Parallel In-situ Data Processing with Speculative Loading

Yu Cheng, Florin Rusu

University of California, Merced
Outline

- Background
- Scanraw Operator
- Speculative Loading
- Evaluation
SAM/BAM Format

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

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

External table vs. Loading

External Table

Loading

Execution time

Loading time

Query time

Q1
Q2
Q3
Q4

Loading

Loading

External Table
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
Comparison

External table vs. Loading

<table>
<thead>
<tr>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Table</td>
</tr>
<tr>
<td>Q1</td>
</tr>
<tr>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
</tr>
<tr>
<td>Q4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loading</td>
</tr>
<tr>
<td>Loading</td>
</tr>
</tbody>
</table>

Optimal

Loading

loading time

optimal

External Table

Query time
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
Raw File Processing

Execution Engine

EXTRACT

Tokenize → Parse → Map

READ

line → tuple → page

WRITE

tuple → page

Storage
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
Where Does the Time Go?
Scanraw Operator

- Physical operator
- Parallel super-scalar pipeline

What is the optimal ratio?
Scanraw Operator

- Physical operator
- Parallel super-scalar pipeline

Diagram:
- Raw File
- Read
- Tokenize
- Parse
- Position buffer
- Binary chunks buffer
- Execution Engine
- Write
- DB
Task parallelism
- Stand-Alone threads
  - Read
  - Write
  - Scheduler
  - Tokenize Consumer
- Worker threads
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
Speculative Loading

Database

Raw files

Read

Text chunk cache

raw

raw

raw

Tokenize Consumer

raw

raw

raw

Scheduler

worker threads

donet

get worker

Assign worker

Position chunk cache

full

full

Write

STOP

Binary chunk cache

full

Binary

Binary

Binary

parse consumer

pst

pst

pst

Binary

Binary

Binary

get worker

Assign worker
Speculative Loading

- **Raw files**
- **Scheduler worker threads**
- **Safeguard**
- **Position chunk cache**
- **Parse Consumer**
- **Tokenize Consumer**
- **Text chunk cache**
- **Binary chunk cache**

- **ReadWrite**
- **Last one**
Evaluation

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

Query:

```
SELECT SUM(\sum_{j=1}^{K} C_{ij}) 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
Optimal for the First Query?

- Speculative loading
- Load & process
- External tables

Graph showing execution time (seconds) and percentage of loaded chunks for different numbers of worker threads.
Performance Improvement

Execution time per query (seconds)

Query sequence

- Spec. loading
- Buffer loading
- Load+db
- External tables
Always Optimal?
Resource Utilization

![Graph showing resource utilization over processing progress (%)](image-url)
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
Thank you!

Questions?