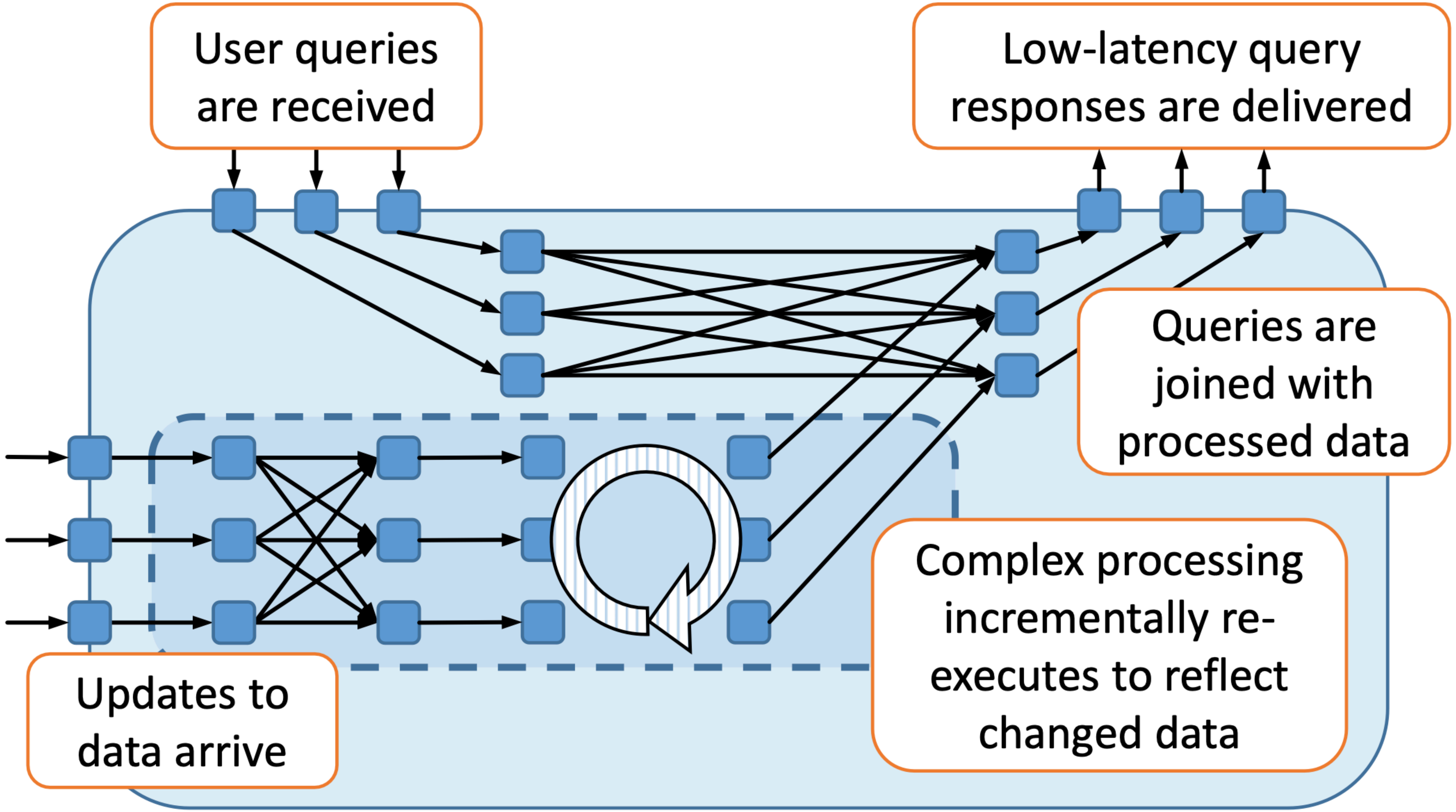
Naiad: A Timely Dataflow System

Daniel Vlasits


User queries are received
Low-latency query responses are delivered
Queries are joined with processed data
Complex processing incrementally re-executes to reflect changed data
Updates to data arrive

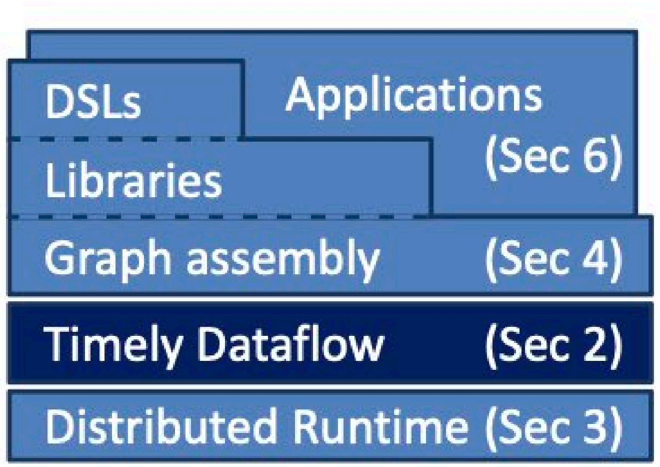
What is it?

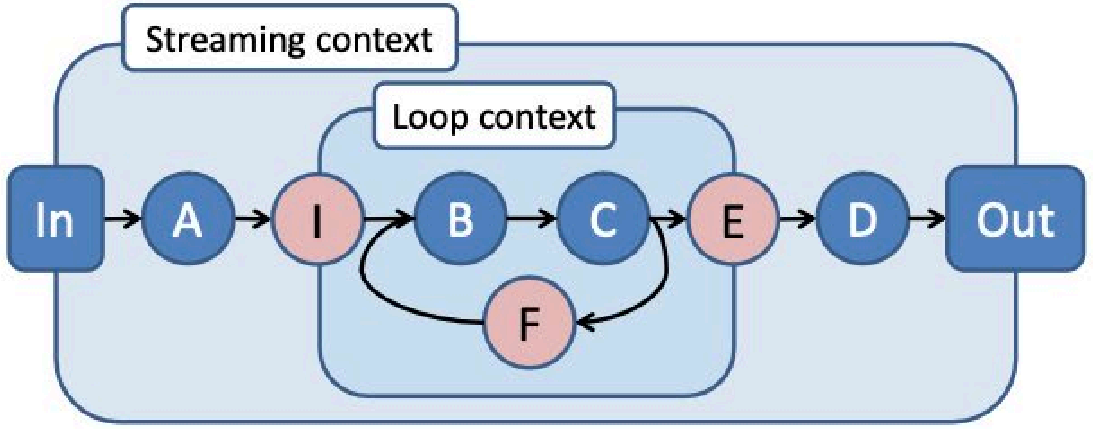
- High throughput
- Low latency
- Iterative + Incremental

- Examples
  - MapReduce
  - Time Stream
  - Giraffe

Timely Dataflow

- Developed to handle continuous data to be processed fast





$v.\textsc{OnRecv}(e : \text{Edge},\ m : \text{Message},\ t : \text{Timestamp})$
$v.\textsc{OnNotify}(t : \text{Timestamp})$.

A vertex may invoke two system-provided methods in the context of these callbacks:

$\textit{this}.\textsc{SendBy}(e : \text{Edge},\ m : \text{Message},\ t : \text{Timestamp})$
$\textit{this}.\textsc{NotifyAt}(t : \text{Timestamp})$.

Example Program – Incrementally updatable MapReduce

```
// 1a. Define input stages for the dataflow.
var input = controller.NewInput<string>();

// 1b. Define the timely dataflow graph.
// Here, we use LINQ to implement MapReduce.
var result = input.SelectMany(y => map(y))
                  .GroupBy(y => key(y),
                     (k, vs) => reduce(k, vs));

// 1c. Define output callbacks for each epoch
result.Subscribe(result => { ... });

// 2. Supply input data to the query.
input.OnNext(/* 1st epoch data */);
input.OnNext(/* 2nd epoch data */);
input.OnNext(/* 3rd epoch data */);
input.OnCompleted();
```

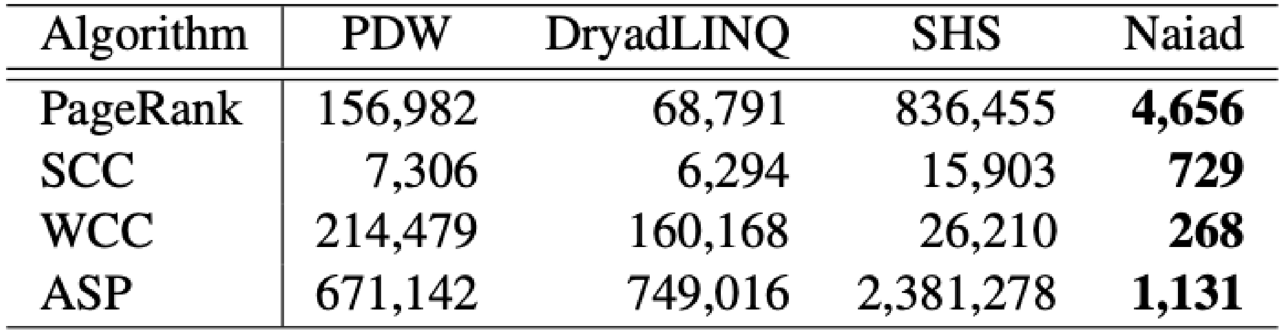
- Constructing the graph

Metrics Considered

- Throughput
- Latency
- Scaling

- Very impressive

- Example given is computing the most popular hashtag.



Algorithm	PDW	DryadLINQ	SHS	Naiad
PageRank	156,982	68,791	836,455	**4,656**
SCC	7,306	6,294	15,903	**729**
WCC	214,479	160,168	26,210	**268**
ASP	671,142	749,016	2,381,278	**1,131**

Table 1: Running times in seconds of several graph algorithms on the Category A web graph. Non-Naiad measurements are due to Najork *et al.* [34].

The Diversity of Applications

- WordCount – Computing word frequencies of Twitter Corpus
- WCC – weakly connected components

- Does a reasonable job of scaling
- Batch computation
- Streaming Computation
- Graph Computation
- All can be expressed at high-level using the Naiad framework

Takeaways

- Decoupling high-level ideas from low-level implementations
- Very fast
- Useful for an incredibly wide range of applications
- Just because we've just covered garbage collection in PL – specific effort made to make the life of collector as easy as possible
- Tradeoffs hardly mentioned, but they prefer performance over restoring from crash – less logging more updating – spark is better