I/O-Efficient Statistical Computing with RIOT

Yi Zhang  Weiping Zhang  Jun Yang
Department of Computer Science, Duke University, USA

New Challenges In Massive Data Analysis

- Exploding data in sciences, business, society, ...
- Increasingly sophisticated analysis
- Lack of easy-to-use, efficient tools for statisticians/analysts
  - Traditional platforms often assume datasets fit in memory
  - Large datasets cause excessive and inefficient I/O

Example in R

- R: a popular open-source language for statistical computing

```r
# n points with coordinates stored in x[1:n], y[1:n]
d <- sqrt((x-x)^2+(y-y)^2+sqrt((x-x)^2+(y-y)^2))
s <- sample(n, 100) # draw 100 samples from 1:n
z <- d[s] # extract elements of d whose indices are in s
```

<table>
<thead>
<tr>
<th>Memory</th>
<th>y-x</th>
<th>x-x</th>
<th>(x-x)^2</th>
<th>y-y</th>
<th>(x-x)^2</th>
<th>sqrt(t(-))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swap</td>
<td>y</td>
<td>y</td>
<td>(x-x)^2</td>
<td>y</td>
<td>(x-x)^2</td>
<td>sqrt(t(-))</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Opportunities for Improvement

- Avoid Intermediate results
  - Multiple large intermediate results take up memory
  + Can we avoid them, but without hand-coding loops?
- Deferred & selective evaluation
  - Each expression gets evaluated in full immediately
  + Can we defer evaluation until necessary?
- Optimize data layouts & algorithms
  - Main-memory model offers few layout/algorithm options
  + Can we have better layouts/algorithms for out-of-core data?
- Algebraic rewrite
  - Operations are carried out in order of appearance
  + Can we use linear algebra laws for more efficient execution?

Déjà Vu for Database Researchers?

- Why don’t they use/extend a database?
- Current database systems are awful at storing arrays, executing numerical code, and optimizing linear algebra expressions
- And good luck with impedance mismatch!
- Try writing this piece of R code in SQL:
  ```r
  w <- solve(t(A) %*% A) %*% t(A) %*% b
  # Least square fit for y = w'x + e, with n covariates
  # b is a column vector of n observations, with corresponding inputs in A (n-by-n)
  ```

RIOT: R with I/O Transparency

- Allow user to stay with R—no new languages/APIs, no SQL
- Get I/O-efficiency without user intervention
- Support integration with database
- Blur boundary between R and backend processing

Achieving Transparency

- Leverage R’s extension mechanism
- Define types and operators to “mask” standard ones
- New types do not require data to be memory-resident
- New operators defer computation when possible

Execution

- Trigger computation when result is needed (e.g., for printing)
- Avoid intermediate results with pipelined execution
- Support I/O-efficient algorithms specialized for linear algebra

Optimization

- Work at the level of linear algebra operations
- Make cost-based choices of data layouts and algorithms

Database Integration

- Allow users to analyze data in databases without “glue” code
- Let users focus what to do instead of how to do it
  ```r
  A <- matrix.db("SELECT i, j, value FROM ... WHERE ...", ...)
  B <- matrix.db("SELECT...", ...)
  A %*% B + C
  ```
  - If A is “-“-shaped and B is “|“-shaped, compute A %*%B inside database
  - If A is “|“-shaped and B is “-“-shaped, load A and B and compute %*% in RIOT

Native Storage

- Flexible disk layout with different linearization options
- Optimized B-tree with special compression and reorganization tricks for arrays
  - Good for varying array sparsity across both space and time