GenStage, Flow & Broadway

Hands on with Flow

Introduction

GenStage

GenStage is a specification for exchanging events between producers and consumers.

Flow

Building computational flows using map-reduce, partitions, windows, and more that run concurrently

Broadway

Building concurrent and multi-stage data ingestion and data processing pipelines to consume events from Amazon SQS, RabbitMQ, and others

The naive approach

Word counting with Enum

```
File.stream!("path/to/some/file")
|> Enum.flat_map(&String.split(&1, " "))
|> Enum.reduce(%{}, fn word, acc ->
  Map.update(acc, word, 1, & &1 + 1)
end)
|> Enum.to_list()
```

Problems: Large memory and no concurrency

The naive approach

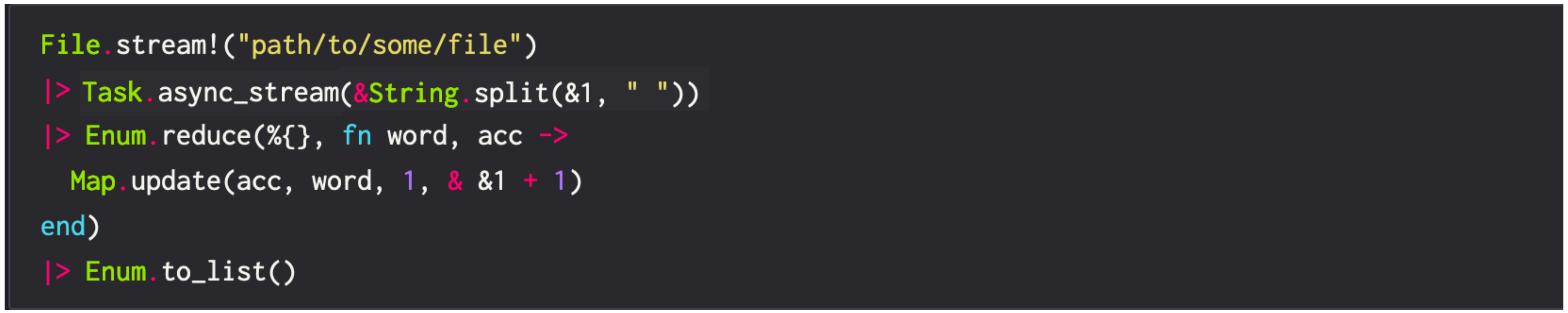
Word counting with Steam

```
File.stream!("path/to/some/file")
|> Stream.flat_map(&String.split(&1, " "))
|> Enum.reduce(%{}, fn word, acc ->
  Map.update(acc, word, 1, & &1 + 1)
end)
|> Enum.to_list()
```

Problems: No concurrency

The naive approach

Word counting with Task.async

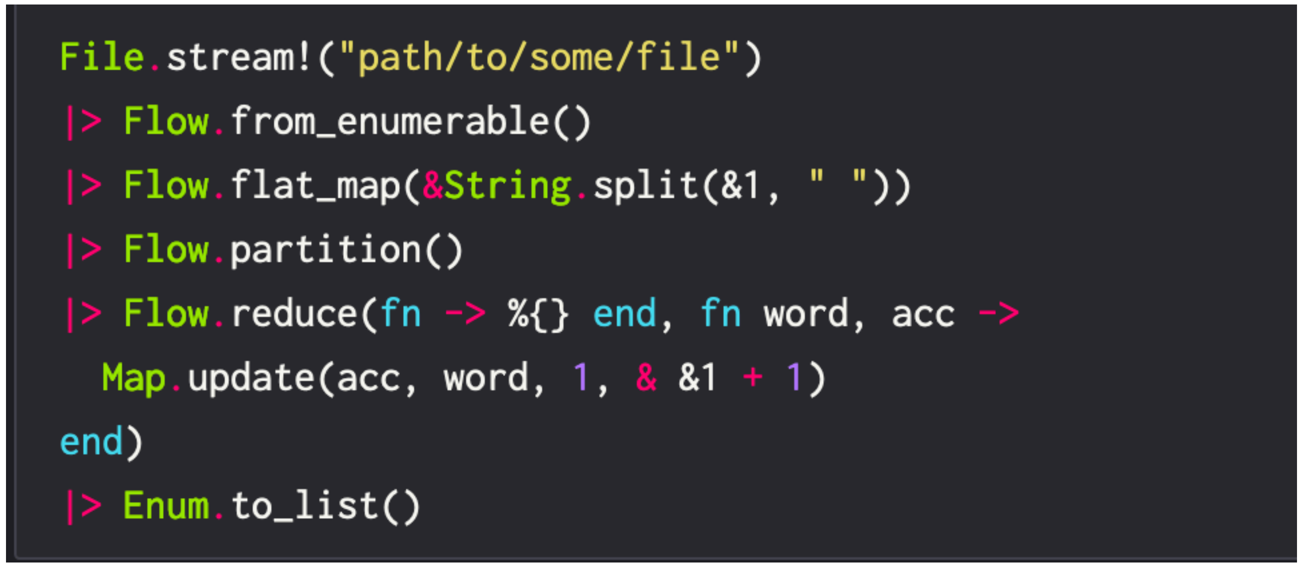


```
File.stream!("path/to/some/file")
|> Task.async_stream(&String.split(&1, " "))
|> Enum.reduce(%{}, fn word, acc ->
  Map.update(acc, word, 1, & &1 + 1)
end)
|> Enum.to_list()
```

Problems: Only partial concurrency

The Flow way

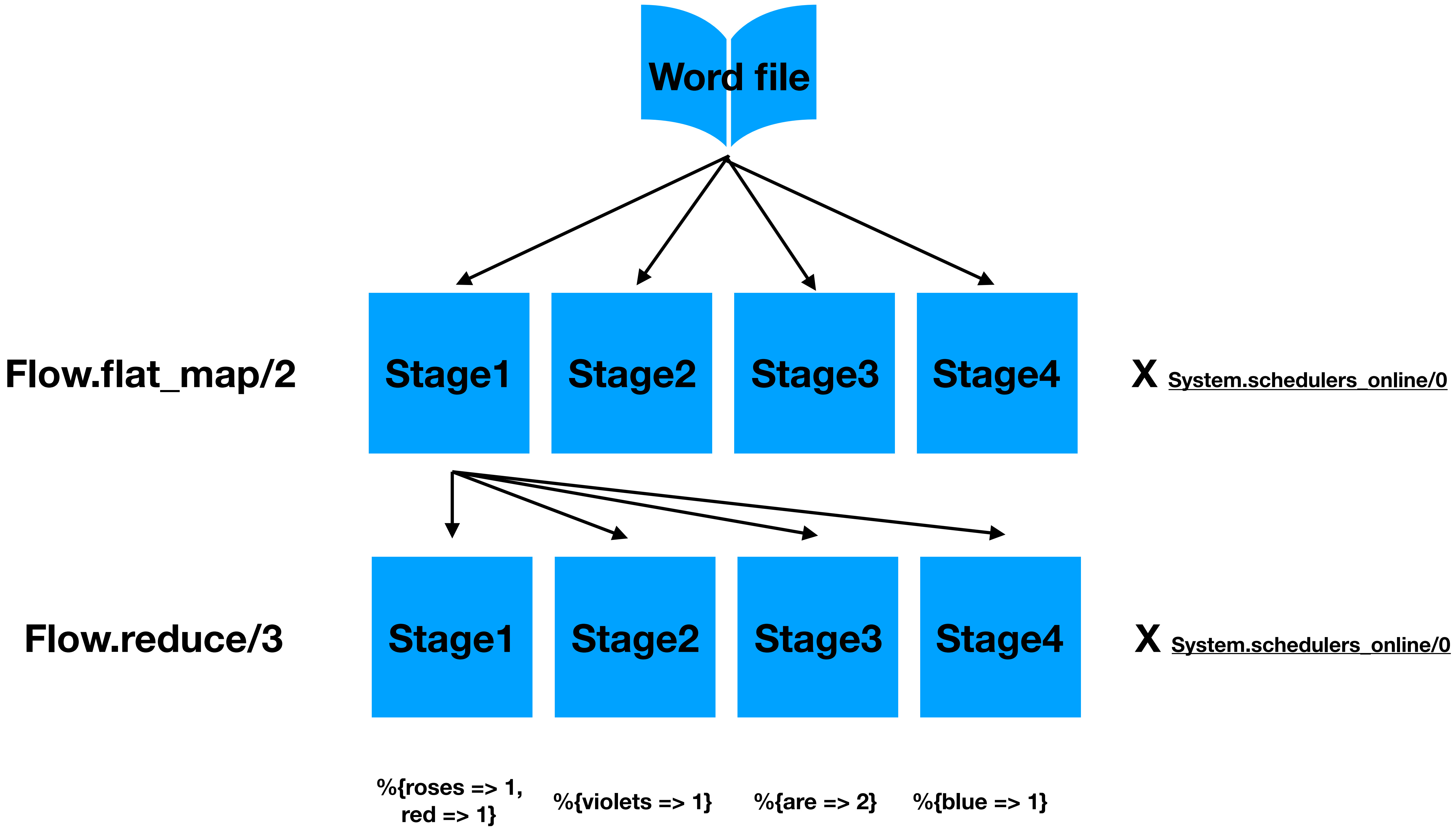
Word counting with Flow



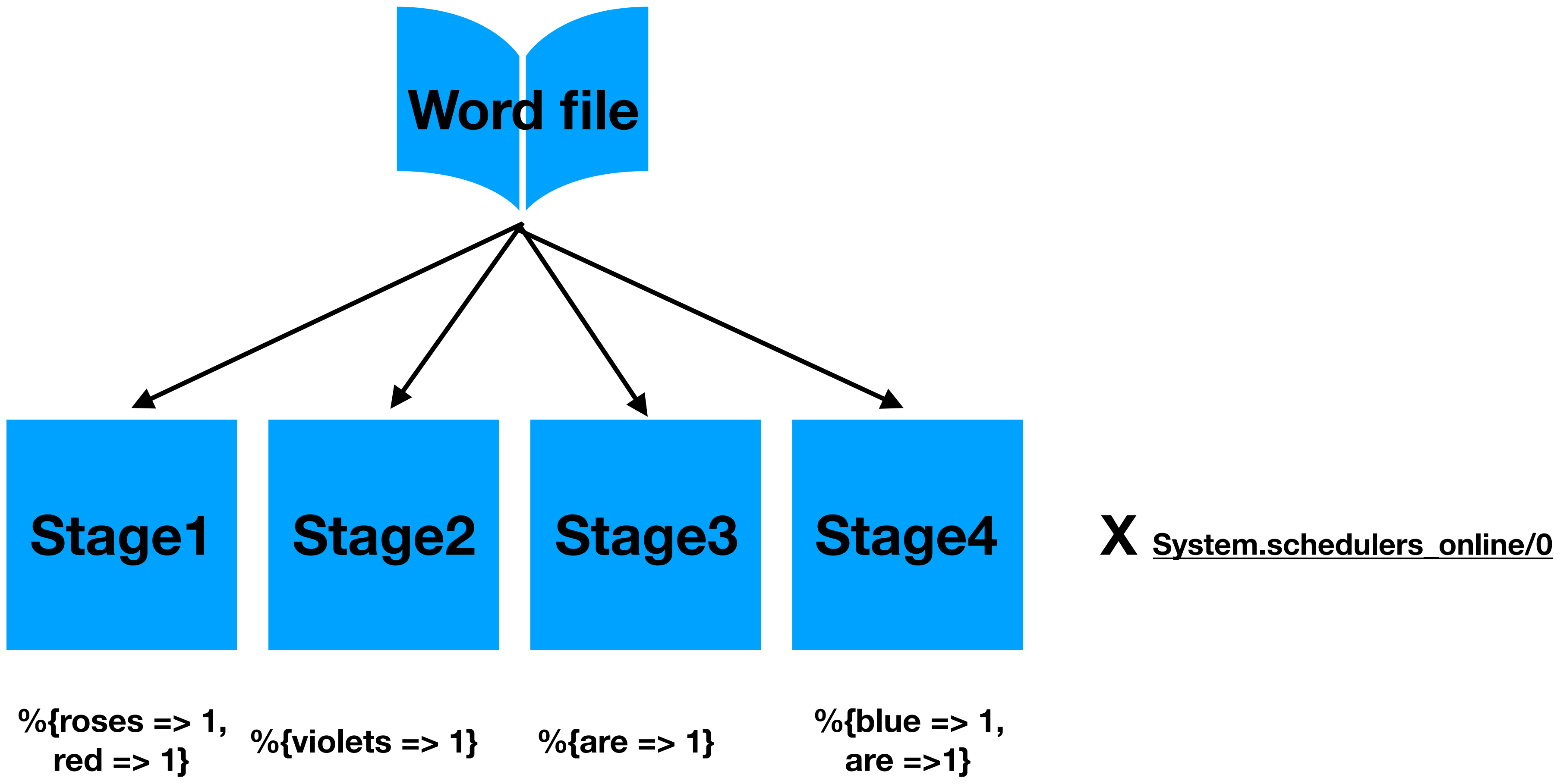
```
File.stream!("path/to/some/file")
|> Flow.from_enumerable()
|> Flow.flat_map(&String.split(&1, " "))
|> Flow.partition()
|> Flow.reduce(fn -> %{} end, fn word, acc ->
  Map.update(acc, word, 1, & &1 + 1)
end)
|> Enum.to_list()
```

Now have both fixed our memory and concurrency problem

How is it working?



Why do we need partitions?



Flow.flat_map/2

Flow.reduce/3

How to configure

Flow.partion & Flow.from_*

- :stages
- :max_demand
- :min_demand

Windows

- Global Window [Default]
- Fixed Windows (Event time)
- Periodic Windows (Processing time)
- Count Windows (event count)

Event window

```
iex> window = Flow.Window.count(10)
iex> flow = Flow.from_enumerable(1..100) |> Flow.partition(window: window, stages: 1)
iex> flow |> Flow.reduce(fn -> 0 end, &(&1 + &2)) |> Flow.emit(:state) |> Enum.to_list()
[55, 155, 255, 355, 455, 555, 655, 755, 855, 955, 0]
```

Flow.emit(:events | :state | :nothing)

Flow.on_trigger(state -> {elements , acc})

Flow.start_link()

Time to get **your** hands dirty

https://github.com/Hanspagh/GenStagePlayground