Online Aggregation for Large MapReduce Jobs

Niketan Pansare\textsuperscript{1}, Vinayak Borkar\textsuperscript{2},
Chris Jermaine\textsuperscript{1}, Tyson Condie\textsuperscript{3}

\textsuperscript{1}Rice University, \textsuperscript{2}UC Irvine, \textsuperscript{3}Yahoo! Research
Outline

- Motivation
- Implementation
- Experiments
- Conclusion
Outline

- Motivation
- Implementation
- Experiments
- Conclusion
select avg(stock_price) from nasdaq_db where company = 'xyz';
(Note: final answer for this query is 1000)
select avg(stock_price) from nasdaq_db where company = 'xyz';

After 1 second,

- Conventional Database:

- With OLA extension:
  - Output range estimate: [0, 2000] with 95% probability
select avg(stock_price) from nasdaq_db where company = 'xyz';

After 2 minutes,

- Conventional Database:

- With OLA extension:
  - Output range estimate: [900, 1100] with 95% probability
select avg(stock_price) from nasdaq_db where company = 'xyz';

After 4 minutes,

- Conventional Database:

- With OLA extension:
  - Output range estimate: [950, 1040] with 95% probability
select avg(stock_price) from nasdaq_db where company = 'xyz';

After 6 minutes,

- Conventional Database:

- With OLA extension:
  - Output range estimate: [990, 1010] with 95% probability
select avg(stock_price) from nasdaq_db where company = 'xyz';

After 10 minutes,

- Conventional Database:

- With OLA extension:
  - Output range estimate: [995, 1005] with 95% probability
SELECT AVG(stock_price) FROM nasdaq_db WHERE company = 'xyz';

After 30 minutes,

- Conventional Database:

- With OLA extension:
  - Output range estimate: [999, 1001.5] with 95% probability
select avg(stock_price) from nasdaq_db where company = 'xyz';

After 2 hours,

- Conventional Database:
  - Output final answer: 1000
- With OLA extension:
  - Output final answer: 1000
Benefit of OLA

- If acceptably accurate answer reached quickly, the query can be aborted

After 6 minutes,
- Conventional Database:

- With OLA extension:
  - Output range estimate: $[990, 1015]$ with 95% probability

STOP EARLY !!!
Why Stop Early?

- Save **human time** (1 hour 54 minutes)
  - 'Answer 1000' v/s 'Estimate 1002.5'
    - For exploratory apps
    - Inaccuracies in ETL process
Why Stop Early?

- Save human time (1 hour 54 minutes)
  - 'Answer 1000' v/s 'Estimate 1002.5'
    - For exploratory apps
    - Inaccuracies in ETL process
- Save machine time → Cost ↓
Why Stop Early?

- Save human time (1 hour 54 minutes)
  - 'Answer 1000' v/s 'Estimate 1002.5'
    - For exploratory apps
    - Inaccuracies in ETL process
- Save machine time → Cost ↓
- Very important when dealing with large data
Why Stop Early?

- Save human time (1 hour 54 minutes)
  - 'Answer 1000' v/s 'Estimate 1002.5'
    - For exploratory apps
    - Inaccuracies in ETL process
- Save machine time → Cost ↓
- Very important when dealing with large data

Online Aggregation
- Introduced in 1997
- Significant research impact (606 citations)
- ACM SIGMOD Test of Time Award

But, limited commercial impact
- Database market (self-managed)
Cost model

- In Self-managed DB: costs are fixed
- In Cloud: You pay for amount of hardware used
  - Less resources → Less cost
  - 10 node cluster: 1h 54min → save $12.92/query on EC2
- User needs to justify the cost to the organization
Self-managed DB → Cloud

- **Cost model**
  - In Self-managed DB: costs are fixed
  - In Cloud: You pay for amount of hardware used
    - Less resources → Less cost
    - 10 node cluster: 1h 54min → save $12.92/query on EC2
  - User needs to justify the cost to the organization

- **Modifying engine** to support randomization
  - Traditional DB: Notoriously difficult
  - Cloud: Much simpler
Self-managed DB → Cloud

- Cost model
  - In Self-managed DB: costs are fixed
  - In Cloud: You pay for amount of hardware used
    - Less resources → Less cost
    - 10 node cluster: 1h 54min → save $12.92/query on EC2
  - User needs to justify the cost to the organization

- Modifying engine to support randomization
  - Traditional DB: Notoriously difficult
  - Cloud: Much simpler

- Therefore, OLA for cloud is an interesting problem
Extend existing approaches

- OLA over single machine
- OLA over multiple machine
- Why it won't work?
- How do we deal with those issues?
Extend existing approaches

- OLA over single machine
  - Confidence interval found using classical sampling theory
  - Tuples are bundled into blocks
  - Blocks arrive in random order
- OLA over multiple machines
- Why it won't work?
- How do we deal with those issues?
OLLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

Note: True answer = 55
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

- Example: Find SUM of below values
  
  5, 9
  7, 4, 2
  8, 3
  1, 10, 6
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

Sample = {}
Estimate = Not available

- 5, 9
- 7, 4, 2
- 8, 3
- 1, 10, 6
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

\[
\begin{align*}
\text{Sample} &= \{\} \\
\text{Estimate} &= \text{Not available}
\end{align*}
\]
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

Example: Find SUM of below values

Sample = {}

Estimate = Not available
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

- **Example:** Find SUM of below values
  
  Sample = \{13\}
  
  Estimate = \(13 \times 4 / 1 = 52\)
OLÁ over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

Sample = {13}
Estimate = $13 \times 4 / 1 = 52$
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

<table>
<thead>
<tr>
<th>Sample = {13}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate = 13 * 4 / 1 = 52</td>
</tr>
</tbody>
</table>

```
7, 4, 2  8, 3
5, 9
7, 4, 2
8, 3
1, 10, 6
```
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

\[
\begin{align*}
7, 4, 2 & \quad 8, 3 \\
5, 9 & \\
7, 4, 2 & \\
8, 3 & \\
1, 10, 6 & \\
\end{align*}
\]

Sample = \{13, 11\}

Estimate = \((13 + 11) \times 4 / 2 = 48\)
OLSA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

Sample = \{13, 11\}

Estimate = \((13 + 11) \times 4 / 2 = 48\)
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

| 7, 4, 2 | 8, 3 | 5, 9 |

Sample = \{13, 11\}

Estimate = \((13 + 11) \times 4 / 2 = 48\)
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

Example: Find SUM of below values

Sample = \{13, 11\}
Estimate = (13 + 11) * 4 / 2 = 48
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

Example: Find SUM of below values

Sample = \{13, 11, 14\}

Estimate = \((13 + 11 + 14) \times 4 / 3 = 50.67\)
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

```
|   | 7, 4, 2 | 8, 3 | 5, 9 | 1, 10, 6 |
```

Sample = \{13, 11, 14\}

Estimate = \((13 + 11 + 14) \times \frac{4}{3} = 50.67\)
OLÁ over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

- Example: Find SUM of below values

Sample = \{13, 11, 14\}

Estimate = (13 + 11 + 14) * 4 / 3 = 50.67
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

Sample = \{13, 11, 14\}

Estimate = (13 + 11 + 14) * 4 / 3 = 50.67
OLÁ over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

Sample = \{13, 11, 14\}

Estimate = \((13 + 11 + 14) * 4 / 3 = 50.67\)
OLA over single machine

- Confidence interval found using classical sampling theory
- Tuples are bundled into blocks
- Blocks arrive in random order

**Example:** Find SUM of below values

| 7, 4, 2 | 8, 3 | 5, 9 | 1, 10, 6 |

Sample = \{13, 11, 14, 17\}

Estimate = \((13 + 11 + 14 + 17) \times 4 / 4 = 55\)
Extend existing approaches

- OLA over single machine
  - Confidence interval found using classical sampling theory
  - Tuples are bundled into blocks
  - Blocks arrive in random order

- OLA over multiple machines
  - Blocks → Non-uniform → Size, Locality, Machine, Network
  - Processing time for block can be large and highly variable

- Why it won't work?
- How do we deal with those issues?
OLÁ over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

So, instead of

Example: Find SUM of below values

\[
\begin{array}{cccc}
7, 4, 2 & 8, 3 & 5, 9 & 1, 10, 6 \\
\end{array}
\]
OLAs over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

\[
\begin{array}{cccc}
7, & 4, & 2 & 8, & 3, & 5, & 9 & 1, & 10, & 6 \\
5, & 9 \\
8, & 3 \\
7, & 4, & 2
\end{array}
\]

X axis = Processing Time
OLA over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

7, 4, 2  8, 3  5, 9  1, 10, 6

- Blocks that take
  - long time to process = RED
  - Short time to process = Green
OLA over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

7, 4, 2  8, 3  5, 9  1, 10, 6

Arrows = Random Time Instances (Polling blocks)
OLA over multiple machines

- Blocks $\rightarrow$ Non-uniform $\rightarrow$ Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

- Example: Find SUM of below values

```
7, 4, 2  8, 3  5, 9  1, 10, 6
```
OLAs over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

\[
\begin{align*}
7, 4, 2 & \quad 8, 3 & \quad 5, 9 & \quad 1, 10, 6
\end{align*}
\]
OLA over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

7, 4, 2  8, 3  5, 9  1, 10, 6
OLA over multiple machines

- Blocks $\rightarrow$ Non-uniform $\rightarrow$ Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

$$
\begin{align*}
7, 4, 2 & \quad 8, 3 & \quad 5, 9 & \quad 1, 10, 6
\end{align*}
$$
OLA over multiple machines

- Blocks $\rightarrow$ Non-uniform $\rightarrow$ Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

- Example: Find SUM of below values

\[ \begin{align*}
7, 4, 2 & \quad 8, 3 & \quad 5, 9 & \quad 1, 10, 6
\end{align*} \]
OLA over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

\[
\begin{array}{c}
7, 4, 2 \\
8, 3 \\
5, 9 \\
1, 10, 6
\end{array}
\]
OLA over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

- 7, 4, 2
- 8, 3
- 5, 9
- 1, 10, 6
OLA over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

```
7, 4, 2  8, 3  5, 9  1, 10, 6
```
OLA over multiple machines

- Blocks → Non-uniform → Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

```
7, 4, 2
8, 3
5, 9
1, 10, 6
```
OLA over multiple machines

- Blocks $\rightarrow$ Non-uniform $\rightarrow$ Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

Notice, there are more arrows on red region than green region
OLA over multiple machines

- Blocks $\rightarrow$ Non-uniform $\rightarrow$ Size, Locality, Machine, Network
- Processing time for block can be large and highly variable

Example: Find SUM of below values

Notice, there are more arrows on red region than green region

**Inspection Paradox:** At any random time $t$, (stochastically) you will be processing those blocks that take long time
Extend existing approaches

- OLA over single machine
  - Confidence interval found using classical sampling theory
  - Tuples are bundled into blocks
    - Arrive in random order
- OLA over multiple machines
  - Blocks → Non-uniform → Size, Locality, Machine, Network
  - Processing time for block can be large and highly variable
- Why it won't work?
- How do we deal with those issues?
Why won't previous approach work?

- Inspection paradox → At the time of estimation, processing longer blocks

- Possible: correlation between processing time and value
  - Eg: count query
Why won't previous approach work?

- Inspection paradox → At the time of estimation, processing longer blocks

- Possible: correlation between processing time and value
  - Eg: count query

- Biased estimates → current techniques won't work
Why won't previous approach work?

- Inspection paradox → At the time of estimation, processing longer blocks.

- Possible: correlation between processing time and value.
  - Eg: count query.

- Biased estimates → current techniques won't work.

This effect is found experimentally in the paper: 'MapReduce Online'.
Why won't previous approach work?

- Inspection paradox → At the time of estimation, processing longer blocks

- Possible: correlation between processing time and value
  - Eg: count query

- Biased estimates → current techniques won't work

- Therefore, need to deal with inspection paradox in principled fashion
Extend existing approaches

- OLA over single machine
  - Confidence interval found using classical sampling theory
  - Tuples are bundled into blocks
    - Arrive in random order
- OLA over multiple machines
  - Blocks → Non-uniform → Size, Locality, Machine, Network
  - Processing time for block can be large and highly variable
- Why it won't work?
- How do we deal with those issues?
How do we deal with Inspection Paradox

- Capture timing information (i.e. processing time of block)
  - Along with values

- Instead of using classical sampling theory, we output estimates using bayesian model that:
  - Allows for correlation between processing time and values
  - And also takes into account the processing time of current block
Outline

- Motivation
- Implementation
- Experiments
- Conclusion
Implementation Overview

- Framework for distributed systems: MapReduce
  - Hadoop
    - Staged processing → Online
  - Hyracks (developed at UC Irvine)
    - Pipelining → "Online"
    - Architecture (and API) similar to Hadoop
    - http://code.google.com/p/hyracks/

- For estimates of "Aggregation",
  - 2 modifications to MapReduce (Hyracks)
  - Bayesian Estimator
Implementation Overview

- Framework for distributed systems: MapReduce
  - Hadoop
    - Staged processing $\rightarrow$ Online
  - Hyracks (developed at UC Irvine)
    - Pipelining $\rightarrow$ "Online"
    - Architecture (and API) similar to Hadoop

- For estimates of "Aggregation",
  - 2 modifications to MapReduce (Hyracks)
  - Bayesian Estimator
Implementation Overview

- Framework for distributed systems: MapReduce
  - Hadoop
    - Staged processing $\rightarrow$ Online
  - Hyracks (developed at UC Irvine)
    - Pipelining $\rightarrow$ "Online"
    - Architecture (and API) similar to Hadoop
    - http://code.google.com/p/hyracks/

- For estimates of "Aggregation",
  - 2 modifications to MapReduce (Hyracks)
  - Bayesian Estimator
Modifications to MapReduce (Hyracks)

- **Master**
  - Maintains random ordering of blocks
    - Logical not physical queue
  - Assigns block from head of queue
  - Block comes to head of queue → Timer starts (processing time)

- Two intermediates set of files
  - Data file → Values
  - Metadata file → Timing information
  - Shuffle phase of reducer
Modifications to MapReduce (Hyracks)

Client → Master

```
select sum(stock_price) from nasdaq_db group by company;
```

<table>
<thead>
<tr>
<th>Blk</th>
<th>Company</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk1</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk1</td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk2</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk3</td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk4</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk5</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk6</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk7</td>
<td>AAPL</td>
<td>4</td>
</tr>
</tbody>
</table>
Modifications to MapReduce (Hyracks)

Time t = 0

Client

Master

<table>
<thead>
<tr>
<th>Blk1</th>
<th>MSFT</th>
<th>AAPL</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk2</td>
<td>ORCL</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Blk3</td>
<td>AAPL</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Blk4</td>
<td>MSFT</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Blk5</td>
<td>ORCL</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Blk6</td>
<td>MSFT</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Blk7</td>
<td>AAPL</td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
Modifications to MapReduce (Hyracks)

Master maintains a logical queue of the blocks

<table>
<thead>
<tr>
<th>Block</th>
<th>MSFT</th>
<th>APPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blk2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blk3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blk4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blk5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blk6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blk7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time $t = 1$
Modifications to MapReduce (Hyracks)

Master randomizes the queue

Time $t = 1$
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Block</th>
<th>Company</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk1</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk2</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk3</td>
<td>AAPL</td>
<td>4</td>
</tr>
</tbody>
</table>

Time \( t = 2 \)

Master forks workers

<table>
<thead>
<tr>
<th>Block</th>
<th>Company</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk4</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk5</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk6</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk7</td>
<td>AAPL</td>
<td>4</td>
</tr>
</tbody>
</table>

Worker 1

Worker 2
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Block</th>
<th>Company</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk1</td>
<td>MSFT</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Blk2</td>
<td>ORCL</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Blk3</td>
<td>AAPL</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Blk4</td>
<td>MSFT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Blk5</td>
<td>ORCL</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Blk6</td>
<td>MSFT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Blk7</td>
<td>AAPL</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Workers request for blocks

Time \( t = 3 \)
Modifications to MapReduce (Hyracks)

Client → Master

Blk 6
MSFT 2
AAPL 4

Blk 5
ORCL 3

Blk 3
AAPL 4

Blk 1
MSFT 2

Blk 4
ORCL 3

Blk 7
MSFT 2
AAPL 4

Blk 2

Masters reads head of queue and assigns it to first worker

Time $t = 4$
Modifications to MapReduce (Hyracks)

Time $t = 5$

Worker 1 starts reading Blk6

Worker 1

Worker 2

Blk1 | MSFT | 2
----|------|---
Blk2 | ORCL | 3
Blk3 | AAPL | 4

Blk4 | MSFT | 2
Blk5 | ORCL | 3

Blk6 | MSFT | 2
Blk7 | AAPL | 4
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Block</th>
<th>Stock</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk1</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk2</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk3</td>
<td>AAPL</td>
<td>4</td>
</tr>
</tbody>
</table>

Time $t = 6$

Assigns Blk5 to Worker2

Worker 1

<MSFT, 2>

Worker 2

Blk6  MSFT  2
Blk7  AAPL  4
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Block</th>
<th>Company</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk 1</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk 2</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk 3</td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk 4</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk 5</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk 6</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk 7</td>
<td>AAPL</td>
<td>4</td>
</tr>
</tbody>
</table>

Worker 1 does its map task

Worker 2

Time \( t = 7 \)
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Blk 1</th>
<th>MSFT</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk 2</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk 3</td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk 4</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk 5</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk 6</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk 7</td>
<td>AAPL</td>
<td>4</td>
</tr>
</tbody>
</table>

Time $t = 8$

Worker 1

<MSFT, 2>

Worker 2

Reducer

Shuffle Phase

Reduce Phase

$t = 4$
Modifications to MapReduce (Hyracks)

Client → Master

Blk 1: MSFT, 2; AAPL, 4
Blk 2: ORCL, 3
Blk 3: AAPL, 4
Blk 4: MSFT, 2
Blk 5: ORCL, 3
Blk 6: MSFT, 2
Blk 7: AAPL, 4

Reducer

Shuffle Phase
<MSFT, 2>

Reduce Phase
<MSFT, 2>

Reducer-MSFT

Worker 1

Worker 2

Blk 6, Blk 5, Blk 3, Blk 1, Blk 4, Blk 7, Blk 2

Time \( t = 9 \)

\( t_{\text{process}} = 4 \)
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Block</th>
<th>Content</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk1</td>
<td>MSFT</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Blk2</td>
<td>ORCL</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Blk3</td>
<td>AAPL</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Blk4</td>
<td>MSFT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Blk5</td>
<td>ORCL</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Blk6</td>
<td>MSFT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Blk7</td>
<td>AAPL</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Reducer

Shuffle Phase

<MSFT, 2>

Reduce Phase

<MSFT, 2>

Reducer-MSFT

Random Time instance: Do estimation

Time t = 9
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Block</th>
<th>Company</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk1</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk2</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk3</td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk4</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk5</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk6</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>AAPL</td>
<td>4</td>
</tr>
</tbody>
</table>

Worker 1

Worker 2

Reducer

Shuffle Phase

Reduce Phase

<MSFT, 2>

Reducer-MSFT

Random Time instance: Do estimation

Time \( t = 9 \)

Random Time instance: Do estimation

\( t_{\text{process}} = 4 \)

\( t_{\text{process}} > 3 \)
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Block</th>
<th>MSFT</th>
<th>AAPL</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 1</td>
<td>MSFT</td>
<td>AAPL</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Block 2</td>
<td>ORCL</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 3</td>
<td>AAPL</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 4</td>
<td>MSFT</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 5</td>
<td>ORCL</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 6</td>
<td>MSFT</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 7</td>
<td>AAPL</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Worker 1

Worker 2

Reducer

Shuffle Phase

Reduce Phase

<MSFT, 2>

Reducer-MSFT

Random Time instance: Do estimation

Time t = 9

$t_{\text{process}} = 4$

$t_{\text{process}} > 3$
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Block</th>
<th>MSFT</th>
<th>AAPL</th>
<th>Process Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk1</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Blk2</td>
<td>ORCL</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Blk3</td>
<td>AAPL</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Blk4</td>
<td>MSFT</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Blk5</td>
<td>ORCL</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Blk6</td>
<td>MSFT</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Blk7</td>
<td>AAPL</td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

Worker 1

Worker 2

Reducer

Shuffle Phase

Reduce Phase

Reducer-MSFT

Random Time instance: Do estimation

Time t = 9

Estimation code

Blk6: \( t_{\text{process}} = 4 \)

Blk5: \( t_{\text{process}} > 3 \)
Modifications to MapReduce (Hyracks)

Client → Master

<table>
<thead>
<tr>
<th>Block</th>
<th>Company</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blk1</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk1</td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk2</td>
<td>ORCL</td>
<td>3</td>
</tr>
<tr>
<td>Blk3</td>
<td>AAPL</td>
<td>4</td>
</tr>
<tr>
<td>Blk4</td>
<td>MSFT</td>
<td>2</td>
</tr>
<tr>
<td>Blk5</td>
<td>ORCL</td>
<td>3</td>
</tr>
</tbody>
</table>

Worker 1

Worker 2

Reducer

Random Time instance: \[5.8, 8\]

Estimation code

Reduce Phase

Shuffle Phase

Reducer-MSFT

Time \(t = 9\)

\(t_{\text{process}} = 4\)

\(t_{\text{process}} > 3\)
Implementation Overview

- Framework for distributed systems: MapReduce
  - Hadoop
    - Staged processing → Online
  - Hyracks (developed at UC Irvine)
    - Pipelining → ”Online”
    - Architecture (and API) similar to Hadoop
    - http://code.google.com/p/hyracks/

- For estimates of ”Aggregation”,
  - 2 modifications to MapReduce (Hyracks)
  - Bayesian Estimator
Bayesian Estimator

- Why? → To deal with Inspection Paradox
Bayesian Estimator

- Why? → To deal with Inspection Paradox
- How?
  - Allows for correlation between processing time and values
  - And also take into account the processing time of current block
Bayesian Estimator

- Why? → To deal with Inspection Paradox
- How?
  - Allows for correlation between processing time and values
  - And also take into account the processing time of current block
- Implementation:
  - C++ code using GNU Scientific Library and Minuit2
  - Input: Data file and Metadata file from Reducer
  - Output: Confidence Interval → Eg:[995, 1005] with 95% prob
Parameterized model:

- Timing Information: $T_{\text{process}}$, $T_{\text{scheduling}}$
- Value: $X$
Bayesian Estimator (Model)

- Parameterized model:
  - Timing Information: $T_{\text{process}}, T_{\text{scheduling}}$
  - Value: $X$
- Underlying distribution
  - Classical sampling theory: $f(X)$
Bayesian Estimator (Model)

- Parameterized model:
  - Timing Information: $T_{\text{process}}$, $T_{\text{scheduling}}$
  - Value: $X$
- Underlying distribution
  - Classical sampling theory: $f(X)$
  - Our approach: $f(X, T_{\text{process}}, T_{\text{scheduling}})$
Bayesian Estimator (Model)

- Parameterized model:
  - Timing Information: $T_{\text{process}}, T_{\text{scheduling}}$
  - Value: $X$
- Underlying distribution
  - Classical sampling theory: $f(X)$
  - Our approach: $f(X, T_{\text{process}}, T_{\text{scheduling}})$
    - Correlation between $X, T_{\text{process}}$ and $T_{\text{scheduling}}$
Bayesian Estimator (Model)

- Parameterized model:
  - Timing Information: \( T_{\text{process}}, T_{\text{scheduling}} \)
  - Value: \( X \)

- Underlying distribution
  - Classical sampling theory: \( f(X) \)
  - Our approach: \( f(X, T_{\text{process}}, T_{\text{scheduling}}) \)

- Correlation between \( X, T_{\text{process}} \) and \( T_{\text{scheduling}} \)

- \( f(X \mid T_{\text{process}} > 100000000, T_{\text{scheduling}} = 22) \neq f(X) \)
Bayesian Estimator (Model)

- Parameterized model:
  - Timing Information: $T_{\text{process}}, T_{\text{scheduling}}$
  - Value: $X$

- Underlying distribution
  - Classical sampling theory: $f(X)$
  - Our approach: $f(X, T_{\text{process}}, T_{\text{scheduling}})$
    - Correlation between $X$, $T_{\text{process}}$, and $T_{\text{scheduling}}$
    - $f(X \mid T_{\text{process}} > 100000000, T_{\text{scheduling}} = 22) \neq f(X)$

- Estimation using Bayesian Machinery
  - Gibbs Sampler
    - Developed probability (or update) equations
Bayesian Estimator (Model)

- Parameterized model:
  - Timing Information: \(T\)
  - Value: \(X\)
  - Underlying distribution
  - Classical sampling theory:
  - Our approach:
    - Correlation between \(X\), \(T_{\text{process}}\) and \(T_{\text{scheduling}}\)
    - \(f(X | T_{\text{process}} > 10000000, T_{\text{scheduling}} = 22) \neq f(X)\)

- Estimation using Bayesian Machinery
  - Gibbs Sampler
    - Developed probability (or update) equations

Detailed discussion in the paper
Outline

- Motivation
- Implementation
- Experiments
- Conclusion
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates

- Experiment 1: (Real dataset)
  - `select sum(page_count) from wikipedia_log group by language`
  - 6 months Wikipedia log (220 GB compressed, 3960 blocks)
  - 11 node cluster (4 disks, 4 cores, 12GB RAM)
  - Uniform configuration: Machines, Blocks
  - 80 mappers and 10 reducer
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates

- Experiment 1: (Real dataset)
  - `select sum(page_count) from wikipedia_log group by language`
  - 6 months Wikipedia log (220 GB compressed, 3960 blocks)
  - 11 node cluster (4 disks, 4 cores, 12GB RAM)
  - Uniform configuration: Machines, Blocks
  - 80 mappers and 10 reducer
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates

- Experiment 1: (Real dataset)
  - 6 months Wikipedia log (220 GB compressed, 3960 blocks)
  - 11 node cluster (4 disks, 4 cores, 12GB RAM)
  - Uniform configuration: Machines, Blocks
  - 80 mappers and 10 reducers

- Experiment 2: (Simulated data set)

Reading the figures

Percentage of data processed
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates
- Experiment 1: (Real dataset)
  - 6 months Wikipedia log (220 GB compressed, 3960 blocks)
  - 11 node cluster (4 disks, 4 cores, 12GB RAM)
  - Uniform configuration: Machines, Blocks
  - 80 mappers and 10 reducers

Experiment 2: (Simulated data set)
Hypothesis:
- Randomized Queue required
- Allow correlation between processing time and value
- Convergence of estimates

Experiment 1: (Real dataset)
- 6 months Wikipedia log (220 GB compressed, 3960 blocks)
- 11 node cluster (4 disks, 4 cores, 12GB RAM)
- Uniform configuration: Machines, Blocks
- 80 mappers and 10 reducers
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates

- Experiment 1: (Real dataset)
  - 10% of data processed → Non-randomized: Inaccurate estimate
Experiments

- **Hypothesis:**
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates

- **Experiment 1: (Real dataset)**
  - 20% of data processed → Non-randomized: Inaccurate estimate

- **Experiment 2: (Simulated data set)**
  - ↑ correlation (Non-uniform configuration)
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates

- Experiment 1: (Real dataset)
  - Non-randomized → Inaccurate estimates
Experiments

- Hypothesis:
  - Randomized Queue required
  - **Allow correlation between processing time and value**
  - Convergence of estimates

- Experiment 1: (Real dataset)
  - Processing large block → no correlation detected
Experiments

- Hypothesis:
  - Randomized Queue required
  - **Allow correlation between processing time and value**
  - Convergence of estimates

- Experiment 1: (Real dataset)
  - Correlation detected → With correlation: Slightly more accurate

![Graph showing correlation](image)
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates

- Experiment 1: (Real dataset)
  - Correlation detected $\rightarrow$ With correlation: Unbiased

![Graph showing the comparison between with and without correlation]
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates

- Experiment 1: (Real dataset) → **Uniform Configuration** (low correlation)

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Correlation</td>
<td>With Correlation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

![Graph](image-url)
Experiments

- Hypothesis:
  - Randomized Queue required
  - **Allow correlation between processing time and value**
  - Convergence of estimates

- Experiment 1: (Real dataset) → **Uniform Configuration** (low correlation) + As ↑ data, likelihood takes over
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates
- Experiment 1: (Real dataset) → **Uniform Configuration** (low correlation) + As ↑ data, likelihood takes over → estimates similar

![Graph showing density and pagecounts](image)
Experiments

- Hypothesis:
  - Randomized Queue required
  - Allow correlation between processing time and value
  - Convergence of estimates
- Experiment 1: (Real dataset)
  - 6 months Wikipedia log (220 GB compressed, 3960 blocks)
  - 11 node cluster (4 disks, 4 cores, 12GB RAM)
  - Uniform configuration: Machines, Blocks
  - 80 mappers and 10 reducer
- Experiment 2: (Simulated data set)
  - ↑ correlation (Non-uniform configuration)
Outline

- Motivation
- Implementation
- Experiments
- Conclusion
Conclusion

- OLA over MapReduce
  - Statistically robust estimates
- Model that accounts for biases that can arise in distributed environment
- Little modification to existing MapReduce architecture
Thanks for your time and attention

Questions ?