Stream Warehousing

Theodore Johnson
AT&T Labs – Research
johnsont@research.att.com
Outline

• What is Stream Warehousing?
• Case Study: Darkstar
• Technical Issues
  – Update propagation
  – Temporal consistency
  – Scheduling
What is Stream Warehousing

• Data Stream Management System
  – Fast processing in main memory
    • Small data windows, 1-pass processing
  – Data reduction and alerting

• Data Warehouse
  – Long-term data storage
  – Curated data sets
  – Fuse data from many sources
  – Derived data products, complex analytics

• Data Stream Warehouse
  – A data warehouse with continual real-time updates.
Data Transformation in the Warehouse
Building Application in the Warehouse
Application: Darkstar

- AT&T Labs – Research project.
- Collect diverse and large-scale data feeds from network elements
- Use for
  - networking research,
  - data mining (e.g. correlate network events with failures),
  - alerting,
  - troubleshooting
- The network is a large and complex system
  - Not just IPV4.
- Argus
  - He Yan, Zihui Ge
- Ptolemy
  - Zihui Ge, Don Caldwell, Bill Beckett
Darkstar: Mining Vast Amounts of Data

Ethernet Access

IPTV

IP Backhaul

Enterprise IP, VPNs

Route monitors

(OSPFmon, BGPmon)

Authentication/ logging (tacacs)

Config

Syslog

SNMP Polling (router, link)

Netflow

Deep Packet Inspection (DPI)

Device service monitoring

(CIQ, MTANet, STREAM)

Active service and connectivity monitoring

Customer feedback – IVR, tickets, MTS

Layer one

Tickets

Alarms

Mobility

Authentication/ logging (tacacs)
ARGUS: Detecting Service Issues...

- **Goal:** detect and isolate *actionable* anomaly events using comprehensive end-to-end performance measurements (e.g. GS tool)
  - Sophisticated anomaly detection and heuristics
  - Spatial localization
  - Accurately accounts for service performance that varies considerably by time-of-day and location
- **Impact:**
  - Reduced detection time from days to approx. 15 mins for detecting data service issues
    - Operational nation-wide monitoring data service performance for 3G and LTE (TCP retransmission, RTT, throughput from GS Tool)
Approach: Mobility Localization Hierarchy

Collect end-to-end Performance Data

SITE
RNC
SITE
RNC
SITE
RNC
SITE
RNC
SITE
RNC

SGSN

GGSN

Market
Sub-Market
Sub-Market
SGSN
SGSN
RNC
RNC
SITE
SITE

SITE
SITE
SITE
SITE
SITE
SITE
SITE
SITE
SITE
SITE
Case Example: Silent CE Overload Condition

- **ARGUS detected event:** 2 Columbia 3G Ericsson SGSN’s impacting RNC’s in West Virginia, Norfolk, and Richmond
- No other indication of issue
- Topology highlighted CE used by only impacted SGSNs

**ARGUS alarm: clmamdorpnn2**
(TCP retransmissions)

**CE Utilization flattening**

- **RCA:** “6148 48 port 1gig card is limited to a shared 1 gig bus for each set of 8 gig ports”
ARGUS As A General Capability...

RTT anomalies (SGSN level)

Spike in call drop rate on MSC hndvacxca1

Social media (Twitter) NY outage

Node metrics, active measurements (CBB, IPAG WIPM delay)
Ptolemy

*Use network visualization and convenient data exploration to help network operators with network health monitoring and service problem troubleshooting*

• **1.** At-a-glance view of network topology and state

![Network Map](image)

• Visualization to summarize important information on network health
  - Color-coded

• Complimentary to ticketing system – reporting issues below “alarming” status

http://ptolemy.research.att.com/
http://ptolemy.research.att.com/mobility
Example 1: Japan Earthquake, March 11th 2011

Assess damage, identify remaining capacity

Loss of many links out of Japan. What’s left?
Example 1: Japan Earthquake, March 11th 2011

Identify traffic shifts, no congestion

Increase in link load as traffic re-routed

Link load
DataDepot

• Data warehousing system developed for stream warehousing
  – (Relatively) independent of the underlying database.

• Technologies for pushing updates through a warehouse
  – Update Propagation
  – Temporal consistency
  – Real-time scheduling in a stream warehouse
  – Lukasz Golab, Vladislav Shkapenyuk
Managing a Stream Warehouse

- Continually arriving data
Managing a Stream Warehouse

- Continually arriving data
- Is loaded into temporally partitioned base tables
Managing a Stream Warehouse

• Continually arriving data
• Is loaded into temporally partitioned base tables
• Updates propagate to higher level data products.
Incremental Updates

- Only propagate the increment.
- Update only those partitions whose sources have new data.
- How can we determine if a source partition has more recent data?
“make” doesn’t work
“make” doesn’t work
“make” doesn’t work
Update Propagation

• We can build complex apps if we’re confident that all updates get propagated.
• 1\textsuperscript{st} version of DD: used \textit{make}-style algorithm
  – Not correct for complex configurations
• Developed update propagation theory
• 2\textsuperscript{nd} version : has scheduling restrictions (read/write locks)
  – Led to poor real-time responsiveness
• 3\textsuperscript{rd} version : no scheduling restrictions
  – Uses a small amount of additional metadata.
  – Similar to a vector timestamp.
  – SSDBM 2011
Effects of Scheduling Restrictions

Update propagation from CPU_RAW to CPU

Starting-timestamp update protocol

Interval-timestamp update protocol
Consistency in a Stream Warehouse

- Traditional notion of consistency: a snapshot of the system.
- In a big, complex system, you can’t take a NOW snapshot.
- In most cases, you eventually reach a point where you are reasonably confident about the state of the system in the recent past.
- CIDR 2011
Data Arrives in a Smear Over Time

- Partially filled partition
- Late arriving data
- Very late data
Number of windows per package

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>Number of Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>100000</td>
<td>0</td>
</tr>
<tr>
<td>200000</td>
<td>2</td>
</tr>
<tr>
<td>300000</td>
<td>4</td>
</tr>
<tr>
<td>400000</td>
<td>6</td>
</tr>
<tr>
<td>500000</td>
<td>8</td>
</tr>
<tr>
<td>600000</td>
<td>10</td>
</tr>
<tr>
<td>700000</td>
<td>12</td>
</tr>
</tbody>
</table>
Many Data Feeds

• The value of a warehouse is the ability to correlate different streams of information.
  – Correlate periods of high packet loss on the active measurement probes with the router CPU and Memory utilization on the routers on the path between the measurement probes.

• Different feeds have different time lags
  – Active measurements: 45 minutes, SNMP: 15 minutes

• Darkstar
  – Warehouse of network performance, configuration, and alert data
  – Used for research, billing, network troubleshooting
  – 100 data feeds, 700 tables as of December 2010.
Query Stability

• How do I know when the data is stable enough to query?

• What is stable enough?
  – Data will never change
  – Data won’t change much.
  – I’ll take whatever is there.
Consistency Levels

• Punctuations on partitions that indicate completeness.
• Vagueness of real-life means that they are best guesses.
• We use the following in our running examples
  – *Open*: The partition should have some data in it.
  – *Closed*: The partition will not change.
  – *Complete*: The partition will not change, and all data has been received.
    • E.g. we know that there are five packages per window, and they will arrive at most 1 hour late.
    • Motivated by specific needs of DataDepot users.

• *Closed* is a guess
  – *WeaklyClosed, StronglyClosed*
• Label each base table partition with a temporal consistency level.

• Use source-specific information to infer how certain we can be that all data for a partition has arrived.
  – Tends to be a hazy notion.

• Sometimes we have a hierarchy
  – Complete > StronglyClosed > Closed > Open
  – But not in general.
• Infer on a partition-wise basis, for each consistency level separately
• **Simple rule**: a partition has consistency level C if all source partitions have consistency level C.
• Can make use of the properties of the defining query to improve the inference.
Update Consistency

• We might know that some tables naturally require their partitions to have a particular consistency (update consistency) to be useful.
  – Router alerts: Open
  – Per-day usage summaries: Closed

• We can reduce update cost by only updating a partition if it would achieve a particular level of consistency
  – Per-day summary fed by 5-minute updates: 288 updates when only 1 is needed.

• Labeling all tables in the warehouse is an excessive burden on the DBA.
  – Label important final-result tables, and infer the update consistency for the others.
Consistency Levels

- Many consistency levels are possible.
- Closed is a guess.
  - WeaklyClosed: probably stable.
  - StronglyClosed: almost certainly stable.
- Other levels
  - MostlyClosed: Few values will change
  - MostlyFull: Most expected records have arrived.
- The consistency levels might not form a hierarchy
Scheduling

• Need to schedule updates to avoid resource thrashing.

• Real Time scheduling problem: some very long jobs, some very short jobs.
  – Global scheduling is the most efficient
  – But, it is easy to generate infeasible task sets with low resource utilization using global scheduling.

• Catch-up processing can generate temporary overloads
  – Due to broken feeds, data quality debugging, etc.
  – Can’t discard updates during overload (unlike DSMS)
  – Need to perform catch-up without affecting real-time tasks.
Conclusions

• Optimization, service quality, and security of large-scale, complex systems require a stream monitoring infrastructure.

• Data Stream Warehousing enables near real-time applications
  – Alerting and troubleshooting using near real-time and historical data

• Next steps:
  – Moving to cloud infrastructure