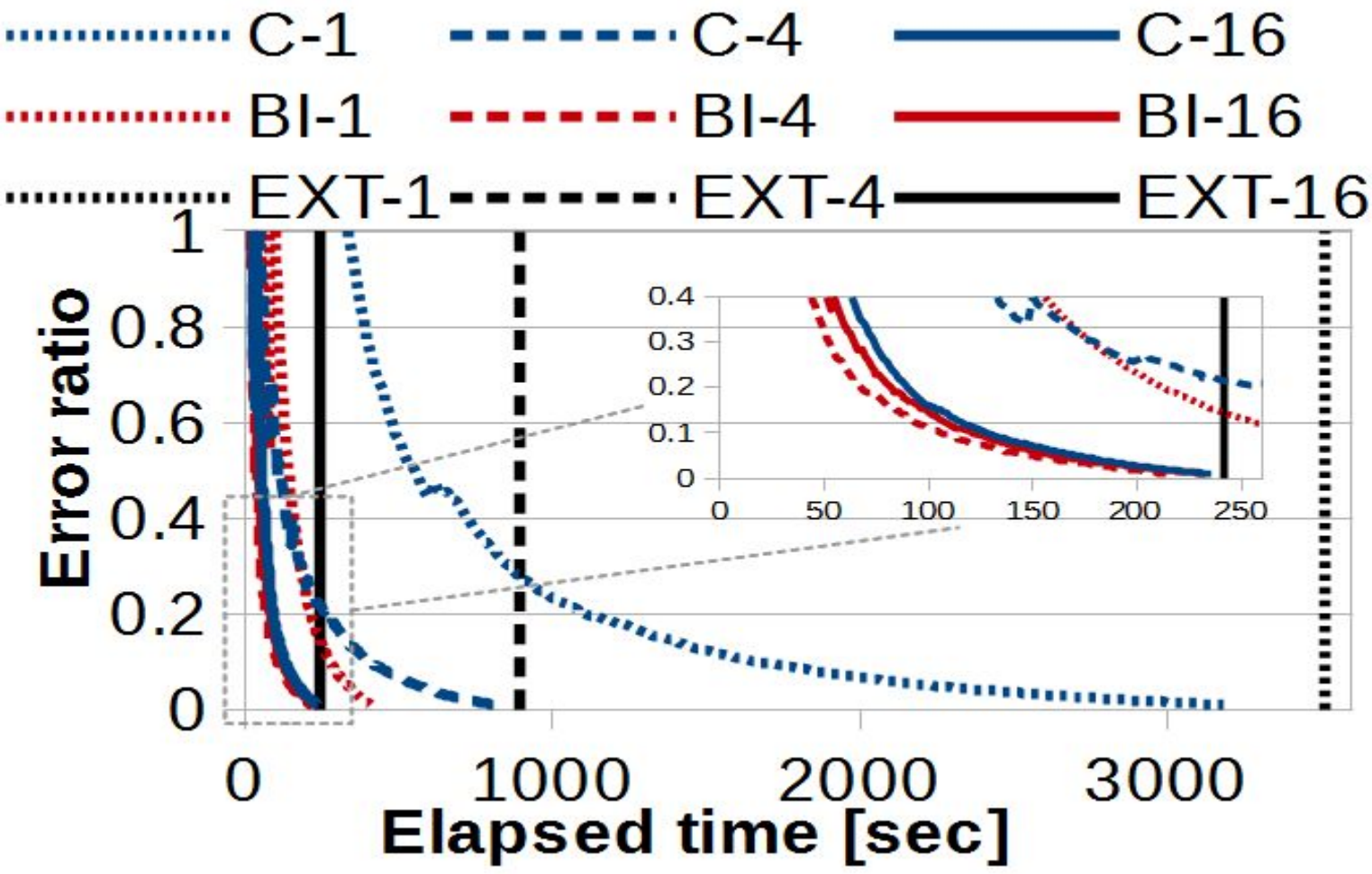
$$\texttt{SELECT SUM}\left(\sum_{j=1}^{K} C_{i_j}\right) \texttt{ FROM FILE}$$

System : 2 AMD 8-core processors, 40 GB of memory, 4 disks in RAID-0 with I/O throughput 450 MB/s

Illustration: 16 attributes, 2^{26} lines, 20GB

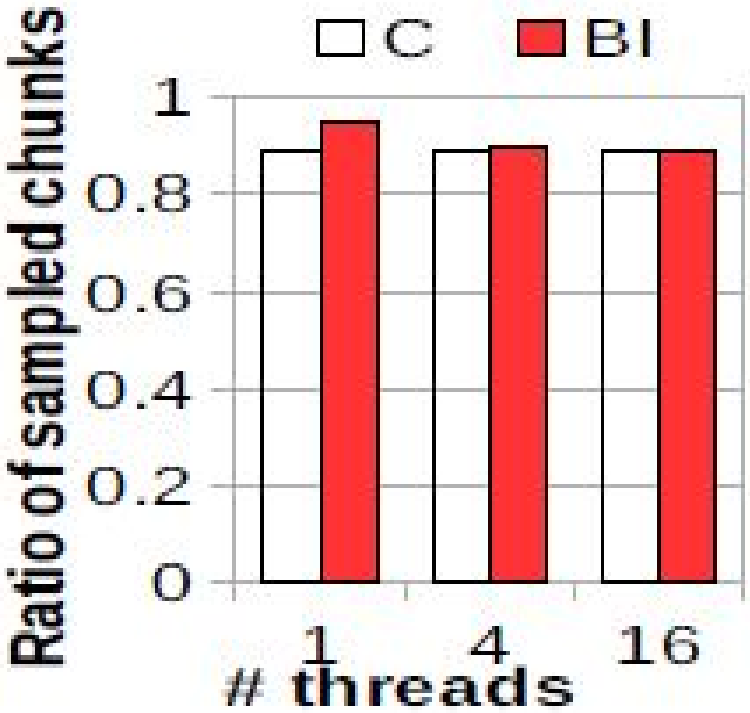
UCMERCED

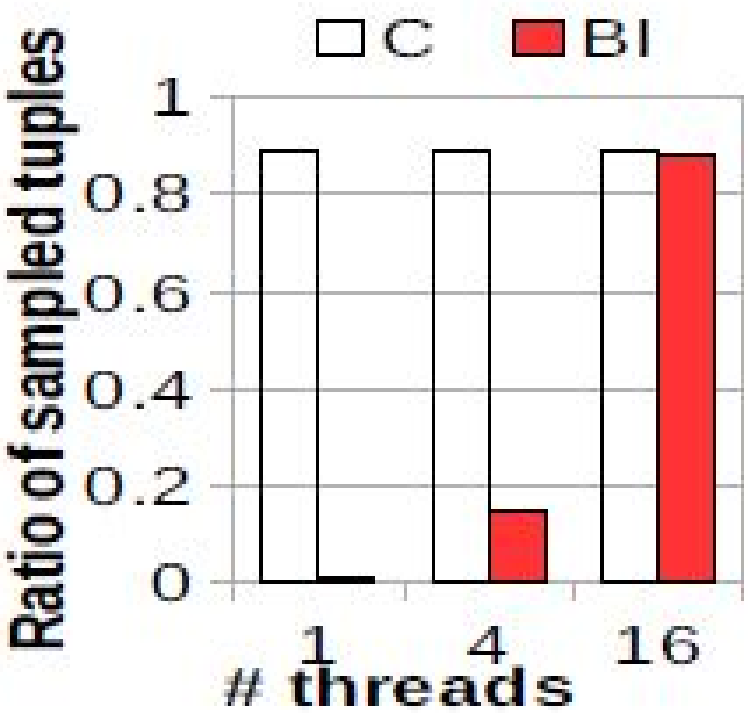
Query Execution Time



UCMERCED

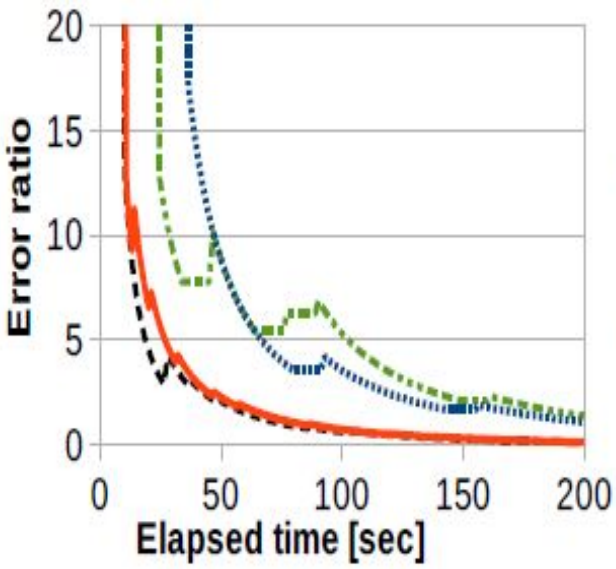
Sample Size



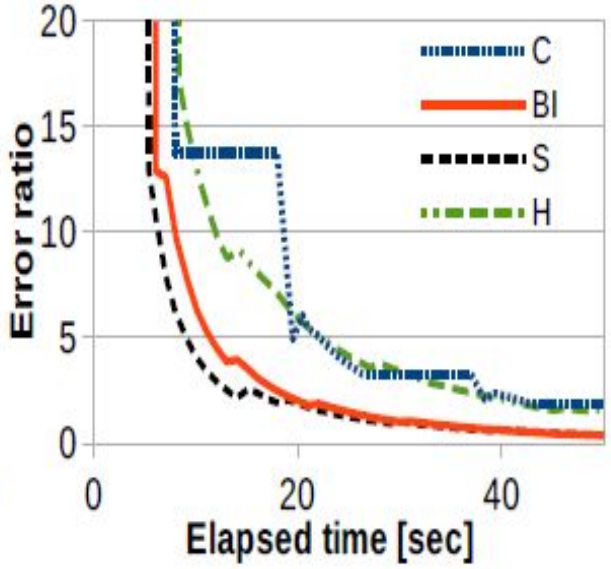


UCMERCED

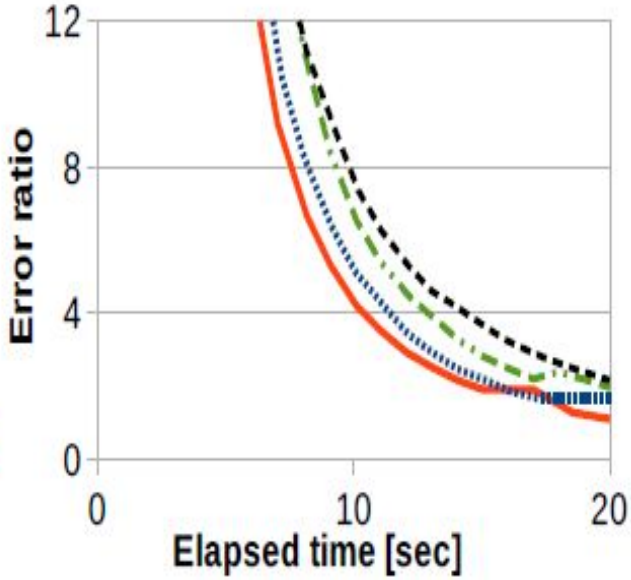
Parallel Sampling Comparison



1 thread



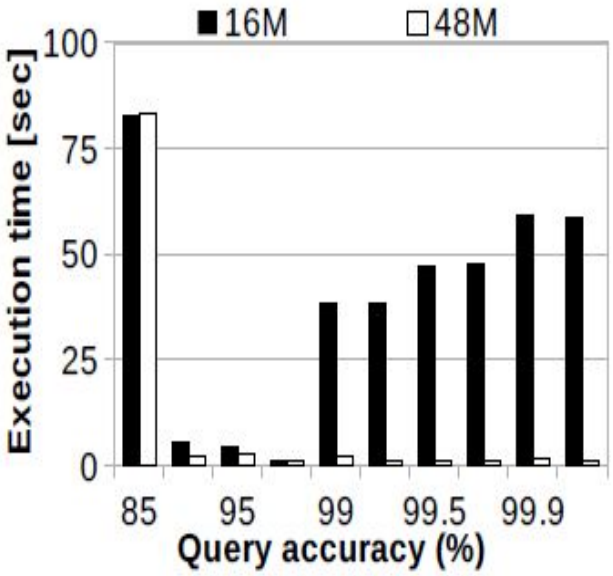
4 threads

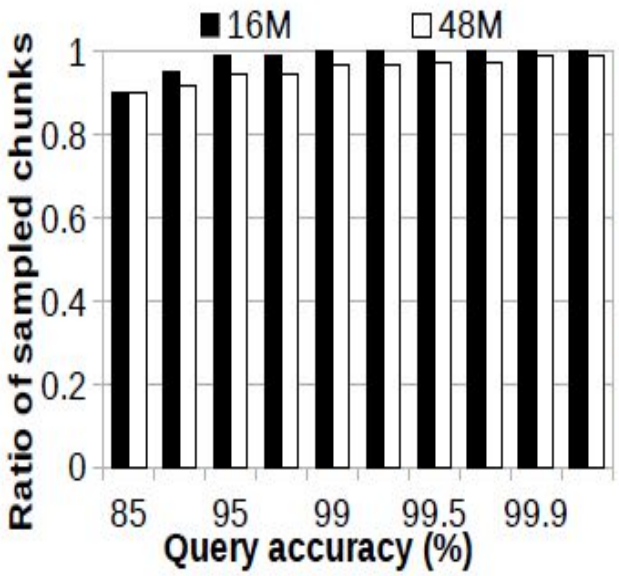


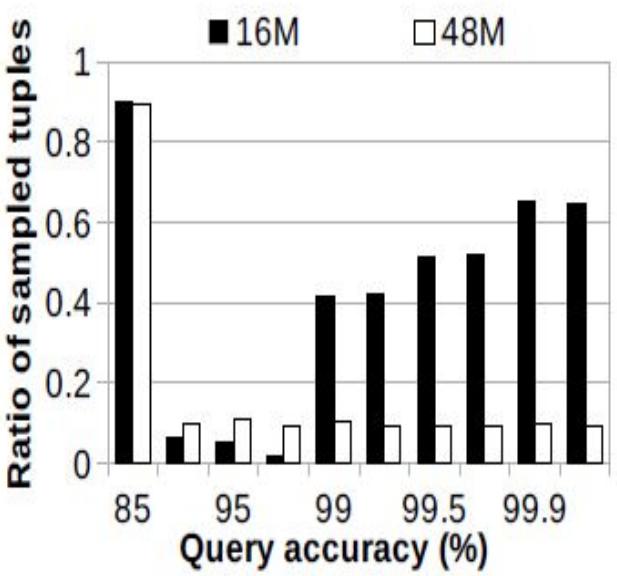
16 threads

UCMERCED

Sample Synopsis

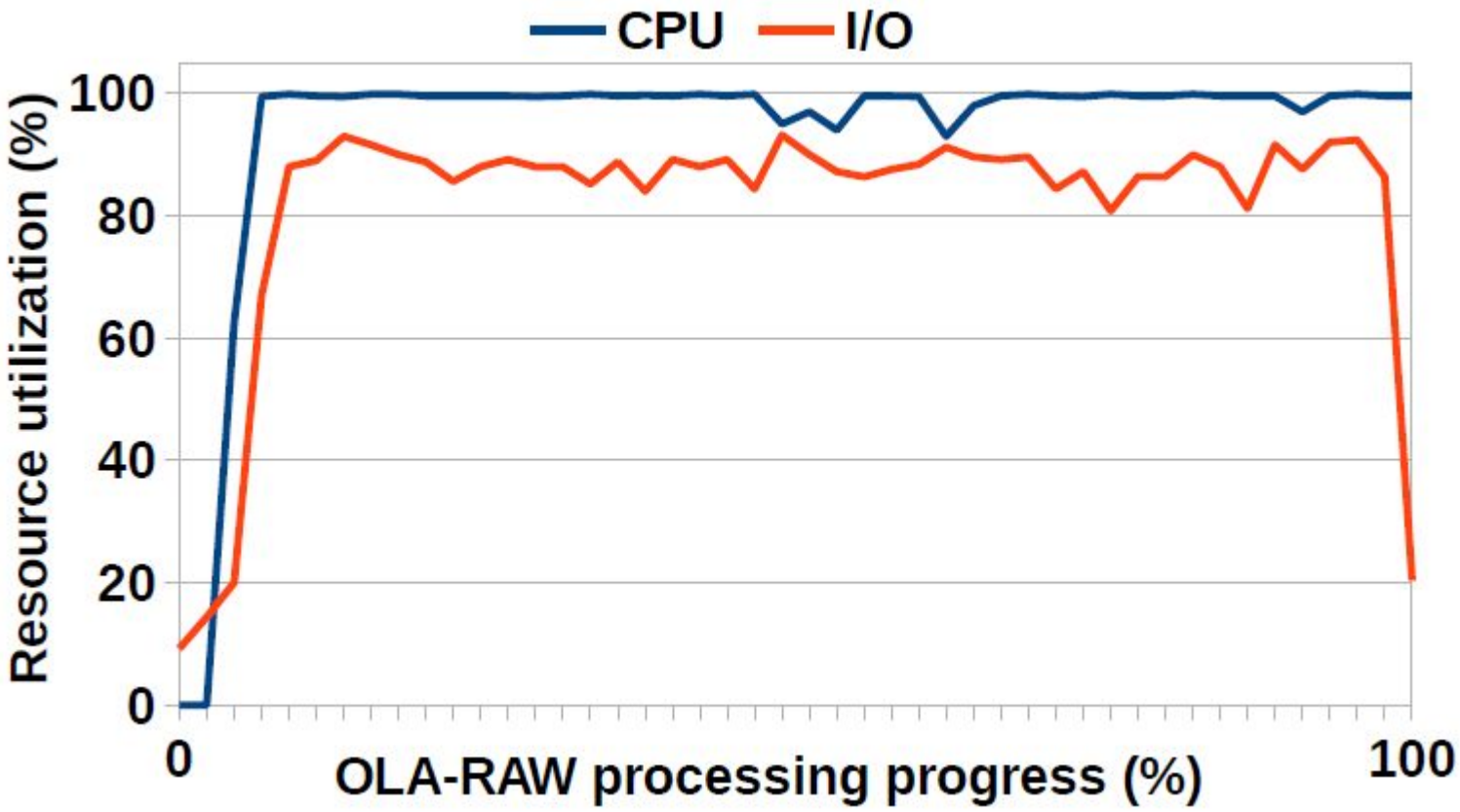




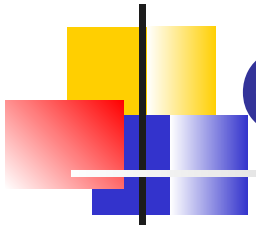


UCMERCED

Resource Utilization

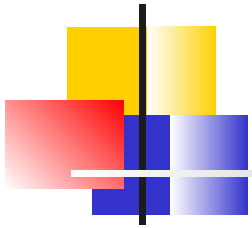


UCMERCED



Conclusions

- OLA-RAW is a novel resource-aware bi-level sampling method for parallel on-line aggregation over raw data
- OLA-RAW is an efficient scheme for data exploration that avoids unnecessary work



Thank you!

Questions?