Big Data Infrastructure at Microsoft: From Research to Production

Lidong Zhou
Microsoft Research
Foundation:
- Large-Scale Distributed Storage
- Data Flow Machinery
- Declarative Data Parallel Language

Big Data Infrastructure: The Evolution
SCOPE/Cosmos in Production: 2010 - 2013

Scale
Maximum Utilization and Throughput with High Reliability At Low Cost

- 110K Commodity machines Across 4 clusters
- 60%-80% CPU Utilization
- 56s Terasort 1000 nodes

Storage growth
64 Petabytes 2010

1.2 Exabytes 2013

Ecosystem
Bing, Ad Center, MSN, Maps, Windows Phone, Xbox Live, Windows Live, Office365, STB, ...

Jobs per day
95K/day 2013

4K Developers

170 Feature Teams

100's Applications

Simplicity
Developers, Researchers, Data Scientists, PM, Product Management, Marketing, and Sales

Customers
Big Data as a Service

Users
Familiar Tools Excel, Web Apps, Reporting

Developers
Scope (20 Lines)

MapReduce (200 Lines)

Courtesy of Big Data Team
SCOPE: Database Meets Map/Reduce

REFERENCE @"/shares/searchDM/SearchLogApi.dll";
USING MS.Internal.Bing.DataMining.SearchLogApi;

//Search Merge Log Impressions
SML =
    VIEW "/shares/searchDM/SearchLogPageView.view"
    PARAMS (Start = @"2013-07-10", End = @"2013-07-11")
;
//Windows Blue distinct users
WindowsBlueClicks =
    SELECT
        Request_ClientId AS Client,
        QueryParser.GetFcsNormalizedQuery(Query_RawQuery) AS Query,
        SUM(PageClicks_Count > 0 ? 1 : 0) AS Clicks,
        MAX(Metrics_DwellTime) AS DwellTime
    FROM
        SMLPageView
    WHERE
        Market == "en-us"
        AND Request_OSInfo.ProductName == "Windows 8.1"
;
//Windows Blue user sessions
WindowsBlueSessions =
    REDUCE WindowsBlueClicks ON Client
    USING MySessionReducer()
;
//Cook for later use
OUTPUT WindowsBlueSessions
    TO SSTREAM @WindowsBlueSessions@@
    CLUSTERED BY Vertical SORTED BY Client
;
SQL relational algebra

Predicates

Custom Reduce Function

Courtesy of Big Data Team
Big Data Infrastructure: The Evolution

Foundation:
• Large-Scale Distributed Storage
• Data Flow Machinery
• Declarative Data Parallel Language

Holistic Code Optimization
• Database Query Optimization
• Program Analysis and Compiler Optimization


OSDI’12
NSDI’12
PeriSCOPE: Pipeline-aware Holistic Code Optimization

Much Deeper!
Optimization Steps

Step 1: Construct inter-procedural flow graph

Step 2: Add safety constraints for skipping shuffling code

Step 3: Transform code for reducing shuffling I/O
Column Reduction: Reduce Number of Columns

Flow Graph

Data Shuffling

Flow Graph

string domain = ExtractURL(row["url"]);

row["domain"] = ExtractURL(row["url"]);
Early Filtering: Reduce Number of Rows

if (row["impr"] > MAX_IMPR) continue;

if (row["impr"] > MAX_IMPR) continue;

if (row["impr"] > MAX_IMPR) continue;
Smart Cut: Reduce Size of Each Row

Flow Graph

Data Shuffling

Flow Graph
Coverage Study*

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Eligible jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Reduction</td>
<td>4,052 (14.05%)</td>
</tr>
<tr>
<td>Early Filtering</td>
<td>3,020 (10.47%)</td>
</tr>
<tr>
<td>Smart Cut</td>
<td>1,544 (  5.35%)</td>
</tr>
<tr>
<td>Overlapped Total</td>
<td>6,397 (22.18%)</td>
</tr>
</tbody>
</table>

* Study on 28,838 jobs collected from SCOPE clusters in 2010/2011.
Significant I/O Reduction Observed
Research to Production

- State-of-art research in OSDI
- Validated with real jobs

Surprise: Not good enough!
- Absolutely do no harm: correctness and performance
- Coverage and overhead
- Complexity and tool maturity

Image credits:
http://m.rgbimg.com/cache1nvK96/users/o/oz/ozetsky/600/mfe0irG.jpg
http://cdn2.everyjoe.com/wp-content/uploads/2013/05/shocked-baby-146x104.jpg
Big Data Infrastructure: The Evolution

Foundation:
- Large-Scale Distributed Storage
- Data Flow Machinery
- Declarative Data Parallel Language

Holistic Code Optimization
- Database Query Optimization
- Program Analysis and Compiler Optimization

Scheduling and Resource Management
- Coordinated scheduling
- Opportunistic tasks
- Corrective actions

OSDI’12
NSDI’12
OSDI’14
Scheduling at Scale

Jobs process gigabytes to petabytes of data and issue peaks of 100,000 scheduling requests/seconds.

Clusters run up to 170,000 tasks in parallel track 14,000,000 pending tasks and each contains over 20,000 servers.

Incrementally rolled out from September to December 2013.
Scheduling Quality

- 60-90% median CPU utilization
- Largely balanced load

- Opportunistic tasks fill the gaps (e.g., during weekends)
- Negligible queuing time for regular tasks
Big Data Infrastructure: The Evolution

Foundation:
- Large-Scale Distributed Storage
- Data Flow Machinery
- Declarative Data Parallel Language

Holistic Code Optimization
- Database Query Optimization
- Program Analysis and Compiler Optimization

Scheduling and Resource Management
- Coordinated scheduling
- Opportunistic scheduling
- Corrective actions

Beyond Batch Processing
- Graph Computation
- Machine Learning and Deep Learning
- Streaming

OSDI’12
NSDI’12 Eurosyst’13
OSDI’14 SoCC’15
NSDI’16
Big Stream Computation

• Continuous input
• Near real-time computation
• Scaling to thousands of nodes
• Fault tolerant
• Strongly consistent

Search data streams

Click prediction, behavior targeting etc.
Streaming Data Flow

- **Input streams**
- **Output streams**
- **Channels**
- **Vertices**

**Timeline**
- X: t₁
- X: t₂

**Events**
- 3, 4, 5, 6, 7
- a
- a, b

**Replay of upstream events**
- Missing or duplicate events
- Rebuild the state

**Symbols**
- R
- X
- M
Decoupling Vertically

*rStream*

Provides the illusion of reliable and asynchronous communication channels
Decoupling Horizontally

Timeline

rVertex
Replayable vertex, can replay from any snapshot

\[ s_1 = \langle 2, \{a\}, t_1 \rangle \]
\[ s_2 = \langle 3, \{b\}, t_2 \rangle \]
\[ s_3 = \langle 4, \{d\}, t_3 \rangle \]
Power of Abstraction

• Easy to reason about correctness

• Enabling powerful optimizations seamlessly
  Move reliable persistent writes off the critical path

• Allowing different instantiations throughout life cycle
  • Offline mode to test, profile, and debug individual vertices
  • Optimized implementation when deployed; simple ones for validation
  • Replication based failure recovery
  • Duplicate execution to handle stragglers and planned maintenance
High complexity
48 stages
18 joins of 5 different types
21.3 TB in-memory state

Massive scalability
Reads 61TB + Write 61TB
7 billions of input events
6 billions of output events
3000+ long-running tasks

Fault tolerance
Handles both planned failures and unplanned outages automatically
Research and Production: Lessons and Experiences

Research
• Deep insights
• Well founded architecture and methodology
• Simple abstractions
• Fundamental principles

Production
• Keep it simple and operation friendly
• Unexpected *will* happen at scale
• Service mindset: test, validate, deploy, and operate at scale
• No regression, no significant complexity, no unpredictable behavior
Big Data Infrastructure: What’s Next

- Convergence of database, systems, programming language, hardware architecture, machine learning and artificial intelligence
- Heterogeneous workloads on heterogeneous hardware: scheduling and resource management
- Continuous, interactive, and rich-structured big data processing

➡️ Research and production better together for greater impact

Image credits:
http://i.telegraph.co.uk/multimedia/archive/02055/babies-hug_2055066i.jpg