Streaming

COS 518: Advanced Computer Systems
Lecture 11

Daniel Suo

What is streaming?

- Fast data!
- Fast processing!
- Lots of data!

Streaming = unbounded data

(Batch = bounded data)

Other defns are somewhat misleading

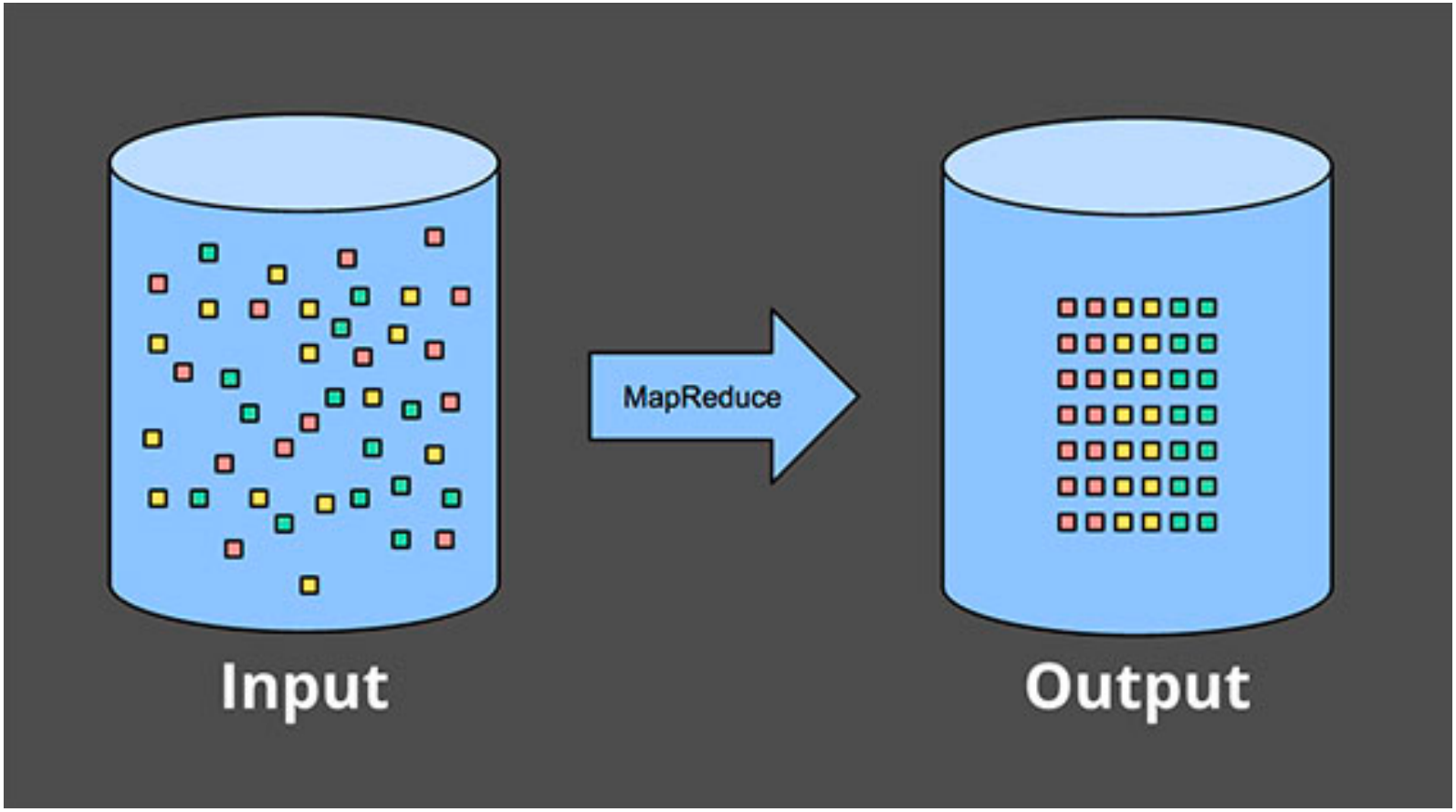
- We can use batch frameworks for stream processing (how?)
- Batch frameworks can also handle scenarios historically covered by stream frameworks (e.g., low-latency, approximate)

Three major challenges

- **Consistency**: historically, streaming systems were created to decrease latency and made many sacrifices (at-most-processing, anyone?)
- **Throughput vs. latency**: typically a trade-off (why?)
- **Time**: as we will soon see, streaming introduces some new challenges

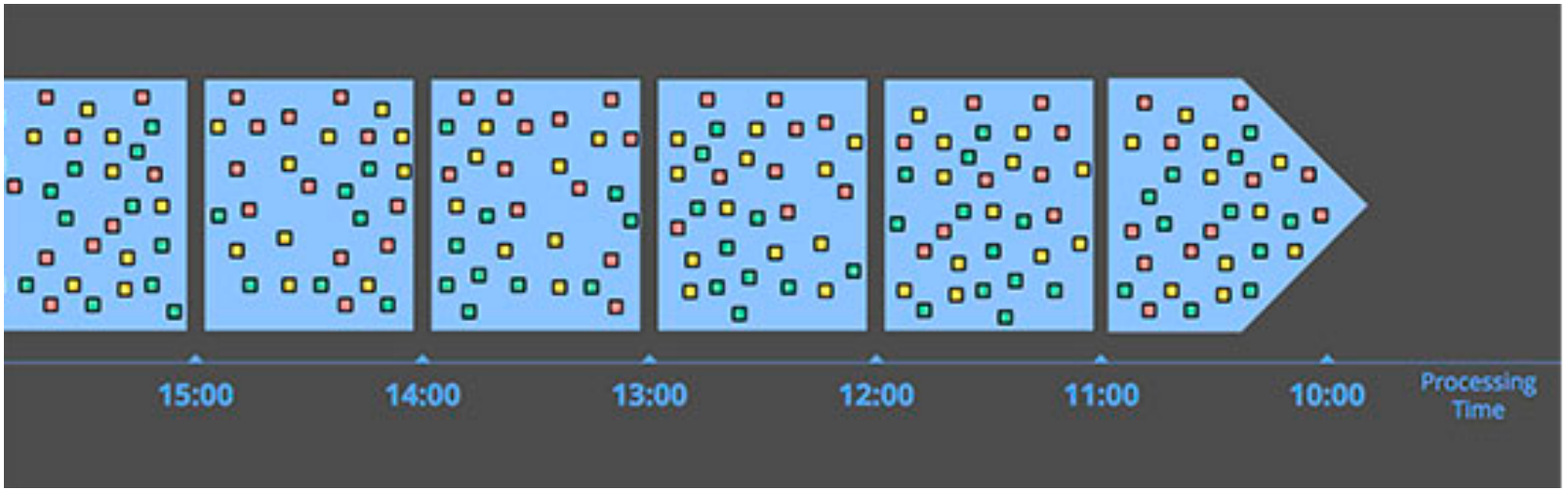
We've covered consistency in a lot of detail, so let's investigate time.

Our lives used to be easy…



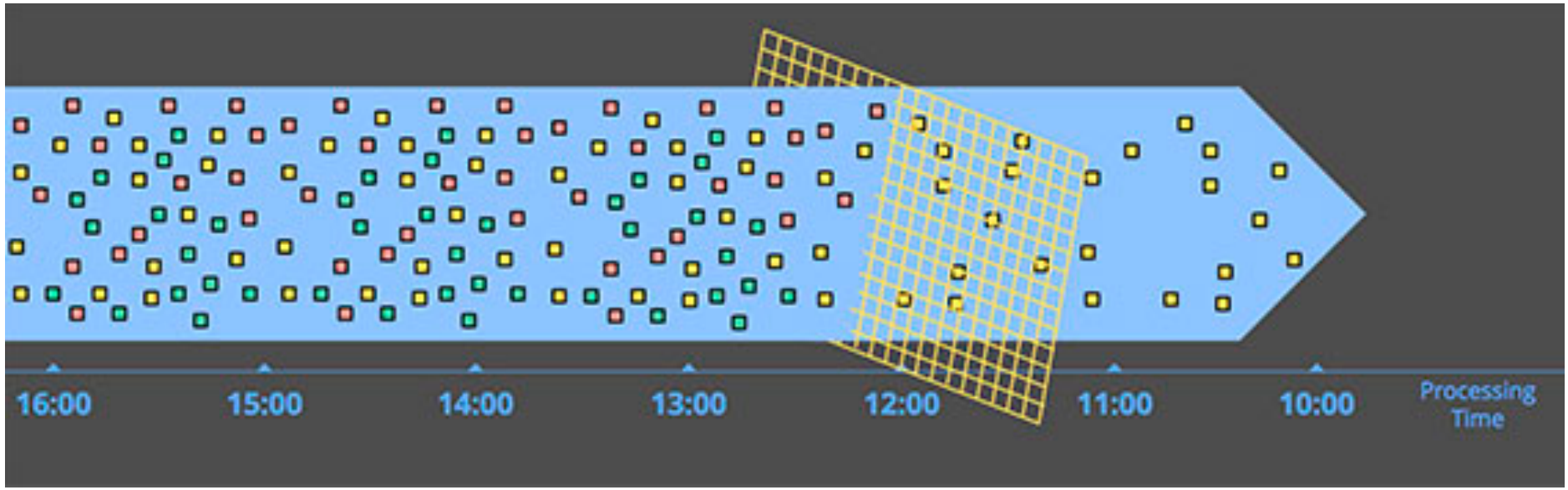
…but if you give a data scientist some data…

- Once we move to unbounded data, we need new methods to process whether for sake of capacity (not enough machines) or availability (data doesn't exist yet)
- Easiest thing to do:



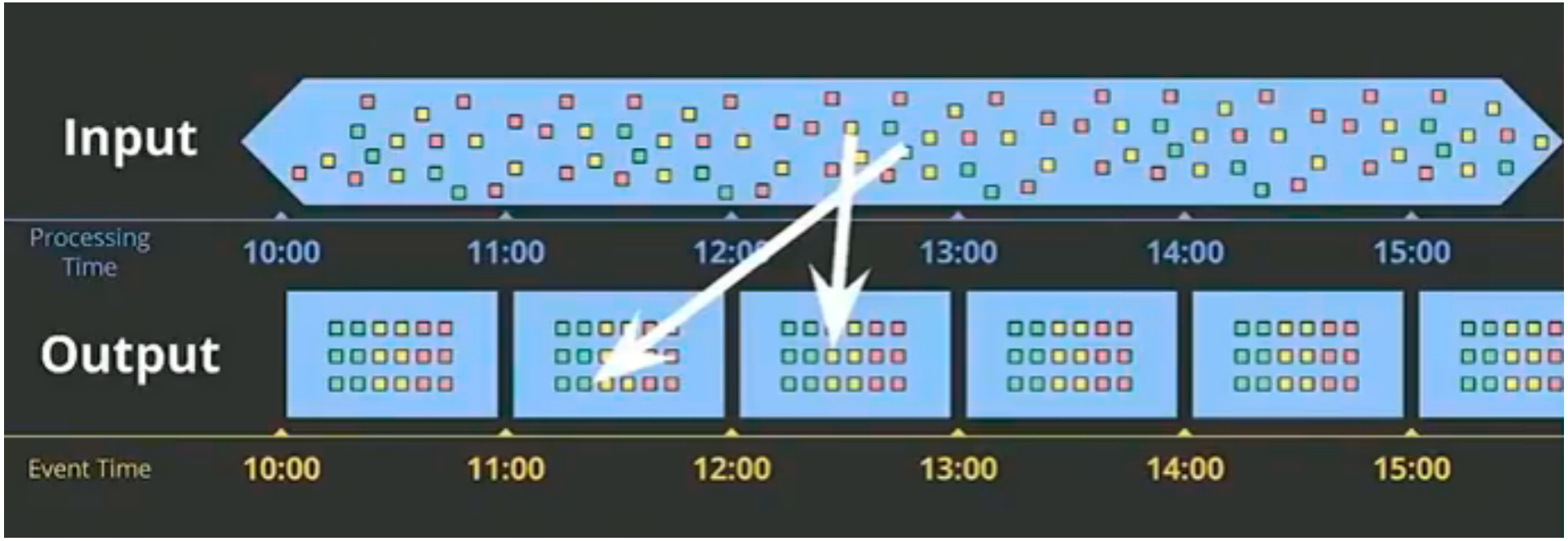
Windowing by processing time is great

- Easy to implement and verify correctness
- Great for applications like filtering or monitoring

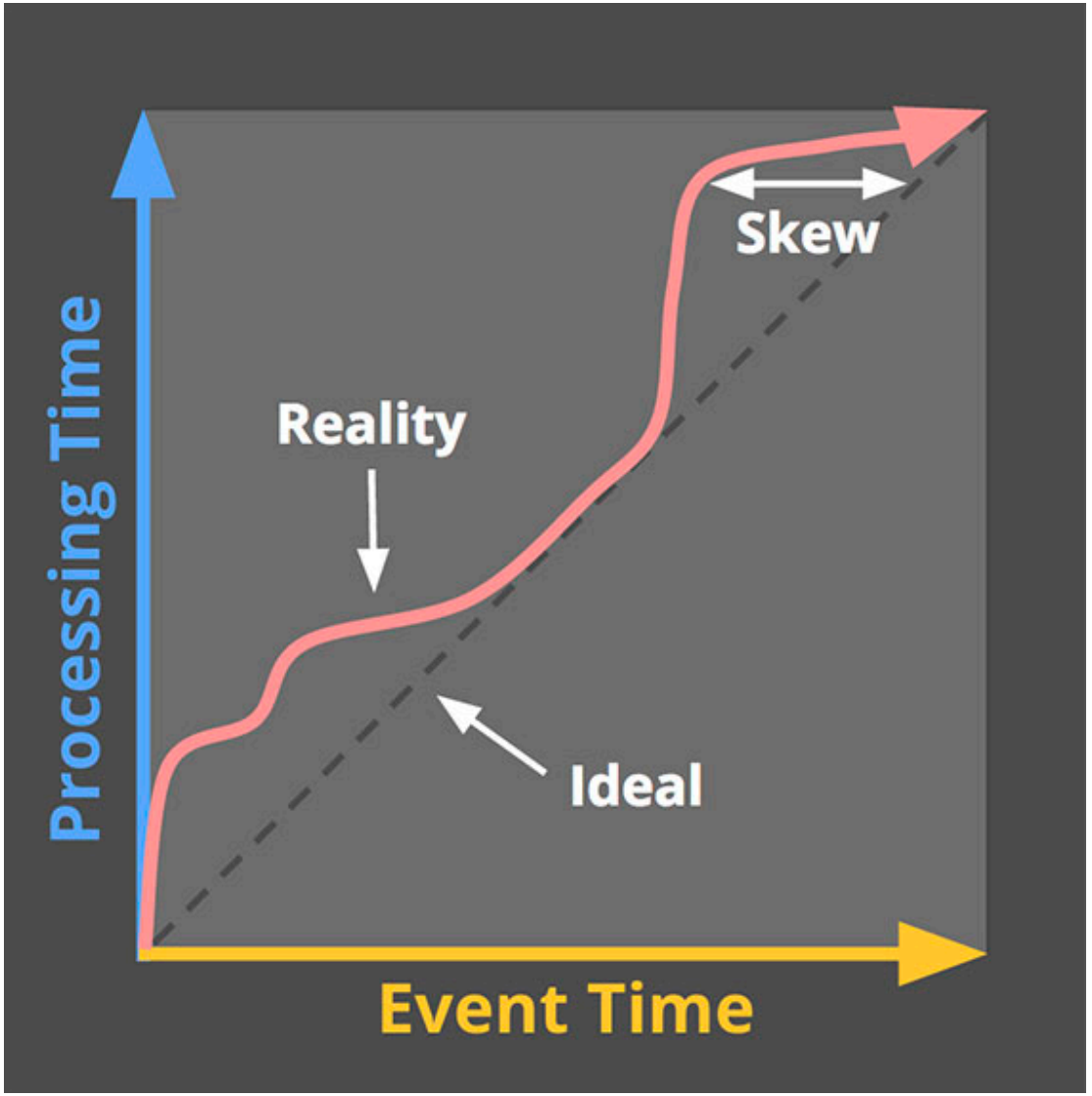


But what if we care about *when* events happen?

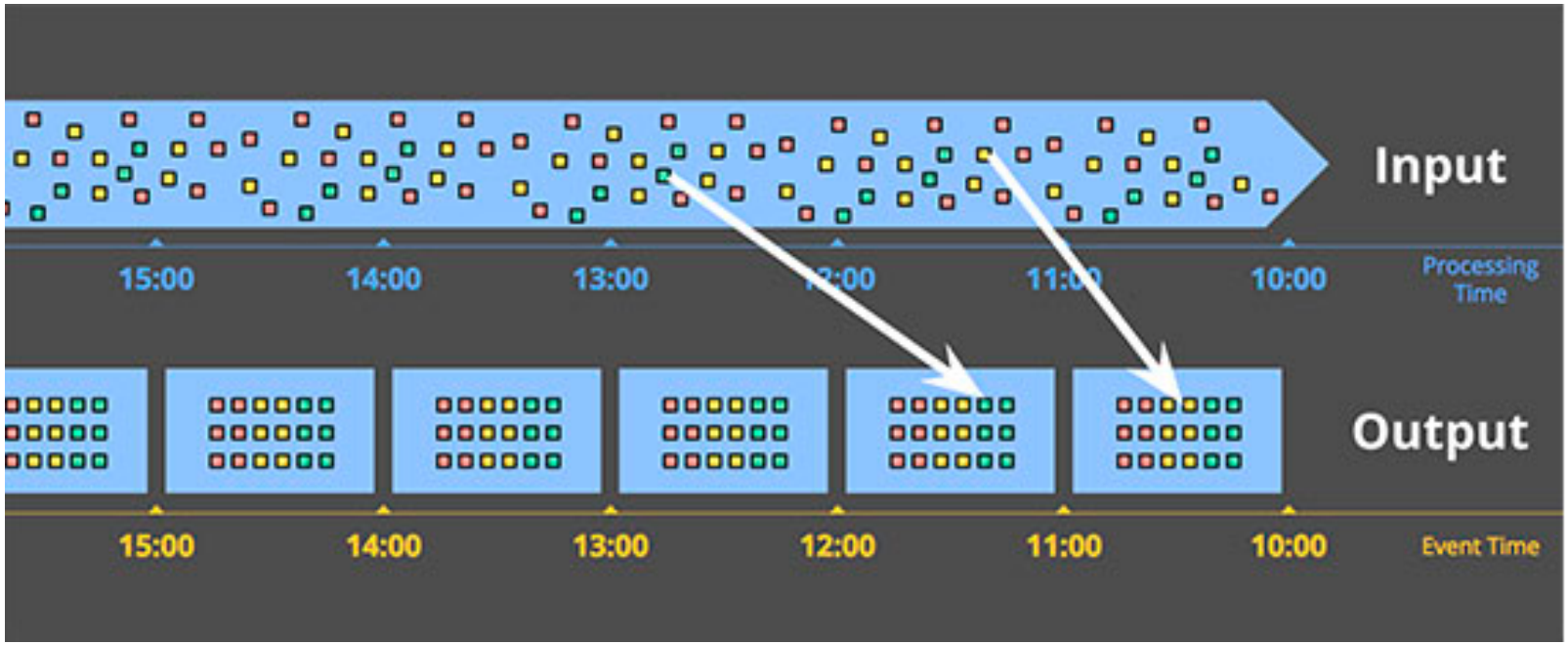
- If we associate event times, then items could now come out-of-order! (why?)



Time creates new wounds



This would be nice



But not the case, so we need tools

- **Windows**: how should we group together data?
- **Watermarks**: how can we mark when the last piece of data in some window has arrived?
- **Triggers**: how can we initiate an early result?
- **Accumulators**: what do we do with the results (correct, modified, or retracted)?

All topics covered in next week's readings!