Performance Optimization for Distributed Intra-Node-Parallel Streaming Systems
8th Int. SMDB Workshop in Conjunction with ICDE 2013, Brisbane

Matthias J. Sax#12  Malu Castellanos*2  Qiming Chen*2  Meichun Hsu*2

#DBIS Group, Humboldt-Universität zu Berlin
*Hewlett-Packard Laboratories, Palo Alto (CA)

1mjsax@informatik.hu-berlin.de  2{firstname.lastname}@hp.com

April 8th 2013
Introduction

Problem Statement

- Increasing the throughput of data-parallel streaming systems.
- Investigation batching techniques to decrease network overhead.
- Optimize the degree of parallelism for each operator in the data-flow.
Introduction

- *Data Intensive Computing*.
- Low latency requirements in many application.
  - Batch systems like Hadoop do not fit.
Introduction

- Data Intensive Computing.
- Low latency requirements in many applications.
- Batch systems like Hadoop do not fit.

- Streaming systems provide low latency, but limited throughput.
- New distributed streaming systems:
  - MapReduce inspired.
  - Introducing intra-node parallelism.
  - Apply user-code to data.
  - Examples: Storm, S4, Muppet.
Streaming systems are . . .

- designed to process infinite input (continuous data).
  - low latency (real time)
Stream Processing Systems (SPS)

Streaming systems are . . .

▶ designed to process infinite input (continuous data).
  ▶ low latency (real time)
▶ execute dataflows (usually DAGs).
Streaming systems are . . .

- designed to process infinite input (continuous data).
  - low latency (real time)
- execute dataflows (usually DAGs).
- send tuples individually from node to node.
Distributed Stream Processing Systems (DSPS)

Intra-node parallelism in DSPS

- Each node in the dataflow has an associated *degree of parallelism* (dop).
Intra-node parallelism in DSPS

- Each node in the dataflow has an associated *degree of parallelism* (dop).
Intra-node parallelism in DSPS

- Each node in the dataflow has an associated *degree of parallelism* (dop).
  - allows for higher throughput due to data-parallel processing
  - difficult to configure dop manually

```
logical operator
user-defined connections
operator instance
```

```
dop=2
---
dop=1
---
dop=2
---
dop=2
---
dop=3
---
dop=2
---
dop=1
```
Intra-node parallelism in DSPS

- Each node in the dataflow has an associated *degree of parallelism* (dop).
  - Allows for higher throughput due to data-parallel processing
  - Difficult to configure dop manually

- Different connection patterns available. (e.g., “random” or “grouping”)
Overview Batching

- Batching is a well known technique to increase throughput.
- Tuples are grouped together in a batch, and processed “at once”.
Overview Batching

- Batching is a well known technique to increase throughput.
- Tuples are grouped together in a batch, and processed “at once”.

![Diagram of batch processing]
Overview Batching

- Batching is a well known technique to increase throughput.
- Tuples are grouped together in a batch, and processed “at once”.

+ Decreases average overhead (per tuple).
- Increases tuple latency.
Key-based batching:
  ▶ Need to batch the correct tuples together.
Batching for Data-Parallel Streaming Systems

Key-based batching:

- Need to batch the correct tuples together.
Key-based batching:

▶ Need to batch the correct tuples together.

Batching scheme is more difficult for multiple consumers using different key-attributes:

▶ Distinct batches.
▶ Shared batches.
Cost Model

Time needed to process a tuple in one operator instance:

- queuing time
- pure processing time (ppt)
- fixed message overhead $n$
- data transfer time $s$ (dependent on message size)
- shipping time

processing time
Cost Model

Time needed to process a tuple in one operator instance:

- queuing time
- shipping time
- ppt
- multiple output tuples
- ppt
- shipping time
- ppt
- shipping time
- ppt
- shipping time
- time
Cost Model

Time needed to process a tuple in one operator instance:
Cost Model

Processing consecutive tuples with different input rates:

No Optimization possible:

Optimization necessary:
Performance Optimization for Distributed Intra-Node-Parallel Streaming Systems

Cost Model

Time needed to process a tuple in one operator instance:

- **queuing time**
- **pure processing time**
- **shipping time**
- **processing time**
- **inter tuple arrival time**
- **idle**
Cost Model

Time needed to process a tuple in one operator instance:

- Queuing time
- Pure processing time
- Shipping time
- Inter tuple arrival time
Decrease shipping time by batching:
Cost Model

Increase consumer *dop* to increase inter tuple arrival time:

\[
\left( \frac{20 \text{ tuples}}{\text{ms}} / 2 \right)^{-1} = 0.1 \text{ms}
\]

\[
\left( \frac{20 \text{ tuples}}{\text{ms}} / 4 \right)^{-1} = 0.2 \text{ms}
\]
Optimization Problem

What is the best \textit{dop} for each node in the dataflow?

What is the best \textit{batch size} for each node in the dataflow?
Optimization Problem

What is the best dop for each node in the dataflow?

What is the best batch size for each node in the dataflow?

▶ Observation: dop and batch size depend on each other.
▶ Claim: We need to optimize both together.
▶ Objective: Minimize average processing latency.
Optimization Algorithm

- Single pass (linear complexity).
Optimization Algorithm

- Single pass (linear complexity).
- Starts at the sources.
Optimization Algorithm

- Single pass (linear complexity).
- Starts at the sources.
  - Does not change dop of sources.

\[ \text{dop: 1} \]
\[ \text{dop: 2} \]
Optimization Algorithm

- Single pass (linear complexity).
- Starts at the sources.
  - Does not change dop of sources.
  - Computes needed batch size for sources.

```
dop: 1
batch: 100

dop: 2
batch: 1
```
Optimization Algorithm

- Single pass (linear complexity).
- Starts at the sources.
  - Does not change dop of sources.
  - Computes needed batch size for sources.
- Computes needed batch size and dop for all other node.

```
dop: 1
batch: 100
```
```
dop: 2
batch: 1
```
```
dop: 10
batch: 50
```
Optimization Algorithm

- Single pass (linear complexity).
- Starts at the sources.
  - Does not change dop of sources.
  - Computes needed batch size for sources.
- Computes needed batch size and dop for all other node.

```
dop: 1  
batch: 100

dop: 10  
batch: 50

dop: 1  
batch: 1

dop: 2  
batch: 1

dop: 1  
batch: 1

dop: 4  
batch: 1
```
Prototype on Top of Storm

- Implemented prototype on top of Twitter’s Storm:
  - Batching layer is completely transparent to system and user-code.
  - Batch is implemented as *batch tuple*.
  - Optimization algorithm computes *batch size* and *dop* and inserts (de-)batching wrappers.
Prototype on Top of Storm

- Implemented prototype on top of Twitter’s Storm:
  - Batching layer is completely transparent to system and user-code.
  - Batch is implemented as *batch tuple*.
  - Optimization algorithm computes *batch size* and *dop* and inserts (de-)batching wrappers.

- Topologies:
  - Adjusted Linear Road Benchmark. (600MB; 12M tuples)
  - Sentiment Analysis on Tweets. (140MB; 1M tweets)
Runtime Results

- Linear: 1145
- Road: 1401
- Sentiment: 3561
- Analysis: 3163

- Not optimized
- Only dop
- Only batching
- Fully optimized
Latency Results Linear Road

Latency reported per batch:
 Parse BS=2048 ⇒ single tuple latency (avg) = 0.15ms.
Future Work

- Consider resource limitations.
- Optimize operator to machine mapping.
- How to deal with Quality of Service constraints?
- Dynamic batching and scaling.
Summary

- Batching techniques for data parallel stream processing.
- Cost model and optimization algorithm that computed optimal:
  - degree of parallelism
  - batch size
- Evaluation shows a performance gain up to a factor of 20.
Summary

- Batching techniques for data parallel stream processing.
- Cost model and optimization algorithm that computed optimal:
  - degree of parallelism
  - batch size
- Evaluation shows a performance gain up to a factor of 20.

Questions?

Thanks!
Example Topologies

Linear Road: (in braces: \textit{dop/batchSize})

Sentiment Analysis: (in braces: \textit{dop/batchSize})
Distinct Batches
Shared Batches
Example: Batching with Error
**Cost Model Formulas**

**Definition**

If a task outputs tuples $t_1, \ldots, t_n$ while processing an input tuple $t$, we call $t_1, \ldots, t_n$ child tuples of $t$. Each input tuple of any node forms a *processing tree*. The processing tree $PT(t)$ of tuple $t$ consists of $t$ and all *recursively* emitted child tuples, i.e., all children of all child tuples and so on.

**Definition**

Let $PT(t)$ be the processing tree of tuple $t$. The *latency* $l(t)$ of a tuple $t$ is: $l(t) = \max(t'.\text{delete}|t' \in PT(t)) - t.\text{create}$.

Objective function:

$$\min \ \text{avg}(l(t)|\forall t \in I)$$
Calculating the DOP

We need a dop, such that the tuple inter arrival time is bigger than the pure processing time:

$$\min\{dop_c|dop_c > \text{ppt} \cdot r\}$$
Calculating the Output Rate and Batch Size

Without batching:

\[ r_o = \left( \max \{ l_i, ppt + n + t \cdot s \} \right)^{-1} \]

With batching:

\[ r_o = \frac{b}{\max \{ b \cdot l_i, n + b \cdot (ppt + t \cdot s) \}} \]

\[ \Leftrightarrow \min \left\{ b \left| b \geq \frac{n}{l_i - ppt - (t \cdot s)} \right. \right\} \]
TopographyOptimizer

\[ P \leftarrow \text{all source nodes} \]

\[ \text{while } P \text{ is not empty do} \]

\[ \text{for all } p \in P \text{ do} \]

\[ \text{if input latency is smaller than processing time then} \]

\[ b \leftarrow \text{calculate batch size to reduce shipping time} \]

\[ \text{if } b > B_{\text{max}} \text{ then } b = B_{\text{max}} \]

\[ \text{increase dop of } p \text{ to increase } l_i \]

\[ \text{end if} \]

\[ \text{end if} \]

\[ \text{calculate output rate } r_o \]

\[ \text{end for} \]

\[ C \leftarrow \text{all unprocessed nodes with known input rate} \]

\[ \text{for all } c \in C \text{ do} \]

\[ \text{dop}_c \leftarrow \text{calculate dop such that } l_i < \text{ppt} \]

\[ \text{end for} \]

\[ P \leftarrow \text{all } c \in C \text{ with outgoing edges} \]

\[ \text{end while} \]
TopographyOptimizer (detailed) I

Input: data flow \( D = (V, E) \)
\( S \leftarrow \{ s \in V | \not\exists (v, s) \in E \} \)
\( P \leftarrow S \)
\( V' \leftarrow V \setminus S \)
while \( P \neq \emptyset \) do
  for all \( p \in P \) do
    if \( l_p < p_p + n_p + t_p \cdot s_p \) then
      \( b \leftarrow \left\lceil \frac{n_p}{l_p - p_p - t_p \cdot s_p} \right\rceil \)
      if \( b > B_{\text{max}} \) and \( p \notin S \) then
        \( b_p \leftarrow B_{\text{max}} \)
        \( P(p) \leftarrow \{ v \in V | \exists e = (v, p) \in E \} \)
        \( \text{dop}_p \leftarrow \left\lceil (\sum_{p' \in P(p)} r_{p'}) \cdot \left( \frac{n_p}{B_{\text{max}}} + p_p + t \cdot s_p \right) \right\rceil \)
      end if
    end if
  end for
  \( r_p \leftarrow \text{dop}_p \cdot b_p \cdot (n_p + b_p \cdot (p_p + t_p \cdot s_p))^{-1} \)
end while
$C \leftarrow \{V' \mid \text{all input rates are calculated already}\}$

**for all** $c \in C$ **do**

$P(c) \leftarrow \{v \in V \mid \exists e = (v, c) \in E\}$

$dop_c \leftarrow \lceil \sum_{p \in P(c)} (p_c \cdot r_p) \rceil$

**end for**

$V' \leftarrow V' \setminus C$

$P \leftarrow \{c \in C \mid \exists e = (c, v) \in E\}$

**end while**