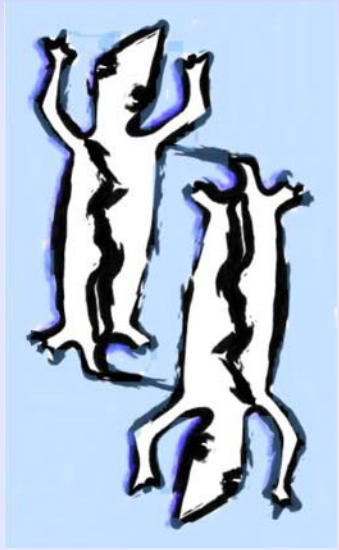


Towards a Data Streams Resource in Grid Services

Beth Plale
Computer Science Dept.
Indiana University

Outline

- Linked Earth Atmospheric Discovery (LEAD) project
- Characterization of data stream systems and relevance in Grid Services
- dQUOB – prototype system for remote query evaluation
- Optimization to join handling for asynchronous streams

http://lead.ou.edu

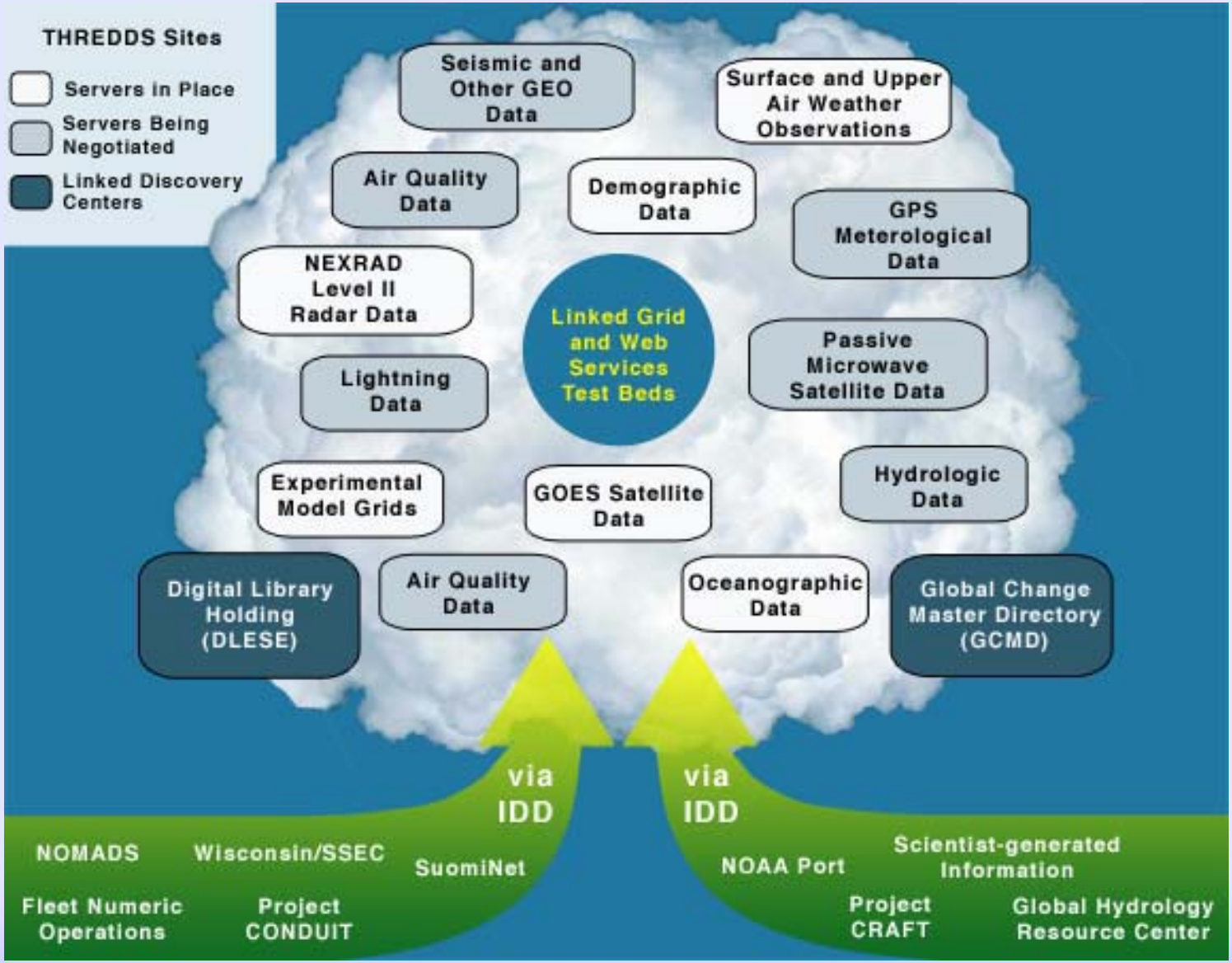
Motivation for LEAD

- Each year, mesoscale weather – floods, tornadoes, hail, strong winds, lightning, and winter storms – causes hundreds of deaths, routinely disrupts transportation and commerce, and results in annual economic losses > $13B.

The Roadblock

- The study of events responsible for these losses is stifled by rigid information technology frameworks that cannot accommodate the
  - its tremendous computational demands, which are among the greatest in all areas of science and engineering
  - its disparate, high volume data sets and streams; and
  - real time, on-demand, and dynamically-adaptive needs of mesoscale weather research;
- Some illustrative examples…

LEAD Data Cloud



The Results Analysis Problem

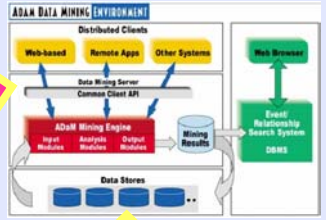
- Specifically,
  - Under what conditions (vertical profiles of horizontal wind, temperature, and humidity) do supercell storms cycle, i.e., produce multiple mesocyclones/tornadoes?
- Critical implications for forecasters
- Clear guidelines exist for distinguishing among types of thunderstorms (supercells, single cells, lines)
- Use numerical model to span desired parameter space
  - 20 vertical profiles of temperature
  - 20 vertical profiles of humidity
  - 100 vertical profiles of the horizontal wind
  - → Yields 4,000 simulations!
  - We have the computer time, but we don't have the ability to analyze the results!

How Will LEAD Help?

Q: When Do Supercell Storms Produce Multiple (Cyclic) Tornadoes?

Create Continuous Climatology From Observations

Data Cloud



Data Mining

Data Assimilation

Gridded Assimilated Data Sets

4000 Storm Simulations

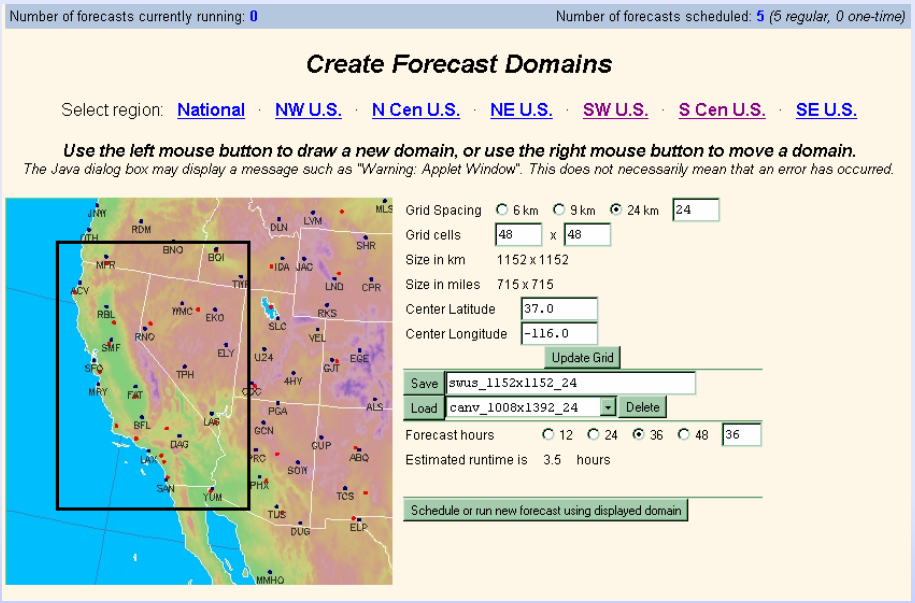
300 Tbytes

Data and Metadata

My LEAD Virtual Public Space

Forecasts on Demand

- Requires a human to determine domain location, other parameters (no automated response to the weather)
- Requires known computational resources (not grid-enabled)
- Receives no special computational priority (no service guarantees)



How Will LEAD Help?

Q: Why Do Some Severe Storms Produce Multiple (Cyclic) Tornadoes?

Create Climatology Based Upon All Observations

Data Assimilation

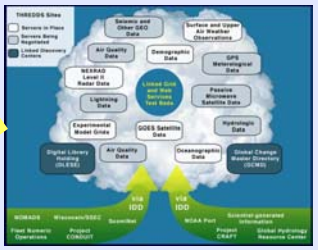
Gridded Assimilated Data Sets

500 Storm Simulations

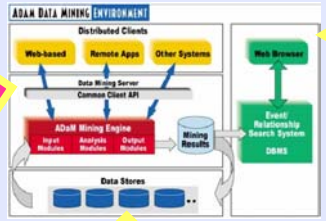
300 Tbytes

Data and Metadata

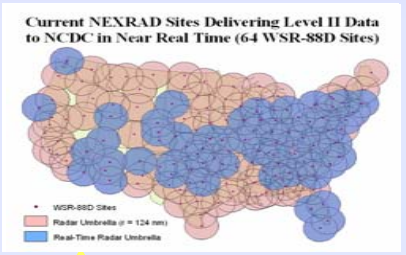
My LEAD Virtual Public Space



Data Cloud



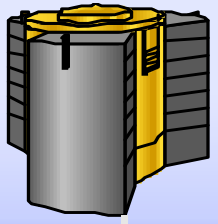
Data Mining



Streaming Radar Data

Real-Time WRF Runs on Grid Only When Environment is Primed and Storms are Present

On-Demand Resource Scheduling



How Will LEAD Help?

Q: Why Do Some Severe Storms Produce Multiple (Cyclic) Tornadoes?

Create Climatology Based Upon All Observations

Data Assimilation

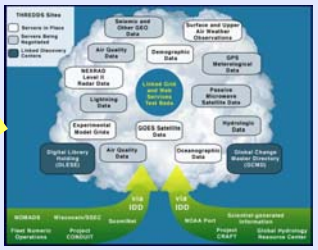
Gridded Assimilated Data Sets

500 Storm Simulations

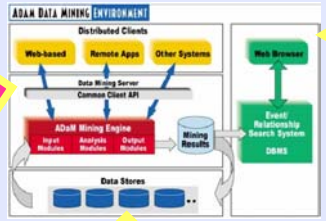
300 Tbytes

Data and Metadata

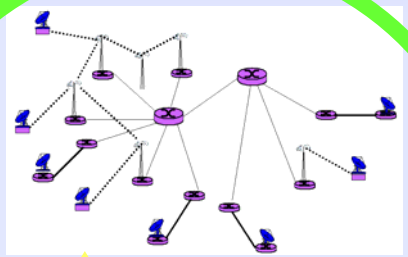
My LEAD Virtual Public Space



Data Cloud



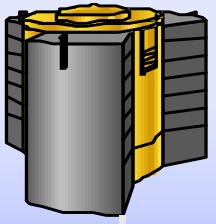
Data Mining



Real Time Control of Radars

Real-Time WRF Runs on Grid Only When Environment is Primed and Storms are Present

On-Demand Resource Scheduling

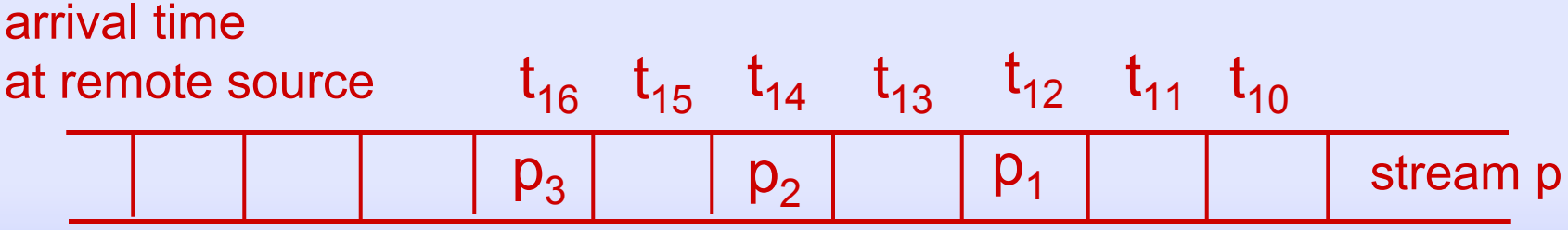


LEAD CS/IT Research

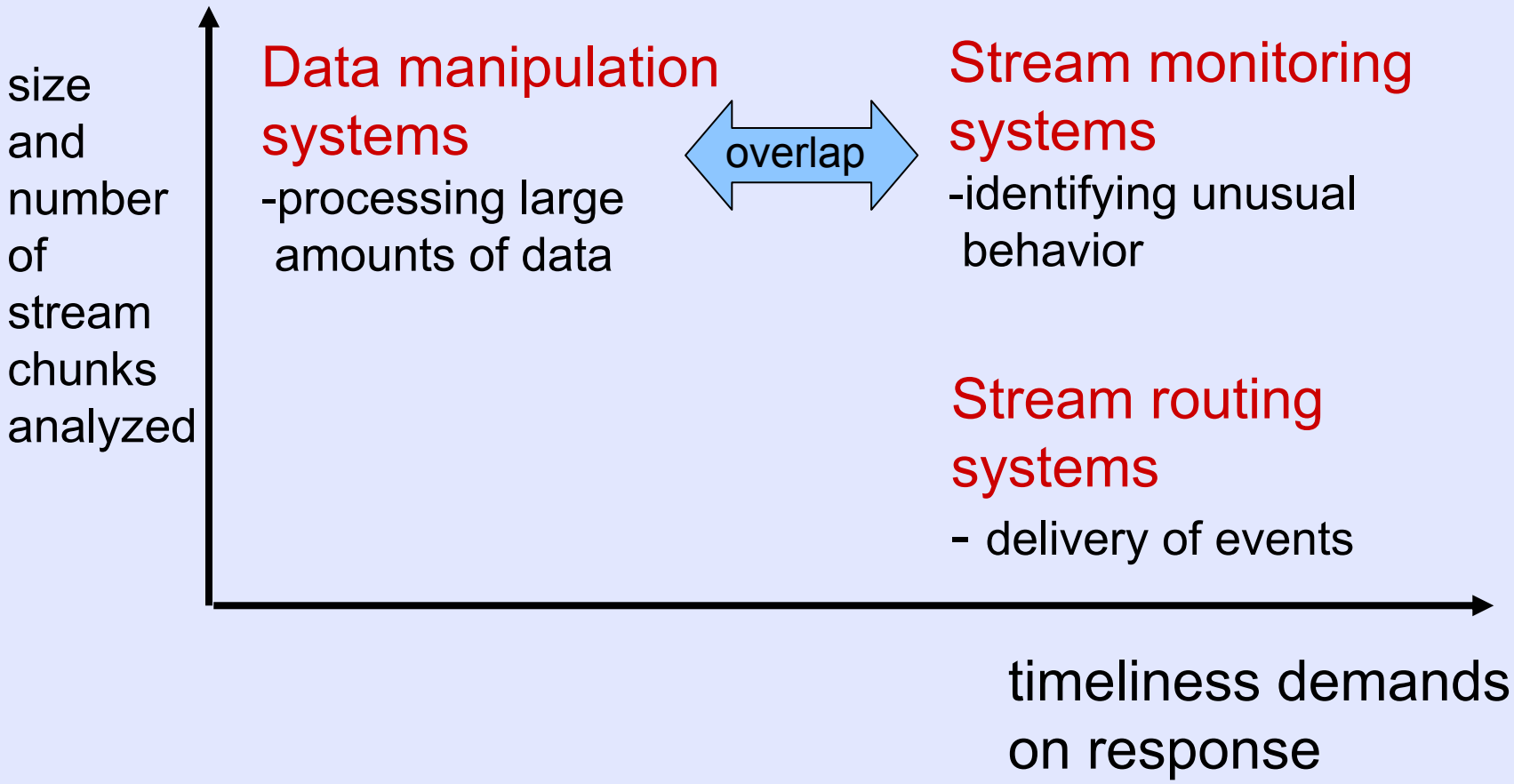
- *Workflow orchestration* – the construction and scheduling of execution task graphs with data sources drawn from real-time sensor streams and outputs
- *Data streaming* – to support robust, high bandwidth transmission of multi-sensor data.
- *Distributed monitoring and performance evaluation* -- to enable soft real-time performance guarantees by estimating resource behavior.
- *Data management* – for storage and cataloging of observational data, model output and results from data mining.
- *Data mining tools* – that detect faults, allow incremental processing (interrupt / resume), and estimate run time and memory requirements based on properties of the data (e.g., number of samples, dimensionality).
- *Semantic and data interchange technologies* – to enable use of heterogeneous data by diverse tools and applications.

Data streams:

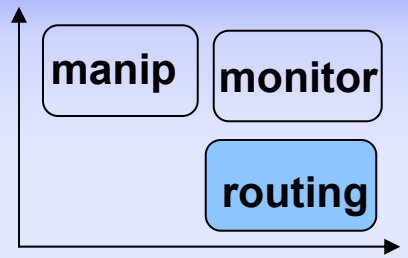
- Indefinite sequence of events, messages, tuples
- Often time marked
  - Generation time, that is, timestamp, and
  - Logical time
- Events continuously generated
  - pushed or pulled from providers to remote consumers
- Because sequence is indefinite, requests are long running, continuously executing



Types of Data Stream Systems



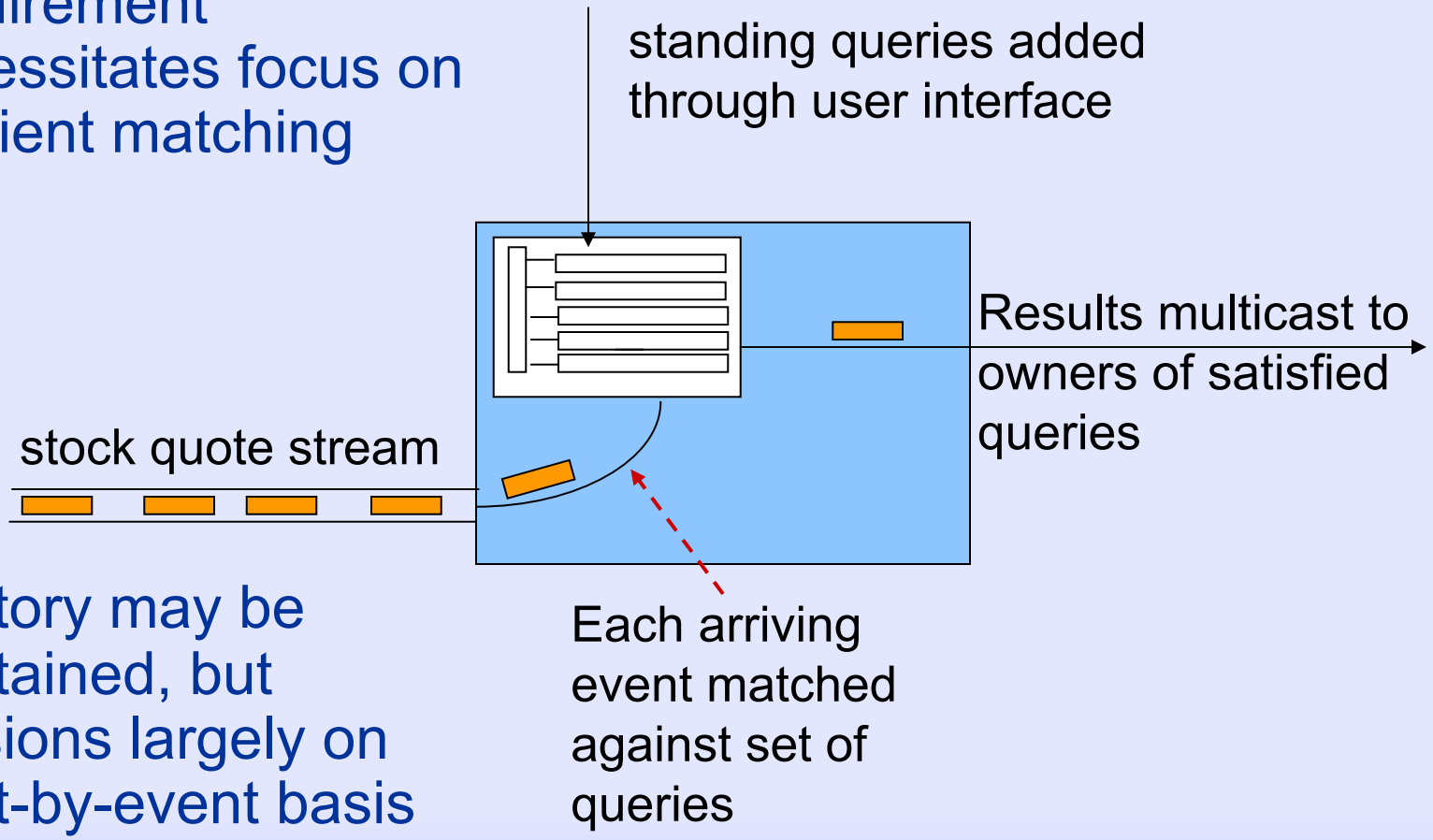
Stream Routing Systems



- Known by various names
  - Publish/subscribe, selective data dissemination
  - document filtering, message oriented middleware (MOM)
- Decisions made event-by-event
  - Set of queries (usually very large) managed over time duration, arriving event matched against set of queries.
- Projects
  - Xfilter (UMaryland), Xyleme (INRIA), XPushMachine (UWashington), NaradaBrokering(IndianaU), Bayou(XeroxParc)

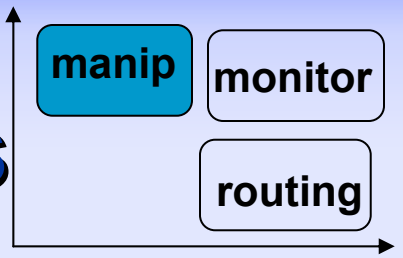
Stream Routing Systems Example

- Timeliness requirement necessitates focus on efficient matching



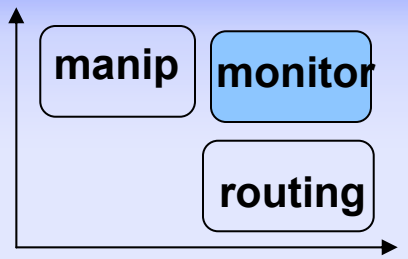
- History may be maintained, but decisions largely on event-by-event basis

Data Manipulation Systems



- Event streams subject to transformation, filtering, aggregation.
- Looser timeliness requirements on results
- Long running queries, often periodic (based on assumption of synchronous streams)
- Results in generation of new streams
- Projects:
  - Antarctic Monitoring(UNottingham), sensor network query layer (Cornell), dQUOB (IndianaU), STREAM (Stanford), Fjords (Berkeley), NiagraCQ (UWisconsin)

Stream Monitoring Systems

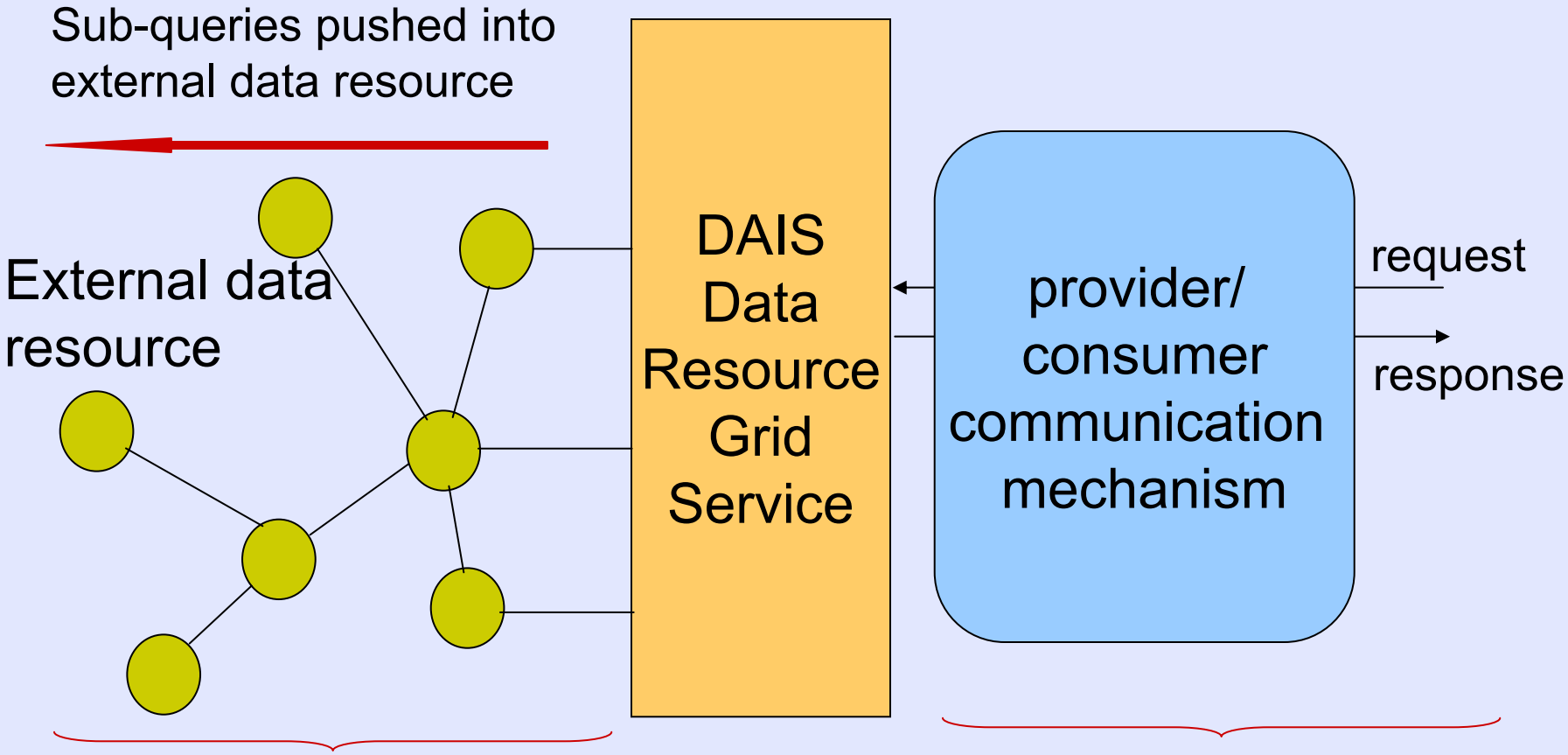


- Event-oriented (versus periodic)
- Less predictable, asynchronous streams
- Intent is to detect anomalous behavior
- Timeliness is critical, time markers key to decision making
- Result is notification message
- Projects:
  - R-GMA (EU DataGrid), dQUOB (IndianaU), Conquer (GeorgiaTech), Gigascope (AT&T), Fjords (Berkeley)

Significance of Distinction

- Different focus in offered service
  - Operator support (i.e., temporal operators in notification systems)
  - Scalability, performance
- Differently suited
  - GGF OGSI Grid Services model distinguishes between data management and data movement
  - GGF DAIS WG defines data resource as a Grid Service that manages an external data resource (usually database.)
  - Data manipulation and monitoring systems are an external data resource amenable to management by a DAIS Data Resource Grid Service.
    - Routing systems are a data movement mechanism

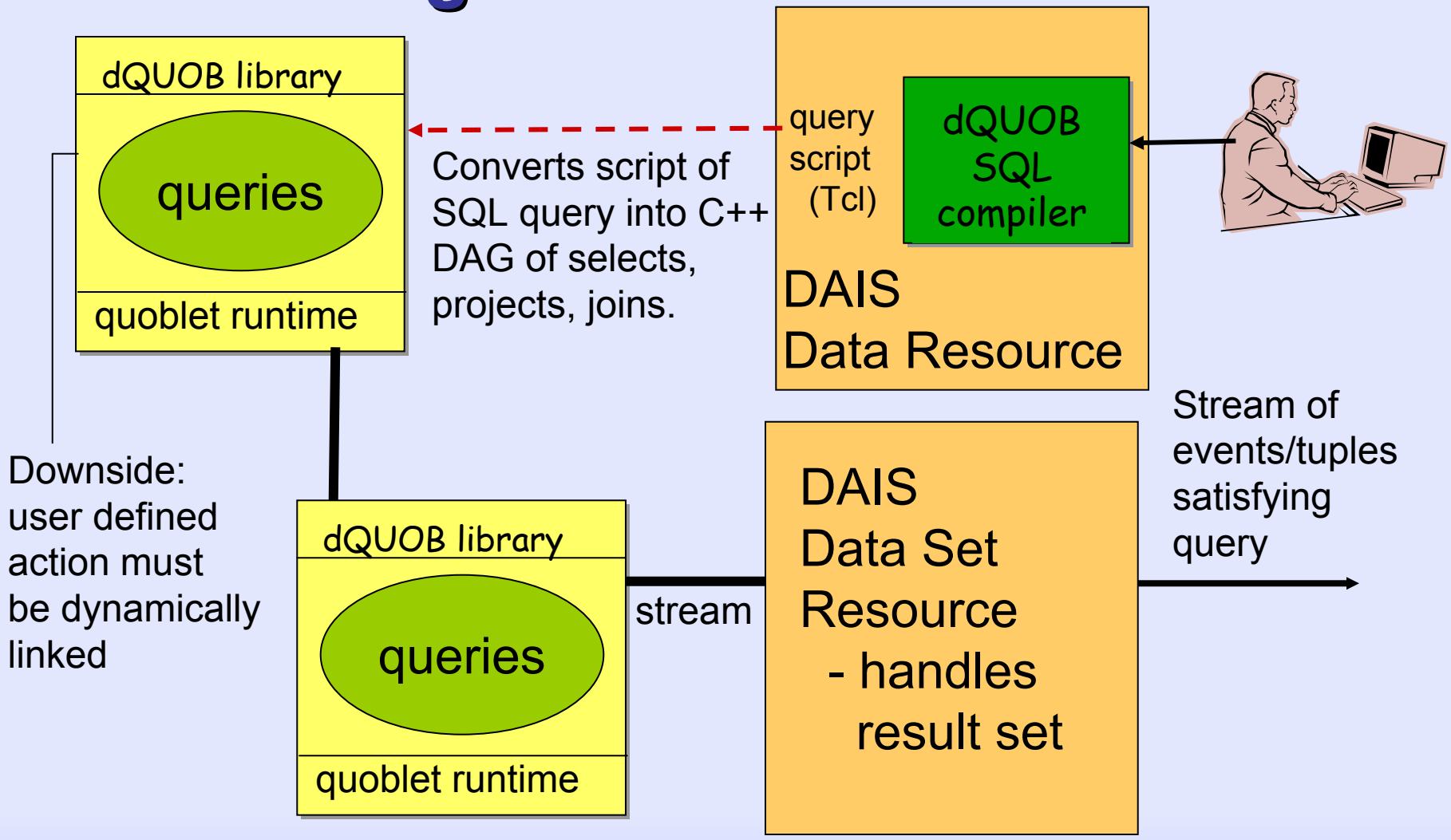
Applicability to Grid Services



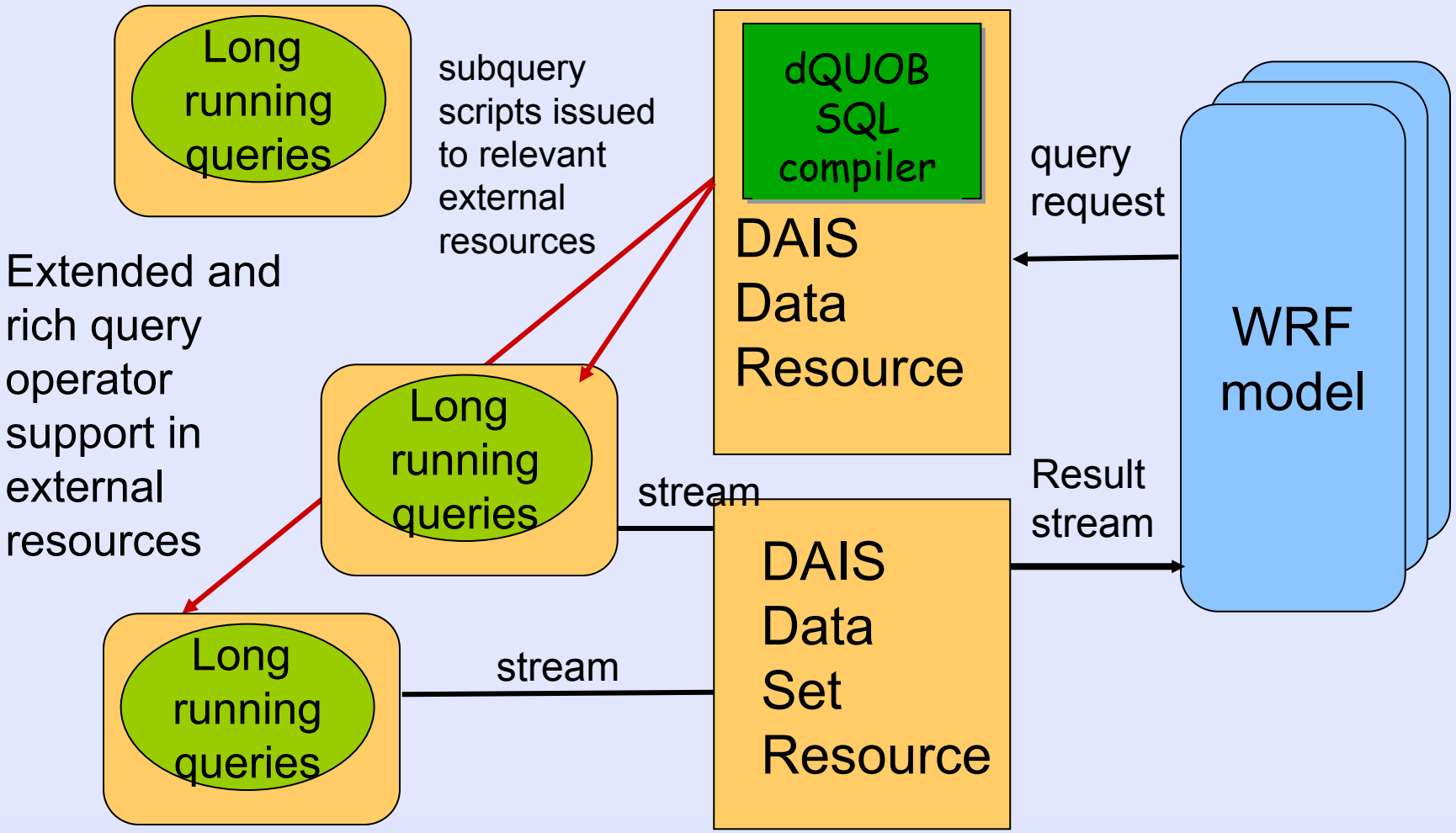
dQUOB: middleware for dynamic query placement at external data resource

- Relational database view of streams
  - Event == tuple, data stream == relation
- Query has rule-based syntax
  - Supports subset of SQL query
  - Coupled with user-defined functions (e.g., mathematical functions (FFT), compression).
- Time-based, two-way join
  - Two events satisfy join if they ‘happen at the same time’
  - Over logical time or timestamp

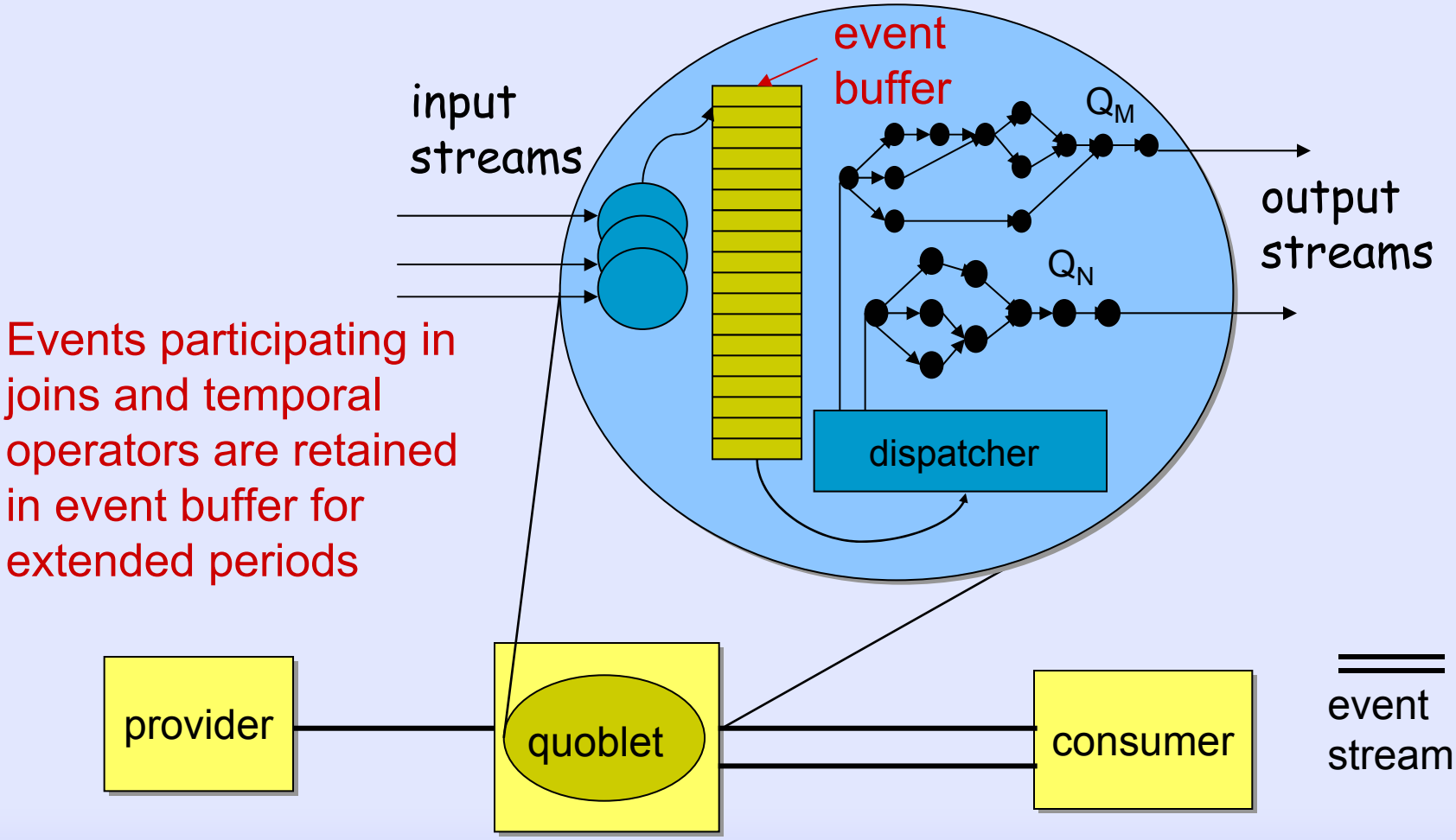
Architecture: pushing queries to remote agents



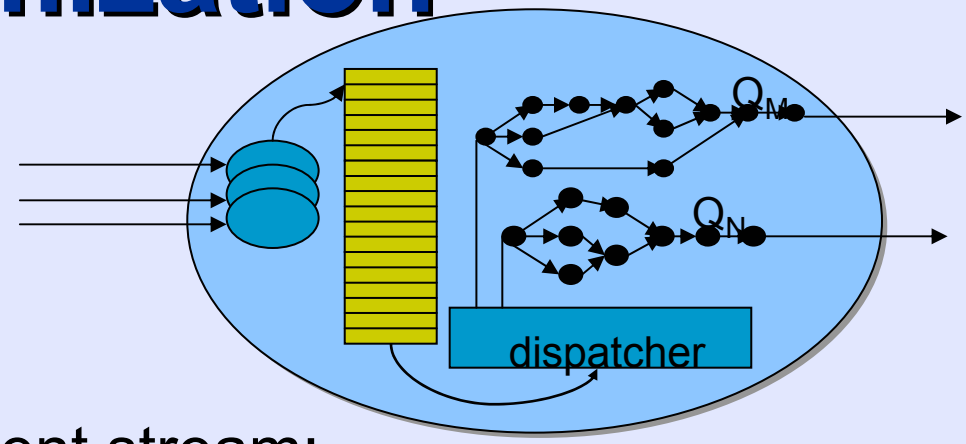
On-demand architecture for LEAD: WRF poses request for updates as query to DAIS



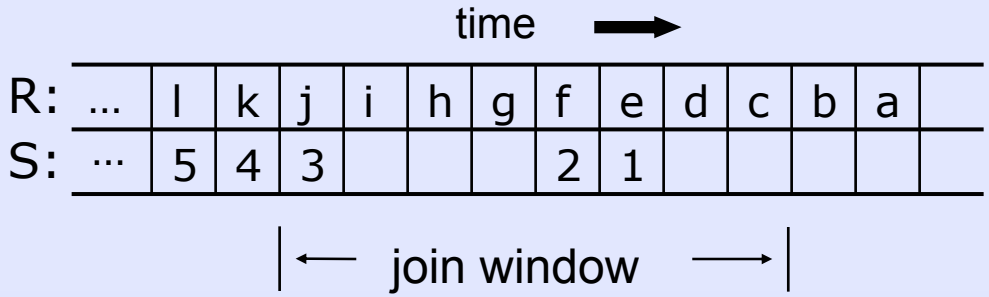
Quoblet architecture: importance of memory management



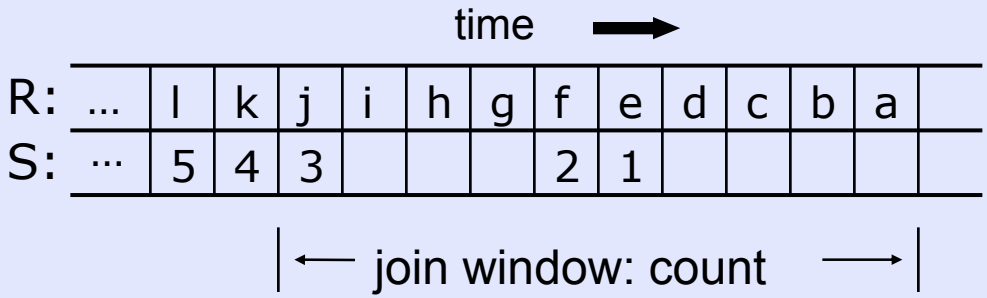
Memory Optimization



- Join window –
  - sliding window over event stream;
  - determines time interval of events over which a join is performed
  - First solution: join window based on integer count

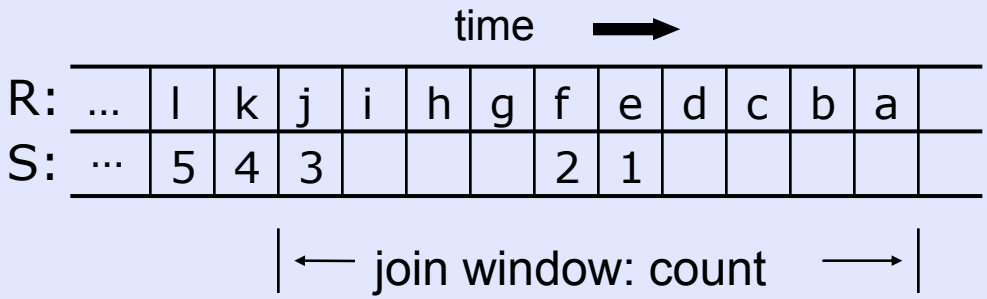


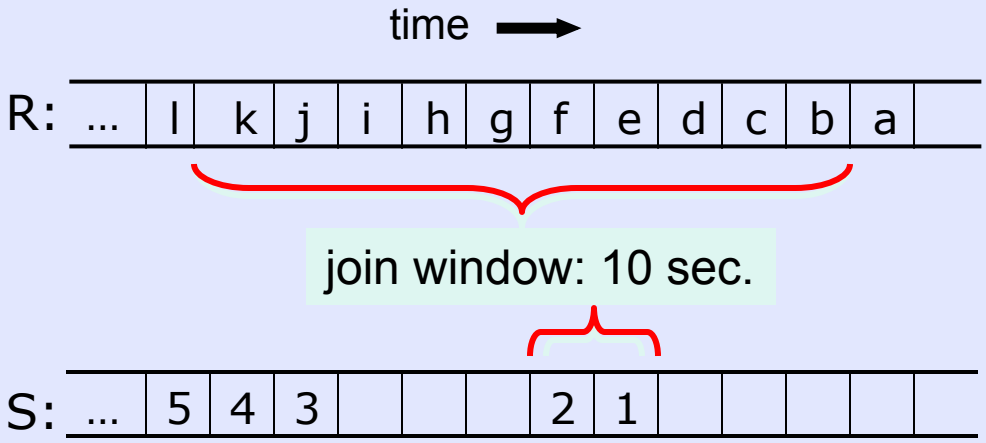
Problem exists with integer count when two streams are asynchronous (that is, one is slow and erratic)



Problem: not intuitive. Difficult for user to pick right join window size. Cost of error is great: too large, consumes memory; too small, increases false negatives

Approach to problem of asynchronous streams: express join window as interval of time





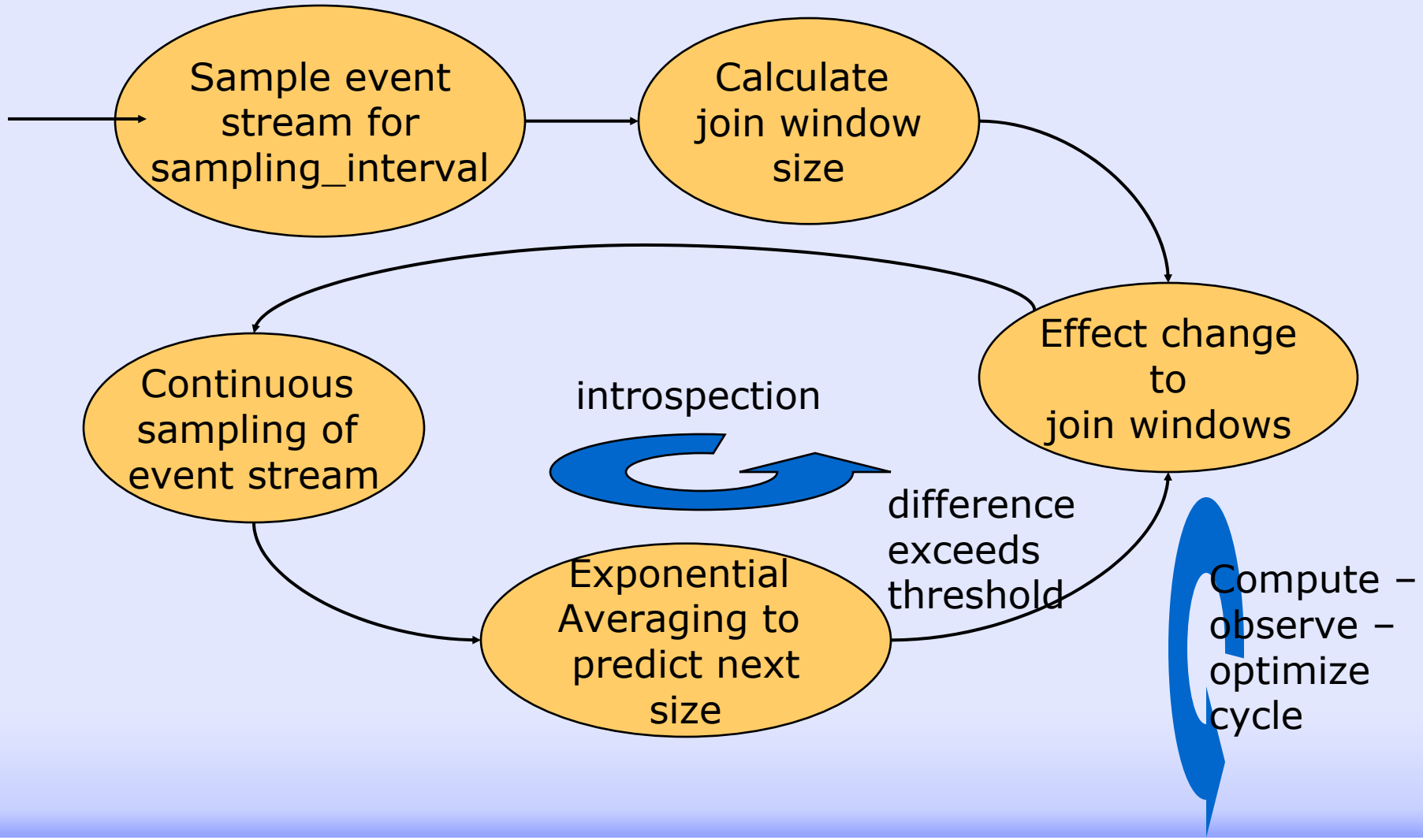
Step 1: user specifies interval in wallclock time. Why? Only interval known for certain at startup.

Step 2: sample during runtime to figure out stream rates. Map wallclock interval into timestamp interval

Step 3: adjust join window sizes

Step 4: use introspection technique to monitor and adapt to changes in event rates

Rate sensitive join window algorithm



Join Algorithm Pseudo Code

```
At_startup (sampling_interval:  integer) {
   for all i concurrently {
      sample event stream[i] for duration of sampling_interval;
      barrier();

      max_timestamp_interval = last_event[i].timestamp –
           first_event[i].timestamp;

      join_window_size[i] = (events_received[i] *
                  sampling_interval) / max_timestamp_interval;
   }
   effect_change[i];
}
```

Example

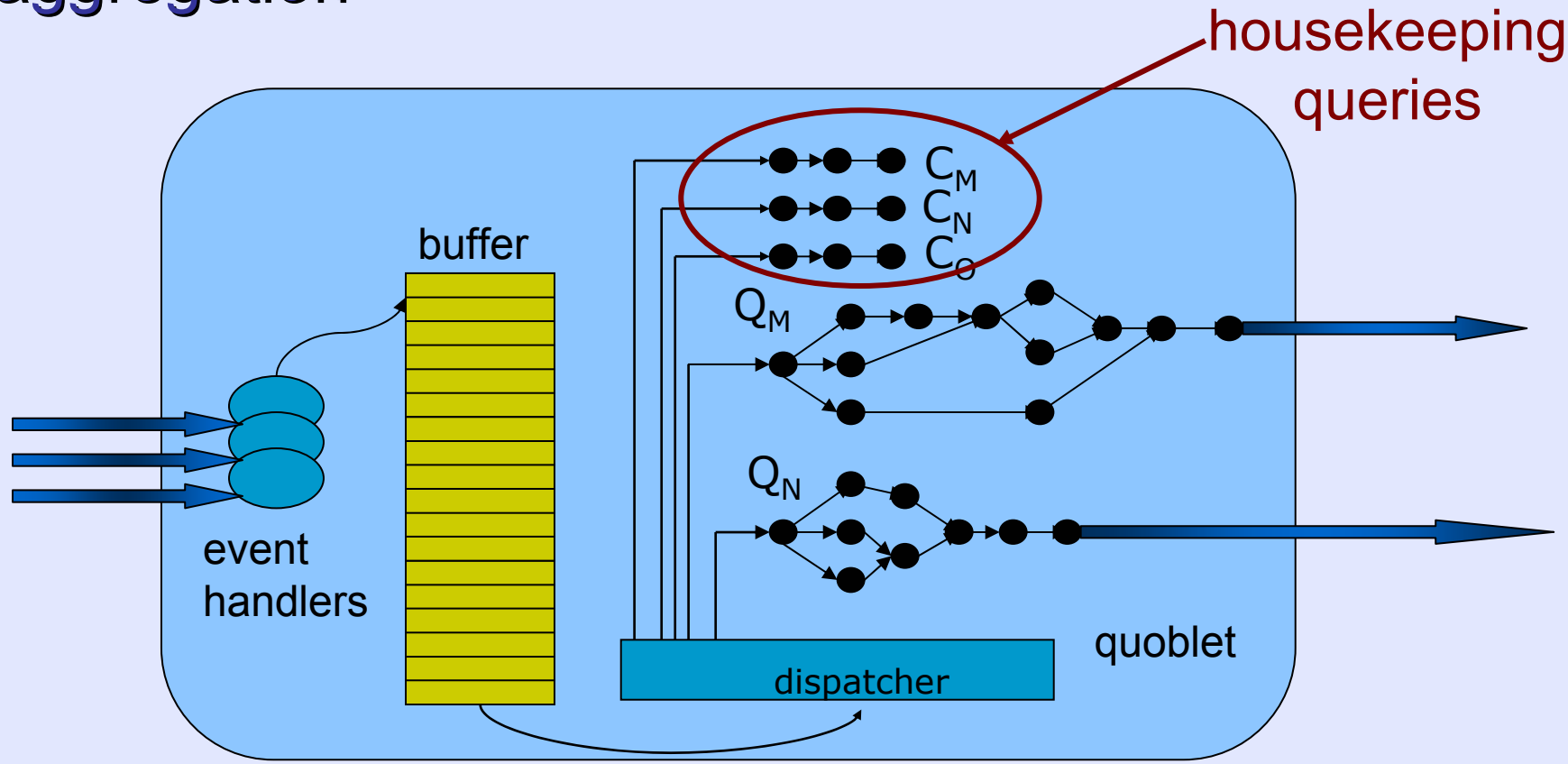
event arrival times

	:30	:28	:22	:18	:15	:12	:11	:10	:07	:03	:01	
α:	:11	:10	:09	:08	:07	:06	:05	:04	:03	:02	:01	
β:	:03						:02				:01	

desired join window size: 30 seconds
timestamp_interval(α) = 10 seconds
timestamp_interval(β) = 2 seconds
max_timestamp_interval = 10 seconds
join_window(α) = (11 * 30) / 10 = 33 events *
join_window(β) = (3 * 30) / 10 = 9 events

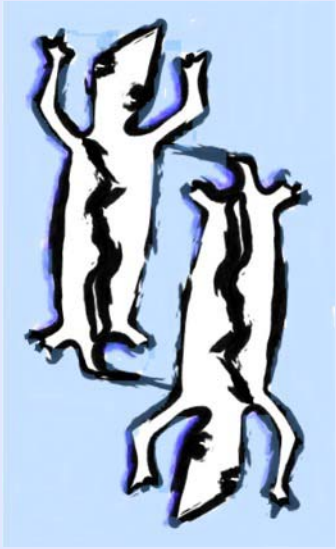
33 events == 30 sec. in wallclock time and 10 seconds in timestamp time

Introspection implemented by new "housekeeping operators" for sampling and aggregation



Current and Future Work

- Complete experimental evaluation of optimized join window algorithm.
- Port to grid services architecture
- Extend operator support to include stream tasks such as file decompress
- Roll current user-defined ‘action’ into supported set of user defined functions
- Determine probability assessment of likelihood of false negatives on a query.
  - When user sets window size of .01 second, we can return warning “probability of false negatives is 90%”



http://www.cs.indiana.edu/~plale/projects/dQUOB