We are all to gather

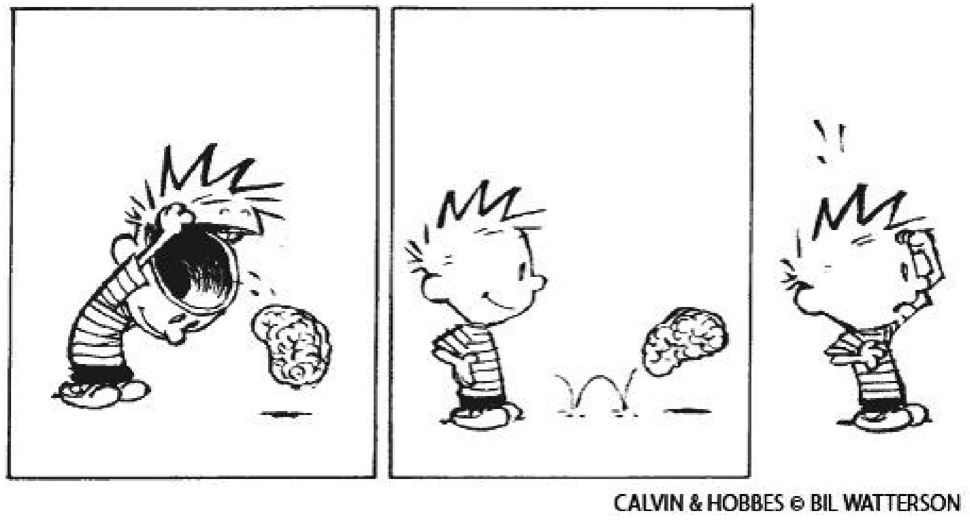
Rémi Forax
Université Gustave Eiffel – January 2024

We are all together

Rémi Forax
Université Gustave Eiffel – January 2024

We are all to **gather**

Rémi Forax
Université Gustave Eiffel – January 2024



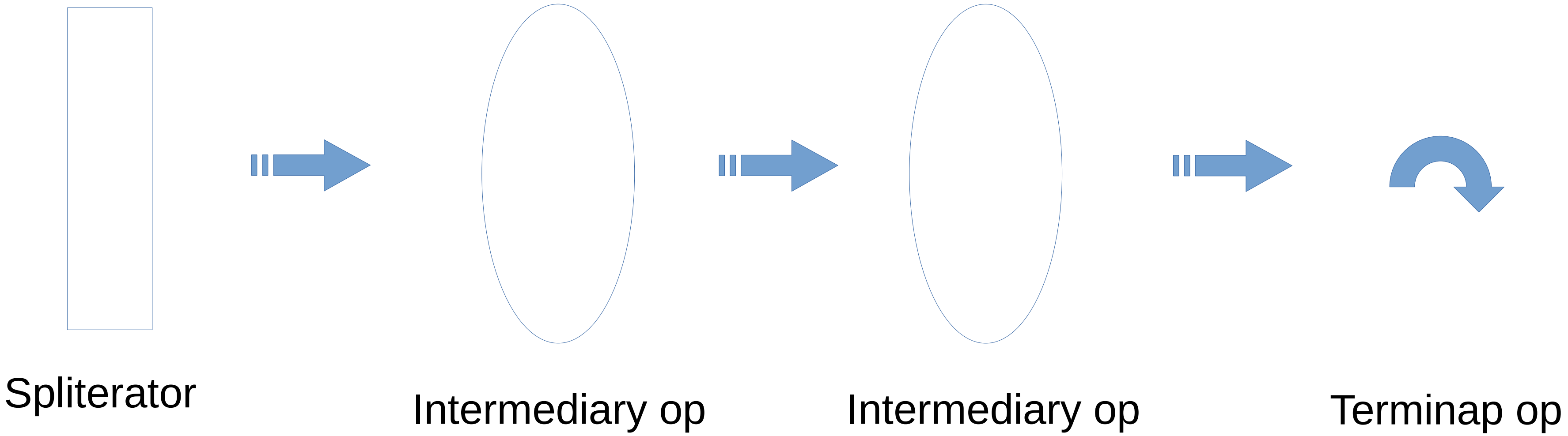
Don’t believe what I’m saying !

Outline

- Stream operations
- The Gatherer API
- Performance and limitations

Stream == pipeline

The terminal operation drives the pipeline



Intermediary Ops

3 axis

- Can stop the computation ? greedy/short-circuiting
- Have an internal state ? stateless/stateful
- Can be parallelizable ? sequential/parallel

Intermediary Ops (examples)

- Operations
  - map() ??

Intermediary Ops (examples)

- Operations
  - map() greedy, stateless, parallelizable

Intermediary Ops (examples)

Operations

- map() greedy, stateless, parallelizable
- filter() ??

Intermediary Ops (examples)

Operations

- map() greedy, stateless, parallelizable
- filter() greedy, stateless, parallelizable
- takeWhile() ??

Intermediary Ops (examples)

Operations

- map() greedy, stateless, parallelizable
- filter() greedy, stateless, parallelizable
- takeWhile() short-circuit, stateless, sequential
- limit() ??

Intermediary Ops (examples)

Operations

- map() greedy, stateless, parallelizable
- filter() greedy, stateless, parallelizable
- takeWhile() short-circuit, stateless, sequential
- limit() short-circuit, stateful, sequential
- reduce() ??

Intermediary Ops (examples)

Operations

- map() greedy, stateless, parallelizable
- filter() greedy, stateless, parallelizable
- takeWhile() short-circuit, stateless, sequential
- limit() short-circuit, stateful, sequential
- reduce() greedy, stateful, parallelizable

Live Code !

Gatherer API

Gatherer<E, A, T>

- Initializer: Supplier<A>
  - Create a state
- Integrator (A state, E element, Downstream<T> downstream) → boolean
  - Accumulate in state and push downstream (back-propagate return type)
- Combiner: BinaryOperator<A>
  - Combine two states, return a new state
- Finisher: BiConsumer<A, Downstream<T>>
  - push downstream

Gatherer API : 3 axis

- Greedy / Short-circuit
  - Integrator.ofGreedy / Integrator
- Stateless / Stateful
  - Integrator / Initializer + Integrator + Finisher?
- Sequential / Parallel
  - Gatherer.ofSequential() / Gatherer.of() + Combiner?

What’s missing ?

Performance ?

Performance issues

No primitive specialization

- mapToInt/flatMapToInt, etc
  - Same issue with collectors
  - Valhalla generics to the rescue ?

Spliterator characteristics are not propagated

- Same issue with collectors
  - For ex: Stream.toList() can presize, not Collectors.toList()

Executive Summary

Gatherer API

User defined intermediary operations

- 3 axis: short-circuitability / statefulness / parallelizable

Gatherers contains predefined Gatherers

Still In preview

– Not enough predefined Gatherers

– Spliterator characteristics should be propagated

– “default operations” design is controversial