MapReduce-Style Processing of Fast Sensor Data

DEBS 2013 Grand Challenge

Authors: Kasper G. S. Madsen, Li Su, Yongluan Zhou
Presentation: Li Su
Outline

• Enorm

• Distributed solution for the grand challenge.

• Experiments
  • Scalability
  • Computation sharing between overlapped windows
  • Latency
Enorm

- Low-latency distributed stream processing system
- Native support for windows
- Sharing of computation between overlapped windows
Enorm: Computation Model

MapReduce-Style Computation Model

- **Map**
  - Input: List(tuple)
  - Output: List(tuplet)

- **Compute**
  - Input: List(tuplet)
  - Output: KV store
  - Internal: update

- **KV store**
  - Input: List(tuplet)
  - Output: List(tuplet)

- **Consolidate**
  - Input: Event
  - Output: List(tuplet)
  - Internal: read
Enorm: Sharing Computation

- **Sharing Strategy:**
  - Calculate disjoint segments
  - Update one segment per input tuple
  - Consolidate multiple segments to generate final window results

- **Non-Sharing Strategy:**
  - Update each window per input tuple
  - No consolidation
Enorm: Sharing Computation

Cost Model:

\[(\text{Sharing Cost}) - (\text{Non-Sharing Cost})\]

\[
E_s = v \cdot t \cdot C_u + t \cdot f \cdot \sum_{i=1}^{n} M_i \cdot C_f
\]

\[
E_w = n \cdot v \cdot t \cdot C_u
\]

\begin{tabular}{|c|p{10cm}|}
\hline
\textbf{n} & number of windows \\
\hline
\textbf{v} & input rate \\
\hline
\textbf{f} & output frequency \\
\hline
\textbf{C_f} & consolidation cost (2 segments) \\
\hline
\textbf{C_u} & update cost (per tuple) \\
\hline
\textbf{M_i} & average number of segments in consolidation (i’th window) \\
\hline
\textbf{t} & elapsed time \\
\hline
\end{tabular}
DEBS 2013: GC solution

- One job to handle four queries
  - Reduce data transferring cost
  - Share computation between queries.

- Query plan design:
  - Components with heavy workload should be scalable
  - Reduce data transferring volume between components
DEBS 2013: GC Solution

**M1 Input partitioning**

- Data in range $R(Y)$ are sent to instance $[Y \mod (#M1)]$
- $(Y) = [t + Y*s, t + (Y + 1)*s + o]$ $(o = 1.5 \text{ s in GC})$

**Query Plan**

Example: $(t=0, s=10, o=5)$

[Diagram showing query execution flow with labels: M1, CP1, CP2, CP3, PS, CA, CS1, CS2, CS3. Psd* = Player sensor data, Ca* = Computed aggregates.]
Experiments

- Experiments conducted on Amazon EC2
- Latency: (0.1% input tuples, average 1524 milliseconds)
- Scalability
- Sharing computation between windows

M1 and CP1 are combined; CP2 are parallelized using 1, 4, 8 instances;
Experiments

* Sharing Computation

* Sharing strategy depends on job and consolidation frequency.
Questions?

Thanks!