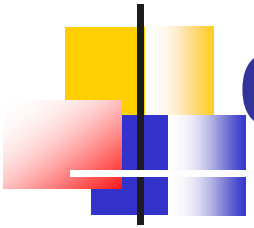
UCMERCED

Parallel In-situ Data Processing with Speculative Loading

Yu Cheng, Florin Rusu

University of California, Merced

UCMERCED



Outline

- Background
- Scanraw Operator
- Speculative Loading
- Evaluation

UCMERCED

SAM/BAM Format

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,255}`	Query template NAME
2	FLAG	Int	`[0,2^16-1]`	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	`[0,2^31-1]`	1-based leftmost mapping POSition
5	MAPQ	Int	`[0,2^8-1]`	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next read
8	PNEXT	Int	`[0,2^31-1]`	Position of the mate/next read
9	TLEN	Int	`[-2^31+1,2^31-1]`	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

- More than 200 TB of genomic data can be downloaded for research

UCMERCED

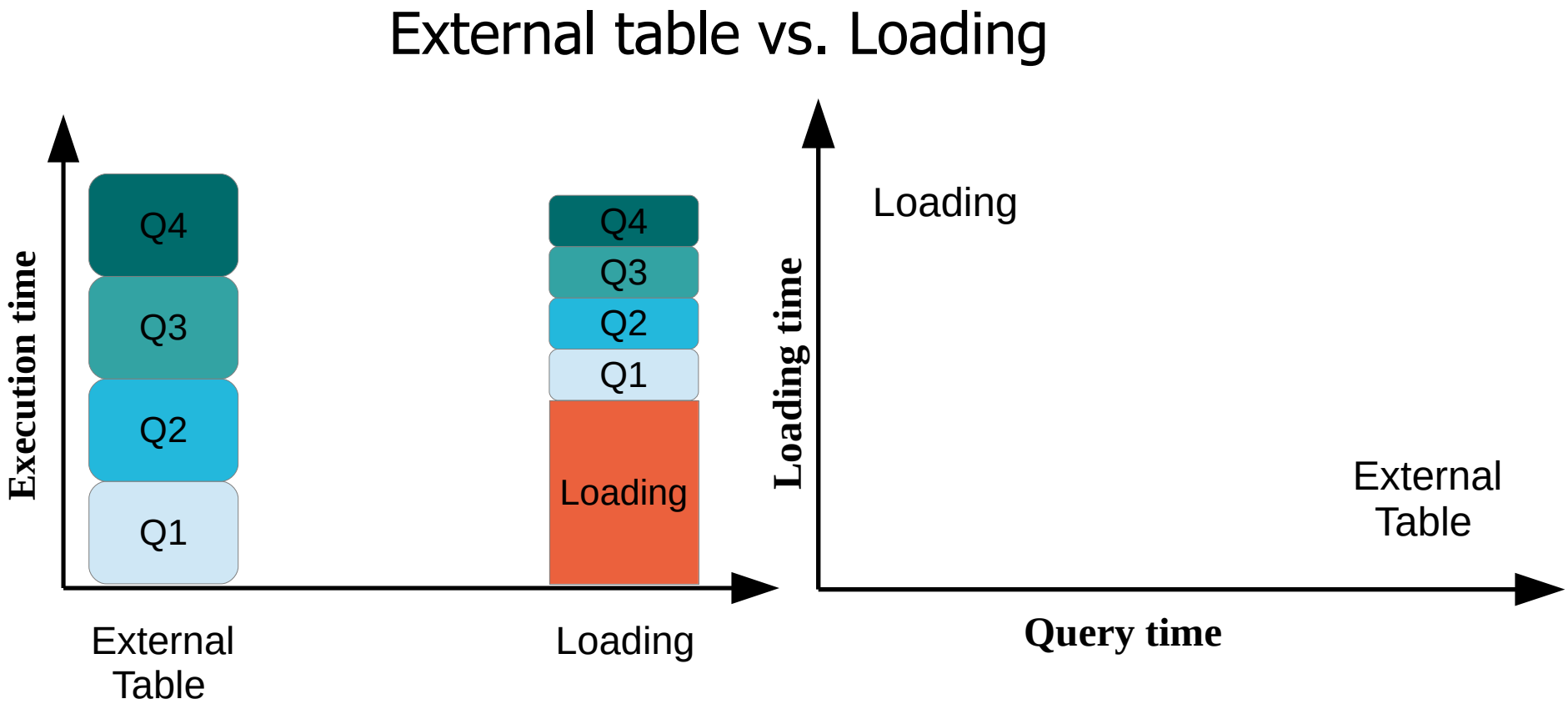
Illustrative Example

- Variant, e.g., genome mutation, identification

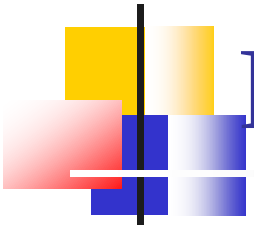

```
SELECT position, count(*) as cnt
FROM genome
WHERE CIGAR IS NOT 'M'
GROUP BY position
HAVING cnt > threshold.
```

UCMERCED

Comparison



UCMERCED

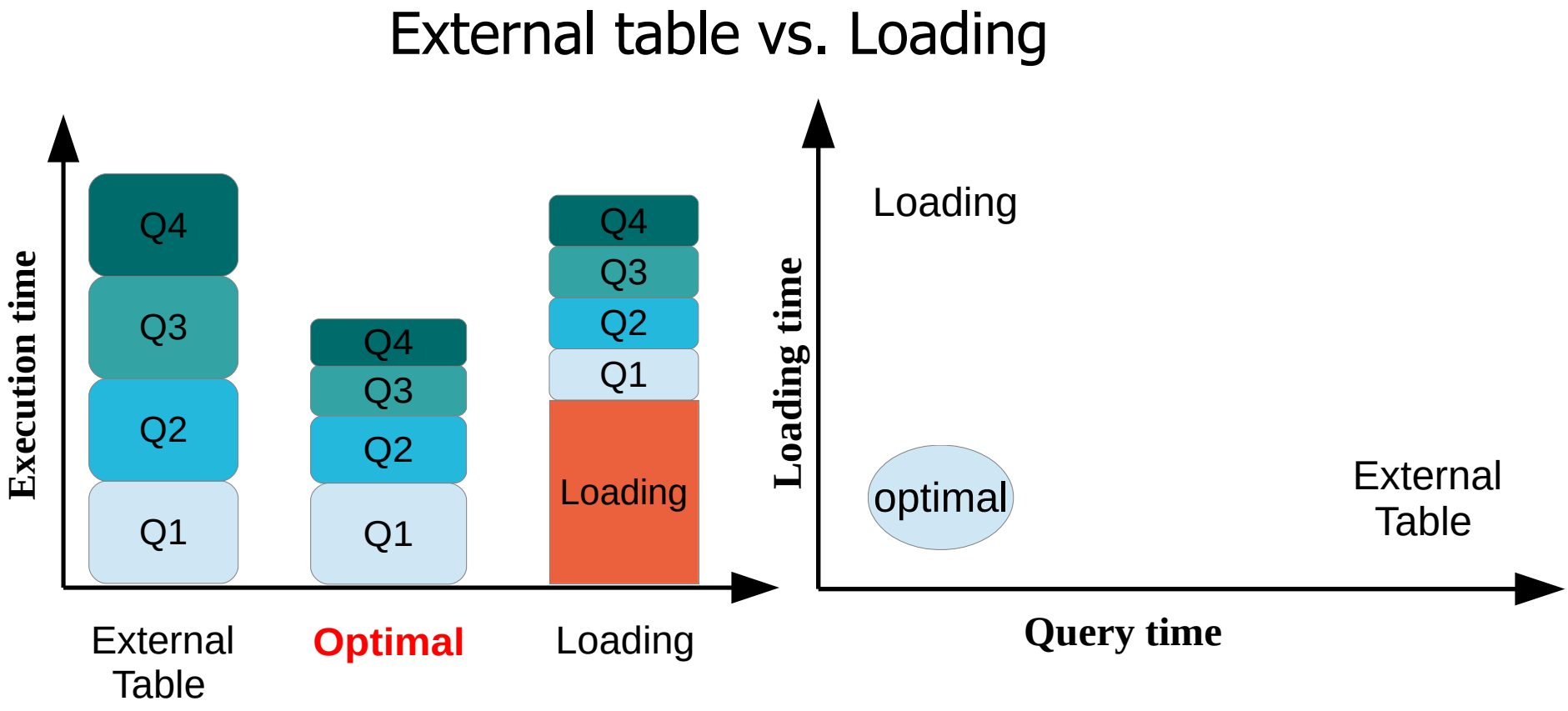


Research Problem

- Can we find an optimal solution to execute SQL-like queries in-situ over raw files?
  - Instant access to data
  - Optimal performance across a query workload

UCMERCED

Comparison



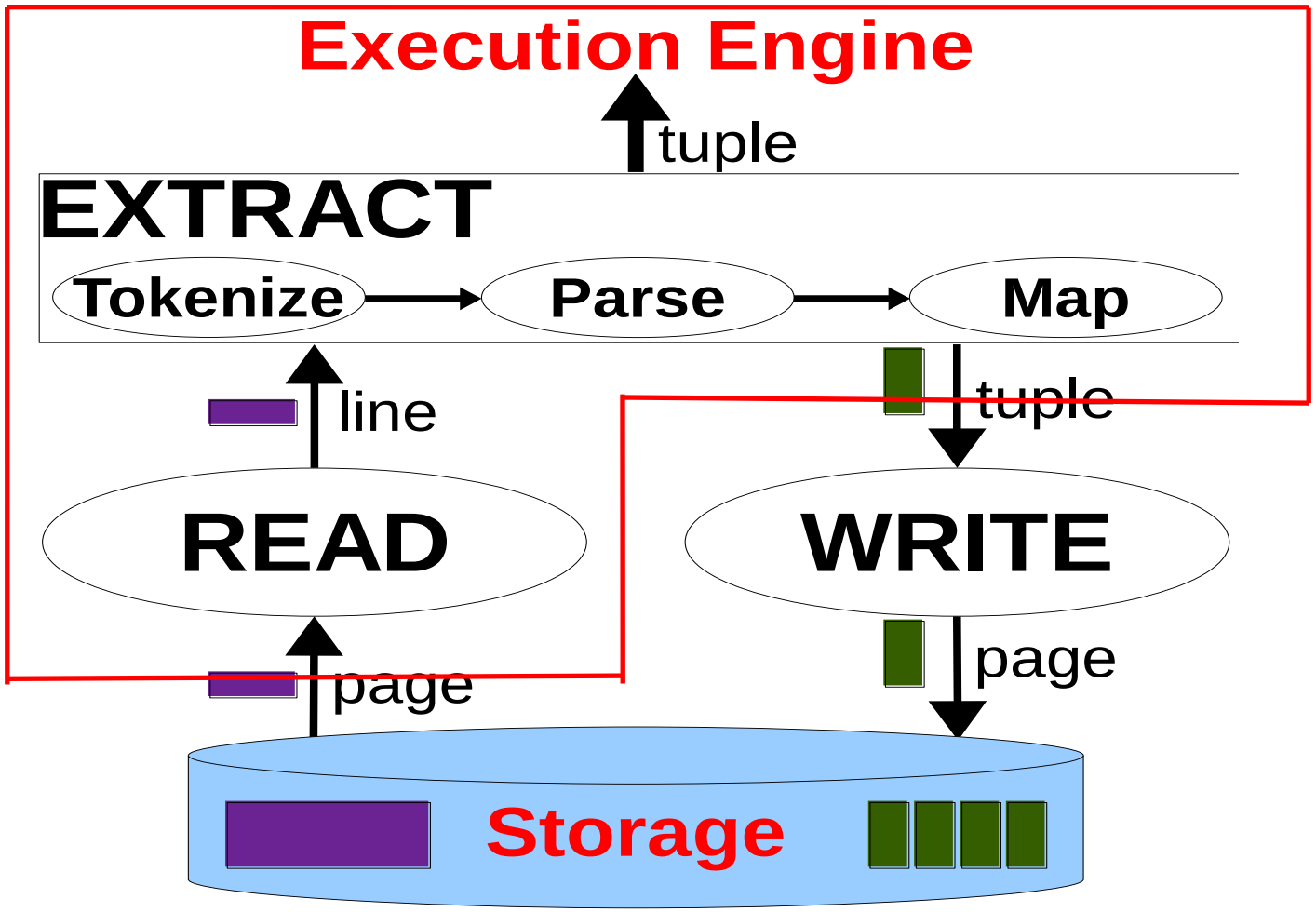
UCMERCED

Related Work

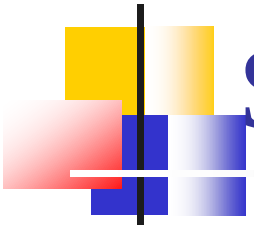
- Adaptive partial loading [Idreos et al., CIDR 2011]
  Only load necessary attributes before query starts
- NoDB [Alagianis et al., SIGMOD 2012]
  Instead of loading, build index and cache necessary attributes in memory
- Invisible loading [Abouzied et al., EDBT/ICDT 2013]
  Portion of necessary data is loaded into database for every query
- Data vaults [Ivanova et al., SSDBM 2012]
  Memory cache for complex data in scientific repositories
- Instant loading [Muhlbauer et al., PVLDB 2013]
  Speed-up loading methods using modern vectorized instructions, e.g., SSE4 and AVX

UCMERCED

Raw File Processing



UCMERCED



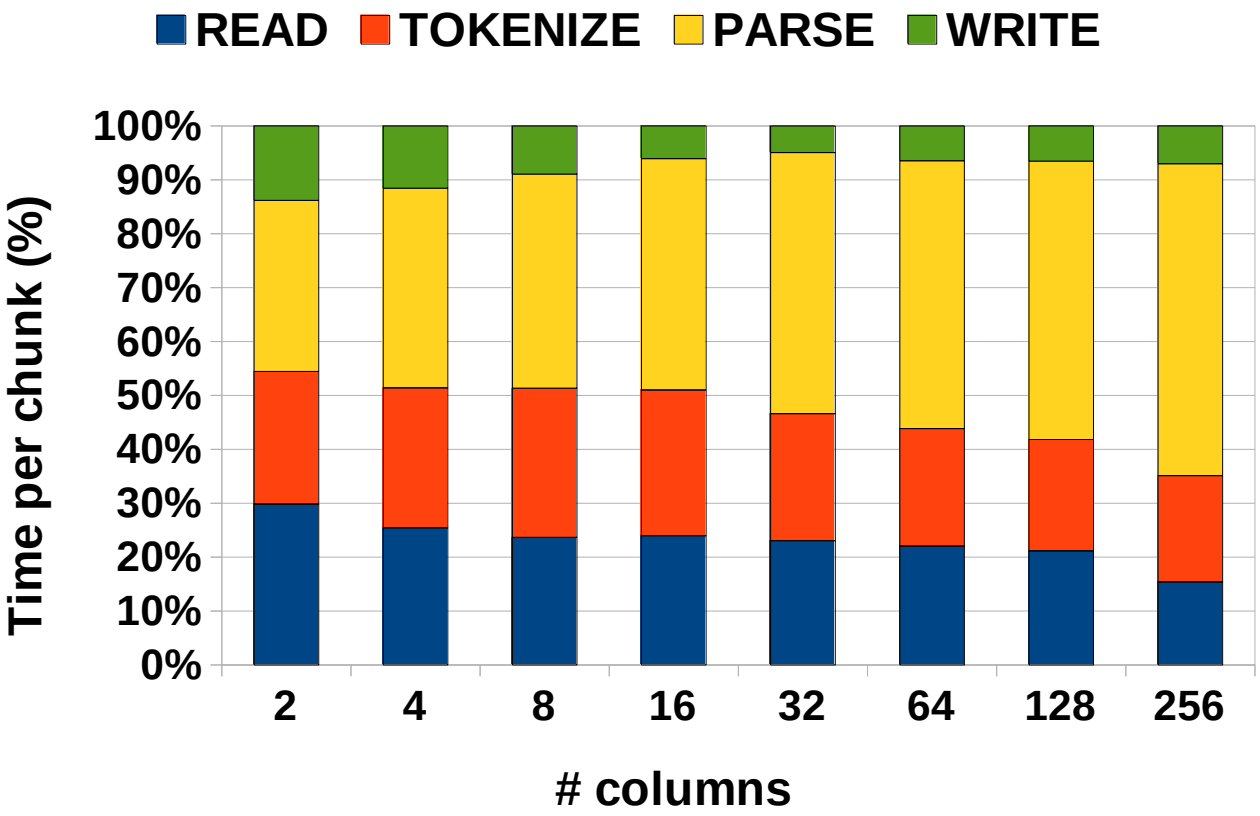
Scanraw Operator

- Can we achieve optimal execution time for the first query?

- How can we design a parallel operator using current multi-core processors?
  Multi-threads
- What kind of architecture can take full advantage of the available parallelism?
  Task-parallelism/Pipeline
- How to integrate the operator with a database system?
  Scanraw operator

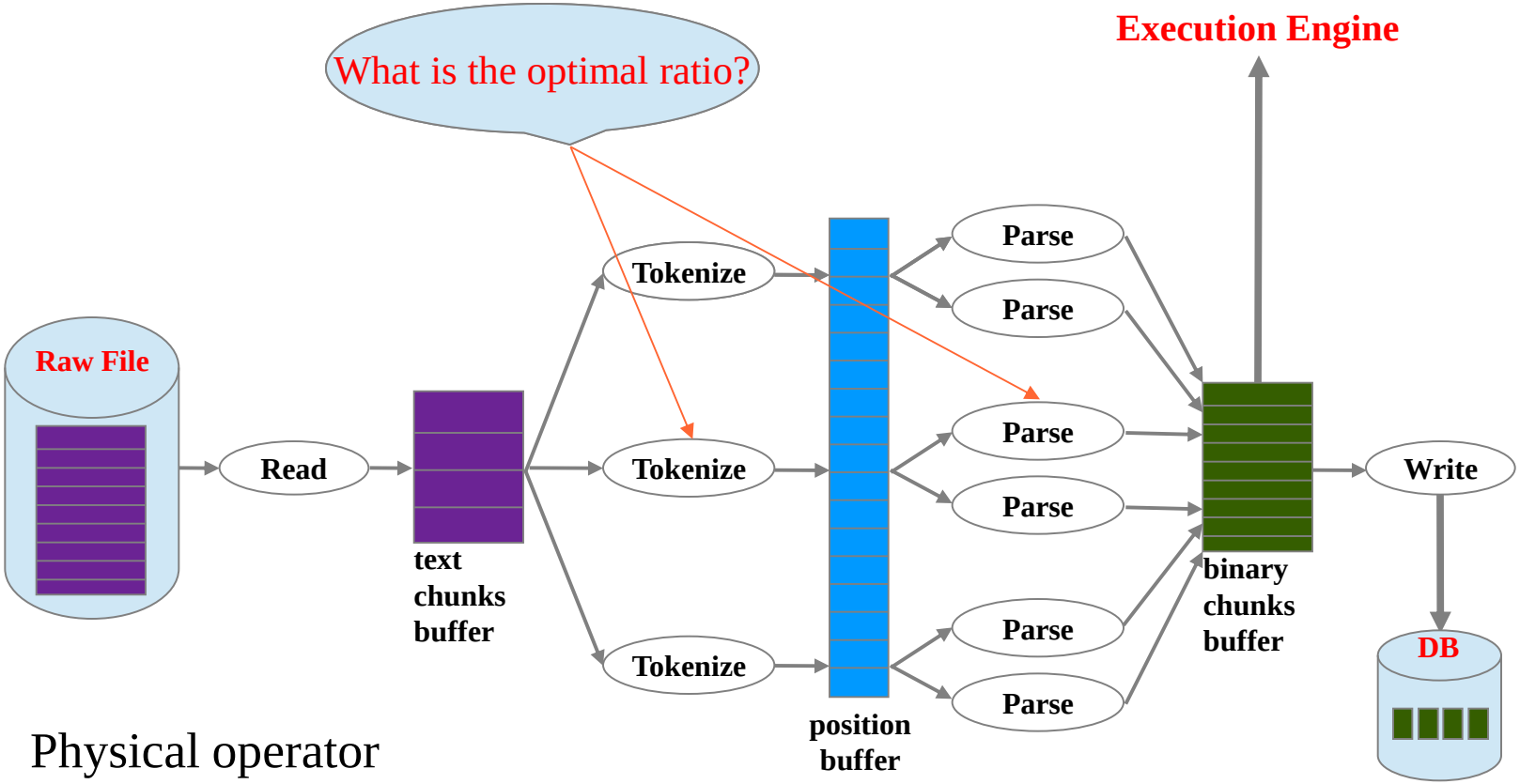
UCMERCED

Where Does the Time Go?



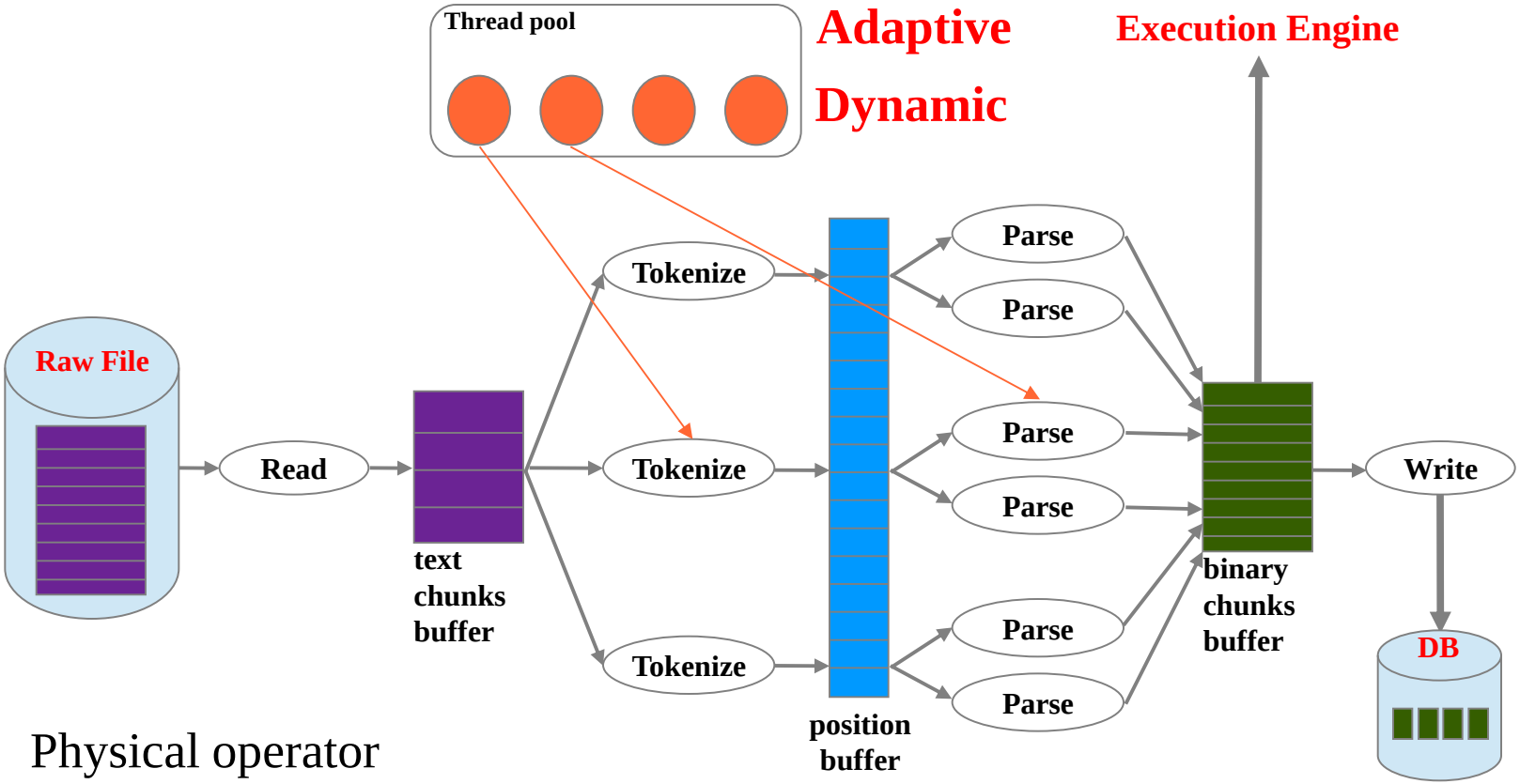
UCMERCED

Scanraw Operator



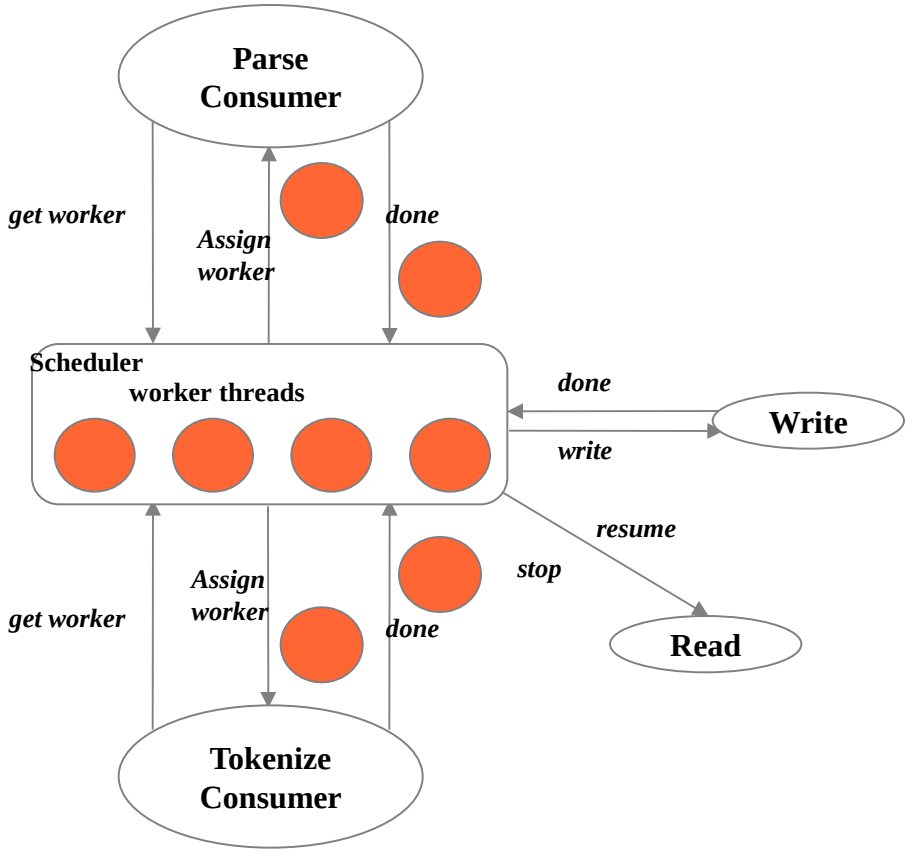
- Physical operator
- Parallel super-scalar pipeline

Scanraw Operator



- Physical operator
- Parallel super-scalar pipeline

Scanraw Operator



Task parallelism

- Stand-Alone threads
  - Read
  - Write
  - Scheduler
  - Tokenize Consumer
  - Parse Consumer
- Worker threads

UCMERCED

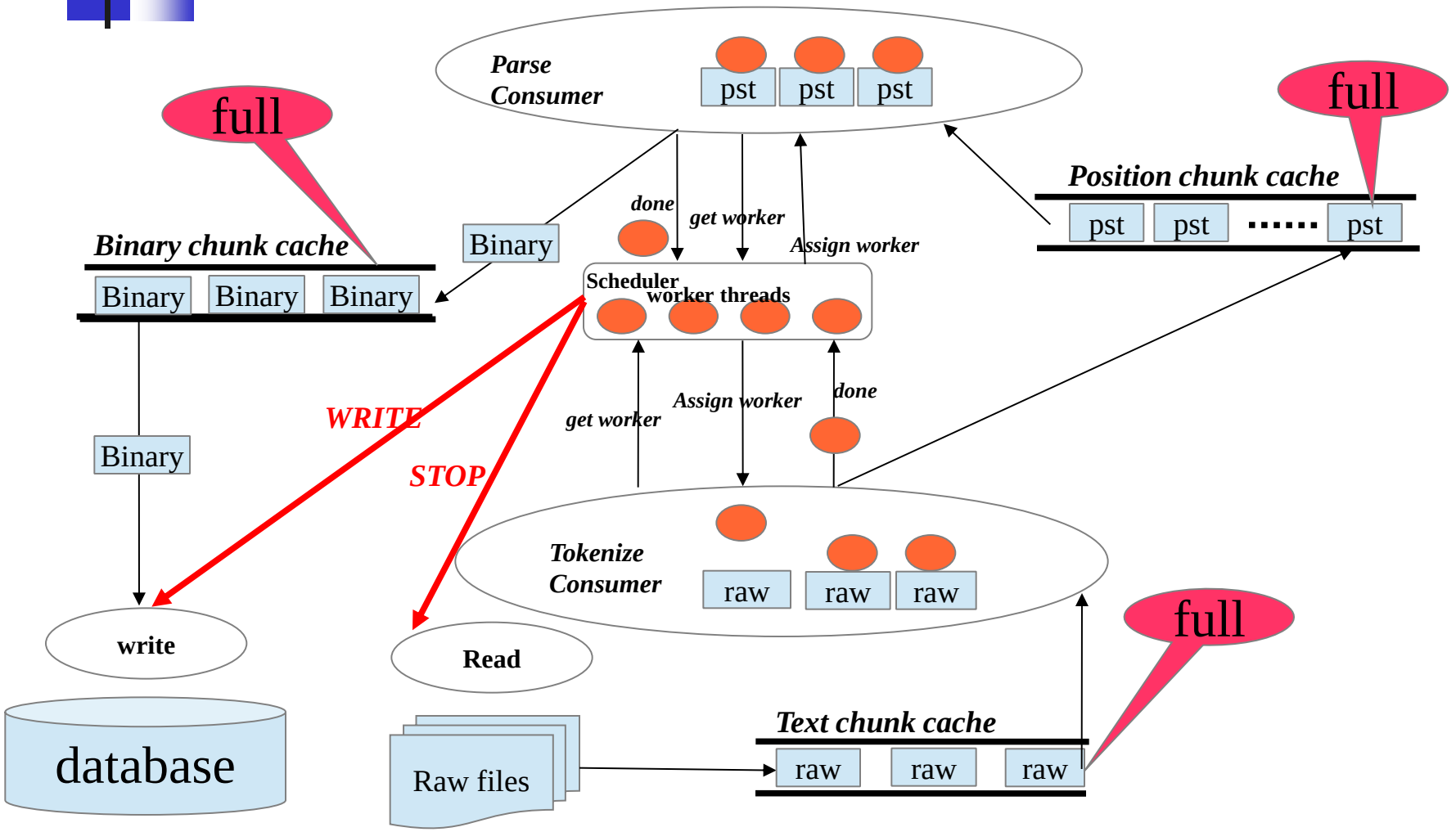
Speculative Loading

- Can we also achieve optimal execution time for a sequence of queries?

- How does speculative loading not interfere with query execution? Task-parallel/Pipeline
- How does speculative loading improve performance for a sequence of queries? Gradual Loading
- How do we guarantee that new chunks are loaded for every query? Safeguard Mechanism

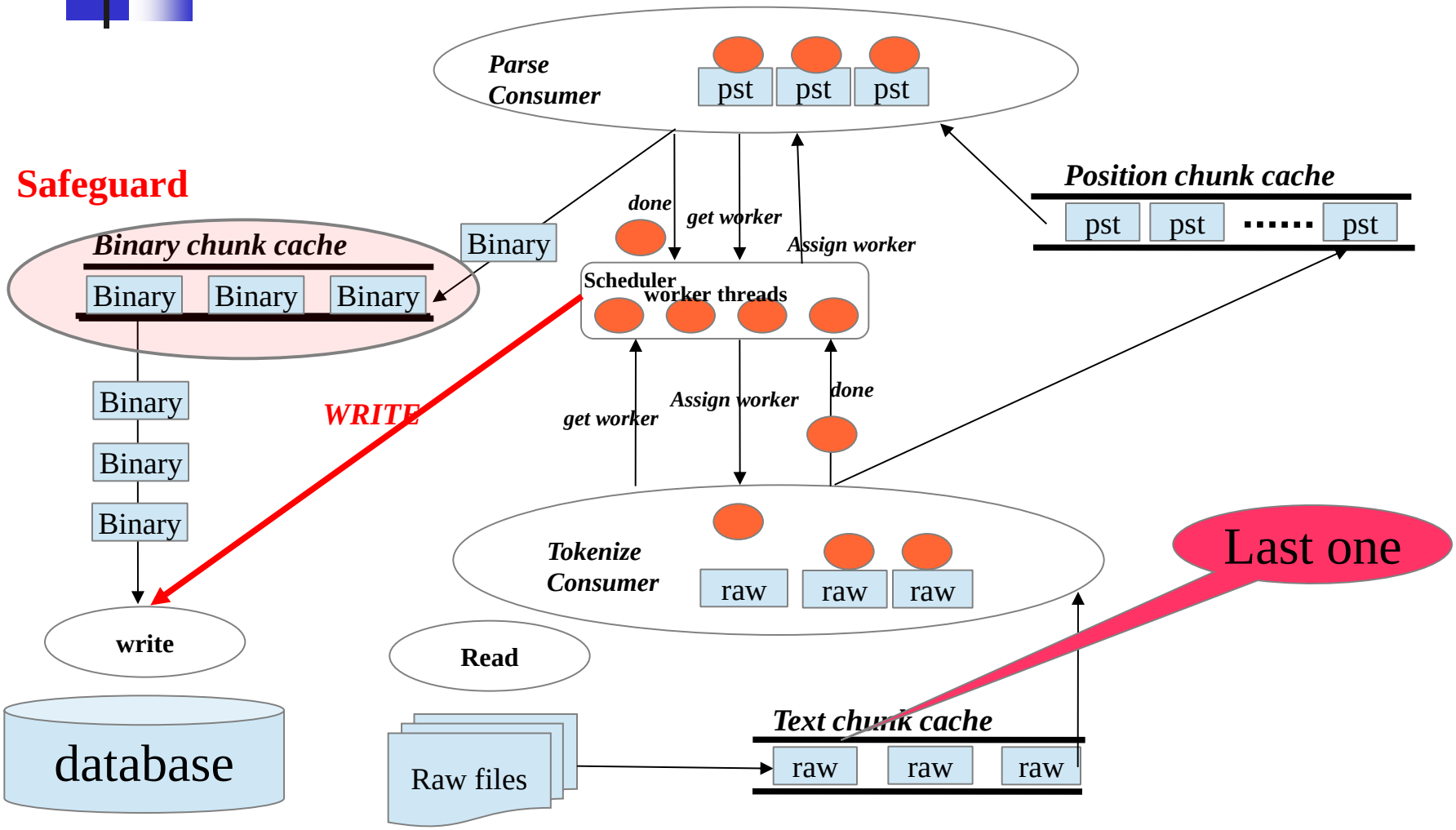
UCMERCED

Speculative Loading



UCMERCED

Speculative Loading



UCMERCED

Evaluation

Data : CSV files with 2 to 256 attrs, 2^{20} to 2^{28} lines; 20MB – 638GB size.

Query :

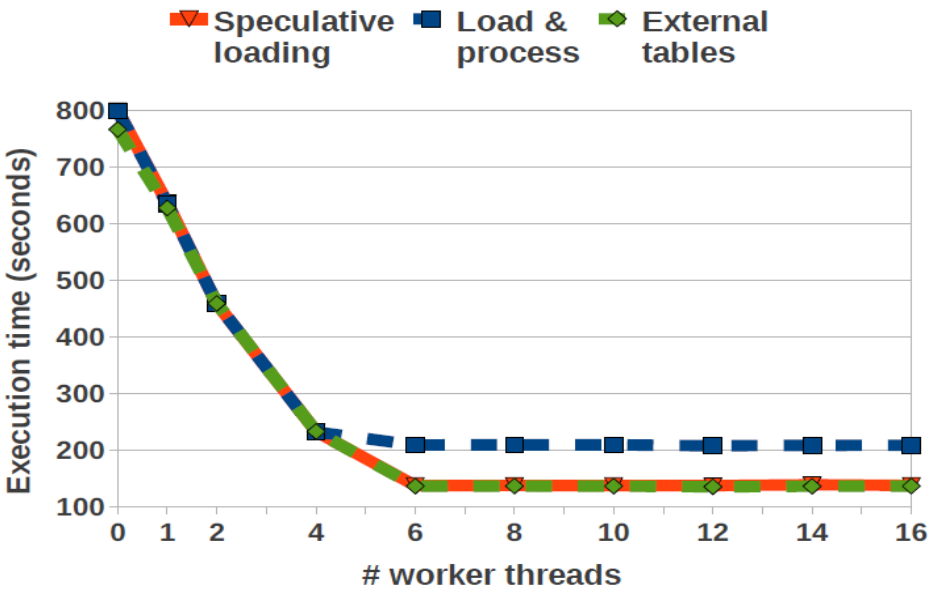
$$\texttt{SELECT SUM}\left(\sum_{j=1}^{K} C_{i_j}\right) \texttt{ FROM FILE}$$

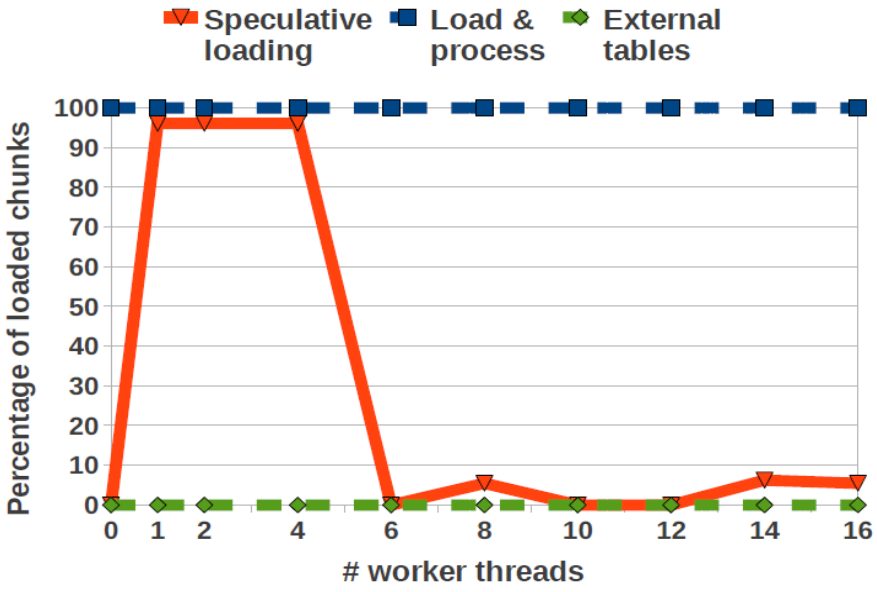
System : 2 AMD 8-core processors – 40 GB of memory, 4 TB 7200 RPM SAS hard-drives with I/O output 450MB/s

Illustration: 64 attrs, 2^{26} lines, 40GB

UCMERCED

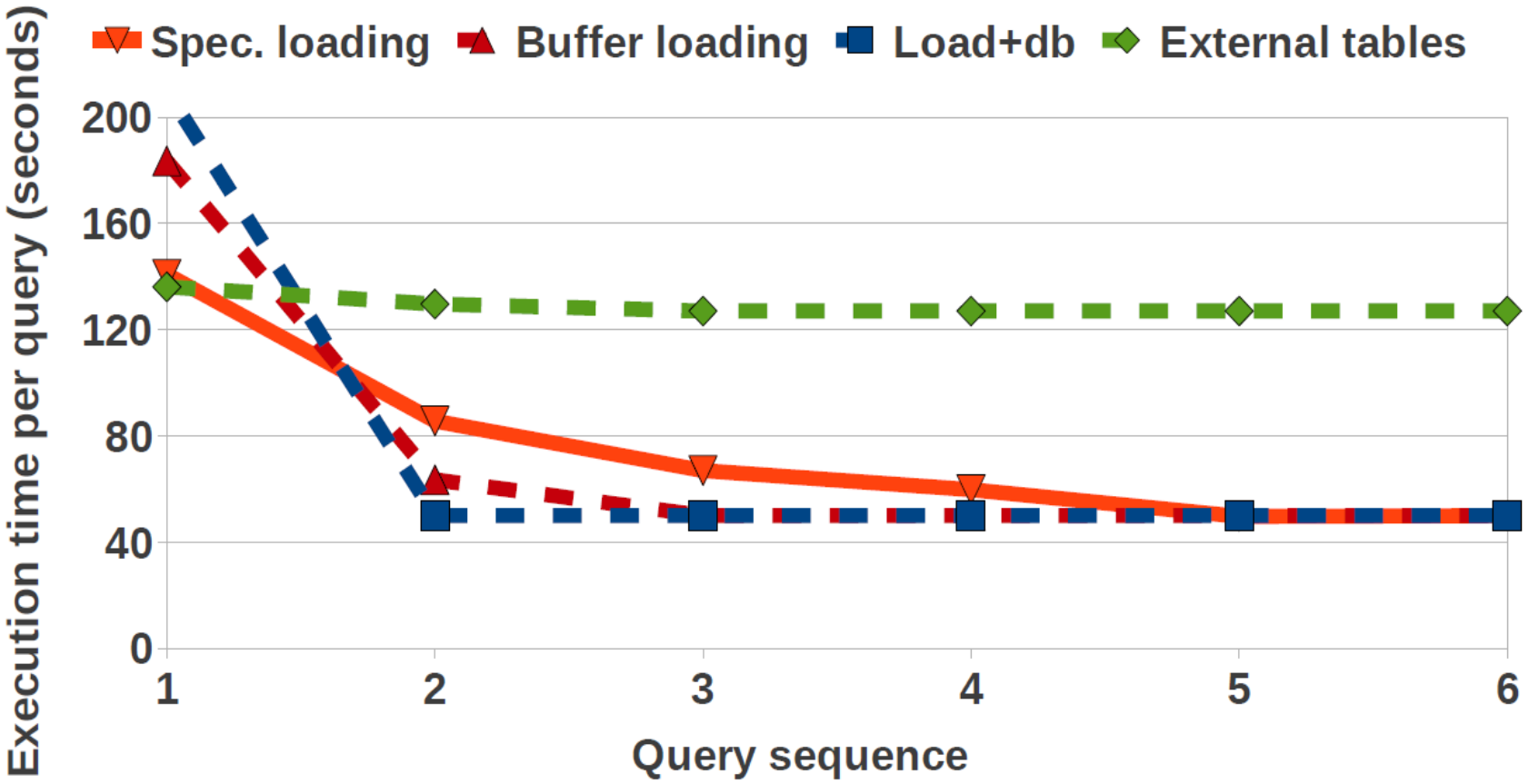
Optimal for the First Query?





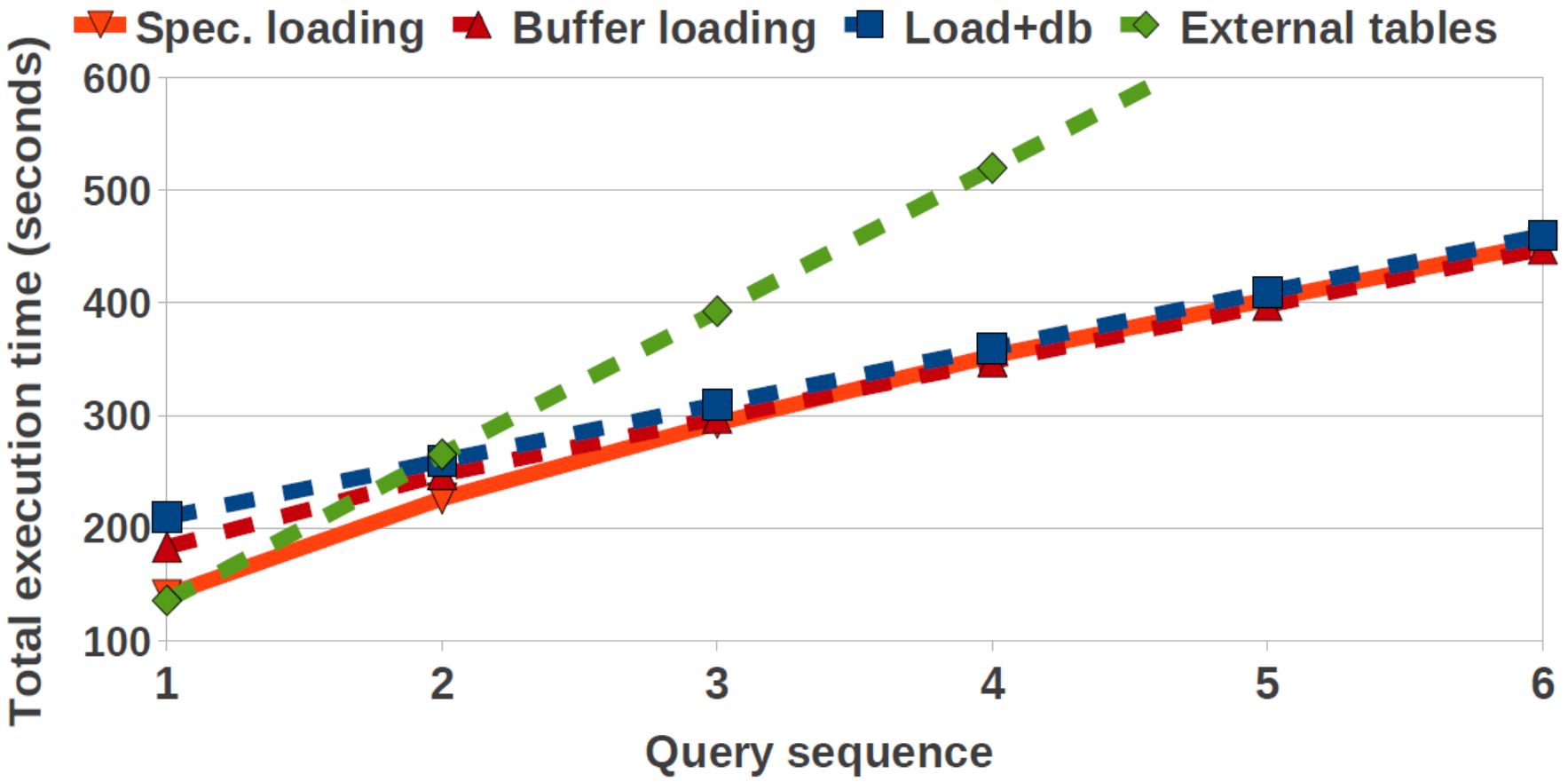
UCMERCED

Performance Improvement



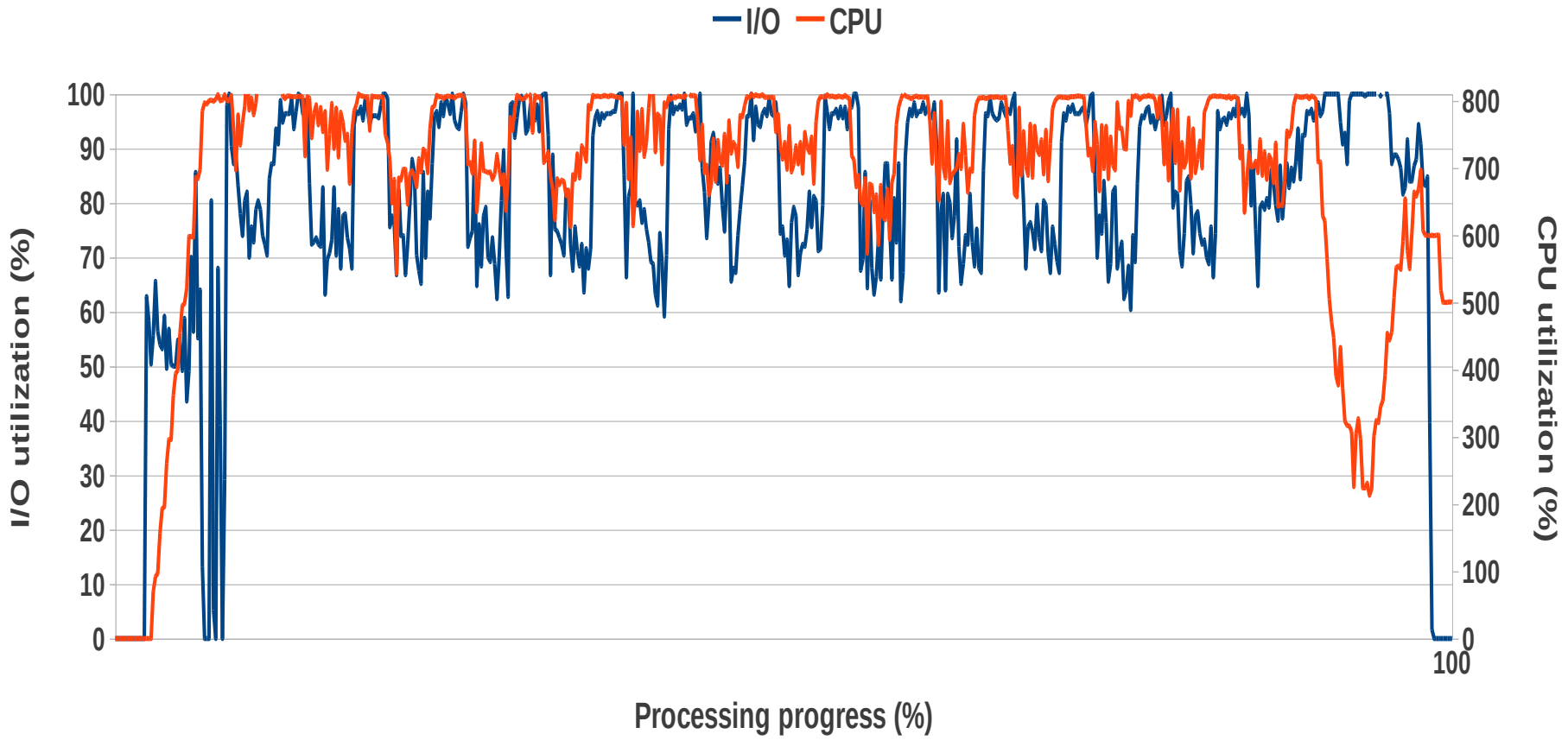
UCMERCED

Always Optimal?



UCMERCED

Resource Utilization

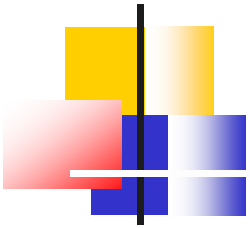


UCMERCED

Conclusions

- Scanraw Operator
  - Super-scalar pipeline architecture
  - No interference with query execution
  - Easy to integrate into database system
- Speculative Loading
  - Make full use of system resource
  - Improve performance for a sequence of queries
  - Always achieve optimal execution time

UCMERCED



Thank you!

Questions?