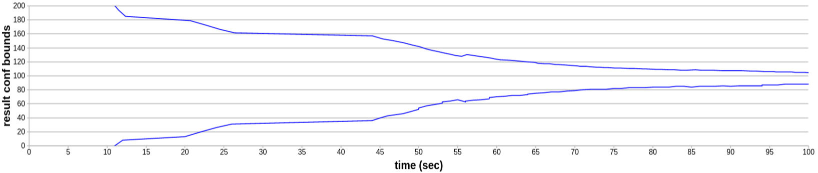
Progressive & Algorithms & Systems

Florin Rusu

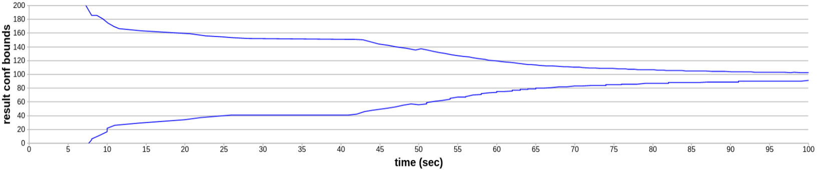
University of California Merced
Lawrence Berkeley National Laboratory

Progressive Computation for Data Exploration

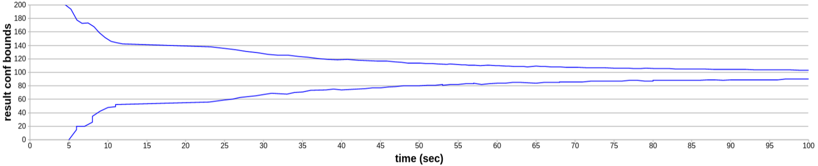
>>> select count(*) from candidates where mag > 10



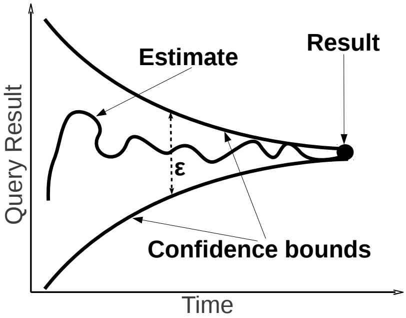
>>> select count(*) from candidates where mag > 10 and mag < 100

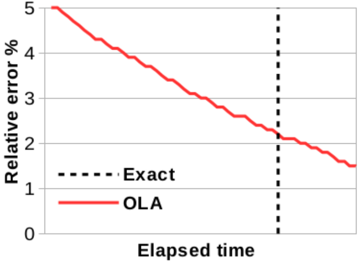


>>> select count(*) from candidates where mag > 10 and mag < 200



Progressive Computation
Online Aggregation (OLA) in DB



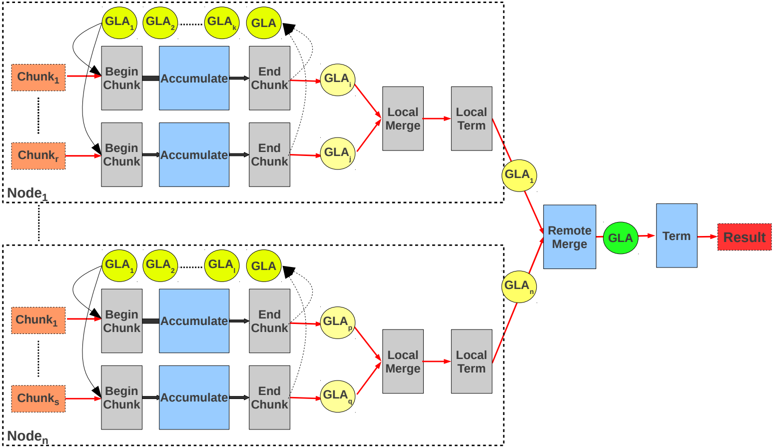


- How to derive confidence bounds that shrink progressively?
- How to generate a meaningful estimate and confidence bounds as early as possible?
- How to minimize the estimation overhead?

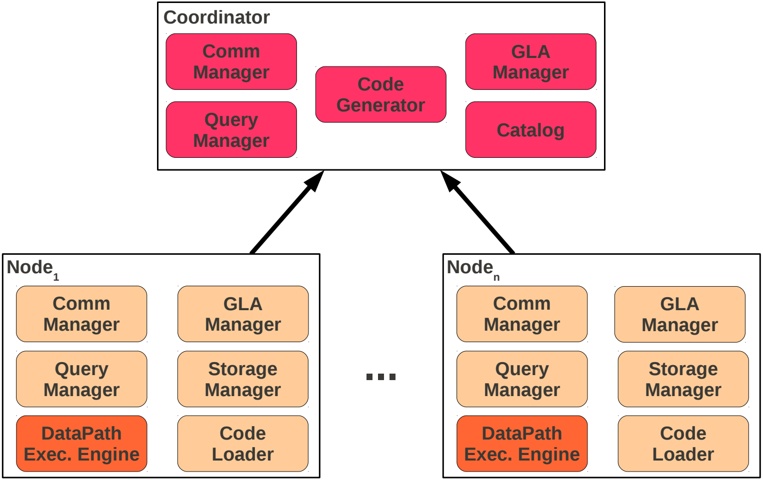
Outline

1. PF-OLA: Parallel Framework for Online Aggregation
2. OLA-GD: Online Aggregation for Gradient Descent Optimization
3. OLA-RAW: Online Aggregation over Raw Data

GLADE


GLA1
GLA2
GLAk
GLA
Chunk1
Chunkr
Begin Chunk
Accumulate
End Chunk
GLAi
GLAj
Local Merge
Local Term
Node1
GLA1
GLA2
GLAl
GLA
Chunk1
Chunks
GLAp
GLAq
Noden
GLA1
GLAn
Remote Merge
GLA
Term
Result

GLADE Architecture



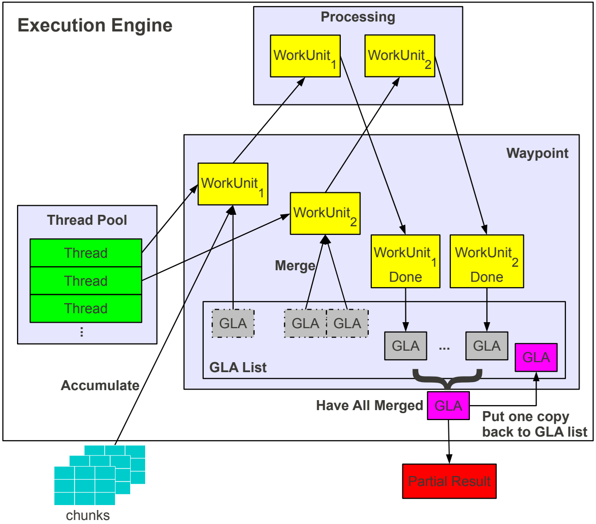
PF-OLA API

Method	Usage

`Init ()`
`Accumulate (Item` *d*`)`
`Merge (UDA` $input_1$`,UDA` $input_2$`, UDA` *output*`)` – local and remote
`Terminate ()` – local and final
	Basic interface

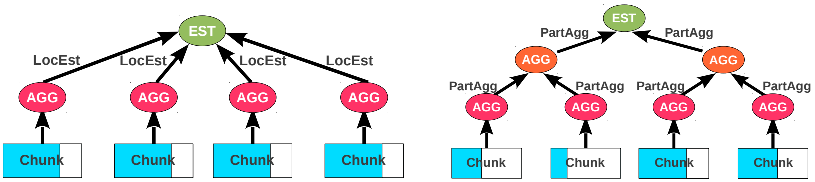
`BeginChunk()` `EndChunk ()`	Chunk processing
`Serialize ()` `Deserialize ()`	Transfer UDA across processes
`EstimatorTerminate ()` `EstimatorMerge (UDA` $input_1$`,UDA` $input_2$`, UDA` *output*`)`	Progressive computation
`Estimate (`*estimator*`,` *lower*`,` *upper*`,` *confidence*`)`	OLA estimation

Partial Aggregation



Parallel Sampling

- Centralized random shuffling
  - Permute data randomly at loading
  - Scan produces larger samples
- Stratified sampling
  - Permute data randomly in each partition
  - Direct extension of random shuffling to partitioned data
- Global data randomization at loading
  - Split data randomly at each node
  - Permute all received data randomly
  - Standard hash-based data partitioning



- Centralized
- Distributed tree

Generic Sampling Estimator

$$\text{AGG} \Longleftarrow \text{SELECT SUM}(f(d))$$
$$\text{FROM } D$$
$$\text{WHERE } P(d)$$

- S is simple random sample without replacement from D
- Estimator $X = \frac{|D|}{|S|} \sum_{s \in S, P(s)} f(s)$

$$E[X] = \text{AGG}$$

$$\text{Var}[X] = \frac{|D| - |S|}{(|D| - 1)|S|} \left[|D| \sum_{d \in D, P(d)} f^2(d) - \left(\sum_{d \in D, P(d)} f(d) \right)^2 \right]$$

$$\text{Est}_{\text{Var}[X]} = \frac{|D|(|D| - |S|)}{|S|^2(|S| - 1)} \left[|S| \sum_{s \in S, P(s)} f^2(s) - \left(\sum_{s \in S, P(s)} f(s) \right)^2 \right]$$

- Data are partitioned across N nodes: $D = D_1 \cup D_2 \cup \cdots \cup D_N$
- Take samples S_i, $1 \leq i \leq N$ independently at each node

Single Estimator

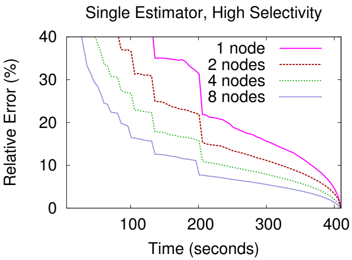
- Guarantee $S = S_1 \cup S_2 \cup \cdots \cup S_N$ is a sample from D
- Synchronized estimator
  - $\frac{S_i}{D_i} = k$ (const), $1 \leq i \leq N$
- Asynchronous estimator
  - Global data randomization

Multiple Estimators

- Stratified sampling
- Build an estimator X_i for each partition D_i, $1 \leq i \leq N$: $X_i = \frac{|D_i|}{|S_i|} \sum_{s \in S_i, P(s)} f(s)$
- $X = \sum_{i=1}^{N} X_i$ is unbiased
- $\text{Var}\left[\sum_{i=1}^{N} X_i\right] = \sum_{i=1}^{N} \text{Var}\left[X_i\right]$

Time to Convergence

```
SELECT n_name, SUM(l_extendprice*(1-l_discount)*(1+l_tax))
FROM lineitem, supplier, nation
WHERE l_shipdate = 1993-02-26 AND l_quantity = 1 AND
 l_discount between [0.02,0.03] AND
 l_suppkey = s_suppkey AND s_nationkey = n_nationkey
GROUP BY n_name
```



- TPC-H scale **8,000 (8TB)**
- Single node: 16 cores @ 2GHz; 16GB RAM; 4 disks @ 110MB/s throughput/disk
- Cluster: 8 X worker + coordinator (9 nodes); Gigabit Ethernet; same rack

Estimation Overhead

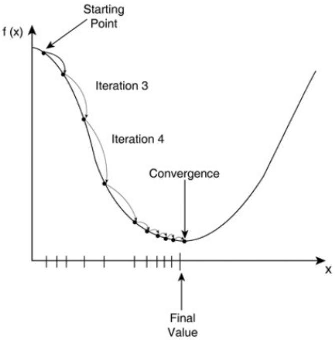
Query	Execution Time (seconds)	
	No estimation	OLA
Aggregate	222	222
$\texttt{Group}_{\texttt{small}}$	344	345
$\texttt{Group}_{\texttt{large}}$	404	407
Join	409	411

References

- Chengjie Qin and Florin Rusu. *Sampling Estimators for Parallel Online Aggregation*. BNCOD 2013, pp. 204–217.
- Chengjie Qin and Florin Rusu. *Parallel Online Aggregation in Action*. SSDBM 2013, pp. 383–386. [Demo]
- Chengjie Qin and Florin Rusu. *PF-OLA: A High-Performance Framework for Parallel Online Aggregation*. Distributed and Parallel Databases (DAPD), August 2013.

Outline

Gradient Descent Optimization



http://www.yaldex.com/game-development/1592730043_ch18lev1sec4.html

$$min_{\vec{w} \in \mathbb{R}^d} \left\{ \Lambda(\vec{w}) = \sum_{(\vec{x}_i, y_i) \in \text{data}} f(\vec{w}, \vec{x}_i; y_i) \right\}$$

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \alpha^{(k)} \nabla \Lambda \left(\vec{w}^{(k)} \right)$$

$\Lambda(\vec{w})$ is the loss

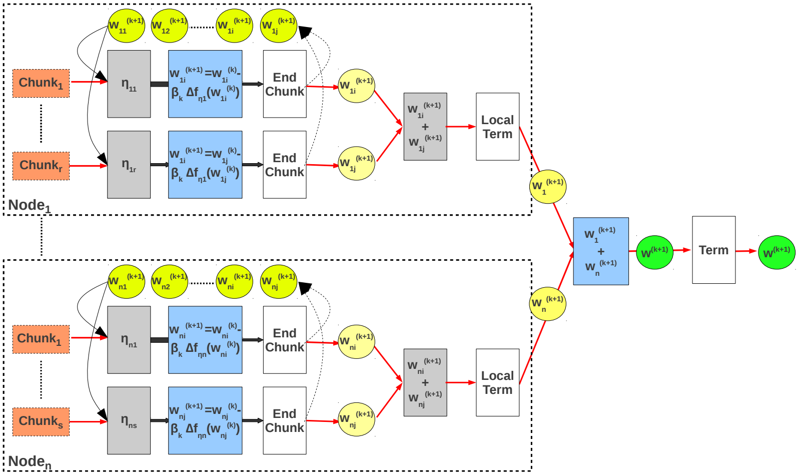
$\nabla \Lambda(\vec{w}) = \left[\frac{\partial \Lambda(\vec{w})}{\partial w_1}, \ldots, \frac{\partial \Lambda(\vec{w})}{\partial w_d} \right]$ is the gradient

$\alpha^{(k)}$ is step size or learning rate

$\vec{w}^{(0)}$ is the starting point (random)

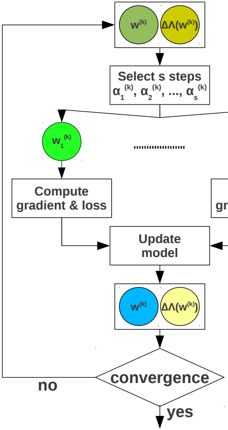
- Convergence to minimum guaranteed for convex objective function

Gradient Descent as GLADE GLA

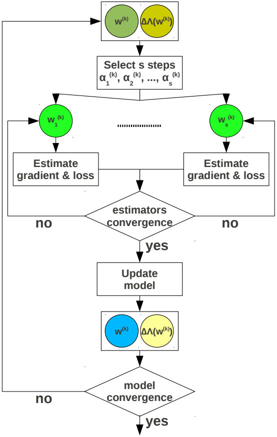


Main idea: apply online aggregation (OLA) sampling to speed-up the execution of a speculative iteration

Speculative Gradient Descent



Approximate Gradient Descent



Approximate BGD

for each example $(\vec{x}_{\eta(j)}, y_{\eta(j)})$ **do**
  for each active model $\vec{w}_i$ **do in parallel**
    Estimate gradient $G_i[est, std]$: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_{\eta(j)}, y_{\eta(j)})\}$
    Estimate loss $L_i[est, std]$: $\Lambda(\vec{w}_i, \vec{x}_{\eta(j)}; y_{\eta(j)})$
  end for
  if estimators_convergence_check(j) **then**
    Prune out models: *Stop Loss*$(\{L_i[est, std]\}_{i \leq s}, \varepsilon_1)$
    Let t be the number of remaining models, i.e., *active*
    if *Stop Gradient*$(\{G_{tl}[est, std]\}_{l \leq d}, \varepsilon_2)$ **then break**
    Let $s = t$
  end if
end for

```
SELECT
   ..., SUM(f_q), ...
FROM data AS D
SELECT
   ..., Z_q=SUM(f_q), ...
   ..., Z_q^2=SUM(f_q^2), ...
FROM sample AS S
```

- estimator: $Z_q \cdot \frac{|D|}{|S|}$
- variance: $\frac{|D|(|D|-|S|)}{|S|^2(|S|-1)} \left(|S| Z_q^2 - (Z_q)^2 \right)$

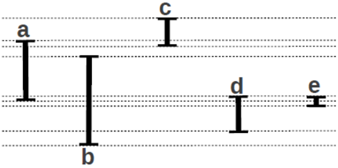
OLA sampling

- no pre-determined sample size
- avoid correlation between estimators by random data shuffling (at runtime)
- continuous sampling at runtime until estimators converge

Approximate BGD

for each example $(\vec{x}_{\eta^{(j)}}, y_{\eta^{(j)}})$ **do**
  for each active model $\vec{w}_i$ **do in parallel**
    Estimate gradient $G_i[est, std]$: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_{\eta^{(j)}}, y_{\eta^{(j)}})\}$
    Estimate loss $L_i[est, std]$: $\Lambda(\vec{w}_i, \vec{x}_{\eta^{(j)}}; y_{\eta^{(j)}})$
  end for
  if estimators_convergence_check(j) **then**
    Prune out models: *Stop Loss*$(\{L_i[est, std]\}_{i \le s}, \varepsilon_1)$
    Let t be the number of remaining models, i.e., *active*
    if *Stop Gradient*$(\{G_{tl}[est, std]\}_{l \le d}, \varepsilon_2)$ **then break**
    Let $s = t$
  end if
end for

- Identify step size with minimum loss based on estimate & confidence bounds
- Interacting estimators
- Exact pruning
- Approximate pruning



Approximate BGD

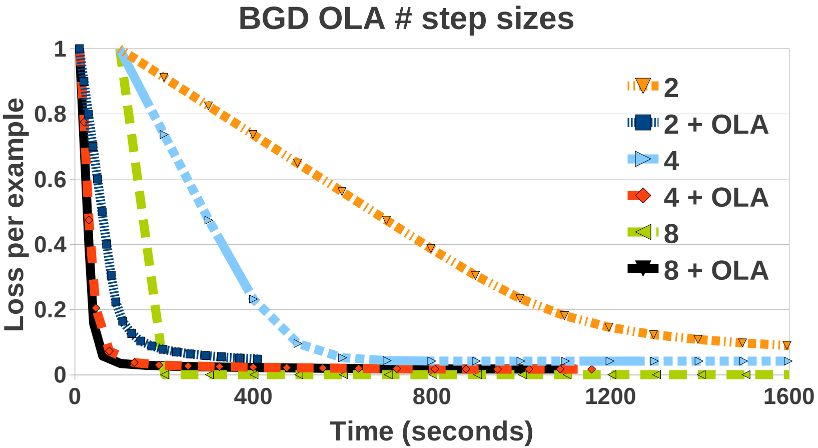
for each example $(\vec{x}_{\eta^{(j)}}, y_{\eta^{(j)}})$ **do**
  for each active model $\vec{w}_i$ **do in parallel**
    Estimate gradient $G_i[est, std]$: $\nabla\Lambda(\vec{w}_i)\{(\vec{x}_{\eta^{(j)}}, y_{\eta^{(j)}})\}$
    Estimate loss $L_i[est, std]$: $\Lambda(\vec{w}_i, \vec{x}_{\eta^{(j)}}; y_{\eta^{(j)}})$
  end for
  if estimators_convergence_check(j) **then**
    Prune out models: *Stop Loss*$(\{L_i[est, std]\}_{i \le s}, \varepsilon_1)$
    Let t be the number of remaining models, i.e., *active*
    if *Stop Gradient*$(\{G_{tl}[est, std]\}_{l \le d}, \varepsilon_2)$ **then break**
    Let $s = t$
  end if
end for

- One estimator for each dimension in model $\vec{w}$
- Grouped estimators

When to stop?

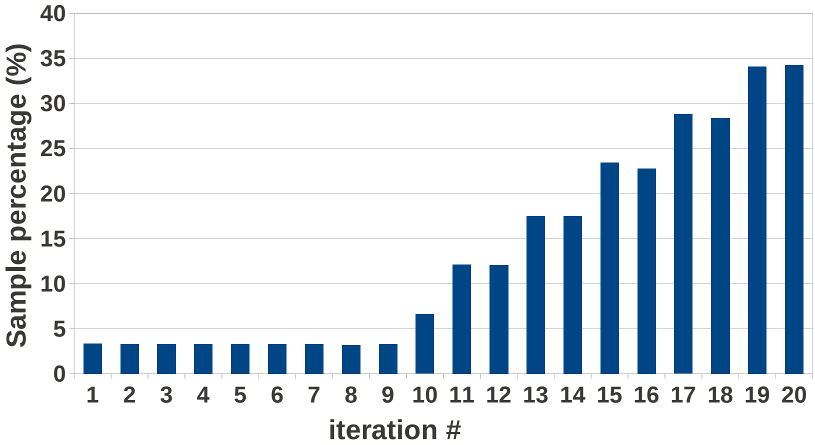
- all estimators converge (million dimensions)
- percentage of estimators converge, e.g., 90%
- unified convergence threshold, e.g., $\varepsilon' = d \cdot \varepsilon$

Train SVM Model



- 50M examples, 200 dimensions, 136GB

Sample Percentage



- 50M examples, 200 dimensions, 136GB

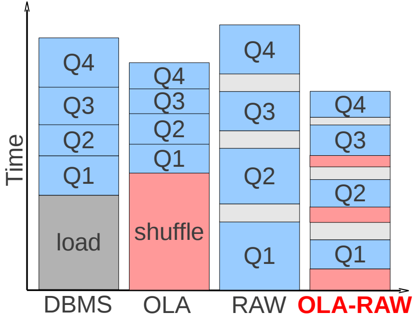
References

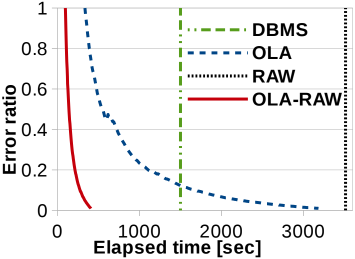
- Chengjie Qin and Florin Rusu. *Speculative Approximations for Terascale Analytics*. CoRR abs/1501.00255, December 2014.
- Chengjie Qin and Florin Rusu. *Speculative Approximations for Terascale Distributed Gradient Descent Optimization*. DanaC@SIGMOD 2015.
- Florin Rusu, Chengjie Qin, and Martin Torres. *Scalable Analytics Model Calibration with Online Aggregation*. IEEE Data Engineering Bulletin, Vol. 38, No. 3, pp. 30–44, September 2015.

Outline

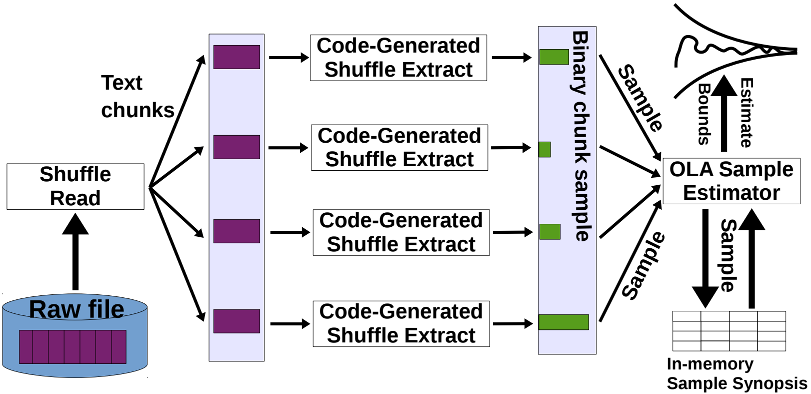
1. PF-OLA: Parallel Framework for Online Aggregation
2. OLA-GD: Online Aggregation for Gradient Descent Optimization
3. OLA-RAW: Online Aggregation over Raw Data

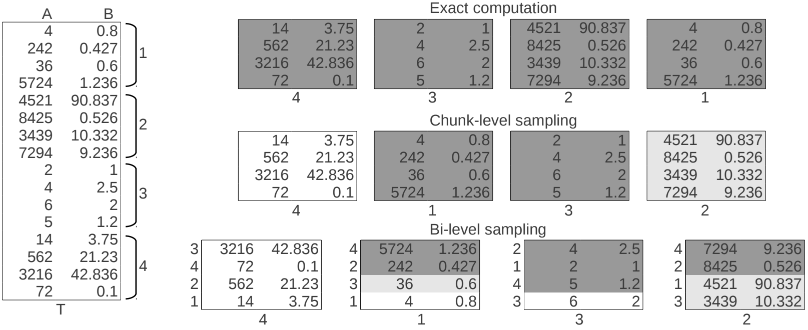
Motivation & Approach





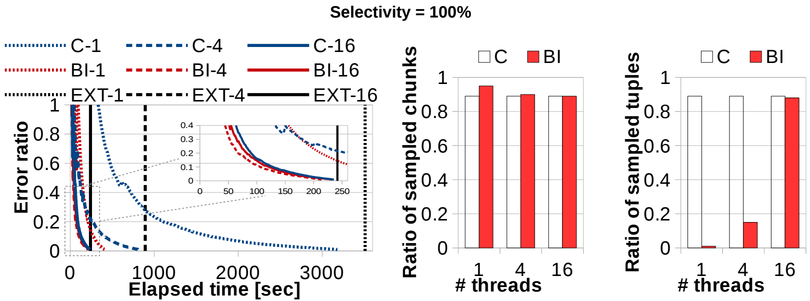
OLA-RAW Architecture

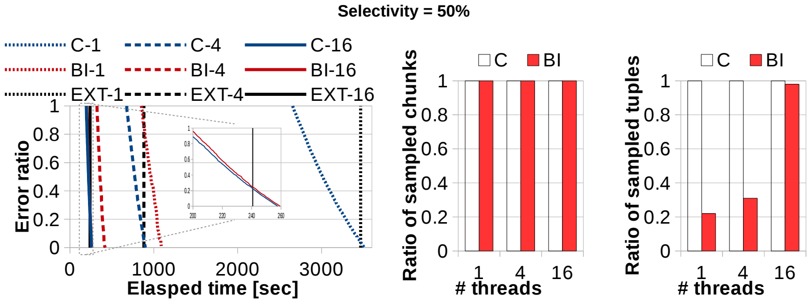




- Inspection paradox: result order $\neq$ random chunk order

Experimental Results





References

- Yu Cheng, Weijie Zhao, and Florin Rusu. *OLA-RAW: Scalable Exploration over Raw Data*. CoRR abs/1702.00358, February 2017.
- Yu Cheng, Weijie Zhao, and Florin Rusu. *Bi-Level Online Aggregation on Raw Data*. SSDBM 2017.

Thank you!

Questions?