SAXONICA.COM
XSLT AND XQUERY PROCESSING

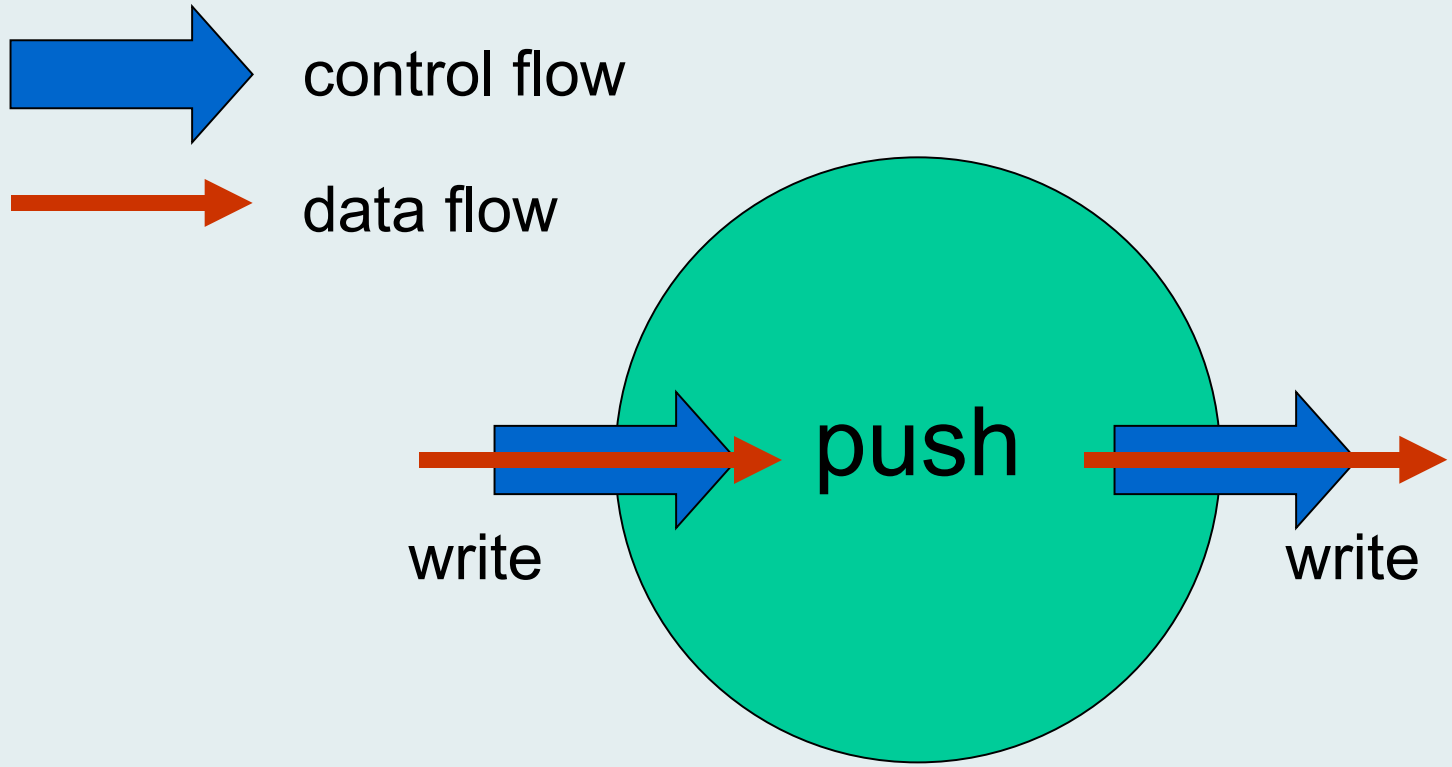
You Pull, I'll Push: on the Polarity of Pipelines

Michael Kay, Saxonica

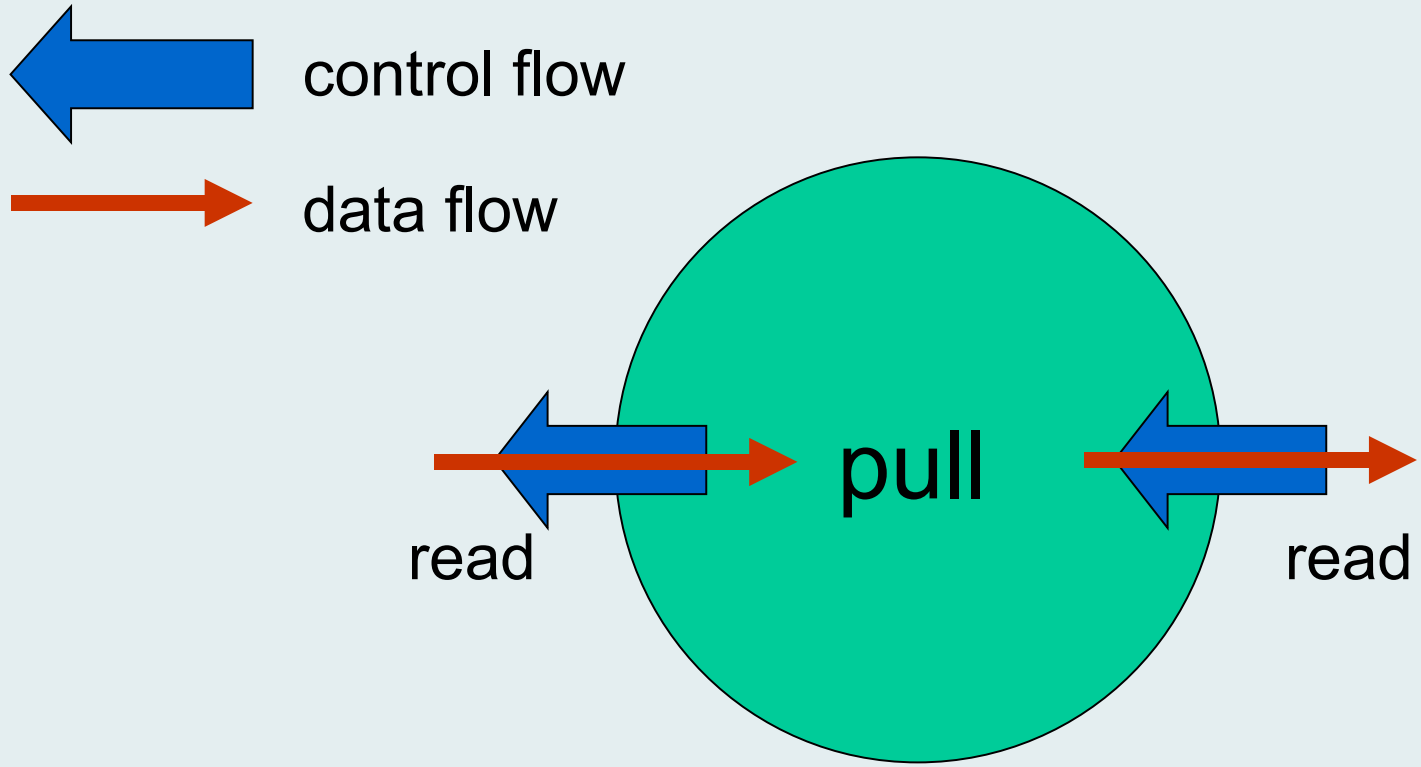
Why pipelines?

- Keeps code simple
- Components are reusable
- Clean way of mixing different technologies
- Flexible deployment
  - distributed
  - asynchronous

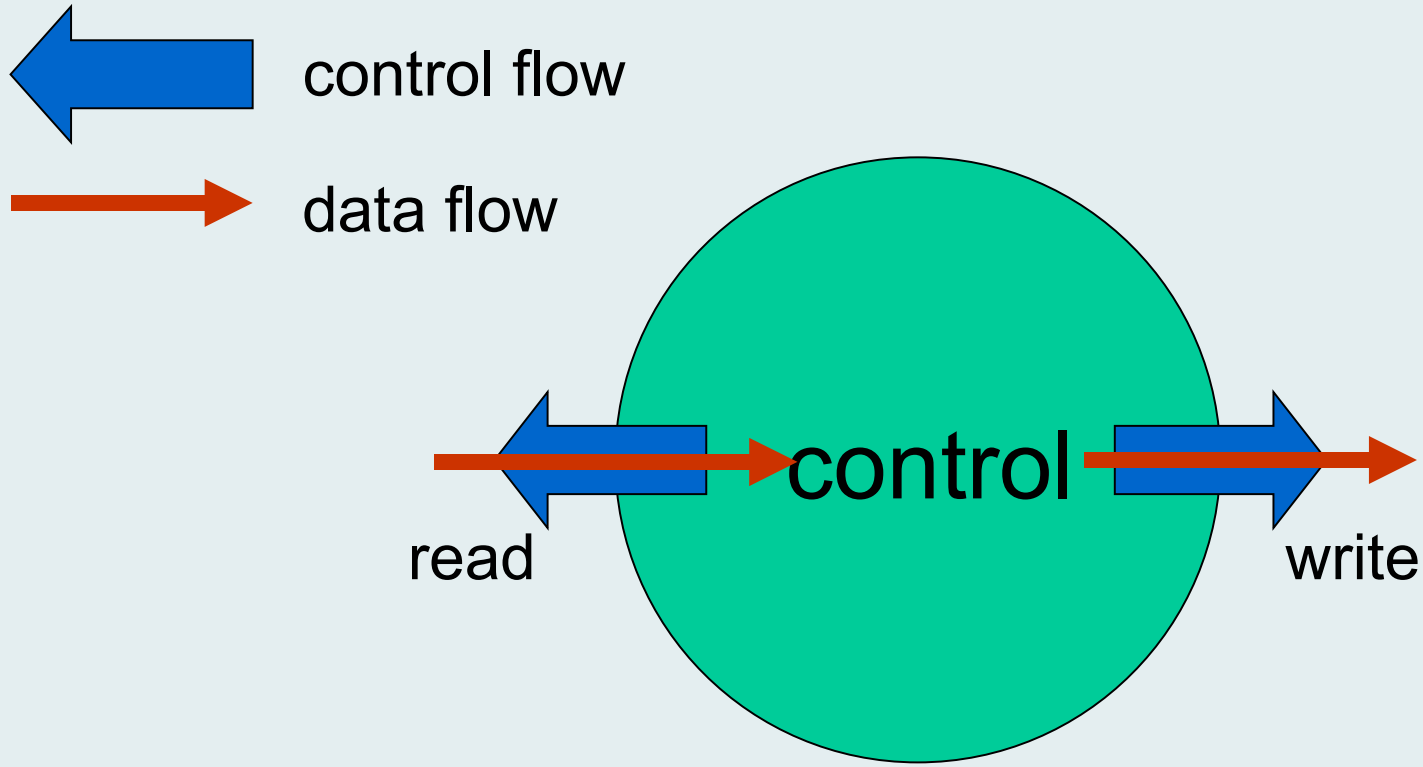
Push components



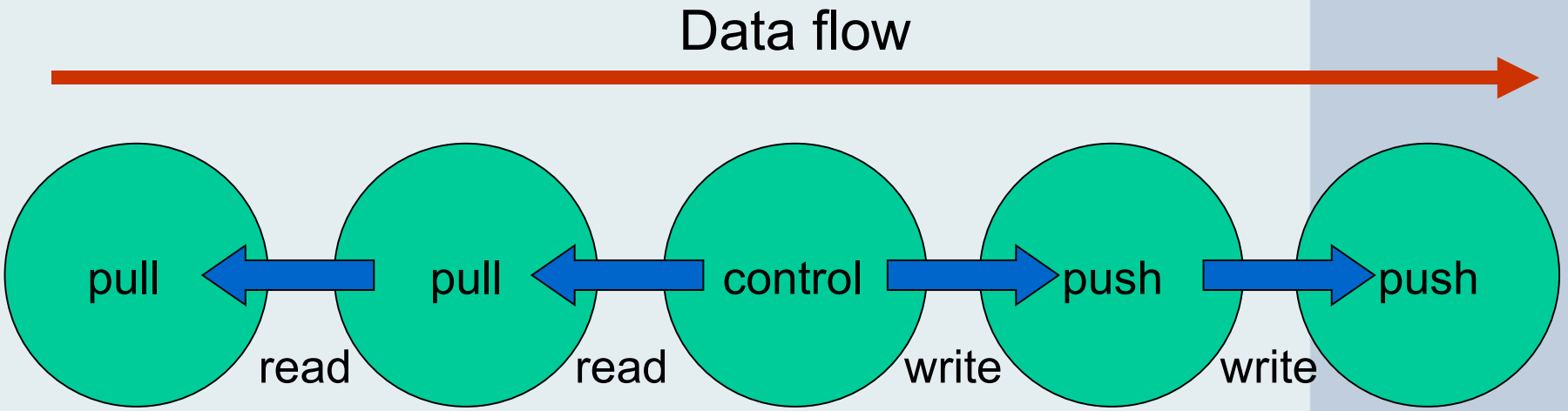
Pull components



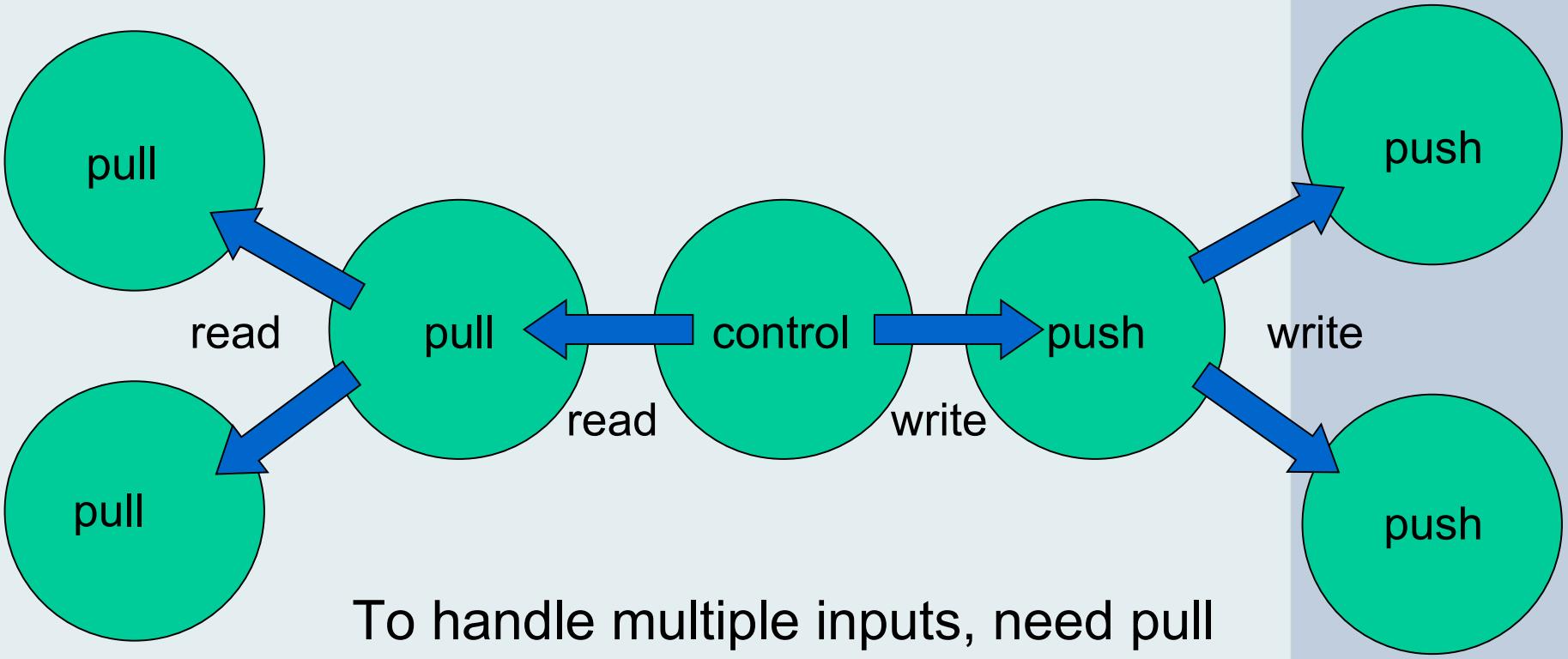
Main-loop components



A fully-streaming pipeline



Branching and Merging



To handle multiple inputs, need pull

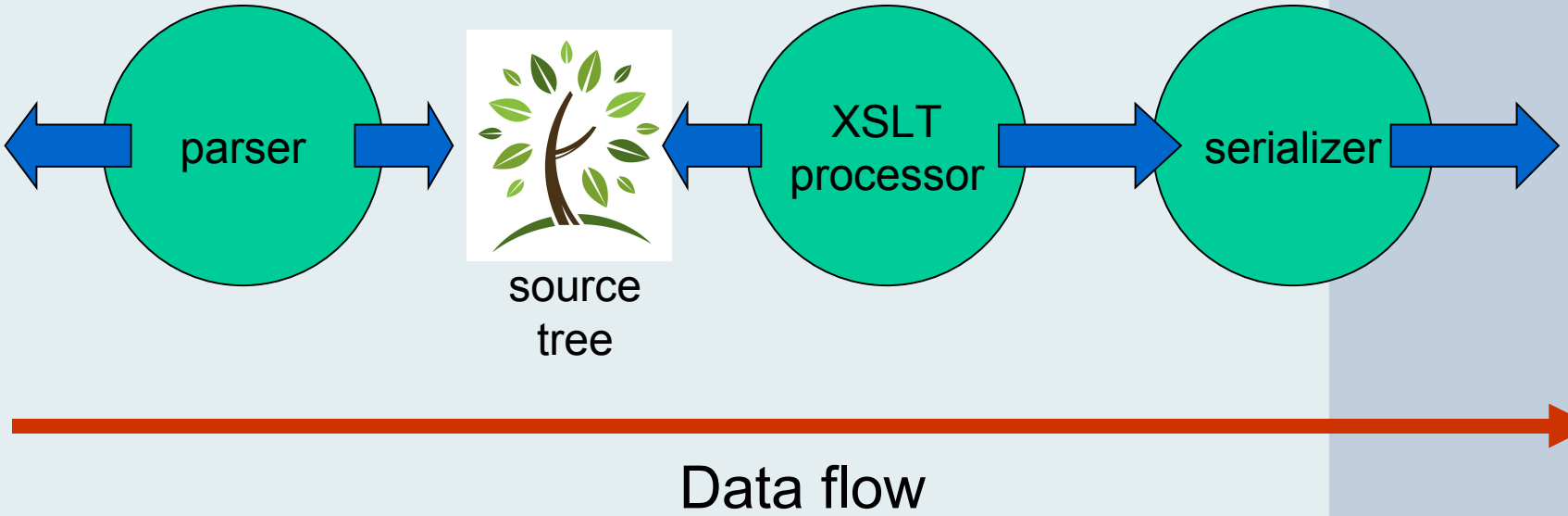
To handle multiple outputs, need push

Data flow

Pipeline Granularity

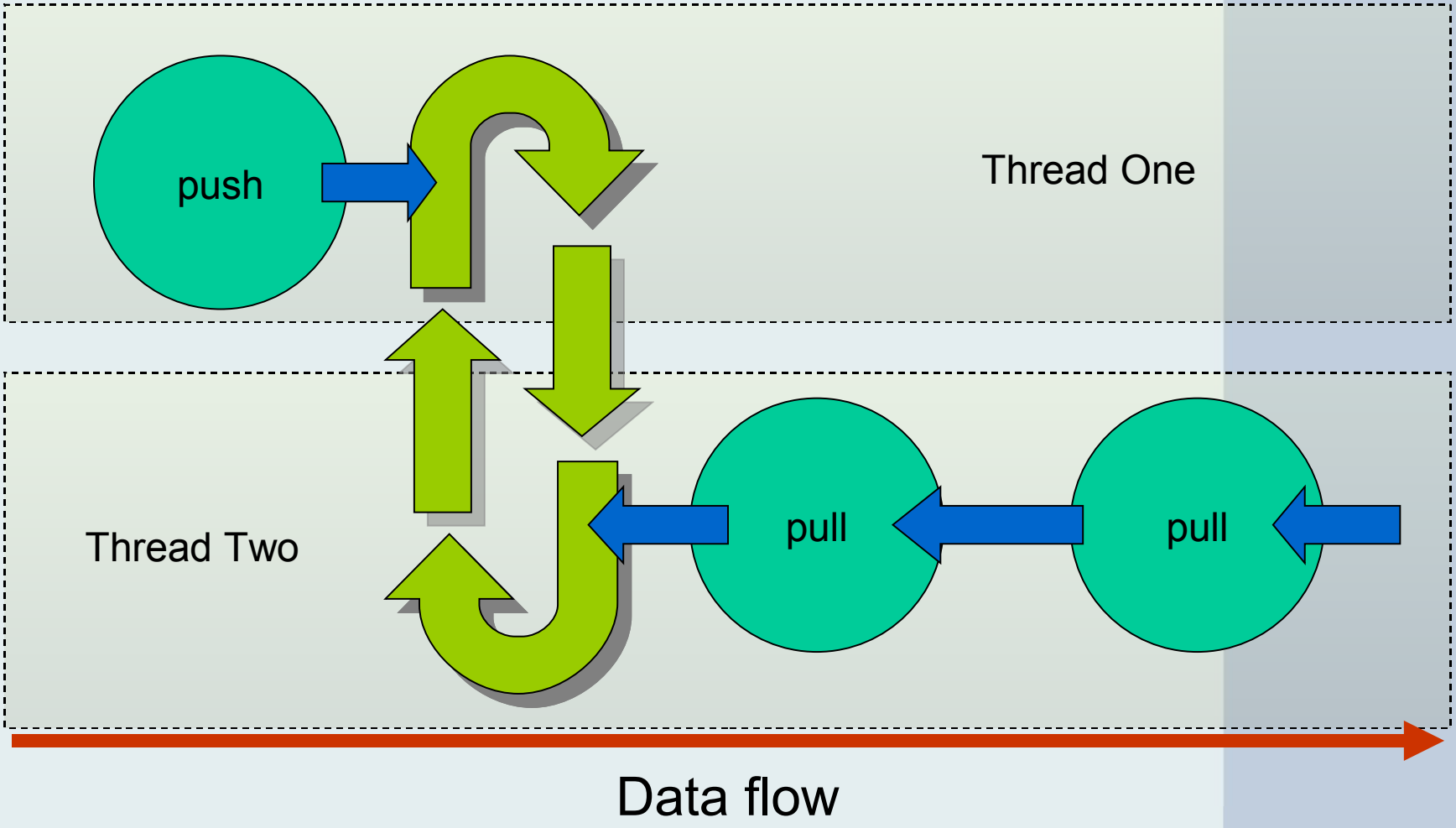
- Event granularity
  - startElement/endElement
- Item granularity
  - node or atomic value
- Operations
  - compose (events -> items)
  - decompose (items -> events)

When pull comes to shove (1) build a tree in memory

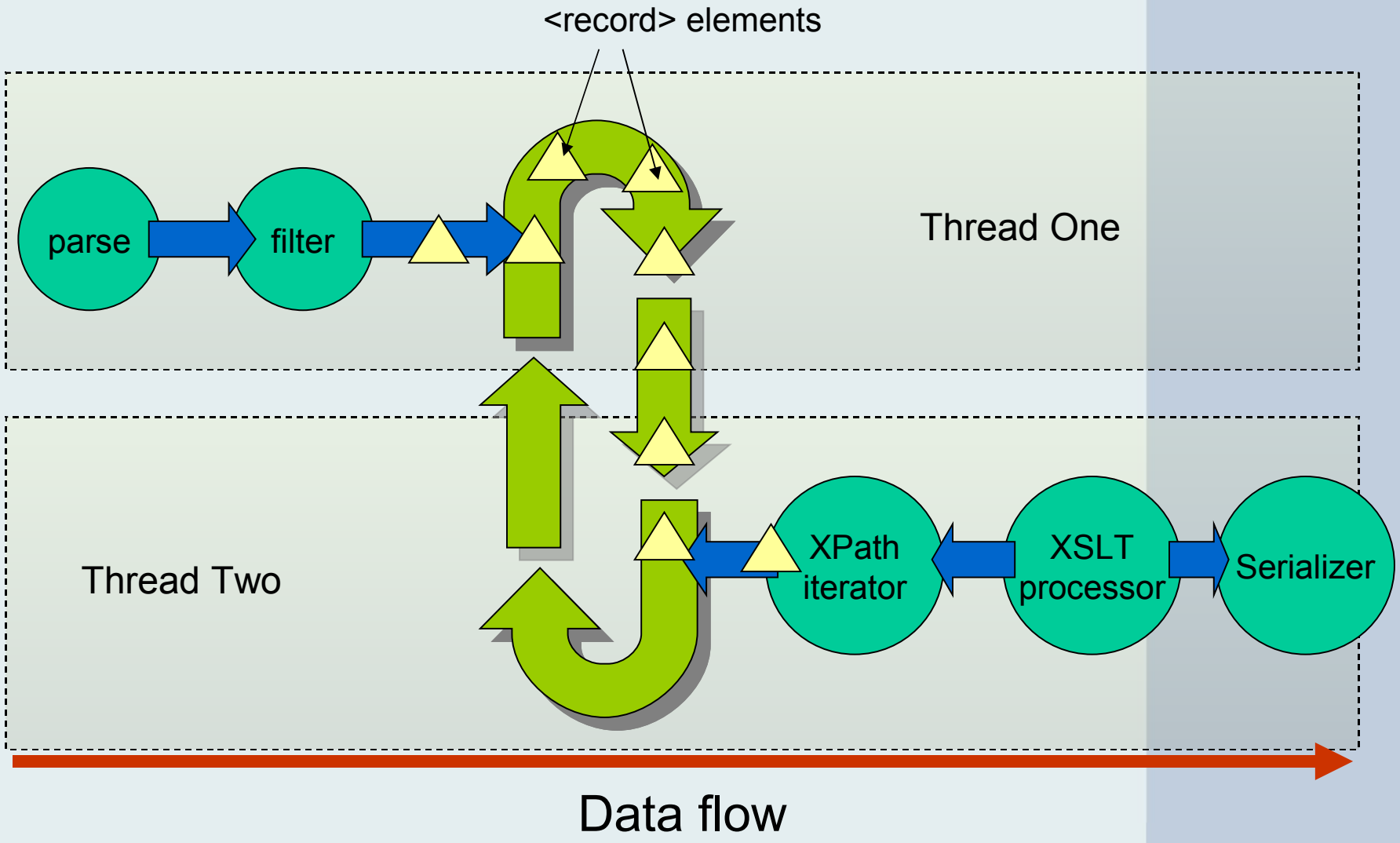


Note: the tree is not only needed because there is a polarity conflict, but also because there is an ordering conflict: many stylesheets do not consume data in the order the parser delivers it.

When pull comes to shove
(2) Two threads and a cyclic buffer



saxon:stream(doc('a.xml')/*/record)



Why is the filter a push filter?

- Answer: Reuse!
  - it was written initially as part of the schema validator (id/key/keyref)
  - the schema validator is a push pipeline
  - it's a push pipeline because it forks
- Message:
  - when you want to reuse components in a pipeline, you might find they have the wrong polarity.

Impact of push-pull conflicts

- Increase memory requirement
- Reduce latency
- Cited XQuery example:
  - increases elapsed time from 1926ms to 3496ms

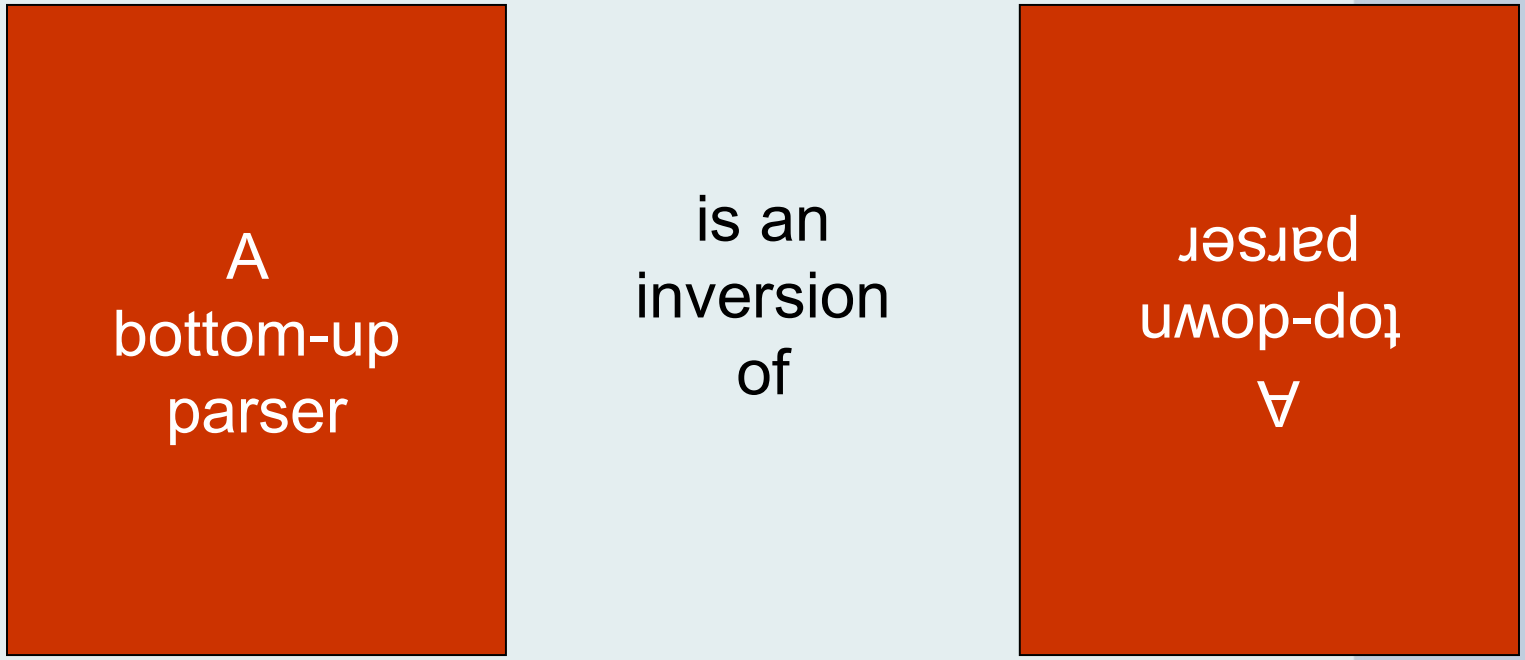
Co-routines

- Two programs each written **as if** they own the main loop
- One of them is **inverted** by the compiler into a push component
- Doesn’t need two threads: only two stacks.

Jackson Structured Programming and the concept of Inversion

- Designed for batch magnetic tape data processing
- How to avoid writing the output of one processing phase to tape, to be read by the next phase?
- Answer: invert the second phase, so it is activated each time data becomes available
- Designed to work with hierarchic data

Compiler analogy



How inversion works

- Replace read() call with
  - save stack
  - save execution point
  - exit
- Add method reenter(X)
  - restore stack
  - resume at saved execution point

Proposition

An XSLT compiler
can generate push or pull code
interchangeably
by applying the technique of inversion

Benefits

- Less turbulence for the intra-stylesheet pipeline
  - (less need to build temporary trees)
- The stylesheet itself can be compiled as a push or pull component for inclusion in an XProc pipeline
- Increased component reusability