Enabling Real Time Data Analysis

Divesh Srivastava, Lukasz Golab, Rick Greer, Theodore Johnson, Joseph Seidel, Vladislav Shkapenyuk, Oliver Spatscheck, Jennifer Yates

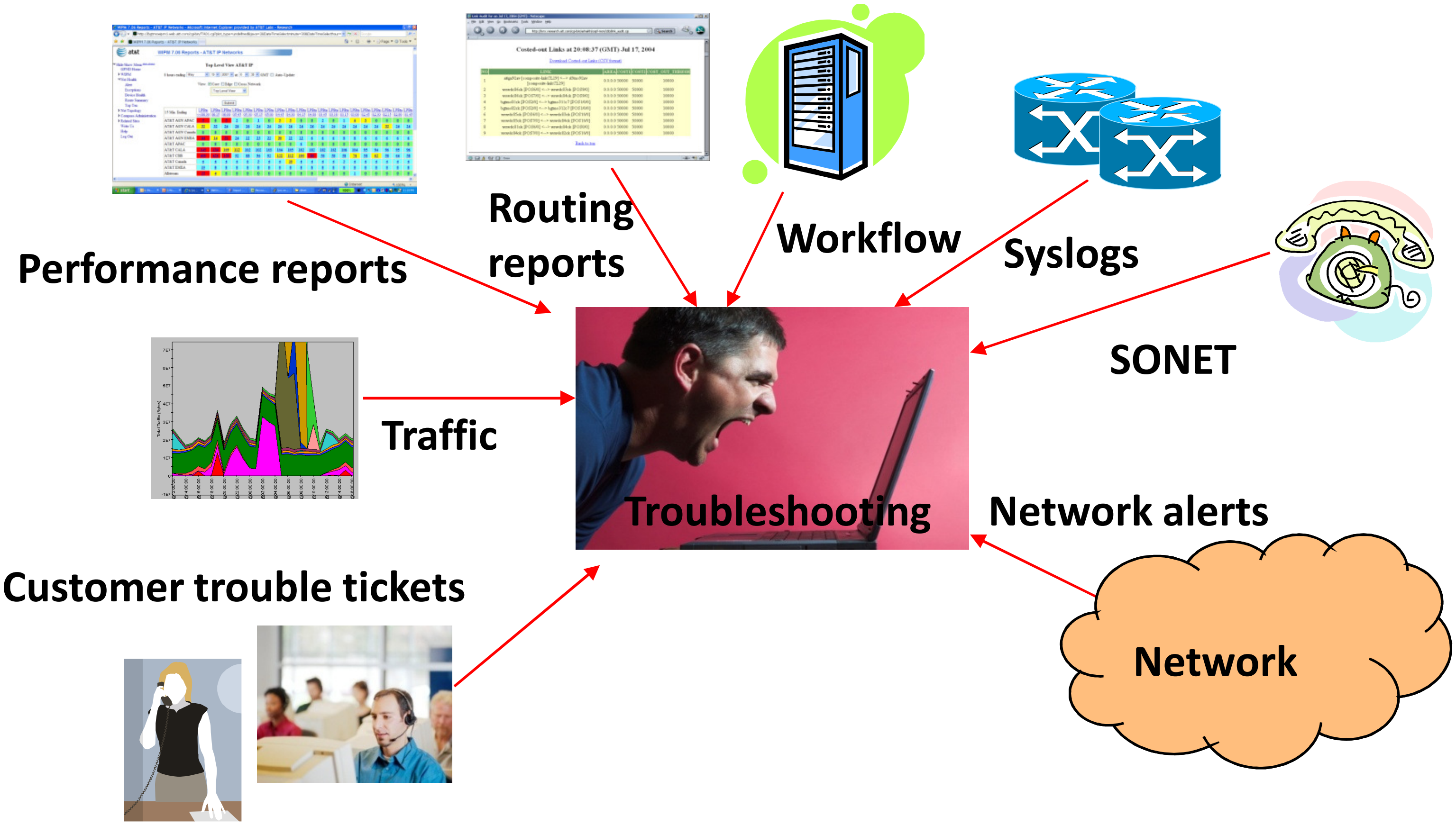
AT&T Labs - Research

Evolution of Data Analysis

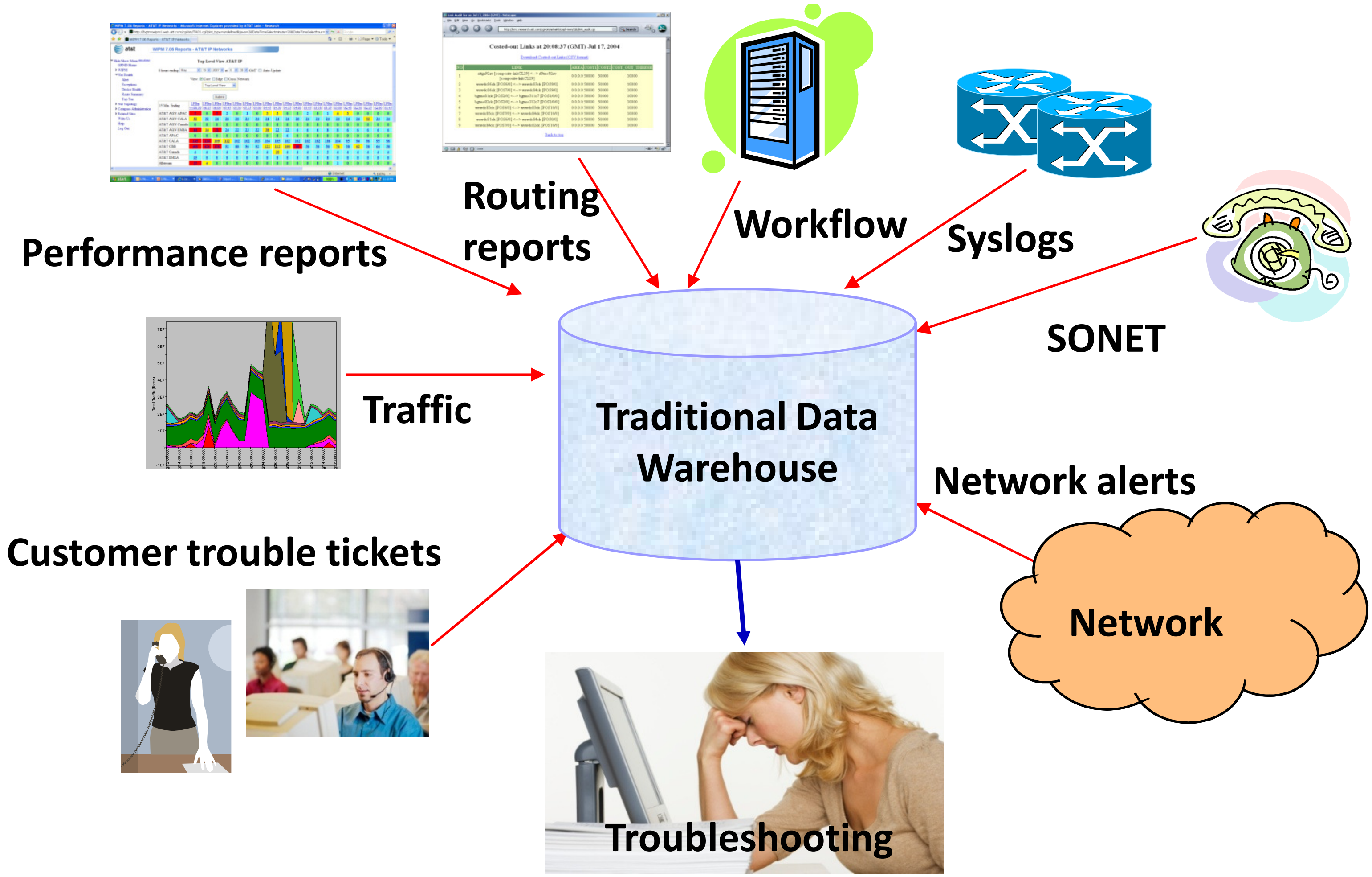
	1980s	1990s	2000s	2010s
Analysis	Offline reports	OLAP, ad hoc analysis	Streaming queries	Real time analysis
Drivers	Banking, airlines	Sales, CRM, Marketing	Alerting, Fraud	Security, Healthcare
	Transactional databases	Data warehouses	Separate DSMS, DW	Integrated DSMS+SDW

at&t

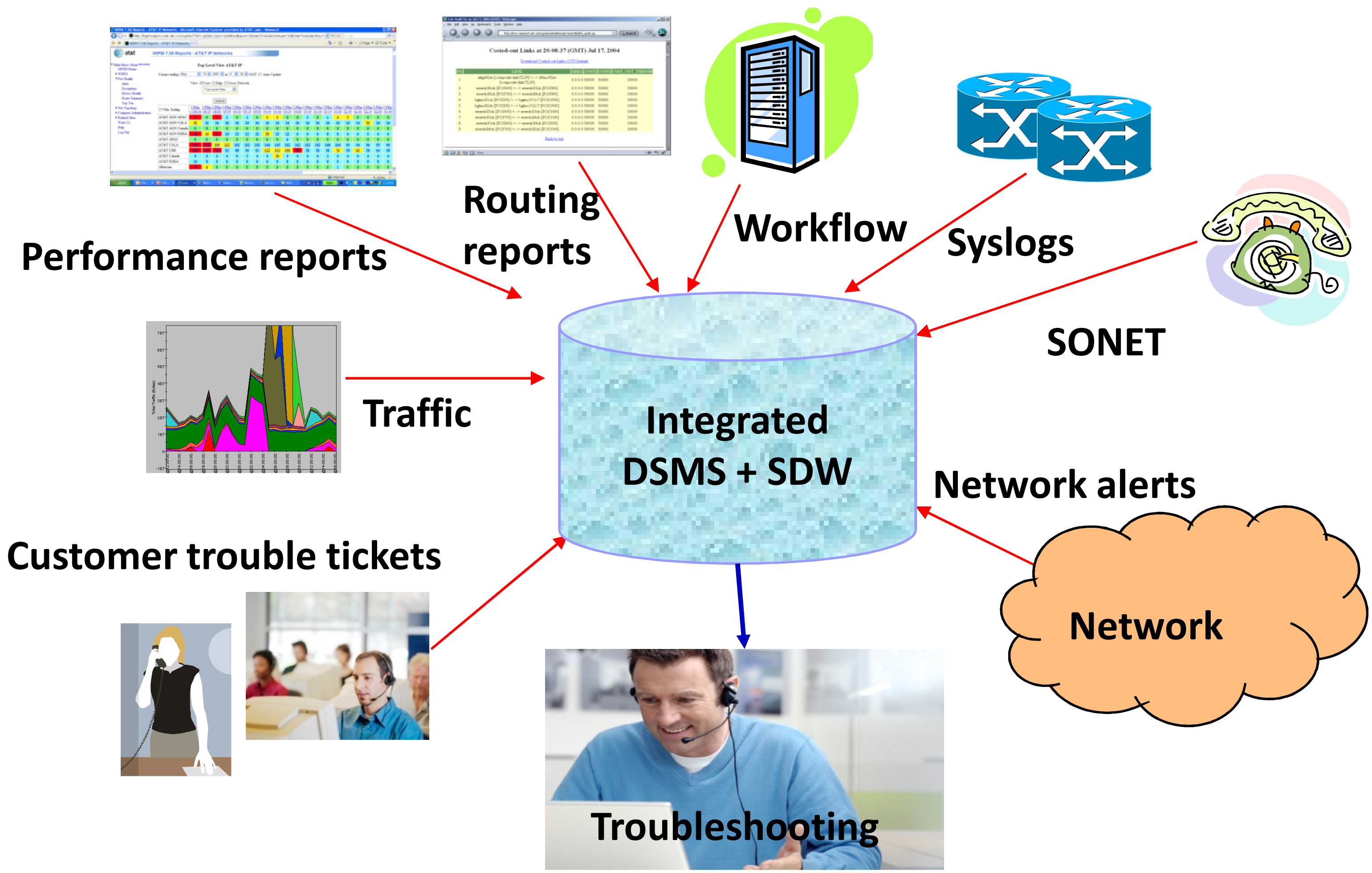
Challenge: Enabling Real Time Analysis



Non-Solution: Traditional Data Warehouses



Solution: Streaming Data Management



Goal: Integrated DSMS + SDW

- Storage
  - Long term storage of historical data in tables, materialized views
  - Update tables, materialized views in real time (minutes, not hours)

- Queries
  - Aggregate massive volumes (Gbit/sec) of streaming data
  - Join across multiple data sources
  - Correlate real-time data with historical data
  - Real-time alerting (reaction time in minutes, not days)

Outline

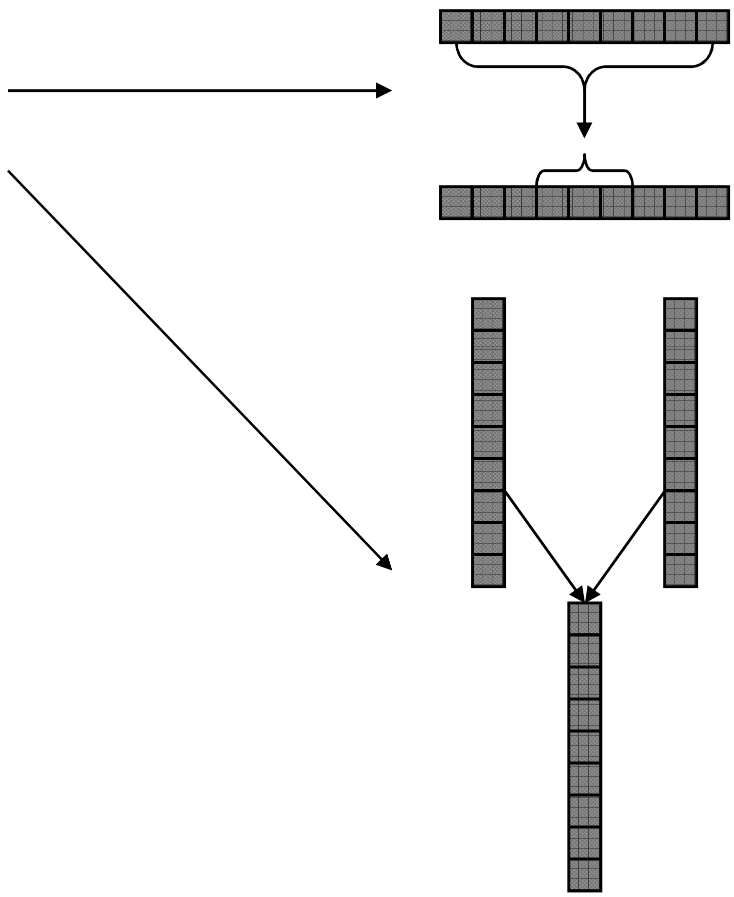
- Motivation
- Data stream management systems
  - GS tool
- Database management systems: the prequel
  - Daytona
- Streaming data warehouses: the sequel
  - Data Depot + Bistro

GS Tool

- GS tool is a fast, flexible data stream management system
  - High performance at speeds up to OC768 (2 x 40 Gbits/sec)
  - GSQL queries support SQL-like functionality
- Monitoring platform of choice for AT&T networks
- Developed at AT&T Labs-Research
  - Collaboration between database and networking research

GS Tool: GSQL Queries

- GSQL queries support
  - Filtering, aggregation
  - Merges, joins
  - Tumbling windows
- Arbitrary code support
  - UDFs (e.g., LPM), UDAFs
- GSQL query paradigm
  - Streams-in, stream-out
  - Permits composability



Example: TCP SYN Flood Detection

- Attack characteristic: exploits 3-way TCP handshake
- Attack detection: correlate SYN, ACK packets in TCP stream

```
define { query_name toomany_syn; }
select A.tb, (A.cnt – M.cnt)
outer_join from all_syn_count A,
   matched_syn_count M
where A.tb = M.tb

define { query_name all_syn_count; }
select S.tb, count(*) as cnt
from tcp_syn S
group by S.tb
```

```
define { query_name matched_syn_count; }
select S.tb, count(*) as cnt
from tcp_syn S, tcp_ack A
where S.sourceIP = A.destIP and
   S.destIP = A.sourceIP and
   S.sourcePort = A.destPort and
   S.destPort = A.sourcePort and
   S.tb = A.tb and
   (S.sequence_number+1) = A.ack_number
group by S.tb
```

GS Tool : Scalability

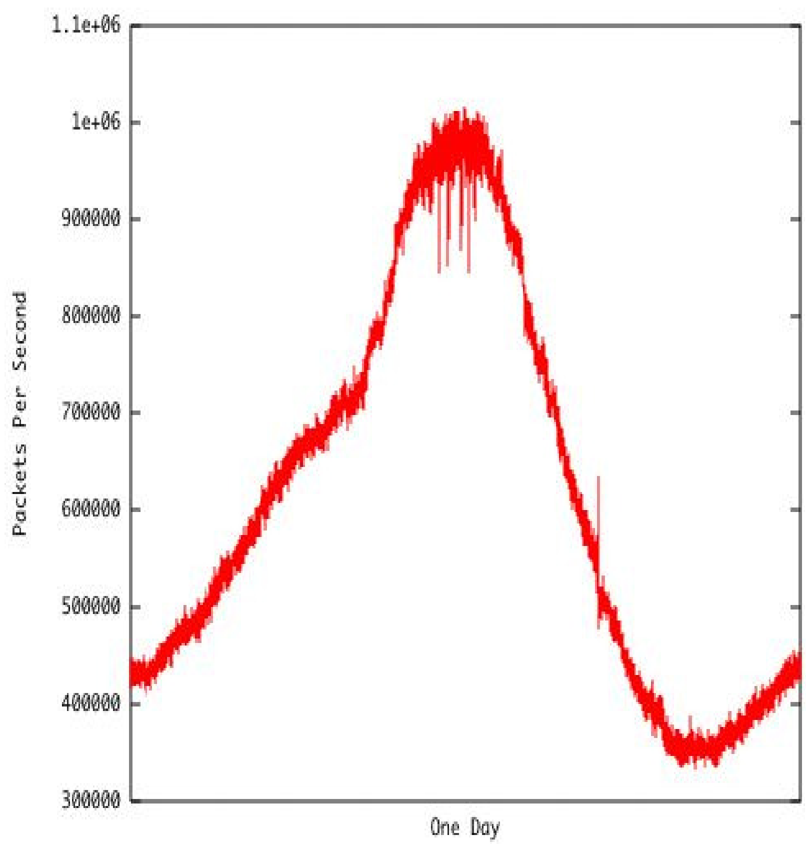
- GS tool is a fast, flexible data stream management system
  - High performance at speeds up to OC768 (2 x 40 Gbits/sec)

- Scalability mechanisms
  - Two-level architecture: query splitting, pre-aggregation
  - Distribution architecture: query-aware stream splitting
  - Unblocking: reduce data buffering
  - Sampling algorithms: meaningful load shedding

Performance of Example Deployment

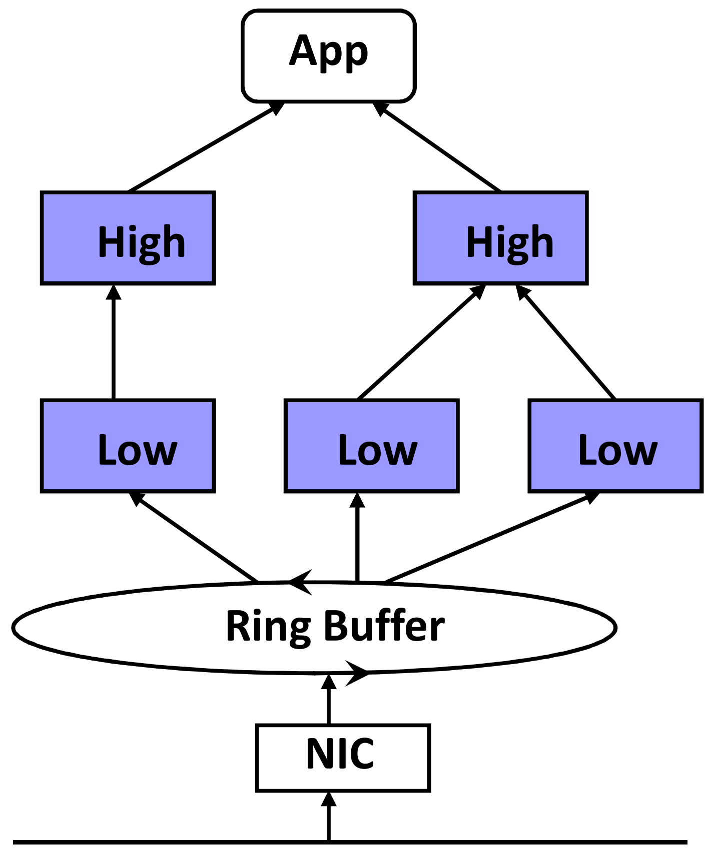
- GS tool is configured to track
  - IP addresses
  - Latency and loss statistics
  - ICMP unreachable messages

- GS tool monitors at peak times
  - 90000 users
  - > 1 million packets/second
  - > 1.4 Gbit/sec



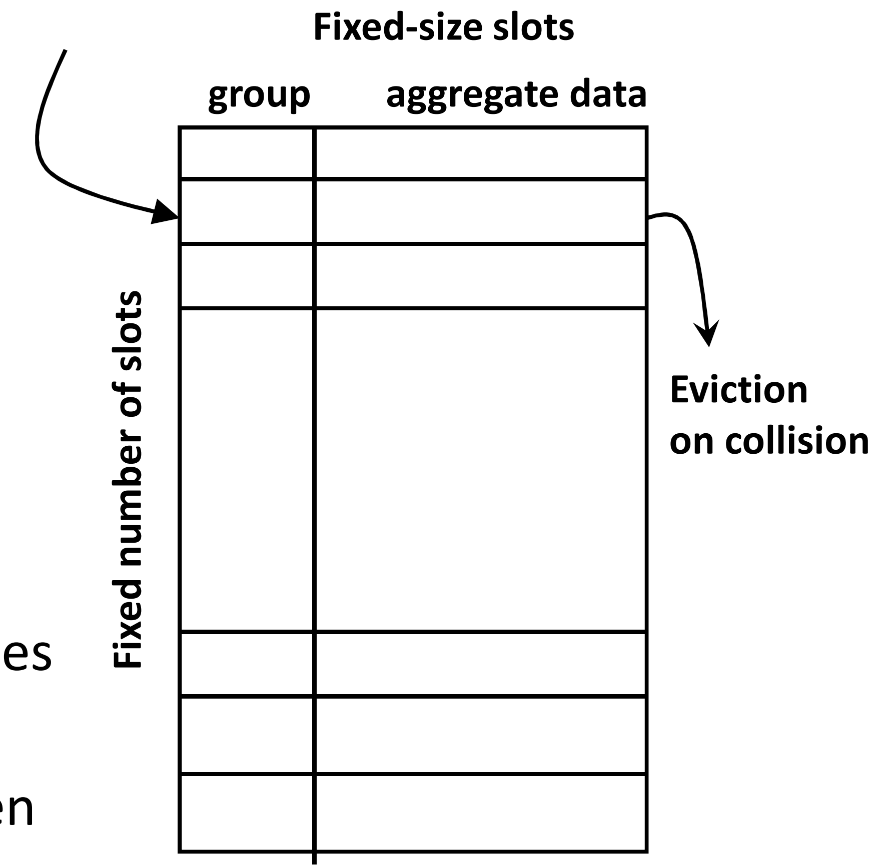
GS Tool: Two-Level Architecture

- Low-level queries perform fast selection, aggregation
  - Significant data reduction
  - Temporal clustering in IP data
- High-level queries complete complex aggregation



GS Tool: Low-Level Aggregation

- Fixed number of group slots, fixed size slot for each group
  - No malloc at run-time
- Direct-mapped hashing
- Optimizations
  - Limited hash chaining reduces eviction rate
  - Slow eviction of groups when epoch changes



GS Tool: Query Splitting

```
define { query_name smtp; }
select tb, destIP, sum(len)
from TCP
where protocol = 6 and
    destPort = 25
group by time/60 as tb, destIP
having count(*) > 1
```

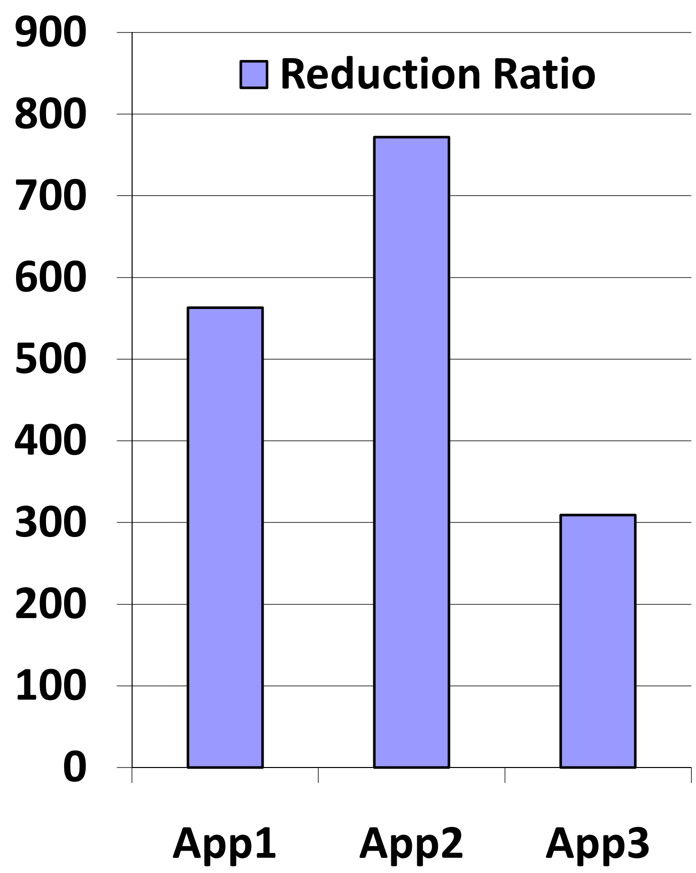
```
select tb, destIP, sum(sumLen)
from SubQ
group by tb, destIP
having sum(cnt) > 1
```

```
define { query_name SubQ; }
select tb, destIP, sum(len) as
   sumLen, count(*) as cnt
from TCP
where protocol = 6 and
    destPort = 25
group by time/60 as tb, destIP
```

GS Tool: Data Reduction Rates

- Significant data reduction through low-level aggregation

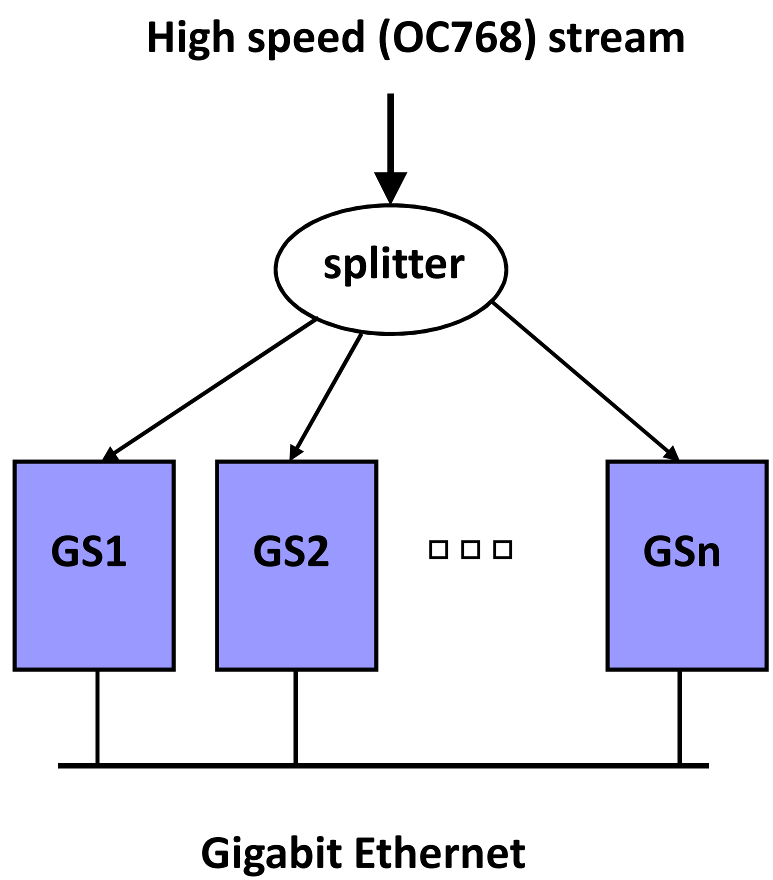
- Typical applications
  - App1: example deployment
  - App2: detailed accounting in 1 minute granularity
  - App3: detailed accounting, P2P detection and TCP throughput monitoring



at&t

Distributed GS Tool

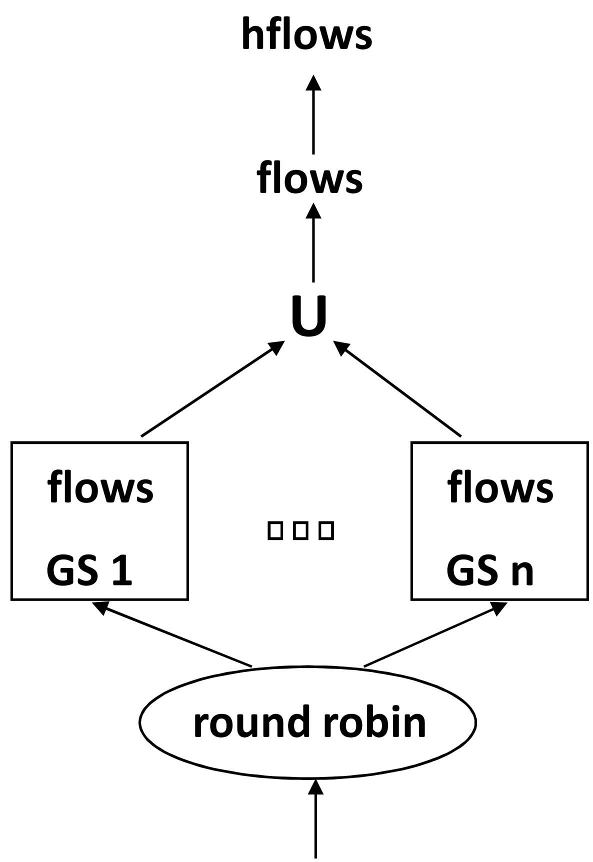
- Problem: OC768 monitoring needs more than one CPU
  - 2x40 Gb/sec = 16M pkts/sec
  - Need to debug splitter, router
- Solution: split data stream, process query, recombine partitioned query results
- For linear scaling, splitting needs to be query-aware



GS Tool : Query-Unaware Stream Splitting

<u>define</u> { query_name flows; }
<u>select</u> tb, srcIP, destIP, count(*)
<u>from</u> TCP
<u>group by</u> time/60 as tb, srcIP,
    destIP

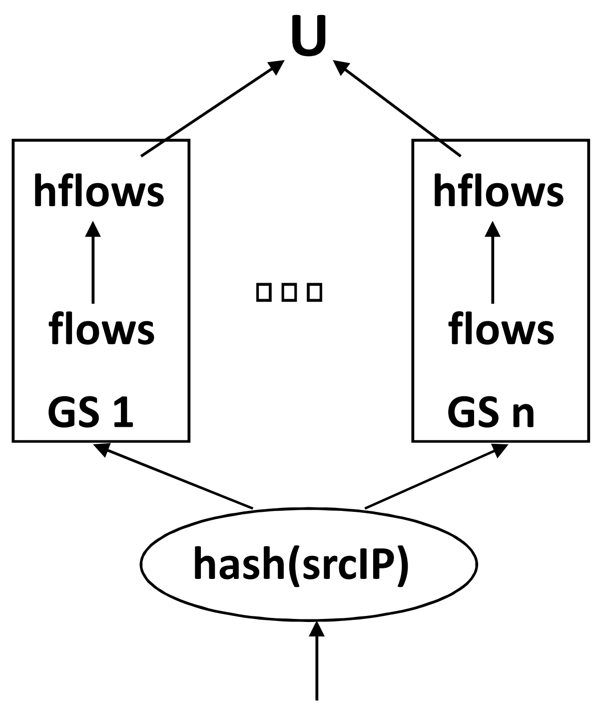
<u>define</u> { query_name hflows; }
<u>select</u> tb, srcIP, max(cnt)
<u>from</u> flows
<u>group by</u> tb, srcIP



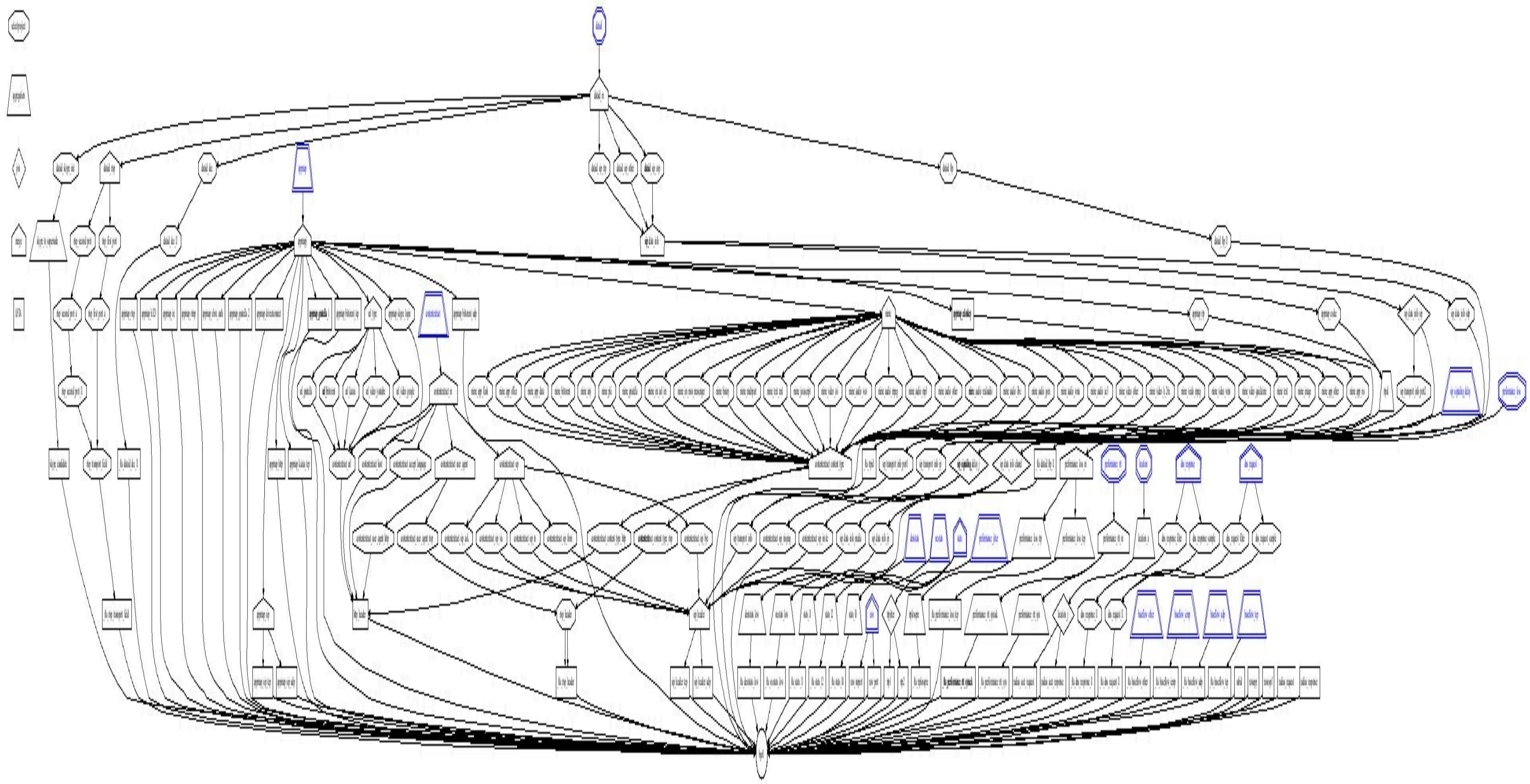
GS Tool : Query-Aware Stream Splitting

<u>define</u> { query_name flows; }
<u>select</u> tb, srcIP, destIP, count(*)
<u>from</u> TCP
<u>group by</u> time/60 as tb, srcIP,
destIP

<u>define</u> { query_name hflows; }
<u>select</u> tb, srcIP, max(cnt)
<u>from</u> flows
<u>group by</u> tb, srcIP



Query-Aware Stream Splitting at Scale



Distributed GS Tool: First Prototype

Outline

- Motivation
- Data stream management systems
  - GS tool
- Database management systems: the prequel
  - Daytona
- Streaming data warehouses: the sequel
  - Data Depot + Bistro

Daytona

- Daytona is a highly scalable and robust data management system
  - Organizes and stores large amounts of data and indices on disk
  - Enables concise, high-level expression of sophisticated queries
  - Provides answers to sophisticated queries quickly
  - Manages data in a concurrent, crash-proof environment
- Data management system of choice for AT&T's largest databases
  - Production quality system since 1989
- Developed at AT&T Labs-Research

Daytona: Cymbal Queries

- High-level, multi-paradigm programming language
  - Full capability of SQL data manipulation language
  - First-order symbolic logic (including generalized transitive closure)
  - Set theory (comprehensions)

- Sophisticated bulk data structures
  - Tables with set- and list-valued attributes on disk
  - Scalar- and tuple-valued multi-dimensional associative arrays
  - Boxes (in-memory collections) with indexing and sorting

- Synergy of complex processing and data access

Daytona: Scalability

- August 2010
  - ~ 1PB norm. data volume
  - 1.25T records in largest table
- Scalability mechanisms
  - Compilation architecture
  - Horizontal partitioning
  - Data compression
  - SPMD parallelization

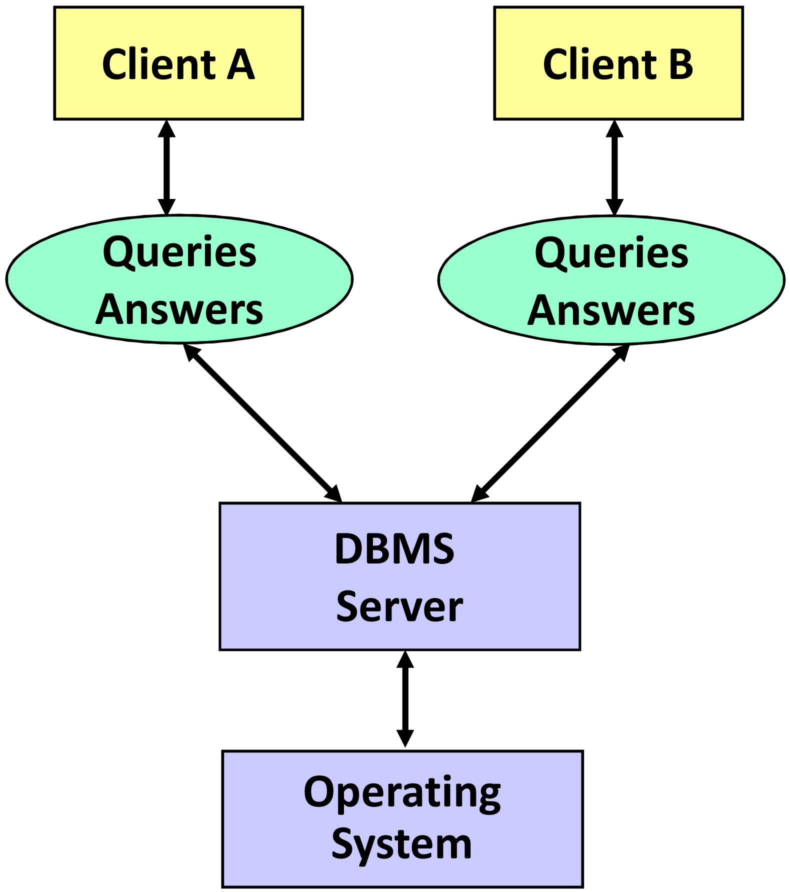
Norm. Data Volume, Unix, DW

Company/Organization	Norm. Data Volume (GB)	DBMS	Platform	Architecture	DBMS Vendor	System Vendor	Storage Vendor
AT&T	330,644	Daytona	UNIX	Federated/SMP	AT&T	HP	HP
AT&T	93,468	Daytona	UNIX	Federated/SMP	AT&T	Sun	Sun
Nielsen Media Research	17,969	Sybase IQ	UNIX	Centralized/SMP	Sybase	Sun	EMC
Yahoo!	17,014	Oracle	UNIX	Centralized/SMP	Oracle	Fujitsu Siemens	EMC
UBS AG	14,177	Oracle	UNIX	Centralized/SMP	Oracle	Sun	EMC
China Telecom Corporation Co.,Ltd. GuangZhou Research Institute	13,241	Sybase IQ	UNIX	Centralized/SMP	Sybase	Sun	Sun
Reliance Infocomm Ltd	11,500	Oracle	UNIX	Centralized/SMP	Oracle	Sun	EMC
Cellcom	10,345	Oracle RAC	UNIX	Centralized/Cluster	Oracle	HP	EMC
Turkcell	9,504	Oracle	UNIX	Centralized/SMP	Oracle	Sun	Hitachi
JPMorganChase	8,875	DB2	UNIX	Centralized/MPP	IBM	IBM	IBM

at&t

Typical DBMS Server Architecture

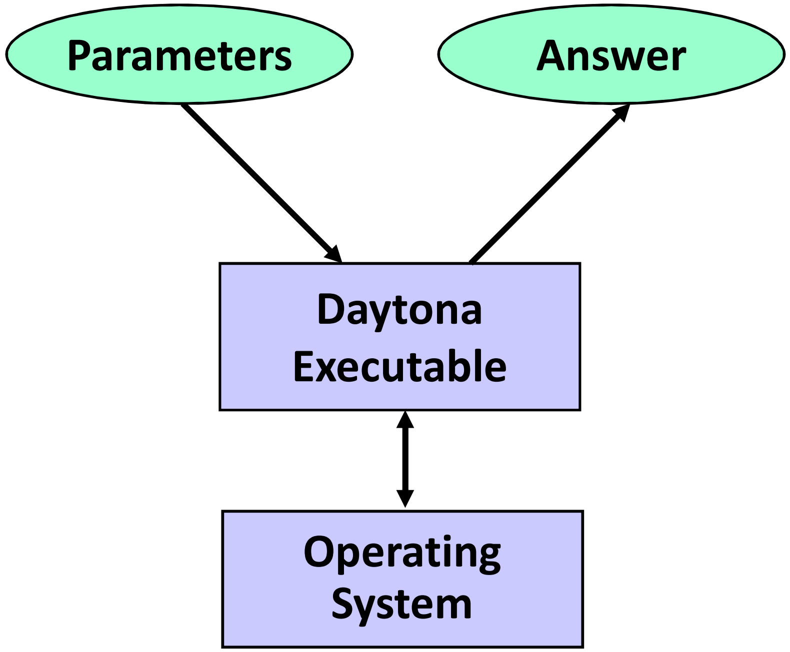
- DBMS server process provides
  - File system
  - Networking
  - Threads
  - Scheduling
  - Memory management
  - Caching
  - …
  - Query processing



Daytona: Compilation Architecture for Speed

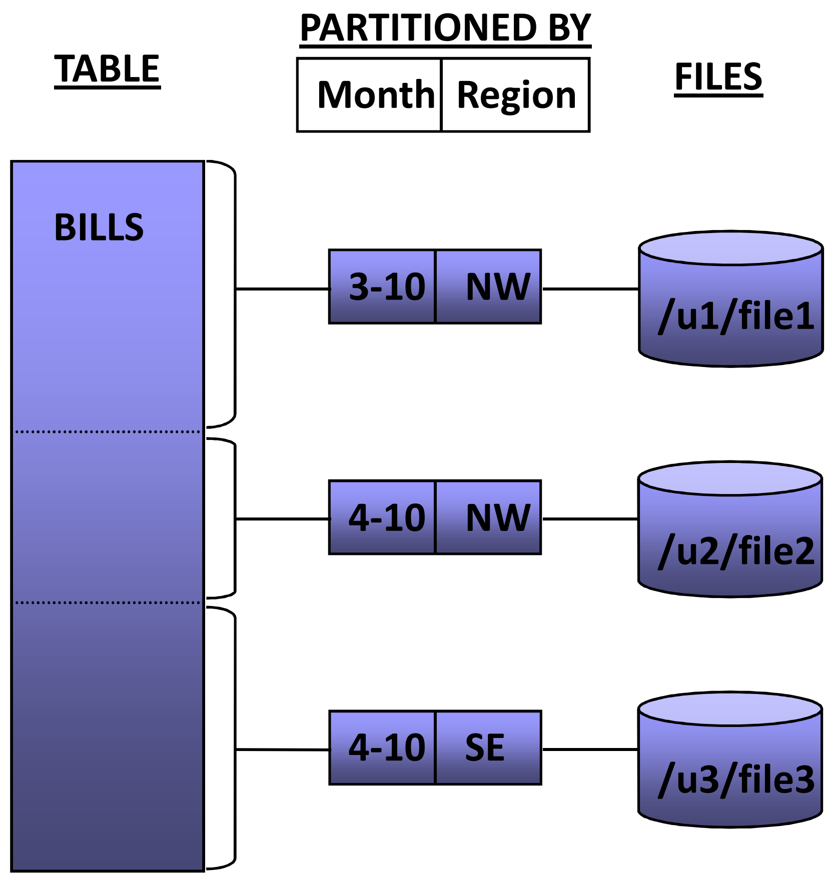
- Highly customized compilation
  - SQL → Cymbal → C
  - Enables using shared libraries

- No DBMS server processes
  - Query executable uses OS



Daytona: Horizontal Partitioning for Capacity

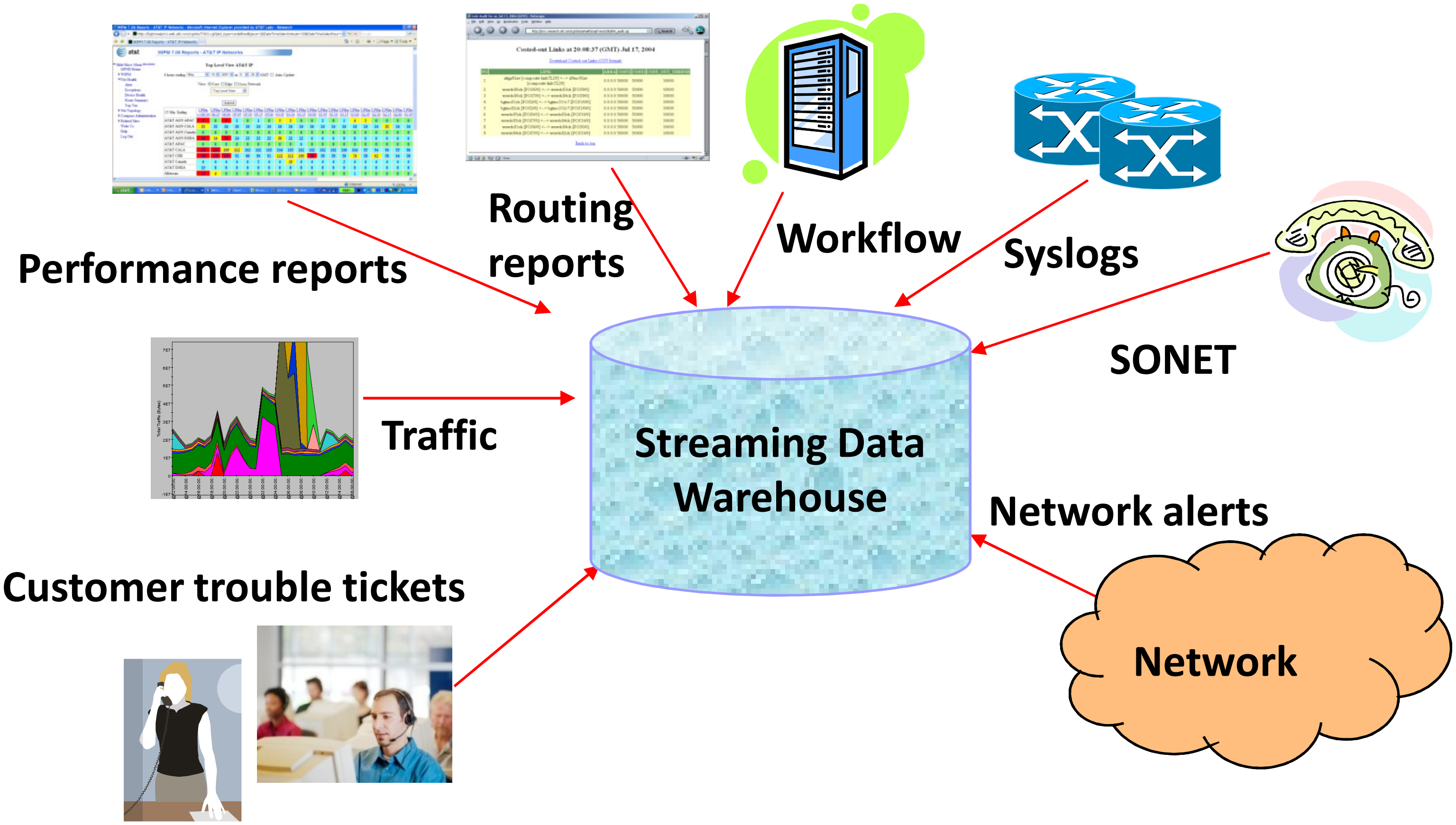
- Unbounded data storage
  - 1.25T records in 135K files
- Fast bulk data adds/deletes
  - Can maintain windows
- Disk load balancing
- Serves as another indexing level
  - Partition directory



Outline

- Motivation
- Data stream management systems
  - GS tool
- Database management systems: the prequel
  - Daytona
- Streaming data warehouses: the sequel
  - Data Depot + Bistro

What do you do with all the Data?



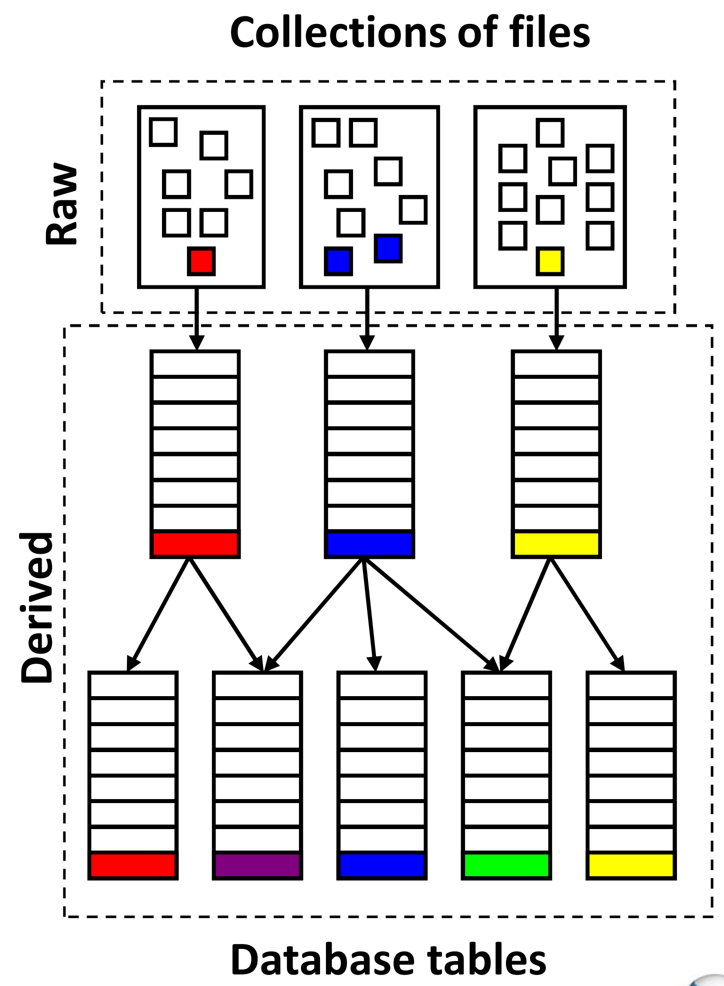
What is a Streaming Data Warehouse?

- Conventional data warehouse
  - Provides long term data storage and extensive analytics facilities
  - Supports deeply nested levels of materialized views
  - Updates to tables and materialized views performed in large batch

- Streaming data warehouse
  - Like a conventional data warehouse, but supports continuous, real time update of tables and materialized views
  - Like a DSMS, but provides long term data storage and supports deeply nested levels of materialized views

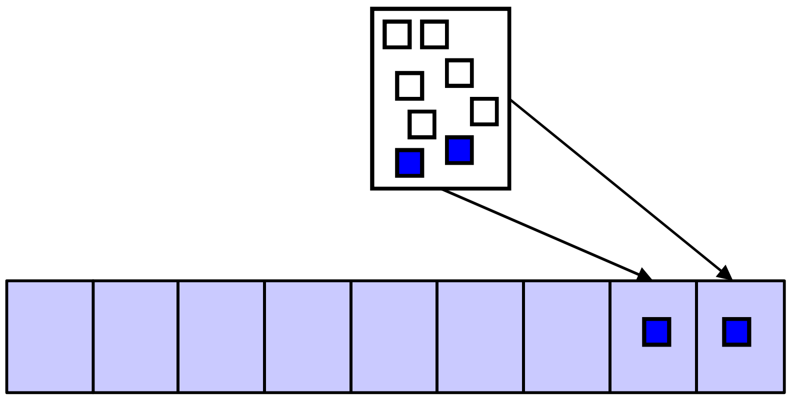
Data Depot

- Simplify stream warehousing
  - Manage Daytona warehouses
- Cascaded materialized views
  - Time-based partitioning
  - Real-time + historical tables
- View update propagation
  - Update scheduling
- Concurrency control



Data Depot: Time-Based Partitioning

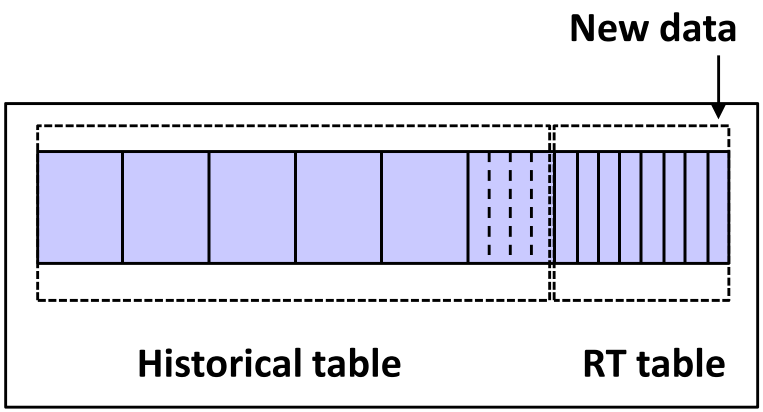
- Time-based partitioning of raw tables, materialized views
  - Daytona horizontal partitions
  - Roll in at leading edge
  - Roll off at trailing edge



- Set partition size to be the real-time update increment
  - Avoid index rebuilding

Data Depot: Real-Time + Historical Tables

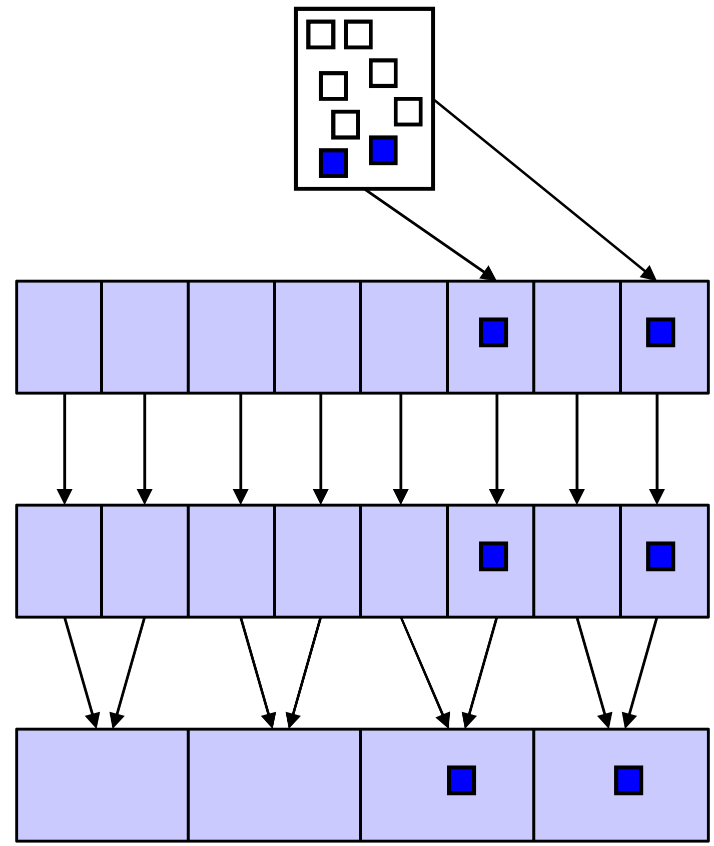
- Issue: RT and historical tables are optimized differently
  - Small partitions for RT tables
  - Large partitions for historical
- Solution: newest part of table is RT, oldest part is historical
  - Combined RT+historical table
  - Transparent query access to data in combined table



Combined RT+historical table

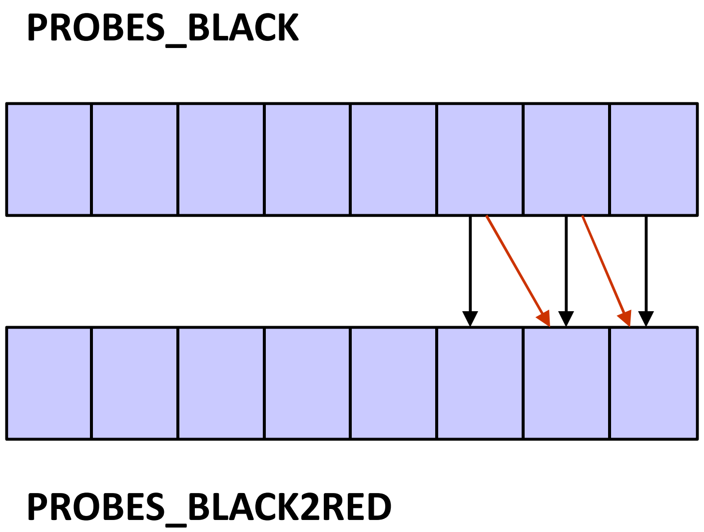
Data Depot: Update Propagation

- Update propagation through partition dependencies
- Overload situation common
  - Need update scheduling
- Basic algorithm
  - Determine source partitions of a derived partition
  - Recompute derived partition if a source partition changes



Data Depot: Partition Dependencies Example

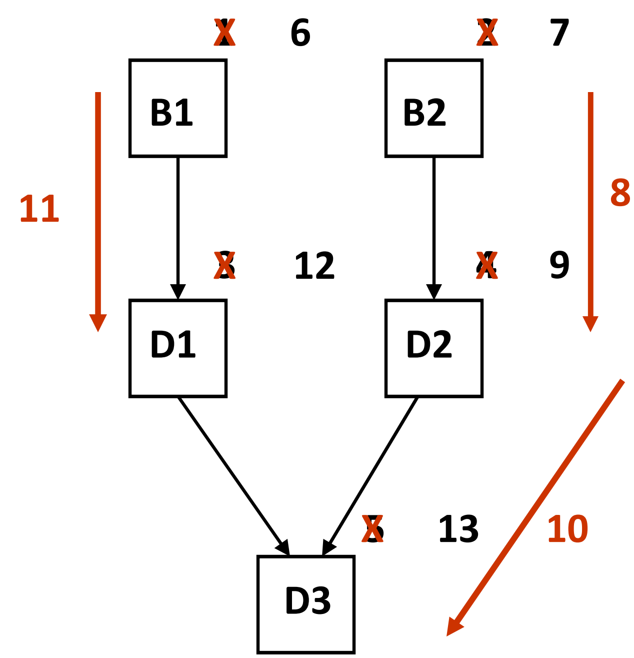
```
CREATE TABLE PROBES_BLACK2RED AS
SELECT TimeStamp, Source, Destination
FROM PROBES_BLACK P
WHERE NOT EXISTS
   (SELECT B.Timestamp, B.Source,
        B.Destination
    FROM PROBES_BLACK B
    WHERE B.Source = P.Source
    AND B.Destination = P.Destination
    AND B.Timestamp = P.Timestamp – 1)
```



Data Depot: Update Scheduling

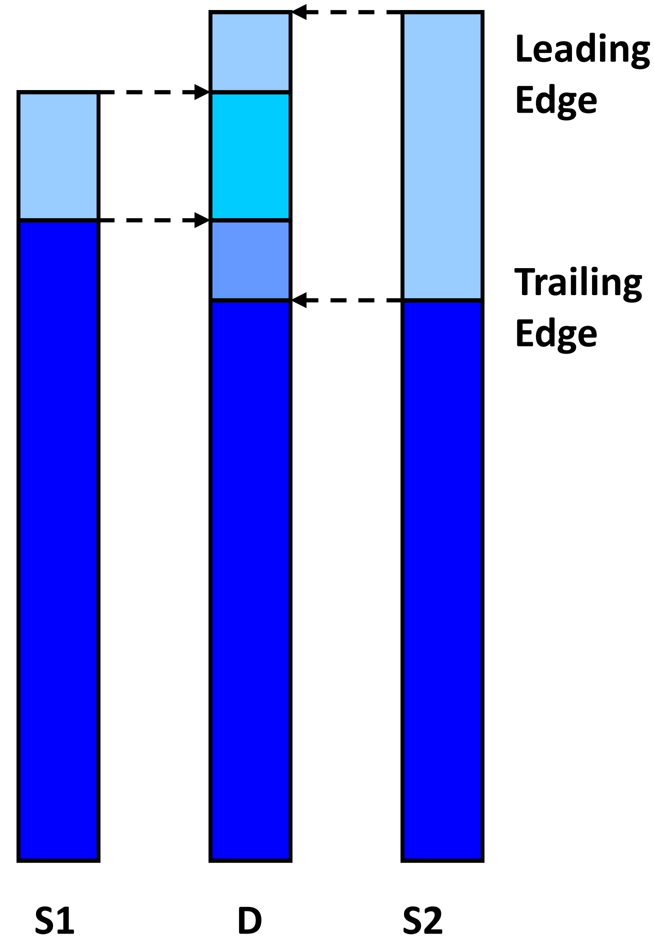
- Make-style protocol is wrong
  - Track last-update timestamp
  - Update if source is newer

- Vector-timestamp style model
  - Record all dependencies
  - Eventual consistency easy
  - Mutual consistency hard
  - Trailing edge consistency is the best compromise



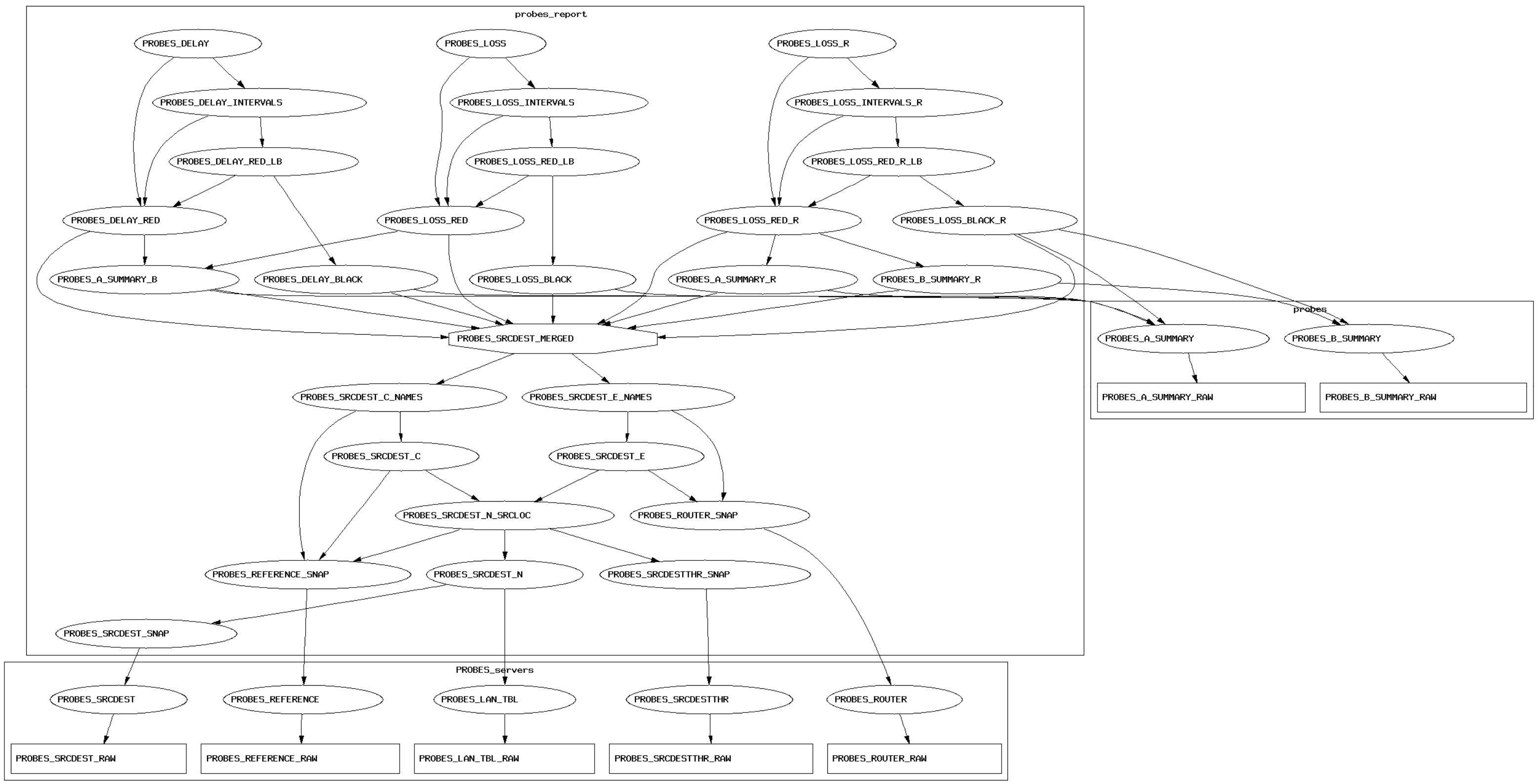
Data Depot: Consistency

- Provides eventual consistency
  - All data depends on raw data
  - Convergence after all updates get propagated
- What about before eventually
  - Leading edge consistency: use all the data that is available
  - Trailing edge consistency: use only stable data



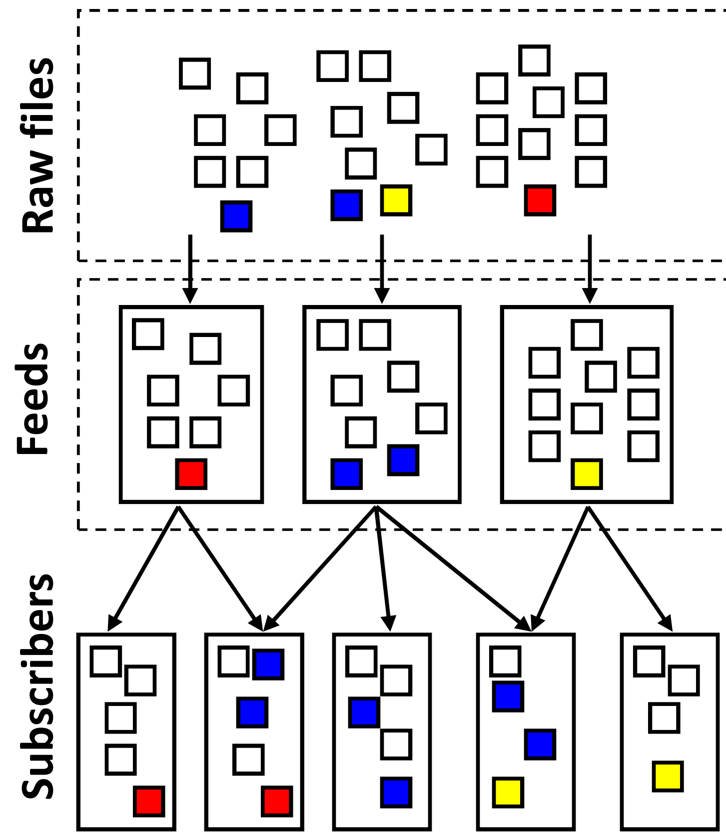
Data Depot: Cascaded Views Example

- Application to measure loss and delay intervals in real time
  - Raw data every 15 min, 97% of update propagation within 5 min



Bistro

- Weakest link: when are base tables ready for a refresh?
  - Every 5 min → 2.5 min delay
- Bistro: data feed manager
  - Pushes raw data from sources to clients with minimal delay
  - Provides delivery triggers
  - Monitors quality, availability
  - Manages resources
- Russian быстро means quickly



at&t

Data Feed Management BB (Before Bistro)

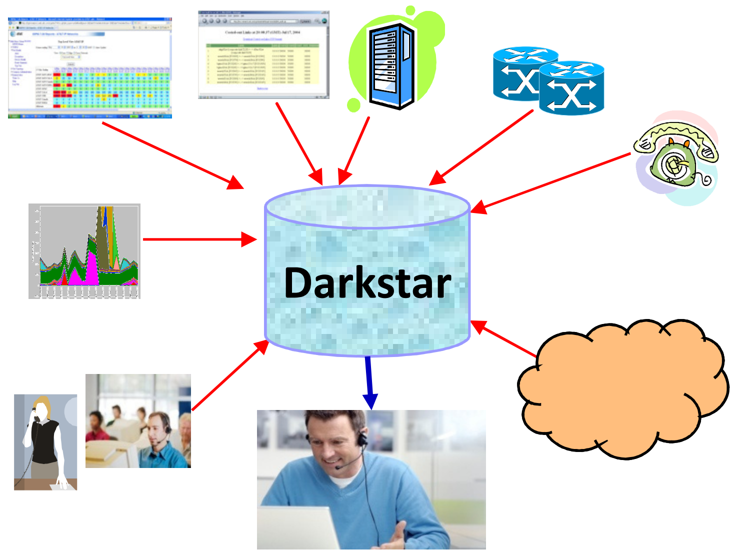
- Data feed management has largely been an ad hoc activity
  - Shell scripts invoked using cron, heavy usage of rsync and find
- Cron jobs
  - Propagation delays, can step on previous unfinished scripts
  - No prioritized resource management
- Rsync
  - Lack of notification on client side (no real-time triggers)
  - Clients must keep identical directory structure and time window
  - No systemic performance and quality monitoring

Bistro: Scalability, Real-Time Features

- Maintain database of transferred raw files
  - Avoid repeated scans of millions of files (e.g., find)
  - Different clients can keep differently sized time windows of data
- Intelligent transfer scheduling
  - Deal with long periods of client unavailability
  - Parallel data transfer trying to maximize the locality of file accesses
- Real-time triggers
  - Notify clients about file delivery, batch notifications

Real-Time Darkstar

- Enabled by Daytona, Data Depot, Bistro
- As of June 2010
  - 183 data feeds, 607 tables
  - 30.9 TB data
- Real-time analysis applications
  - PathMiner: analyze problems on path between 2 endpoints
  - NICE: mine for root causes of network problems



Summary

- What have we accomplished?
  - Built some cool systems: Daytona, Data Depot, GS tool, Bistro
  - Enabled very high-speed streaming analysis using GS tool
  - End-to-end solution for SDW using Daytona, Data Depot, Bistro

- What's next?
  - Existing systems continue to evolve in scale and functionality
  - New systems being built (e.g., Data Auditor for monitoring quality)
  - Integrated DSMS + SDW: challenging research problems

Parting Thoughts

- Enabling real time data analysis is critical
  - Once data is collected, it should be available for analysis right away
  - Need to support artifacts of data analysis in the database
- Need to have an end-to-end solution for RT data management
  - Our focus has been on managing data once it is in the database
  - We should do research on the entire pipeline from data generation to data management to data usage
- Exciting journey: should keep us busy for quite a while!